End-to-end Service Failure Diagnosis Using Belief Networks

M. Steinder, A. S. Sethi
Computer and Information Sciences Department
University of Delaware
Newark, DE
USA
{*steinder,sethi*}*@cis.udel.edu*

Abstract

We present fault localization techniques suitable for diagnosing end-to-end service problems in communication systems with complex topologies. We refine a layered system model that represents relationships between services and functions offered between neighboring protocol layers. In a given layer, an end-to-end service between two hosts may be provided using multiple host-to-host services offered in this layer between two hosts on the end-to-end path. Relationships among end-to-end and host-to-host services form a bipartite probabilistic dependency graph whose structure depends on the network topology in the corresponding protocol layer. When an end-to-end service fails or experiences performance problems it is important to efficiently find the responsible host-to-host services. Finding the most probable explanation (MPE) of the observed symptoms is NP-hard. We propose two fault localization techniques based on Pearl's iterative algorithms for singly connected belief networks. The probabilistic dependency graph is transformed into a belief network, and then the approximations based on Pearl's algorithms and exact bucket tree elimination algorithm are designed and evaluated through extensive simulation study.[1]

Keywords

Fault localization, probabilistic inference, belief networks, event correlation

1 Introduction

Fault localization [15, 18, 35], a central aspect of network fault management, isolates the most probable set of root problems based on their external manifestations, called

[1]Prepared through collaborative participation in the Communications and Networks Consortium sponsored by the U. S. Army Research Laboratory under the Collaborative Technology Alliance Program, Cooperative Agreement DAAD19-01-2-0011. The U. S. Government is authorized to reproduce and distribute reprints for Government purposes notwithstanding any copyright notation thereon.

symptoms. This paper focuses on the diagnosis of availability and performance problems associated with end-to-end services provided in a given protocol layer by means of intermediate nodes invisible to the layers above. When an end-to-end service fails, one needs to locate host-to-host services responsible for the end-to-end service failure. This paper considers end-to-end service failure diagnosis to be a crucial step towards multi-layer fault localization. We present probabilistic iterative fault localization techniques capable of isolating multiple-simultaneous root problems responsible for end-to-end service failures in a given layer. The proposed solutions allow the management system to perform fault localization iteratively in real-time.

In the past, fault diagnosis efforts concentrated mostly on detecting, isolating, and correcting faults related to network connectivity [9, 18, 33, 35]. The diagnosis focused on lower layers of the protocol stack (physical and data-link layers) [24, 35], and its major goal was to isolate faults related to the availability of network resources, such as broken cable, inactive interface, etc. Modern enterprise environments increasingly demand support for quality of service (QoS) guarantees. This paper, in addition to dealing with resource availability problems, focuses on isolating the causes of notifications that indicate violations of the QoS.

Uncertainty about dependencies within the managed network and about the set of observed symptoms has been recognized as a major obstacle in performing fault localization [9, 18, 19]. In performance problem diagnosis, uncertainty becomes ubiquitous. This paper presents an application of belief networks [25] as a representation of uncertain knowledge about relationships among network entities, and fault localization techniques based on this representation. The large number of dependencies among network components (both physical and abstract ones) makes it reasonable to perform fault localization in a hierarchical fashion. It may be argued that fault localization should be performed starting from a macro-view (high level or abstract view) of the problem to select a potential spot of the problem, and then it should focus on the micro-view (low-level or detailed view) of the chosen spot. The modeling schema used in this paper allows such a hierarchical representation. The fault localization techniques proposed in this paper are suitable for fault models represented by bipartite graphs with particular focus on diagnosing end-to-end service failures.

In Section 2, we describe the layered dependency graph model for multi-layer fault diagnosis refined to expose the end-to-end service model and to allow non-deterministic reasoning about both availability and performance related problems. In Section 3, we present belief networks concepts used in this paper. Section 4 presents the mapping of the layered dependency graph into a belief network. In Section 5, we introduce three algorithms for finding the best symptoms' explanation using a bipartite belief network, which include bucket tree elimination [6] and two approximations based on Pearl's iterative algorithms [25]. The performance and accuracy of the algorithms were evaluated through an extensive simulation study described in Section 6. A comparison of our solutions with other event correlation techniques is presented in Section 7.

2 Layered model for alarm correlation

For the purpose of fault diagnosis, communication systems are frequently modeled in a layered fashion imitating the layered architecture of the modeled system [11, 35]. This approach provides a natural abstraction of the modeled system's entities, reusability of the model's modules, and ability to divide the fault management task into separate, simpler subtasks. Because of fault propagation, the effects of an abnormal operation of functions or services provided by lower layers may be observed in higher layers. Fault management systems model fault propagation by representing either causal relationships among events [12, 35] or dependencies among system entities [11, 18].

In the layered fault model, the definition of entity dependencies is based on real-life relationships between layers on a single host and among network nodes communicating within a single protocol layer. The model presented in this paper is based on the model proposed in [11]. However, our model does not include representation of protocols, since we assume that protocols are implemented correctly. We divide the fault model components into *services* and *functions*. A service offered by protocol layer L between nodes **a** and **c** (*Service*$_L$*(a,c)*) is implemented in terms of layer L functions on hosts **a** and **c** (*Network Functions*$_L$*(a)* and *Network Functions*$_L$*(c)*), and services that layer $L-1$ offers between hosts **a** and **c**. Layer L functions on node **a** depend on layer $L-1$ functions on node **a**. The recursive dependencies among services and functions constitute a dependency graph as presented in Figure 1.

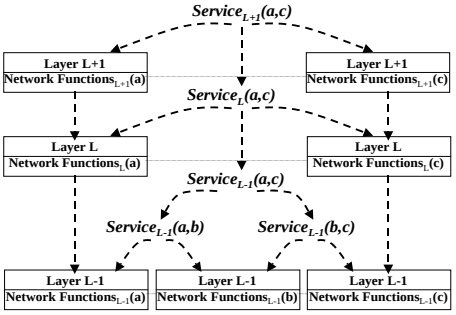

Figure 1: Layered dependency model

The general dependency graph template obtained from services, protocols and functions in different layers provides a macro-view of the relationships that exist in the system. To incorporate the micro-view of the relationships within particular model components, the layered model should be further refined to include possibly complex relationships within services and functions in the same layer. In particular, an end-to-end service offered by layer L between hosts **a** and **c** is implemented in terms of multiple host-to-host services offered by layer L between subsequent hops on the path of a layer L packet from node **a** to node **c** (such as *Service*$_{L-1}$*(a,c)* in Figure 1). Ability to reason about failures observed in an end-to-end service, i.e., symptoms, and trace them down to particular host-to-host service failures, i.e., faults, is the primary focus of the presented research.

With every dependency graph node we associate multiple failure modes $F_1, \ldots, F_k$, which represent availability and performance problems pertaining to the service or function represented by the dependency graph node. In real-life systems, the following conditions are typically considered a service/function failure: F_1 – service/function

ceases to exist (e.g., a cable connection is broken), F_2 – service/function introduces unacceptable delay (e.g., one of the host-to-host links is congested), F_3 – service/function produces erroneous output (e.g., bit errors are introduced in a link between routers), and F_4 – service/function occasionally does not produce output (e.g., packets are lost due to buffer overflow). The micro-model of *Service$_L$(a,c)* or *Network Function$_L$(a)* defines which failure mode occurs in *Service$_L$(a,c)* or *Network Function$_L$(a)* when a particular failure mode occurs in a service or function on which *Service$_L$(a,c)* or *Network Function$_L$(a)* depends. To create a micro-model for *Network Function$_L$(a)*, the knowledge of its definition and implementation is required. The micro-model of *Service$_L$(a,c)* is built based on the knowledge of the network protocol used to provide *Service$_L$(a,c)*. For example, knowing that layer L protocol implements an error detection mechanism, one can predict that erroneous output produced by *Service$_{L-1}$(a,b)* (condition F_3) results in data loss in *Service$_L$(a,b)* (condition F_4). When layer L does not implement an error detection mechanism, condition F_3 in *Service$_{L-1}$(a,b)* results in condition F_3 in *Service$_L$(a,b)*.

Uncertainty about dependencies among communication system entities is represented by assigning probabilities to the links in the dependency or causality graph [18, 19]. Some commonly accepted assumptions in this context are that (1) given fault **a**, the occurrences of faults **b** and **c** that may be caused by **a** are independent, (2) given occurrence of faults **a** and **b** that may cause event **c**, whether **a** actually causes **c** is independent of whether **b** causes **c** (the OR relationship among alternative causes of the same event), and (3) root faults are independent of one another. We take advantage of these approximating assumptions throughout the paper.

Unlike in other publications on this subject [17, 18], in this paper, the dependency graph nodes have multiple failure modes. Therefore, instead of a single probability value, we assign probability matrices to the dependency links. Let F_X denote a set of failure modes related to service or function X, and F_Y denote a set of failure modes related to the dependent service or function Y. The label assigned to dependency link $X \rightarrow Y$ is a two-dimensional matrix $|F_Y| \times |F_X|$, $\mathcal{P}$, such that $\mathcal{P}(F_j, F_i) = P\{$*service/function* Y *is in failure mode* $F_j |$ *service/function* X *is in failure mode* $F_i\}$, where $F_j \in F_Y$ and $F_i \in F_X$.

To model end-to-end services one needs to determine the set of host-to-host services on which an end-to-end service depends. Because of the dynamic routing protocols (such as Spanning Tree Protocol [26] in the data-link layer or any dynamic routing protocol in the network layer), an end-to-end service may depend on different sets of host-to-host services at different times. Current dependencies may be obtained using network management protocols such as SNMP [3], which provide the means to dynamically determine dependencies established using configuration or real-time algorithms. In this paper, we use an example of a data link layer topology in which end-to-end connectivity is achieved through a network of spanning-tree bridges. A current spanning tree, which uniquely shows a set of MAC-layer links used for communication between any two hosts may be obtained from `dot1dBase` Group of *Bridge MIB* [8]. Updates

of the spanning tree may be triggered by `newRoot` and `topologyChange` traps [8]. Obtaining dependency information is, is general, still an open research problem. Some of the available techniques are discussed in [30].

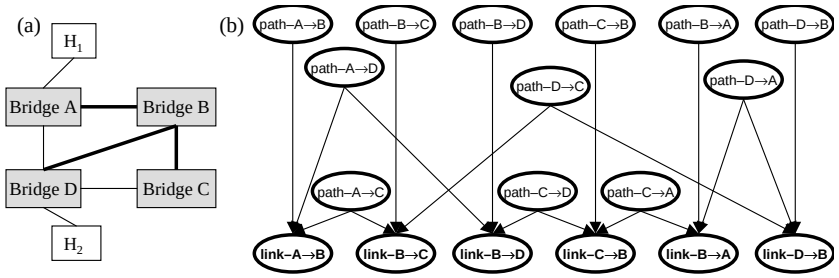

Figure 2: (a) Example bridge topology with the current spanning tree marked in bold; (b) Dependency graph built based on the spanning tree in (a)

Figure 2 presents a dependency graph for data link layer services in the network topology composed of four spanning-tree bridges [26]. The current spanning tree is marked in bold lines. In the dependency graph, we distinguish between *links*, which provide bridge-to-bridge delivery service, and *paths*, which provide delivery service from the first to the last bridge on the packet route from the source node to the destination node. The delivery service provided by paths is built of services provided by links.

3 Belief network concepts

A *belief network* [6, 25] is a directed acyclic graph [6] (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 variable V_i will be denoted by D_i. The set of directed edges E represents causal relationships among the variables and the strengths of these influences are specified by conditional probabilities. An evidence set e is a partial assignment of variables in V.

Belief networks are used to make four basic queries given evidence set e: (1) belief assessment, (2) most probable explanation, (3) maximum a posteriori hypothesis, and (4) maximum expected utility [6]. 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 a complete assignment of values to random variables in V that best explains the observed evidence e. It is known that these tasks are NP-hard in general belief networks [4]. A belief updating algorithm, polynomial with respect to $|V|$, is available for *polytrees*, i.e., directed graphs without undirected cycles [25]. 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, various approximation techniques have been investigated [7, 25]. To the best of our

knowledge, no approximation has been proposed that works well for all types of networks. Moreover, some approximation schemas have been proven to be NP-hard [5].

In this paper, we focus on a class of belief networks representing a simplified model of conditional probabilities called *noisy-OR gates* [25]. The simplified model contains binary-valued random variables. The noisy-OR model 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. The model assumes that all inhibitory mechanisms are independent [13, 25]. This assumption of independence is ubiquitous in probabilistic fault localization approaches reported in the literature [18, 19]. It indicates that all alternative causes of the same effect are independent. This simplification helps avoid exponential time and memory otherwise needed to process and store conditional probability matrices associated with random variables in the belief network. Furthermore, belief assessment in polytrees with the noisy-OR model has polynomial complexity, which makes it attractive to use with our problem as an approximation schema. The solution presented in this paper may be easily extended to incorporate other canonical belief network models, e.g., an AND-gate, which may be useful to represent some specific fault localization problems. The details of fault localization with other belief network models are beyond the scope of this paper.

4 Mapping layered model into belief network

We designed a mapping from the layered dependency graph (Section 2) into a belief network as follows.

• For every node of the layered dependency graph and for every failure mode associated with this node, we create a random variable, whose domain is {*true, false*} (we will also use 1 and 0 to represent values *true* and *false*, respectively). Let V_i be a belief network node created for failure mode F_j of the dependency graph node representing *Service*$_L$*(a,b)* or *Network Function*$_L$*(a)*. Assignment V_i=*true* (V_i=*false*) indicates that *Service*$_L$*(a,b)* or *Network Function*$_L$*(a)* is (is NOT) in condition F_j.
• For every dependency graph edge X→Y and for every failure mode of node Y, F_i, determine F_j, the failure mode of node X that results from condition F_i in node Y. Let V_i be the belief network node corresponding to dependency graph node Y and failure mode F_i. Let V_j be the belief network node corresponding to dependency graph node X and failure mode F_j. Add a belief network edge pointing from V_i to V_j.
• Let $\mathcal{P}$ be the probability matrix associated with dependency link X→Y. Matrix P_j associated with node V_j represents the following conditional probability distribution.

$P(V_j$=*false* $\mid V_i$=*false*$) = 1$ $\quad$ $P(V_j$=*true* $\mid V_i$=*false*$) = 0$
$P(V_j$=*false* $\mid V_i$=*true*$) = 1 - \mathcal{P}(F_i, F_j)$ $\quad$ $P(V_j$=*true* $\mid V_i$=*true*$) = \mathcal{P}(F_i, F_j)$

To complete the mapping of the fault localization problem into the problem of computing queries in belief networks, we need to define the interpretation of faults and symptoms in the theory of belief networks. A symptom is defined as an observation that a dependency graph node X, which typically corresponds to a higher-level service, is in condition F_j (*negative* symptom), or is NOT in condition F_j (*positive* symptom).

We will denote by $\mathcal{S}$ the set of all possible symptoms. If V_i is the belief network node corresponding to the dependency graph node X and its failure mode F_i, then the negative symptom and positive symptom are interpreted as an instantiation of V_i with value *true* and *false*, respectively. Thus, as a result of this mapping, the set of all observed symptoms, which will be denoted by $\mathcal{S}_o \subseteq \mathcal{S}$, becomes the evidence set e. The dependency graph node X, which corresponds to a lower-level service or function, is at fault, if it is in any of the conditions $F_1, \ldots, F_4$, say condition F_i. The set of all possible faults is denoted by $\mathcal{F}$. The fact that the service or function corresponding to X is in failure mode F_i is represented by value *true* in the domain of the random variable V_i. The problem of finding the set of faults, $\mathcal{F}_c \subseteq \mathcal{F}$ that best explains the set of observed symptoms $\mathcal{S}_o$ is equivalent to computing the MPE query based on the evidence set e.

5 Algorithms

In this section, three algorithms are presented to find the best symptom explanation with causal dependencies between events represented by graphs described in Section 4: *bucket-tree elimination* [6] and algorithms for polytrees [25], i.e., *iterative belief propagation in polytrees* and *iterative MPE in polytrees*. Iterative belief propagation [25] is augmented here to provide a complete symptom explanation hypothesis rather than the marginal posterior probability distribution. For iterative MPE, an approximation is proposed that allows polynomial-time complexity to be achieved. We use n to denote the number of nodes (bridges, routers, etc.) in the managed system.

5.1 Most probable explanation through bucket elimination

Bucket elimination [6] (**Algorithm 1**) is one of the most popular algorithmic frameworks for computing queries using belief networks. For the purpose of fault localization we use MPE query. The detailed description of the algorithm is beyond the scope of this paper; the algorithm explanation may be found in [6, 30].

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 in bipartite graphs representing the problem of end-to-end service failure diagnosis (Figure 2-(b)) is bound by $\mathcal{O}(n^2 exp(n))$ assuming that (1) the graph nodes are processed according to the optimal ordering [6, 30], and (2) the belief network contains all possible *path* nodes (there are $\mathcal{O}(n^2)$ such nodes possible). Sections 5.2 and 5.3 present two algorithms of polynomial complexity.

5.2 Iterative inference in Bayesian polytrees

Recall from Section 3 that in singly-connected networks (polytrees) representing the noisy-OR-gate model of conditional probability distribution, Bayesian inference may be computed in polynomial time using the algorithm presented in [25]. The graph in Figure 2-(b) is not a polytree because it contains an undirected loop.

Networks with loops violate certain independence assumptions based on which the

local computation equations were derived for polytrees. As suggested in [25], the iterative algorithm in loopy networks may not converge. 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 [23].

Iterative belief propagation utilizes a message passing schema in which the belief network nodes exchange λ and π messages (Figure 3). 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 sub-graph 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 probability that $X = x$ given the entire body of evidence in the subgraph containing X created by removing edge $X \rightarrow U_i$. Based on the messages received from its parents and children, node X computes $bel(x)$, i.e., the probability that $X{=}x$ given the entire evidence.

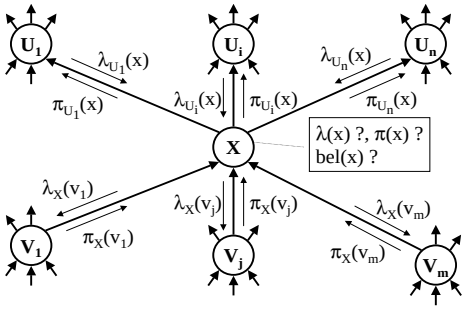

Figure 3: Message passing in Pearl's belief propagation

The belief propagation algorithm in polytrees starts from the evidence node and propagates the changed belief along the graph edges by computing $bel(x)$, $\lambda_X(v_i)$'s and $\pi_X(u_i)$'s in every visited node. The complete description of the iterative algorithm for polytrees including expressions for $\lambda_X(v_j)$, $\pi_{U_i}(x)$, and $bel(x)$ along with some illustrative examples may be found in [25]. In noisy-OR gate belief networks, functions $\lambda_X(v_j)$, $\pi_{U_i}(x)$, and $bel(x)$ may be evaluated in linear time with respect to the number of node X's neighbours. In reported applications of iterative belief updating to loopy graphs, several iterations are performed in which the entire graph is searched according to some pre-defined ordering [20, 23].

This paper adapts the iterative belief propagation algorithm to the problem of fault localization with fault models represented by bipartite graphs as in Figure 2-(b). In this application, we perform one traversal of the entire graph for every observed symptom. For every symptom we define a different ordering that is equivalent to the breadth-first order started in the node representing the observed symptom.

It may be noticed that parents of an unobserved *path* node are independent of one another; therefore, unobserved *path* nodes constitute a belief propagation barrier, i.e., change of belief passed from one unobserved *path* node's parent may not change the belief of another parent. This makes it reasonable to stop the traversal of the graph any time an unobserved *path* node is reached and thereby avoid the calculations in the

unobserved *path* nodes altogether.

Algorithm 2 (MPE through 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 *such as* X *is not an unobserved path node, along ordering* o
compute $\lambda_X(v_j)$ *for all* X*'s parents,* V_j*, and for all* $v_j \in \{0, 1\}$
compute $\pi_{U_i}(x)$ *for all* X*'s children,* U_i*, and for all* $x \in \{0, 1\}$
Symptom analysis phase:
for every symptom $S_i \in \mathcal{S}_O$ *run inference iteration starting from* S_i
compute $bel(v_i)$ *for every node* V_i*,* $v_i \in \{0, 1\}$
Fault selection phase:
while $\exists$ *link node* V_j *for which* $bel(1){>}0.5$ *and* $S_O{\neq}\emptyset$ *do*
take V_j *with the greatest* $bel(1)$ *and mark it as observed to have value of 1*
run inference iteration starting from V_j
remove all S_i *such that* V_j *may cause* S_i *from* S_O
compute $bel(v_i)$ *for every node* V_i*,* $v_i \in \{0, 1\}$

The computations described above produce the marginal posterior distribution resulting from the observation of the evidence. Based on this distribution we approximately calculate the most probable explanation of the observations as follows. We choose a *link* node with the highest posterior probability, place the corresponding fault in the MPE hypothesis, mark the node as observed with value 1, and perform one iteration of the belief propagation starting from the chosen *link* node. This step is repeated until both of these conditions hold: (1) the posterior distribution contains *link* nodes whose probability is greater than 0.5, and (2) unexplained symptoms remain in S_O.

Local computations in *path* nodes require $\mathcal{O}(k)$ operations, where k is the number of links that constitute the path. Since in an n-node network, a path may be composed of at most n links, local computations in *path* nodes require $\mathcal{O}(n)$ operations. Thus, in a single iteration processing *path* nodes requires $\mathcal{O}(n(|\mathcal{S}|) \subseteq \mathcal{O}(n^3)$ operations. Local computations in *link* nodes require $\mathcal{O}(k)$ steps, where k is a number of node's children. Thus, processing all *link* nodes is $\mathcal{O}(\sum k)$. Observe that $\sum k =$ the number of all causal links in the bipartite graph, i.e., n^3 because there are at most n^2 *path* nodes and every path may be composed of at most n links. We may conclude that a single iteration of the algorithm is $\mathcal{O}(n^3)$, and the complexity of the entire algorithm is $\mathcal{O}(|\mathcal{S}_o|n^3) \subseteq \mathcal{O}(n^5)$.

5.3 Iterative most probable explanation in Bayesian polytrees

In this section, we introduce an augmentation of the iterative MPE algorithm for polytrees [25] to networks with undirected loops. The iterative belief updating algorithm presented in Section 5.2 computes the marginal posterior probability of the Bayesian network nodes given the observed evidence. In Algorithm 2 we used this distribution to select the most probable explanation. The MPE algorithm in every iteration produces the most probable value assignment to the belief network nodes. This allows us to

eliminate the *fault selection* phase in Algorithm 2, which contributes to the complexity and is an additional source of the inaccuracy.

Algorithm 3 (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
compute $\lambda^*_X(v_j)$ *for all* X*'s parents,* V_j*, and for all* $v_j \in \{0,1\}$
compute $\pi^*_{U_i}(x)$ *for all* X*'s children,* U_i*, and for all* $x \in \{0,1\}$
Symptom analysis phase:
for every symptom $S_i \in \mathcal{S}_O$ *run inference iteration starting from* S_i
compute $bel^*(v_i)$ *for every node* V_i*,* $v_i \in \{0,1\}$
Fault selection phase:
choose all link nodes with $bel^*(X{=}1){>}bel^*(X{=}0)$

Similarly 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 the nodes located in the subgraph containing X resulting from the removal of link $X \rightarrow U_i$, which include the proposition $X = x$. The belief metric $bel^*(x)$ stands for the probability of the most probable explanation of evidence e that is consistent with the proposition $X = x$. The equations for calculating $\lambda^*_X(v_j)$ and $\pi^*_{U_i}(x)$, and belief metric $bel^*(x)$ are presented in [25]. However, unlike in iterative belief updating described in Section 5.2 the message computation requires exponential time. In [30], we introduce an approximation that reduces the complexity to polynomial. In this paper, we take advantage of this approximation.

The algorithm for computing MPE calculates λ^* and π^* for every network node traversing the graph starting from the observed symptom in the 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 *link* nodes with $bel^*(x{=}1){>}bel^*(x{=}0)$.

Local computations in *path* nodes require $\mathcal{O}(n^2)$ operations, where n is the maximum path length. Thus, in a single iteration, processing *path* nodes requires $\mathcal{O}(n^2(|\mathcal{S}|) \subseteq \mathcal{O}(n^4)$ operations. Similarly to belief updating, local computations in all *link* nodes require $\mathcal{O}(n^3)$ operations. We may conclude that a single iteration of the algorithm is $\mathcal{O}(n^4)$, and the complexity of the entire algorithm is $\mathcal{O}(|\mathcal{S}_o|n^4) \subseteq \mathcal{O}(n^6)$.

6 Simulation Study and Comparison of Algorithms

The algorithms presented in Section 5 were implemented in Java. We used JavaBayes [1] package to obtain an implementation of Algorithm 1. The algorithms were

evaluated through a set of comprehensive experiments. As a real-life application domain, we chose the data-link layer in a bridged network in which the path ambiguity is resolved using Spanning Tree Protocol [26]. As a result, the shape of the considered graphs is reduced to trees, thus making random generation of dependencies resembling real-life scenarios easier. We tested the algorithms on randomly generated network topologies, whose size ranged from 5 to 50 nodes in the case of Algorithm 2. Because of high computation time of Algorithms 1 and 3, we had to limit the scope of experiments to graphs of size ≤ 10 and ≤ 25 (see Table 1).

Table 1: Comparison of Algorithms 1- 3

Algorithm	Bucket Elimination (Alg. 1)	Iterative Belief Updating (Alg. 2)	Iterative MPE (Alg. 3)
Theoretical bound	$exp(n)$	n^5	n^6
Detection rate	96-100%	93-98%	95-100%
False positive rate	0-4%	2-5%	0-8%
Max. network size with localization time <10s	10	50	25
Lost and spurious symptoms	yes	yes	yes
Is algorithm iterative?	no	yes	yes
Prediction capabilities	yes	extension required	yes

For every graph size, we randomly generated spanning tree connections, link failure probabilities, and conditional probabilities on causal links between *link* and *path* nodes. The link failure probabilities were uniformly distributed random values of the order of 10^{-6}, and the conditional probabilities on causal links were uniformly distributed random values in the range $[0.5, 1)$. For every graph size, one hundred different graphs were generated. For each randomly generated graph, we performed 200 experiments. In every experiment, we randomly generated the set of malfunctioning links, $\mathcal{F}_c$, based on their failure probabilities. Then, based on the conditional probabilities on causal links between *link* and *path* nodes, the set of observed symptoms, $\mathcal{S}_o$, resulting from the faults in $\mathcal{F}_c$ was generated. The observed symptoms were then randomly ordered. The ordered set $\mathcal{S}_o$ was supplied as an input to the algorithms presented in Section 5. Their output, the set of detected faults, $\mathcal{F}_d$, was compared with $\mathcal{F}_c$. We used the following two metrics to represent the accuracy of the algorithms.

$$\text{detection rate} = \frac{|\mathcal{F}_d \cap \mathcal{F}_c|}{|\mathcal{F}_c|} \qquad \text{false positive rate} = \frac{|\mathcal{F}_d - \mathcal{F}_c|}{|\mathcal{F}_d|}$$

Detection rate represents the percentage of faults that occurred in the network in a given experiment that were detected by an algorithm. *False positive rate* represents the percentage of faults proposed by an algorithm that were not occurring in the network in a considered experiment, i.e., they were false fault hypotheses. Table 1 shows detection rate and false positive rate intervals of the analyzed algorithms.

Figure 4 presents the relationship between detection rate and graph size. The mean for a particular graph size is an average over the mean detection rates for particular graphs of that size, within statistically computed confidence intervals. We observe that

Algorithms 1 and 3 outperform Algorithm 2 by 1-2%. The shape of the graphs in Figure 4 indicates a strong dependency of the detection rate on the graph size. For small (5-node) graphs, the number of symptoms observed is typically small (less than 10), which in some cases is not sufficient to precisely pinpoint the actual fault. Since in small graphs the size of $\mathcal{F}_c$ is also small, any mistake in fault detection significantly reduces the detection rate. When the graph gets larger, the number of observed symptoms increases, thereby increasing the ability to precisely detect the faults. On the other hand, as the graph size grows, the multi-fault scenarios are becoming more and more frequent. In multi-fault experiment, it is rather difficult to detect all actual faults, which leads to partially correct solutions and the decreasing accuracy of Algorithm 2.

Figure 5 presents the relationship between false positive rate and the graph size. The false positive rate for a particular graph size is calculated as a mean of average false positive rates for particular graphs of that size. As depicted in Figure 5, 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 graphs 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.

Figures 6, 7, and 8 present the dependency of the correlation time on the graph size in the presence of 1, 2, and 4 network faults, respectively. Although in the tested graph size range, Algorithm 1 exhibited the best accuracy, the difference between the accuracy of Algorithm 1 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 correlation time of the order of several seconds (Figure 9), even for large networks and multi-fault scenarios, is encouraging.

To make the evaluation of algorithms presented in Section 5 complete, one also needs to compare them with respect to other features. We believe that the following factors should be taken into account in this evaluation: (1) potential for dealing with lost and spurious symptoms, (2) ability to work in the event-driven environment and (3) usability for prediction and test planning. Comparison of algorithms with respect to these factors is presented in Table 1. All algorithms presented in Section 5 have a potential for working in the environment in which lost and spurious symptoms occur. A solution that addresses the problem of lost and spurious symptoms for Algorithm 2 is described in [32]. All presented algorithms, except Algorithm 1, are iterative and allow an event-driven building of fault hypotheses. Algorithms 1 and 2 may be used to check the existence of unobserved symptoms, to calculate the system components affected by fault or to improve testing procedures.

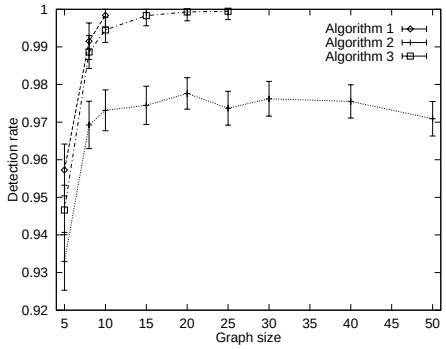

Figure 4: Comparison of accuracies achievable with algorithms presented in Section 5 for different network sizes

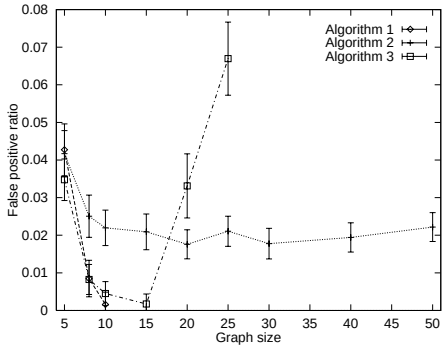

Figure 5: Comparison of false-positive metric values for algorithms presented in Section 5 using different network sizes

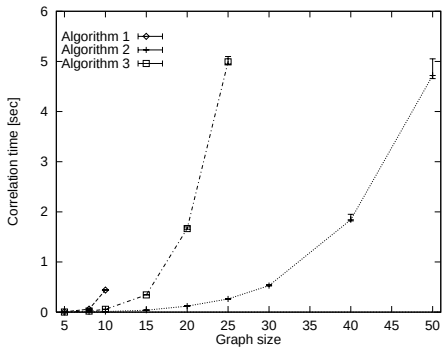

Figure 6: Comparison of single fault detection time for algorithms presented in Section 5 vs. network size

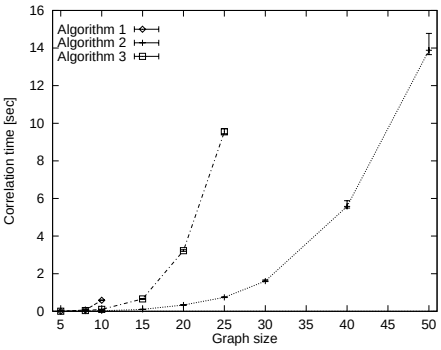

Figure 7: Comparison of correlation time vs. network size in the presence of two network faults

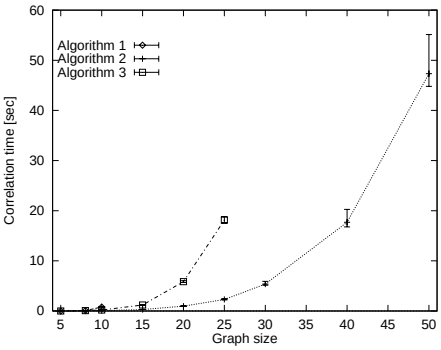

Figure 8: Comparison of correlation time vs. network size in the presence of four network faults

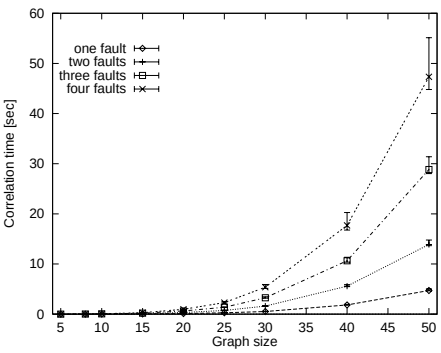

Figure 9: Fault localization time with Algorithm 2

7 Related work

In the past, various event correlation techniques were proposed including rule-based systems [22, 34], model-based reasoning systems [15, 24], model traversing techniques [16], case-based systems [21], fault propagation models [12, 18], and the codebook approach [35]. Most of the above approaches utilize deterministic reasoning. This paper focuses on non-deterministic event correlation which is unavoidable in fault diagnosis related to quality of service degradation particularly in upper protocol layers.

Katzela et al. [18] proposed an $\mathcal{O}(N^3)$ algorithm that finds the most probable explanation of a set of symptoms in an N-node dependency graph. The algorithm utilizes the maximum mutual dependency heuristics which is not suitable for diagnosing end-to-end service failures, because in end-to-end services model, all host-to-host services are independent of one another. In addition to the above problem, the approach presented in [18] does not allow lost or spurious symptoms and the correlation may not be performed in event-driven fashion. Kliger et al. [19] propose a probabilistic model to be used with the codebook approach but they do not present the non-deterministic decoding schema. The approach of Algorithms 2 and 3 can be used for this purpose. Statistical data analysis methods are used for non-deterministic fault diagnosis in bipartite-graphs in [10]. The technique detects link failures in wireless and/or battlefield networks based on the observed set of broken end-to-end connections.

Previous applications of belief networks to fault diagnosis are limited to rather narrow applications. Deng et al. [9] present a polynomial algorithm for updating belief in a restricted Bayesian network used as a model for fault diagnosis in linear light-wave networks. In [13] tree-shaped belief networks are applied to troubleshoot printing services. Wang et al. [33] applies Bayesian theory to identifying faulty links in communication networks. Their approach does not include conditional probabilities, and therefore is suitable for the diagnosis of availability related problems only.

In our previous work on fault localization, we proposed an incremental algorithm for diagnosing faults whose propagation pattern may be modeled by bipartite graphs [31]. The algorithm has lower computational complexity than the algorithms proposed in this paper but is not suitable for other fault management tasks such as the prediction of affected services or test planning.

The system model used in this paper constitutes a refinement of the layered model proposed in [11]. The hierarchical modeling uncertainty with belief networks is consistent with other work on constructing diagnostic models [29]. Unlike in [29], our model allows representation of different types of influences caused by one component on its dependent components.

8 Conclusions and future work

In this paper, we present and evaluate several fault localization algorithms using fault propagation models represented by bipartite graphs. We show that not only are exact

algorithms theoretically unacceptable because of their exponential complexity bound, they are not usable in practice even for relatively small networks either. Algorithms based on iterative message propagation (Algorithms 2 and 3) allow us to find a solution in an event-driven fashion and have very promising acuracy and performance.

In this paper, we implicitly assume that the set of observed symptoms is accurate. In reality, spurious symptoms may occur, which do not indicate any abnormal condition, or symptoms may be lost. Moreover, the reasoning is performed based only on the negative information, i.e., observed end-to-end service failures. We do not take into account positive information that some end-to-end services did not fail. Modifications to Algorithm 2 that address these problems are described in [32]. In our simulation study, we consider the case in which the conditional probability distribution represented by a belief network is known accurately. The study described in [32] indicates that Algorithm 2 offers very good performance even for approximate values of conditional probabilities in the belief network model.

The algorithms presented in this paper were evaluated on a restricted class of network topologies. While we find no reason to believe that in arbitrary network topologies the performance or accuracy of these algorithms would be substantially different, we think that the algorithms should also be evaluated on arbitrary topologies resembling real-life networks. It is of particular interest to investigate means of exploiting domain semantics of real-life networks to improve the scalability of the algorithms.

In this paper, we consider the situation in which the routing information necessary to build a dependency model for end-to-end services is available. However, to obtain this information may be time consuming and require substantial amount of resources. In future research, we would like to investigate diagnosing end-to-end service failures without access to the accurate routing information.[2]

References

[1] http://www.cs.cmu.edu/ javabayes/Home/.

[2] C. Berrou, A. Glavieux, and P. Thitimajshima. Near Shannon limit error-correcting coding and decoding: Turbo Codes. In *Proceedings of Int'l Communications Conference*, 1993, pp. 1064–1070.

[3] 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, 1996. RFC 1905.

[4] G. F. Cooper. Probabilistic inference using belief networks is NP-Hard. Technical Report KSL-87-27, Stanford University, 1988.

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

[6] R. Dechter. Bucket Elimination: A unifying framework for probabilistic inference. In E. Horvitz and F. V. Jensen, eds, *Proc. of the Twelfth Conference on Uncertainty in Artificial Intelligence*, Portland, Oregon, Aug. 1996. Morgan Kaufmann Publishers.

[7] R. Dechter and I. Rish. A scheme for approximating probabilistic inference. In D. Geiger and P. Shenoy, eds, *Proc. of the Thirteenth Conference on Uncertainty in Artificial Intelligence*, Brown University, Providence, Rhode Island, Aug. 1997. Morgan Kaufmann Publishers.

[2] 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.

[8] E. Decker, P. Langille, A. Rijsinghani, and K. McCloghrie. *Definition of Managed Objects for Bridges*. IETF Network Working Group, 1993. RFC 1493.

[9] R. H. Deng, A. A. Lazar, and W. Wang. A probabilistic approach to fault diagnosis in linear lightwave networks. In Hegering and Yemini [14], pp. 697–708.

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

[11] R. Gopal. Layered model for supporting fault isolation and recovery. In *Proc. of Network Operation and Management Symposium*, Honolulu, Hawaii, 2000.

[12] M. Hasan, B. Sugla, and R. Viswanathan. A conceptual framework for network management event correlation and filtering systems. In Sloman et al. [28], pp. 233–246.

[13] D. Heckerman and M. P. Wellman. Bayesian networks. *Communications of the ACM*, 38(3):27–30, Mar. 1995.

[14] H. G. Hegering and Y. Yemini, eds. *Integrated Network Management III*. North-Holland, Apr. 1993.

[15] G. Jakobson and M. D. Weissman. Alarm correlation. *IEEE Network*, 7(6):52–59, Nov. 1993.

[16] J. F. Jordaan and M. E. Paterok. Event correlation in heterogeneous networks using the OSI management framework. In Hegering and Yemini [14], pp. 683–695.

[17] S. Kätker and M. Paterok. Fault isolation and event correlation for integrated fault management. In A. Lazar, R. Saracco, and R. Stadler, eds, *Integrated Network Management V*. Chapman and Hall, May 1997, pp. 583–596.

[18] I. Katzela and M. Schwartz. Schemes for fault identification in communication networks. *IEEE Transactions on Networking*, 3(6):733–764, 1995.

[19] S. Kliger, S. Yemini, Y. Yemini, D. Ohsie, and S. Stolfo. A coding approach to event correlation. In Sethi et al. [27], pp. 266–277.

[20] F. R. Kschischang and B. J. Frey. Iterative decoding of compound codes by probability propagation in graphical models. *Journal on Selected Areas in Communications*, 16(2):219–230, Feb. 1998.

[21] L. Lewis. A case-based reasoning approach to the resolution of faults in communications networks. In Hegering and Yemini [14], pp. 671–681.

[22] G. Liu, A. K. Mok, and E. J. Yang. Composite events for network event correlation. In Sloman et al. [28], pp. 247–260.

[23] R. J. McEliece, D. J. C. MacKay, and J.-F. Cheng. Turbo decoding as an instance of Pearl's "belief propagation" algorithm. *Journal on Selected Areas in Communications*, 16(2):140–151, Feb. 1998.

[24] Y. A. Nygate. Event correlation using rule and object based techniques. In Sethi et al. [27], pp. 278–289.

[25] J. Pearl. *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. Morgan Kaufmann Publishers, 1988.

[26] R. Perlman. *Interconnections, Second Edition: Bridges, Routers, Switches, and Internetworking Protocols*. Addison Wesley, 1999.

[27] A. S. Sethi, F. Faure-Vincent, and Y. Raynaud, eds. *Integrated Network Management IV*. Chapman and Hall, May 1995.

[28] M. Sloman, S. Mazumdar, and E. Lupu, eds. *Integrated Network Management VI*. IEEE Publishing, May 1999.

[29] S. Srinivas. Building diagnostic models from functional schematics. Technical Report KSL 95-15, Knowledge Systems Laboratory, Dept. of Computer Science, Stanford University, Feb. 1994.

[30] M. Steinder and A. S. Sethi. Multi-layer fault localization using probabilistic inference in bipartite dependency graphs. Technical Report 2001-02, CIS Dept., Univ. of Delaware, Feb. 2001.

[31] M. Steinder and A. S. Sethi. Non-deterministic diagnosis of end-to-end service failures in a multi-layer communication system. In *Proc. of ICCCN*, Scottsdale, AR, 2001. pp. 374–379.

[32] M. Steinder and A. S. Sethi. Increasing robustness of fault localization through analysis of lost, spurious, and positive symptoms. In *Proc. of IEEE INFOCOM*, New York, NY, 2002. (to appear).

[33] C. Wang and M. Schwartz. Identification of faulty links in dynamic-routed networks. *Journal on Selected Areas in Communications*, 11(3):1449–1460, Dec. 1993.

[34] P. Wu, R. Bhatnagar, L. Epshtein, M. Bhandaru, and Z. Shi. Alarm correlation engine (ACE). In *Proc. of Network Operation and Management Symposium*, New Orleans, LA, 1998. pp. 733–742.

[35] S. A. Yemini, S. Kliger, E. Mozes, Y. Yemini, and D. Ohsie. High speed and robust event correlation. *IEEE Communications Magazine*, 34(5):82–90, 1996.