Non-deterministic Diagnosis of End-to-End Service Failures in a Multi-layer Communication System

Małgorzata Steinder and Adarshpal S. Sethi
Department of Computer and Information Sciences
University of Delaware, Newark, DE
{steinder,sethi}@cis.udel.edu

***Abstract*– Fault localization is a process of isolating faults responsible for the observable malfunctioning of the managed system. Until recently, fault localization efforts concentrated mostly on diagnosing faults related to the availability of network resources in the lowest layers of the protocol stack. This paper focuses on end-to-end service failure diagnosis as a critical step towards multi-layer fault localization in an enterprise environment. By refining a previously proposed modeling technique, we present a universal method of modeling both availability and performance related problems associated with end-to-end services in a non-deterministic fashion. We introduce and evaluate a novel algorithm that allows an event-driven, incremental diagnosis of end-to-end service failures.**

1 Introduction

Fault diagnosis is a central aspect of network fault management. The core of fault diagnosis is fault localization – a process of analyzing external symptoms of network disorder to isolate possibly unobservable faults responsible for the symptoms' occurrences. Traditionally, fault localization has been performed by an expert based on disorder symptoms displayed on a management console. As systems grew larger and more complex, automated fault localization techniques became critical. Until recently, fault localization effort concentrated mostly on diagnosing faults related to network connectivity in lower layers of the protocol stack (typically the physical and data-link layers), and its major goal was to isolate faults related to the availability of resources, such as broken cable, inactive interface, etc. Recent advances in the deployment of enterprise services such as virtual private networks and application service provision require that fault localization also focus on diagnosing performance problems in multiple layers of the protocol stack including the application and service layers. Modern enterprise environments impose several challenges on the fault localization problem, which include modeling and reasoning about (1) the system state in various protocol layers, (2) interactions between protocol layers, (3) versatile types of failures, and (4) non-determinism within the system structure and its observed state.

This paper addresses the above issues by proposing a fault management technique suitable for diagnosing both availability and performance problems across multiple layers of the protocol stack. We describe a system model created by refining the layered dependency graph introduced in [5]. The refined model allows not only vertical but also horizontal failure propagation to be seamlessly represented by including micro-models of end-to-end services (such as IP end-to-end communication) provided in a protocol layer by means of multiple intermediate hosts (such as routers). Non-deterministic dependencies between the model components make it possible to represent uncertainty associated with relationships between the communication system entities. The algorithm introduced in this paper allows end-to-end service failures to be diagnosed in an iterative and incremental fashion.

In Section 2, we present related work on multi-layer fault diagnosis, and motivate the need for end-to-end service failure diagnosis. Section 3 presents the layered dependency graph template as the system model for multi-layer fault diagnosis. In Section 4, we present the mapping of the dependency graph template into a causality graph. Section 5 introduces an iterative and incremental algorithm for diagnosing end-to-end service failures. In Section 6, the incremental algorithm is evaluated.

2 Motivation and Related Work

Various fault localization techniques were investigated in the past. A comparative study of these techniques has been presented in [18, 20]. In this section, we survey recent publications that address the problem of multi-layer fault diagnosis.

Gopal et al. [5] presented a model in which the definition of entity dependencies is based on real-life relationships between layers on a single host and between network nodes in a single protocol layer. The model addresses vertical failure propagation by allowing relationships between abstract functions and services between neighboring layers to be modeled. It does not, however, provide means to represent multiple possible failure types of a particular function or service, nor does it provide any support for modeling non-deterministic system behavior.

Yemanja [1] defines a hierarchical set of entity models, which correspond to physical or conceptual network entities, e.g., network layers. Entity models are instantiated with the data obtained from the arriving event attributes or from the configuration database, and communicate using internal composite events. As events propagate up the entity model hierarchy, their semantics becomes more and more abstract so that higher-level scenarios can correlate higher-level symptoms with lower-level failures without the detailed knowledge of the low-level network state.

Both Yemanja [1] and the layered system model [5] address the issue of vertical fault propagation. They do not discuss the situation in which an explanation of a higher-level problem (e.g., end-to-end service failure) must be selected from a large set of possible lower-level failures (i.e., host-to-host link failures). This situation results from horizontal fault propagation and is a very common problem in large communication systems.

SMARTS InCharge system [21] allows the multi-layer system model to be built using a high-level language, called MODEL [14]. (MODEL would be suitable as a language for defining dependencies within the layered model used in this paper.) For fault localization, InCharge utilizes the codebook technique [21], which is very efficient and is resilient to the noise in the alarm data. However, the codebook technique is difficult to apply to the correlation of transport and application layer events since relationship changes between managed objects, which are frequent in higher layers, require reconfiguration of the codebook. Although Kliger et al. [10] proposed a model that allows a probabilistic code-book to be built, the code-book technique lacks a non-deterministic decoding schema. One of the applications of the algorithm introduced in this paper is to provide a decoding schema for the probabilistic code-book approach [21].

In our previous work, we investigated the application of belief networks to the problem of end-to-end service failure diagnosis [19]. The algorithm proposed in this paper is conceptually simpler and has lower computational complexity. However, it does not possess certain properties that the belief network approach does, e.g., it may not predict service failures, nor is it useful for the planning of testing procedures.

3 LAYERED MODEL FOR ALARM CORRELATION

In communication systems, a failure of a system component may affect the operation of other system components. The ability of a fault in one entity to change the state of other entities is referred to as fault propagation. Because of fault propagation, the effects of abnormal operation of functions or services provided by lower layers may be observed in higher layers (vertical propagation). Similarly, a fault may be observable on network hosts, which are distant from the location of the fault occurrence (horizontal propagation). In order to find explanations of higher-layer problems, it is useful to create a fault propagation model. Fault management systems model fault propagation by representing either causal relationships between events [6, 21] or dependencies between communication system entities [5, 7, 9].

For the purpose of fault diagnosis, communication systems are frequently modeled in a layered fashion imitating the layered architecture of the modeled system [5, 12, 21]. In this paper, we adopt the layered dependency graph template proposed in [5] and refine it to enable the representation of end-to-end services, non-determinism and performance problem modeling.

3.1 Layered model template

In the layered fault model [5], components are generally divided into *services*, *protocols*, and *functions* [5]. A service offered by protocol layer L between nodes **a** and **b** (*Service*$_L$*(a,b)*) is implemented in terms of layer L functions on hosts **a** and **b** (*Network Functions*$_L$*(a)* and *Network Functions*$_L$*(b)*), and the layer L protocols through which hosts **a** and **b** communicate. The layer L protocols running between hosts **a** and **b** use layer $L-1$ functions on hosts **a** and **b**, and services that layer $L-1$ offers between hosts **a** and **b**. Layer L functions on node **a** depend on layer $L-1$ functions on node **a**. The recursive dependencies between services, protocols and functions constitute a dependency graph as described in [5]. In this paper, we find it useful to eliminate the protocol nodes. This model simplification is justified, since it may be assumed that the protocols are implemented correctly; under this assumption, protocols cannot contribute explanations to service failures. Fig. 1 shows the resultant general dependency graph for a layered network, in which *Service*$_L$*(a,c)* directly depends on *Service*$_{L-1}$*(a,c)*.

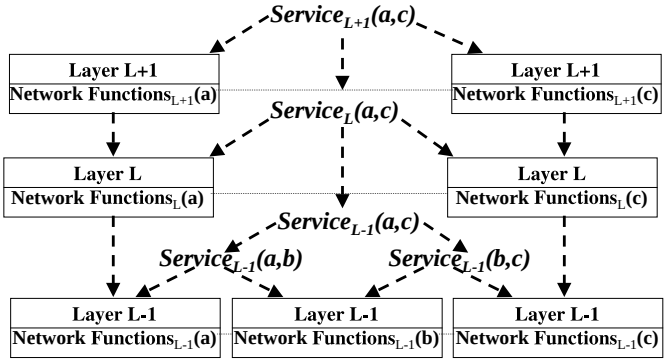


Figure 1: Refined layered network dependency model

The 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, protocols and functions in the same layer. This paper introduces a micro-model of end-to-end services; an end-to-end service offered by layer L between hosts **a** and **c** is implemented in terms of multiple hop-to-hop services offered by layer L between subsequent hops on the path of the layer L packet from node **a** to node **c** (such as *Service*$_{L-1}$*(a,c)* in Fig. 1). The 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 critical in order to perform fault diagnosis in complex network topologies, and is the primary focus of the presented research.

In this paper, besides the elimination of the protocol nodes, the model presented in [5] is refined as follows. 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., the cable connection is broken),
- F_2 – service/function introduces unacceptable delay (e.g., one of the hop-to-hop links in network layer is congested),
- F_3 – service/function produces erroneous output (e.g., bit errors are introduced in a serial link between routers),
- F_4 – service/function occasionally does not produce output (e.g., packets are lost due to buffer overflow).

The knowledge of communication protocols makes it possible to predict which of these conditions will occur in a higher-level service/function if any of these conditions occur in one or more lower-layer services/functions.

3.2 Representation of non-determinism in the layered model

Most fault localization techniques proposed to date [6, 8, 13, 21] utilize a deterministic fault model in which the dependency link from **a** to **b** implies that if **a** fails, then **b** also fails. The deterministic model is typically sufficient to represent faults in lower layers of the protocol stack related to the availability of services offered by these layers. However, these fault localization techniques are rather difficult to apply when faults are Byzantine [16], e.g., related to service performance. In the upper layers, frequent reconfigurations of service dependencies make it impossible to keep such a deterministic model up-to-date.

Uncertainty about dependencies between communication system entities is represented by assigning probabilities to the links in the dependency or causality graph [9, 10]. 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 the occurrences of faults **a** and **b** that may cause event **c**, whether **a** actually causes **c** is independent of whether **b** causes **c** (OR relationship between alternative causes of the same event), and (3) faults are independent of one another. We take advantage of these assumptions throughout the paper.

Unlike other publications on this subject [9], this paper uses the dependency graph whose nodes have multiple failure modes. Therefore, instead of a single probability value, we assign probability matrices to the dependency links. Let $\mathcal{F}_X$ denote a set of failure modes related to service or function X, and $\mathcal{F}_Y$ denote a set of failure modes related to the dependent service or function Y. The label assigned to dependency link $Y \rightarrow X$ is a two-dimensional matrix $|\mathcal{F}_Y| \times |\mathcal{F}_X|$, $\mathcal{P}$, 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 $F_j \in \mathcal{F}_Y$ and $F_i \in \mathcal{F}_X$.

3.3 Obtaining the dependency graph

The dependency graph described in Section 3.1 records two types of dependencies between services and functions in neighboring protocol layers: *static* and *dynamic* dependencies. Dynamic dependencies result from, e.g., run-time addition and deletion of services (such as establishment and termination of TCP sessions). Another source of dynamic dependencies is the usage

of routing protocols (such as the Spanning Tree Protocol [16] in the data-link layer or any dynamic routing protocol in the network layer). Because of the dynamic routing protocols, an end-to-end service may depend on different sets of host-to-host services at different times. In order to reason about the causes of the end-to-end service failures, we need to determine the currently used set of host-to-host services. Network management protocols such as SNMP [3] provide the means to determine dependencies established using configuration or real-time algorithms. For example, the list of the data link layer links belonging to the current spanning tree may be obtained from SNMP agents on the bridges/switches using the data contained in `dot1dBase` Group of *Bridge MIB* [4]. Updates of the spanning tree may be triggered by `newRoot` and `topologyChange` traps [4].

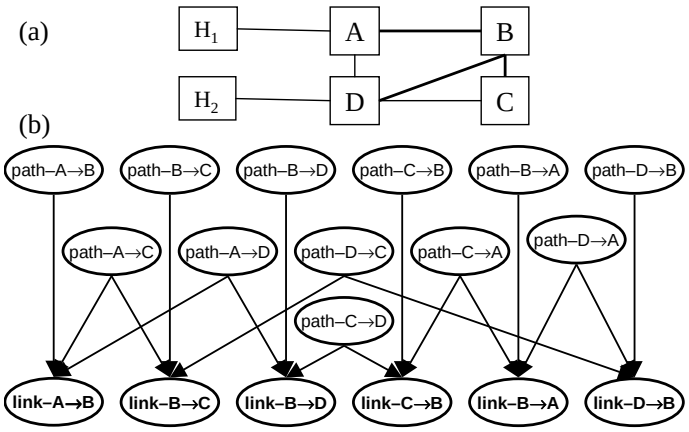


Figure 2: (a) Example bridge topology with the current spanning tree marked in bold; (b) Dependency graph built based on the spanning tree in (a)

In Fig. 2, we present a dependency graph for data link layer services in the simple network topology composed of four learning bridges [16], A, B, C, and D. In the dependency graph, we distinguish between *links*, which provide bridge-to-bridge delivery service, and *paths*, which provide packet 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 delivery services provided by links. We find it reasonable to consider unidirectional communication between two hosts a service since it is possible for a service between two hosts to fail only in one direction, while in the opposite direction it remains intact. By distinguishing between opposite directions, it becomes possible to detect these situations.

In the non-deterministic fault-model, locating a faulty link service when the path service fails may be rather complex. In large networks, testing all link services is time consuming even if it is technically possible. Therefore, before any tests are scheduled, the link services that are the most likely to have caused the path problem should be determined based on the analysis of the observed symptoms, i.e., path service failures. To build such a fault hypothesis, in Section 5, we introduce a novel algorithm using an *incremental hypothesis update*, which utilizes a causality graph as a fault propagation model.

4 Mapping Layered Dependency Graph into Causality Graph

A causality graph is a directed acyclic graph $G(E, C)$ whose nodes E correspond to events and whose edges C describe cause-effect relationships between events. Causality graph edges are labeled with the probability of the causal implication between events at the head and at the tail of an edge. Parentless causality graph nodes, which represent root problems, are labeled with the probabilities of their independent occurrence.

While dependency graphs seem more natural and easier to build and modify than causality graphs, by dealing with events rather than the managed system components, causality graphs are frequently more suitable as a fault propagation model for fault diagnosis than dependency graphs [6, 21]. A causal model of the system may be created based on the layered system model presented in Section 3 with the following steps.

- For every node of the layered dependency graph and for every failure mode associated with this node, we create a causality graph node, which may be assigned values *true*, *false*, or may be assigned no value. Let V_i be a causality graph node created for failure mode F_j of the dependency graph node representing *Service$_L$(a,b)* or *Network Function$_L$(a)*. Node V_i represents an event associated with the fact that *Service$_L$(a,b)* or *Network Function$_L$(a)* is in condition F_j. Assignment V_i=*true* indicates that event V_i occurred. Assignment V_i=*false* indicates that event V_i did not occur. When V_i is not assigned, it is interpreted as the fact that whether event V_i did or did not occur is not known.
- 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. This determination may be performed based on the knowledge of communication protocols. 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)*. Let V_i be the causality graph node corresponding to dependency graph node Y and failure mode F_i. Let V_j be the causality graph node corresponding to dependency graph node X and failure mode F_j. Add an edge pointing from V_i to V_j.
- Let $\mathcal{P}$ be the probability matrix associated with dependency link X→Y. Label $V_i \rightarrow V_j$ with $\mathcal{P}(F_i, F_j)$.
- If X is a parentless node, label V_j with $\mathcal{P}(F_j)$.

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 causality graph node corresponding to the dependency graph node X and its failure mode F_i, then the negative symptom is interpreted as an instantiation of V_i with value *true*, and the positive symptom is interpreted as an instantiation of V_i with value *false*. 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* assigned to the node V_i. Fault localization task is to find the set of root problems, i.e., parentless causality graph nodes, that best explains the set of observed symptoms, i.e., leafless causality graph nodes.

5 Fault Localization Algorithms for End-to-end Service Failure Diagnosis

In this section, we address the problem of finding the set of root problems that best explains the set of observed positive and negative symptoms using a causality graph as the fault propagation model. In general causality graphs, the problem is known to be NP-hard [15]; the exact calculation of the best explanatory hypothesis requires the number of steps that is exponential with respect to the number of causality graph nodes. Moreover, to

the best of our knowledge no heuristic technique has been proposed so far that would allow an approximate calculation of the symptom explanation hypothesis in polynomial time based on a causality graph of any shape. Some researchers address this problem by reducing a causality graph to a much simpler bipartite graph using serial-parallel edge reduction operators [21]. Without the complex graph structure, it becomes easier to design an effective fault localization algorithm. (The technique presented in this section is suitable for this purpose.) Unfortunately, the causality graph reduction to a bipartite graph makes it difficult to modify the fault propagation model when network configuration or the probability distribution changes. We believe that the fault localization problem should be simplified by dividing it into smaller subproblems; each subproblem focuses on a subgraph of the original causality graph, typically representing some level of abstraction.

In this section, we present two algorithms for failure diagnosis with fault models represented by bipartite causality graphs: a combinatorial algorithm, sometimes considered optimal [2] and a novel technique capable of calculating symptom explanation in an effective, iterative, and incremental fashion.

5.1 Combinatorial algorithm

The combinatorial algorithm [2] presented in this section assumes a naive approach by evaluating all possible combinations of faults for their ability to explain the observed symptoms. For a given combination of faults $\mathcal{F}_i$ and a set of observed symptoms $\mathcal{S}_o$, the measure of goodness $g(\mathcal{F}_i, \mathcal{S}_o)$ is computed as follows.

$$\begin{aligned} g(\mathcal{F}_i, \mathcal{S}_o) = {} & P\{\text{all faults in } \mathcal{F}_i \text{ occurred}\} \cdot \\ & P\{\text{each symptom in } \mathcal{S}_o \text{ is caused by at least one fault in } \mathcal{F}_i\} \\ = {} & \prod_{f \in \mathcal{F}_i} P(f) \cdot \prod_{s \in \mathcal{S}_o} \Big(1 - \prod_{f \in \mathcal{F}_i} \Big(1 - P(s \mid f)\Big)\Big) \end{aligned} \quad (1)$$

While correlating real-life symptoms, it is frequently assumed that the number of faults that occurred is small. This suggests that in the combinatorial algorithm we should start evaluating fault combinations from those that contain the fewest faults and terminate the search as soon as an explanation of all symptoms is known. This leads to the following combinatorial algorithm.

Algorithm 1 (Combinatorial Algorithm)

for $i = 1$ *until* $i < |\mathcal{F}|$ *do*
  for all i*-fault combinations from* $\mathcal{F}$*,* $\mathcal{F}_i$
    compute $g(\mathcal{F}_i, \mathcal{S}_o)$
  if at least one $\mathcal{F}_i$ *is found such that* $g(\mathcal{F}_i, \mathcal{S}_o) > 0$
    return $\mathcal{F}_i$ *such that* $g(\mathcal{F}_i, \mathcal{S}_o)$ *is maximum*

It may be easily calculated that Algorithm 1 performs $\sum_{i=1}^{|\mathcal{F}|} \binom{|\mathcal{F}|}{i} \cdot i \cdot |\mathcal{S}_o| = \mathcal{O}(2^n)$ operations. However, when multiple concurrent faults are unlikely, the algorithm's practical complexity may be polynomial. In the simulation study presented in the following section, we determine if the exponential bound is a significant factor in practical applications and if implementation of other, more complicated algorithms is justified.

5.2 Incremental Hypothesis Update

The technique we describe in this section creates a number of alternative fault hypotheses ranked using a belief metric. The algorithm proceeds iteratively and after every symptom observation, it is able to output the set of the most probable hypotheses. The iteration triggered by the ith symptom, S_i, creates the set of hypotheses, $\mathcal{H}_i$, based on the set of hypotheses resulting from the previous iteration, $\mathcal{H}_{i-1}$, and the information about causal relationships between faults and symptoms stored in the causality graph. Every hypothesis $h_j \in \mathcal{H}_i$ is a subset of $\mathcal{F}$, and is able to explain all symptoms in $\{S_1, \ldots, S_i\}$. We define H_{S_i} as the set $\{F_k \in \mathcal{F}\}$ such that F_k may cause S_i, i.e., the causality graph contains a directed path from F_k to S_i. The set of hypotheses $\mathcal{H}_i$ is created from $\mathcal{H}_{i-1}$ by incorporating the explanation, H_{S_i}, of the last observed symptom, S_i. Every hypothesis $h_j \in \mathcal{H}_i$ is minimal, i.e., if any fault $F_l \in h_j$ is removed from h_j, hypothesis h_j is no longer able to explain all the observed symptoms.

The belief metric b_i, similarly to the measure of goodness $g()$ in Algorithm 1, represents the probability that all faults belonging to h_j have occurred, and that every observed symptom $S_k \in \{S_1, \ldots, S_i\}$ is explained by at least one of the faults in h_j. Formally, $b_i(h_j) = g(h_j, \{S_1, \ldots, S_i\})$.

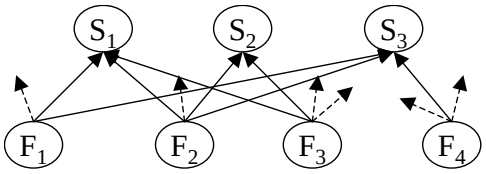
To incorporate the explanation of symptom S_i into the set of fault hypotheses, in the i-th iteration of the algorithm, we analyze every hypothesis $h_j \in \mathcal{H}_{i-1}$. If h_j is able to explain symptom S_i, we put it into $\mathcal{H}_i$. The hypotheses in $\mathcal{H}_{i-1}$ that do not explain S_i have to be extended by adding to each of them a fault from H_{S_i}; in the greedy approach, a new hypothesis may be created for every fault from H_{S_i}. Unfortunately, this would result in very fast growth to the size of $\mathcal{H}_i$ and, in consequence, would make the computational complexity of the algorithm unacceptable. Instead, we adopt the following heuristics. Fault $F_l \in H_{S_i}$ may be added to $h_j \in \mathcal{H}_{i-1}$ only if the size of h_j, $|h_j|$, is smaller than the size of any hypothesis in $\mathcal{H}_{i-1}$ that contains F_l and explains symptom S_i. The usage of this heuristics is derived from the fact that the probability of multiple simultaneous faults is small. Therefore, of any two hypotheses containing F_l, the hypothesis that contains the fewest faults is more likely to constitute the optimal symptom explanation. Thus, since it is not efficient to keep all possible hypotheses, we remove the hypotheses that are bigger in size. In the following Algorithm 2, $\mu(F_l)$ denotes the minimum size of a hypothesis that contains fault F_l calculated over all hypotheses in the current hypotheses set.

Algorithm 2 (Incremental Hypothesis Update)

let $\mathcal{H}_0 = \{\emptyset\}$ *and* $b_0(\emptyset) = 1$
for every observed symptom S_i*:*
  let $\mathcal{H}_i = \emptyset$
  for all $F_l \in \mathcal{F}$ *let* $\mu(F_l) = |\mathcal{F}|$
  for all $h_j \in \mathcal{H}_{i-1}$ *do*
    for all $F_l \in h_j$ *such that* $F_l \in H_{S_i}$ *do*
      $\mu(F_l) = \min(\mu(F_l), |h_j|)$
    add h_j *to* $\mathcal{H}_i$ *and calculate* $b_i(h_j)$
  for all $h_j \in \mathcal{H}_{i-1} - \mathcal{H}_i$ *do*
    for all $F_l \in \mathcal{F} \cap H_{S_i}$ *such that* $\mu(F_l) > |h_j|$ *do*
      add $h_j \cup \{F_l\}$ *to* $\mathcal{H}_i$ *and compute* $b_i(h_j \cup \{F_l\})$
choose $h_j \in \mathcal{H}_{|\mathcal{S}_o|}$ *such that* $b_{|\mathcal{S}_o|}(h_j)$ *is maximum*

We illustrate the algorithm with the following example. The fault model in Fig. 3 presents causal relationships between faults F_1, F_2, F_3, and F_4 and symptoms S_1, S_2, and S_3. Suppose the symptoms are observed in order S_1, S_3 and S_2. Initially, the only available hypothesis is $\emptyset$, which indicates that, given no symptom observations, we conclude that no faults occurred. Then, symptom S_1 arrives, whose explanation is H_{S_1}. As a result of extending $\emptyset$, we obtain $\mathcal{H}_1$. The explanation for symptom S_3 is H_{S_3}. Since F_1 and F_2 belong to hypotheses $\{F_1\}$ and $\{F_2\}$, respectively, $\{F_1\}$ and $\{F_2\}$ are placed in $\mathcal{H}_2$ and both $\mu(F_1)$ and $\mu(F_2)$ are set to 1. Hypothesis $\{F_3\}$ does not explain S_3; therefore, it has to be extended with faults in H_{S_3}. However, we cannot use F_1 and F_2 since their $\mu(.)$s $\leq |\{F_3\}|$. The only extension possible is $\{F_3, F_4\}$. In the next iteration, after symptom S_2 has been observed, we are allowed to extend $\{F_1\}$ by adding fault F_3 since $\mu(F_3) = |\{F_3, F_4\}| = 2$ while $|\{F_1\}| = 1$, but we are not allowed to extend $\{F_1\}$ by adding fault F_2, because $\mu(F_2) = |\{F_1\}| = 1$.

An important problem to solve is the efficient computation



$S_1 : H_{S_1} = \{F_1, F_2, F_3\} \rightarrow \mathcal{H}_1 = \{\{F_1\}, \{F_2\}, \{F_3\}\}$
$S_3 : H_{S_3} = \{F_1, F_2, F_4\} \rightarrow \mathcal{H}_2 = \{\{F_1\}, \{F_2\}, \{F_3, F_4\}\}$
$S_2 : H_{S_2} = \{F_2, F_3\} \rightarrow \mathcal{H}_3 = \{\{F_1, F_3\}, \{F_2\}, \{F_3, F_4\}\}$

Figure 3: Example of incremental hypothesis updating: (a) Example causality graph; (b) Sets of hypotheses created after observing symptoms S_1, S_3, and S_2.

of $b_i(h_j)$. We observe that $b_i(h_j)$ may be calculated iteratively based on $b_{i-1}(h_j)$ as follows:

1. If $h_j \in \mathcal{H}_i$ and h_j explains S_{i+1}

$$b_{i+1}(h_j) = b_i(h_j)(1 - \prod_{F_l \in h_j \cap H_{S_{i+1}}} (1 - P(S_{i+1}|F_l)) \quad (2)$$

2. Otherwise, if F_l explains S_{i+1}

$$b_{i+1}(h_j \cup \{F_l\}) = b_i(h_j)P(F_l)P(S_{i+1}|F_l) \quad (3)$$

To calculate the upper bound for the worst case computational complexity, we observe that the calculation of $b_i(h_j)$ is $\mathcal{O}(|h_j \cap H_{S_i}|) \subseteq \mathcal{O}(|H_{S_i}|) \subseteq \mathcal{O}(n)$, since in an n-node network a path may be composed of at most n links. The calculation of $b_i(h_j \cup \{F_l\})$ is $\mathcal{O}(1)$. The algorithm performs $|\mathcal{S}_o|$ iterations. In every iteration we execute two *for* loops. The first loop requires $\mathcal{O}((\max_i(|\mathcal{H}_i|)|H_{S_i}|)$ steps. The second loop requires $\mathcal{O}(\max_i(|\mathcal{H}_i|)|H_{S_i}| \cdot 1)$ operations. Therefore, the complexity of the entire algorithm is $\mathcal{O}(|\mathcal{S}_o| \max_i(|\mathcal{H}_i|)n)$. To get the precise bound we need to determine the bound for $\max_i(|\mathcal{H}_i|)$. It turns out that in rare cases the size of the hypothesis set may grow exponentially. To avoid this problem we set a limit on the number of hypotheses that may be created in each iteration; the least likely hypotheses are rejected when the limit is exceeded. The price we pay for this modification is that the best hypothesis is no longer guaranteed to be minimal. If the limit set on the size of the hypothesis set is $\mathcal{O}(n)$, operations involved in controlling the size of $\mathcal{H}_i$ do not increase the theoretical bound on the complexity of the entire algorithm. In the simulation study described in Section 6, we used the limit of $2n$. Thus, the complexity is $\mathcal{O}(|\mathcal{S}_o|n^2)$, i.e., $\mathcal{O}(|\mathcal{S}_o|n^2)$, and in the worst case it is $\mathcal{O}(n^4)$.

6 Simulation Study and Comparison of Algorithms

The algorithms presented in Section 5 were implemented in Java and evaluated through simulation on a set of randomly generated network topologies. For evaluation purposes, as a real-life application domain, we chose the end-to-end service failure diagnosis in the data link layer in a bridged network in which the path ambiguity is resolved using the Spanning Tree Protocol [16]. As a result, the shape of the considered network topologies is reduced to trees, thus making random generation of dependencies resembling real-life scenarios easier. We varied network size from 5 to 100 network nodes (25 nodes in the case of the Combinatorial algorithm). For every graph size, we randomly generated 100 spanning trees along with 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 $[0.5, 1)$.

To evaluate the algorithms, 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, and on $\mathcal{F}_c$ symptom probability distribution has been calculated. Then, the set of observed symptoms, $\mathcal{S}_o$, resulting from the faults in $\mathcal{F}_c$ was randomly 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 evaluate the algorithms.

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

In the above equations, *detection rate* represents the percentage of faults occurring in the network that were detected by an algorithm. *False positive rate* represents the percentage of faults proposed by the algorithm that were not occurring in the network in a considered experiment, i.e., they were false fault hypotheses. For every generated network topology, we executed 200 such experiments calculating the mean detection and false positive rates.

Figures 4 and 5 present the relationship between detection rate and false positive rate, respectively, and network size. The mean for a particular network size is an average over the mean detection (or false positive) rates for particular network topologies of that size, within statistically computed confidence intervals. We observe that there is no statistically significant difference in the detection and false positive rates between the Incremental and Combinatorial algorithms.

The experiments revealed that the detection and false positive rates depend on the network size. For small (5-node) networks, the number of symptoms observed is typically small (less than 10), which in some cases is not sufficient to precisely pinpoint the actual fault. When the network gets bigger, the number of observed symptoms increases, thereby increasing the ability to precisely detect the faults. Therefore, with the growing number of network nodes, the detection rate increases and the false positive rate decreases. On the other hand, as the network size grows, the multi-fault scenarios are getting more and more frequent. In multi-fault scenarios, it is rather difficult to detect all actual faults, which leads to partially correct solutions. When the number of alternative explanations is large, the algorithms are likely to choose a very probable, but not correct solution. Thus, for even bigger networks, detection rate decreases and false positive rate increases (Figures 4 and 5). The gradual drop of the detection rate observed in the case of Algorithm 2 suggests that this drop may be asymptotic. One can also conclude that both analyzed algorithms have the very satisfactory detection rate of at least 95% (for networks larger than 5 nodes), and that false positive rate for Algorithm 2 does not exceed 4%.

Fig. 6 presents a comparison of execution times for the Combinatorial and Incremental algorithms in the presence of one and two network faults. In the figure, the confidence intervals were omitted because of their negligible size. The Incremental algorithm performs better than the Combinatorial algorithm regardless of the number of faults and network size, and the difference between the algorithms increases sharply with the increasing number of faults in the system. The correlation time of the Incremental algorithm measured over the entire tested network size range is presented in Fig. 7. We find the execution time of the order of several seconds even for large networks and multi-fault scenarios very encouraging.

7 Conclusions and Future Work

In this paper, we presented a refinement of a layered model template proposed in [5] that allows non-deterministic modeling

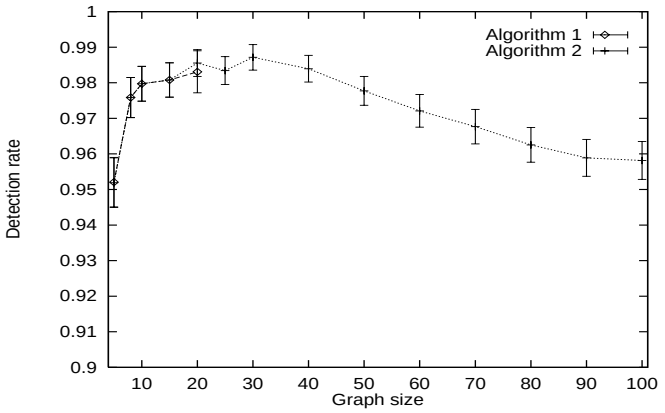


Figure 4: Comparison of accuracies achievable with Combinatorial and Incremental algorithms

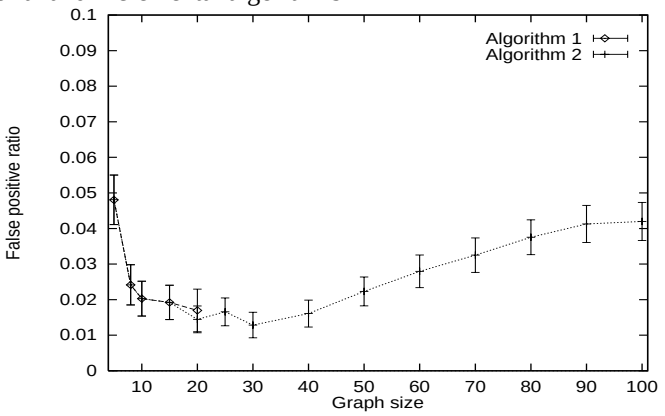


Figure 5: Comparison of false-positive metric values for Combinatorial and Incremental algorithms

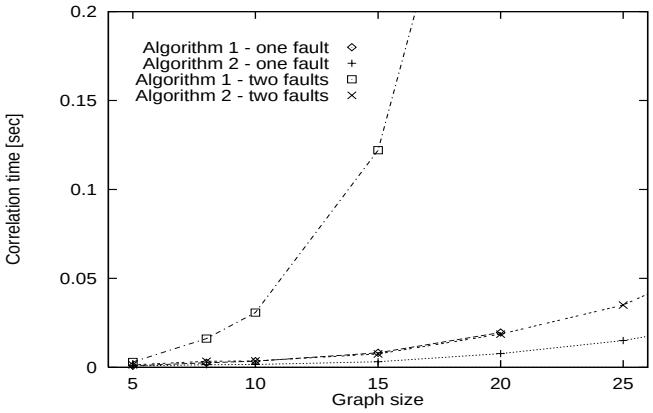


Figure 6: Correlation time for Combinatorial and Incremental algorithms in the presence of one and two network faults

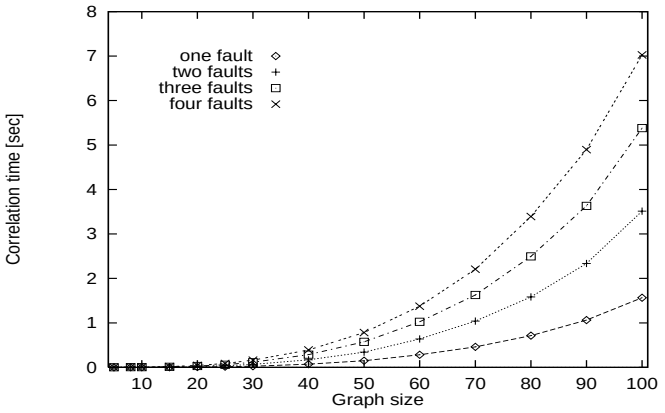


Figure 7: Fault localization time with the Incremental algorithm

of both availability and performance related problems associated with physical and abstract system components. The refinement also includes micro-models of end-to-end services provided in given protocol layers between end hosts via multiple intermediate hosts. We introduced and evaluated an event-driven fault localization algorithm capable of creating a failure explanation hypothesis in an incremental fashion. We showed that the proposed algorithm has close to the optimal accuracy and very promising performance.

Our future research will include the analysis of positive symptoms (i.e., the lack of failure observations), which may be used to decrease confidence in the failure of those hop-to-hop services for which many resultant end-to-end service failures did not occur. We will also equip the algorithm with the ability to deal with lost and spurious symptoms, i.e., symptoms which do not indicate any existing faults. We also plan to evaluate the algorithm on network topologies resembling the topology of the Internet.

In this paper, we considered 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 needed to install and run management agents on network devices, which collect the management information, and to regularly transmit the routing information over the network. In future research, we would like to investigate diagnosing end-to-end service failures without access to the accurate routing information.

REFERENCES

[1] K. Appleby, G. Goldszmidt, and M. Steinder. Yemanja – a layered event correlation engine for multi-domain server farms. In *IFIP/IEEE International Symposium on Integrated Network Management VII*, Seattle, WA, May 2001. IEEE Publishing.

[2] A. T. Bouloutas, S. Calo, and A. Finkel. Alarm correlation and fault identification in communication networks. *IEEE Transactions on Communications*, 42(2/3/4):523–533, 1994.

[3] J. Case, M. Fedor, M. Schoffstall, and J. Davin. *A Simple Network Management Protovol (SNMP)*. IETF Network Working Group, 1990. RFC 1157.

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

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

[6] M. Hasan, B. Sugla, and R. Viswanathan. A conceptual framework for network management event correlation and filtering systems. In M. Sloman, S. Mazumdar, and E. Lupu, eds, *IFIP/IEEE International Symposium on Integrated Network Management VI*, pp. 233–246, Boston, MA, May 1999. IEEE Publishing.

[7] S. Kätker. A modeling framework for integrated distributed systems fault management. In C. Popien, ed, *Proceeding of the IFIP/IEEE International Conference on Distributed Platforms*, pp. 187–198, Dresden, Germany, 1996.

[8] S. Kätker and M. Paterok. Fault isolation and event correlation for integrated fault management. In Lazar et al. [11], pp. 583–596.

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

[10] S. Kliger, S. Yemini, Y. Yemini, D. Ohsie, and S. Stolfo. A coding approach to event correlation. In Sethi et al. [17], pp. 266–277.

[11] A. Lazar, R. Saracco, and R. Stadler, eds. *IFIP/IEEE International Symposium on Integrated Network Management V*, San Diego, CA, May 1997. Chapman and Hall.

[12] D. M. Meira and J. M. S.Nogueira. Modelling a telecommunication network for fault management applications. In *Proceedings of IEEE Network Operation and Management Symposium*, pp. 723–732, 1998.

[13] Y. A. Nygate. Event correlation using rule and object based techniques. In Sethi et al. [17], pp. 278–289.

[14] D. Ohsie, A. Mayer, S. Kliger, and S. Yemini. Event modeling with the MODEL language. In Lazar et al. [11], pp. 625–637.

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

[16] R. Perlman. *Interconnections, Second Edition: Bridges, Routers,*

Switches, and Internetworking Protocols. Addison Wesley, 1999.

[17] A. S. Sethi, Y. Reynaud, and F. Faure-Vincent, eds. *IFIP/IEEE International Symposium on Integrated Network Management IV*, Santa Barbara, CA, May 1995. Chapman and Hall.

[18] M. Steinder. Fault localization in communication networks - a survey. Technical report, CIS Dept., Univ. of Delaware, Feb. 2001.

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

[20] M. Steinder and A. S. Sethi. The present and future of event correlation: A need for end-to-end service fault localization. In N. Callaos et al., ed, *World Multi-Conf. Systemics, Cybernetics, and Informatics*, vol. XII, pp. 124–129, Orlando, FL, 2001.

[21] 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.