The present and future of event correlation: A need for end-to-end service fault localization

Małgorzata Steinder, Adarshpal S. Sethi
Computer and Information Sciences Department
University of Delaware, Newark, DE

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. Modern enterprise environments require that fault diagnosis be performed in integrated fashion in multiple layers of the protocol stack and that it include diagnosing performance problems. This paper reviews the existing approaches to fault localization and presents its new facets revealed by the demands of modern enterprise systems. We also present end-to-end service failure diagnosis as a critical step towards multi-layer fault localization in an enterprise environment.

Keywords: Fault diagnosis, event correlation

1 INTRODUCTION

Fault diagnosis is a central aspect of network fault management. Since faults are unavoidable in communication systems, their quick detection and isolation is essential for systems' robustness, reliability and accessibility. 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 manually by an expert or a group of experts experienced in managing communication networks based on network disorder symptoms displayed on a management console. As systems grew larger and more complex, automated fault localization techniques became critical. The benefits of designing automated fault localization techniques include: decreasing labor cost, improving fault diagnosis performance, and improving the accuracy of fault diagnosis process.

Automated fault localization has been recognized, in both research and industrial environments, as a necessity in large communication systems. Until recently, fault localization efforts concentrated mostly on diagnosing faults related to network connectivity in lower layers of the protocol stack (typically physical and data-link layers), and its major goal was to isolate faults related to the availability of network 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 application and service layers. This paper discusses the resultant approaches to automated fault localization, their benefits and shortcomings. We also discuss new challenges of the fault localization problem revealed by the demands of modern enterprise systems along with already existing attempts to address them.

In Section 2, we define fault localization and the most common difficulties it involves. In Section 3, available approaches to fault localization are discussed and compared. The challenges of the fault localization problem in an enterprise environment are discussed in Section 4. In Section 5, we describe how these challenges have been addressed so far and justify the need for further research in this area, in particular in the area of end-to-end service failure diagnosis.

2 FAULT LOCALIZATION PROBLEM

Fault localization is a process of isolating faults responsible for the observable malfunctioning of the managed system. In this section, we define the vocabulary related to fault localization and describe the most common problems it is associated with.

Event is an exceptional condition occurring in the operation of the hardware or software of the managed network [22, 41].

Faults (also referred to as *root problems*) constitute a class of network events that can be handled directly [22, 41]. Faults may be classified according to their duration time as: (1) permanent, (2) intermittent, and (3) transient [40]. Permanent fault exists in a network until a repair action is taken. Intermittent faults occur on a discontinuous and periodic basis, causing degradation of service for short periods of time. However, frequently re-occurring intermittent faults significantly jeopardize service performance. Transient faults cause a temporary and minor degradation of service. They are usually repaired automatically [40].

Error, a consequence of fault, is defined as a discrepancy between a computed, observed, or measured value or condition and a true, specified, or theoretically correct value or condition [40]. Faults may cause one or more errors. Some errors result in a deviation of a delivered service from the specified service that is visible to the outside world. The term *failure* is used to denote this type of error. Other errors are not visible externally. However, an error in a network device or software may cause the malfunctioning of dependent network devices or software. Thus, errors may propagate within the network causing failures of faultless hardware or software. In order to correct an error, the fault which caused the error has to be resolved; therefore, errors are typically not handled directly.

Symptoms are external manifestations of failures [22]. They are observed as *alarms* – notifications of a potential failure [22, 25, 41]. These notifications may originate from management agents via management protocol messages (e.g., SNMP trap [6]), management systems, which monitor the network status, e.g., using command *ping*, system log-files, or character streams sent by external equipment.

Some faults may be directly observable, i.e., they are problems and symptoms at the same time. However, many types of faults are unobservable due to (1) their intrinsically unobservable nature, (2) local corrective mechanisms built into the management system that destroy evidence of fault occurrence, or (3) the lack of management functionality necessary to provide indications of fault existence [8]. Examples of intrinsically unobservable faults include livelocks and deadlocks. Some faults may be partially-observable – the management system provides indications of fault occurrence, but the indications are not sufficient to precisely locate the fault.

In a communications network, a single fault may cause a number of alarms to be delivered to the network management center. Multiple alarms may be a result of (1) fault re-occurrence, (2) multiple invocations of a service provided by a faulty component, (3) generation of multiple alarms by a device for a single fault, (4) detection of and issuing a notification about the same network fault by many objects (hardware or software network components) simultaneously, and (5) error propagation to other network objects causing them to fail and generate additional alarms [15]. It may be argued that typical networked systems provide plenty

of information necessary to infer the existence of faults [41].

The following paragraphs present some common problems that have to be addressed by a fault localization technique.

> *Fault evidence may be ambiguous, inconsistent and incomplete* [8, 14, 28].

Ambiguity in the observed alarm set stems from the fact that the same alarm may be generated as an indication of many different faults. Inconsistency is a result of a disagreement between different devices with regard to the facts related to network operation; one object may have the perception that a component is operating correctly, while another may consider the component faulty. Incompleteness is a consequence of alarm loss or delay. It is essential that the fault management system be able to create a consistent view of network operation even in the presence of ambiguous, inconsistent and incomplete information.

> *The available system knowledge and fault evidence may contain uncertain information* [7, 8, 14, 30].

A suit of alarms generated by a fault may depend on many factors such as dependencies between network devices, current configurations, services in use since fault occurrence, presence of other faults, values of other network parameters, etc. Due to this nondeterminism, system knowledge available to the management application may be subject to inaccuracy and inconsistency. Fault evidence may also be inaccurate because of spurious alarms generated by transient problems; once the problem disappears, the generated alarms do not correspond to any fault.

> *Multiple unrelated faults may occur simultaneously generating overlapping sets of alarms* [8, 37].

This additional complication in the fault localization problem results from the fact that different related and unrelated faults may happen within a short time of one another. The event management system should be able to detect unrelated simultaneous problems even if they generate overlapping sets of alarms.

> *Multiple different fault hypotheses may exist that explain the set of observed symptoms* [28].

Given that a single alarm may indicate different types of faults that occurred in different communication devices, fault localization may be unable to give a definite answer. Some approaches discussed in Section 3 combine fault localization with testing in order to resolve these ambiguities. These approaches are usually tailored to locating specific network faults. In the general case, the lack of automated testing techniques makes it impossible to verify a possible answer in real-time. Therefore, most existing techniques try to isolate a set of probable fault hypotheses that may be later verified on- or off-line depending on the available testing techniques. Preferably, a confidence measure should be associated with every formulated hypothesis based on some belief metric, e.g., the information cost of a hypothesis, the probability that it is valid, etc. The optimality criteria may include minimizing the size of the hypotheses set, the solution cost, the error probability, etc.

> *In large systems, performing the fault localization process and maintaining the available knowledge base within a single management application may be computationally infeasible* [27, 40, 41].

In large communication systems, it is also impractical to assume that the fault localization process has access to and the computational capability of processing the information about the entire system. Many researchers have concluded that the fault localization process in large networks should be performed in a distributed fashion by a group of event management nodes with data and processing complexity divided among them. Each of several managers governs a subset of network hardware and/or software components within boundaries marked by protocol layers or network domains. While they propagate across the network, errors cross boundaries of management domains. As a result, the fault management system may be provided with indications of faults that did not happen in its management domain and/or be unable to detect all symptoms of faults existing in its management domain. Distributed fault localization schemas should allow the management nodes to reach the solution collectively.

> *The strength of causal correlation between events may change with the length of the period of time between the events' occurrences* [23, 32].

The important aspect related to fault localization is representation of time. Events are related not only causally but also temporally. Therefore, the correlation process has to provide means to represent and interpret time associated with event occurrence as well as a technique of correlating events related with respect to the time of their occurrence and duration.

3 FAULT LOCALIZATION TECHNIQUES

In the past, numerous fault localization paradigms were proposed. They derive from different areas of computer science, including artificial intelligence, graph theory, neural networks, information theory, and automata theory. In Figure 1, a classification of the existing solutions is proposed. The most common approaches are as follows [26]:

- model-based reasoning tools
- fault propagation models
- model traversing techniques
- case-based reasoning tools

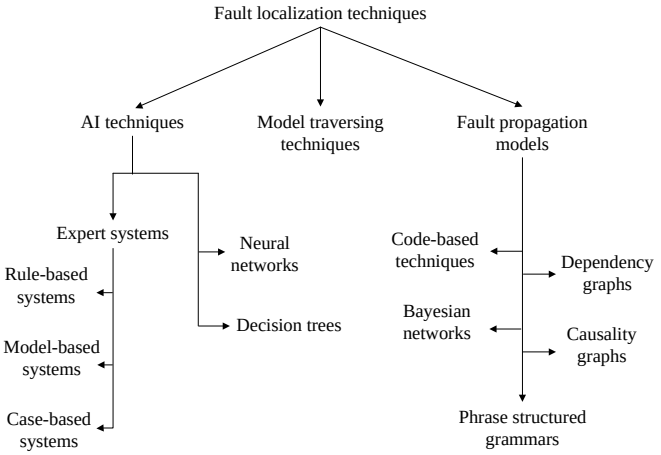

Figure 1: Classification of fault localization techniques

In this section, some of the approaches presented in Figure 1 will be described and compared.

Artificial Intelligence techniques for fault localization
The most widely used Artificial Intelligence (AI) techniques in the field of fault localization and diagnosis are expert systems. Expert systems try to reflect actions of a human expert when solving problems in a particular domain. Their knowledge base imitates knowledge of a human, which may be either surface – resulting from experience, or deep – resulting from understanding the system behavior from its principles. Expert systems applied to the fault localization problem differ with respect to the structure of the knowledge they use. Approaches that rely solely on surface knowledge are referred to as rule-based reasoning systems [32, 33], which in the domain of fault localization exploit a forward-chaining inferencing mechanism.

Rule-based systems do not require profound understanding of the architectural and operational principles of the underlying system and, for small systems, may provide a powerful tool for eliminating the least likely hypotheses. The downsides of rule-based systems include the inability to learn from experience, inability

to deal with unseen problems, and difficulty in updating the system knowledge [31]. Rule-based systems are difficult to maintain because the rules frequently contain hard-coded network configuration information. Although approaches have been proposed to automatically derive correlation rules based on the statistical analysis of alarm databases [29], it is still necessary to regenerate a large portion of correlation rules when the system configuration changes. Rule-based systems are also unable to deal with inaccurate information and get convoluted if timing constraints are included in the reasoning process. Also, rule interactions may result in unwanted side-effects, difficult to verify and change.

Model-based approaches [5, 9, 22, 35] incorporate deep knowledge in the form of a model of the underlying system. They constitute a class of expert systems that are the most widely used for fault diagnosis. The system model provides information on network topology [41] and on how a failure condition or alarm in one component is related to failure conditions or alarms in other components [15]. The model-based approaches differ with regard to the technology used to define the system model. Thanks to representing deep knowledge of the network connectivity and operation, model-based approaches have the potential to solve novel problems and their knowledge may be organized in an expandable, upgradeable and modular fashion. However, the models may be difficult to obtain and keep up-to-date, and their manipulation is computationally complex.

Case-based systems [31] make their decisions based on experience and past situations. They try to acquire relevant knowledge of past cases and previously used solutions to propose solutions for new problems. If a direct match for an open problem is found in the database, the solution from the matching case is returned. Otherwise, the case is chosen that most closely matches the open problem. Then, the solution applied to the chosen past case is adapted for the open problem. Case-based systems are well suited for learning correlation patterns. They are resilient to changes in network configuration. However, case-based systems require an application-specific model for the resolution process and are computationally complex.

Model traversing techniques

Model traversing techniques [11, 15, 24, 26] use a formal representation of a communication system with clearly marked relationships between network entities. Failures in the communication system propagate along relationships between network entities. By exploring these relationships, starting from the network entity that reported an alarm, the fault identification process is able to determine which alarms are correlated and locate faulty network elements. Model traversing techniques are robust against frequent network configuration changes [26]. They are particularly attractive when automatic testing of a managed object may be done as a part of the fault localization process. Model traversing techniques seem natural when relationships between objects are graph-like and easy to obtain. They naturally enable the design of distributed fault localization algorithms. However, they are unable to model situations in which the failure of a device may depend on a logical combination of other device failures.

Fault propagation models

Fault propagation models require a-priori specification of how a failure condition or alarm in one component is related to failure conditions or alarms in other components [15]. Based on this information, the fault localization algorithm tries to isolate the root cause of the observed set of symptoms. Given a set of observed symptoms, the algorithm solving the correlation problem should return a number of fault hypotheses along with some measure of their correctness. It has been shown that the correlation problem is NP-hard [3, 28].

Most algorithms use the minimum number of faults as a measure of hypothesis goodness, based on the assumption that simultaneous occurrence of multiple faults is an unlikely event. In some cases, however, independent multiple faults happen simultaneously with a probability that may not be ignored. In these cases the appropriate heuristics is to find the explanation that offers the greatest level of confidence [15]. As a measure of confidence, the probability of fault occurrence or information cost associated with fault occurrence are used.

A number of different techniques based on fault propagation models have been proposed [3, 15, 28]. They include causality and dependency graph models, context-free grammars, and code-based techniques.

Fault localization techniques based on dependency and causality graph fault models: A *causality graph* is a directed acyclic graph $G_c(E, C)$ whose nodes E correspond to events and whose edges C describe cause-effect relationships between events. An edge $(e_i, e_j) \in C$ represents the fact that event e_i causes event e_j, which is denoted as $e_i \rightarrow e_j$ [12]. A *dependency graph* is a directed graph $G = (O, D)$, where O is a finite, nonempty set of objects in use and D is a set of edges between these objects. The directed edge $(o_i, o_j) \in D$ denotes the fact that an error or fault in o_i may cause an error in o_j. Probabilities may be assigned to nodes and edges in both causality and dependency graph. Dependency and causality graphs are equivalent when used as a representation of a communication system in which every object may experience only one type of failure. Otherwise, causality graph has the potential to provide a more detailed view of the system by allowing multiple causality graph nodes per single network object.

Causality graph is used in [12] as a framework for developing fault localization algorithms based on a deduction system. The technique assumes 'AND' relationships between errors caused by the same event, i.e., if an event e_i happens, then all events e_j, such that $e_i \rightarrow e_j$, happen. Because of this determinism, when a significant number of symptoms is lost, the algorithm may be unable to output a solution.

The *divide and conquer* $\mathcal{O}(N^3)$ algorithm [28] uses a dependency graph as a fault model. Before the correlation algorithm is started, a set of all problems that may cause one or more alarms in the observed alarm cluster, alarm cluster domain, is determined. The actual correlation procedure proceeds recursively dividing the alarm cluster domain into two subsets characterized by maximum mutual dependency. Maximum mutual dependency of two sets means that the label assigned to any edge between two nodes in the same set is higher than the label assigned to any edge connecting two nodes belonging to different sets. If the subset with a higher probability that one of its members was a primary source of failure is able to explain all alarms, the next step of the recursion is invoked for the entire alarm cluster and the higher-probability alarm cluster sub-domain. Otherwise, the alarm cluster domain is divided into two sub-clusters corresponding to the maximum mutual dependency sets. The next step is executed for both alarm cluster sub-domains and their corresponding alarm sub-clusters. The algorithm always explains all the observed alarms, but may fail to give their best explanation [28]. It does not handle lost or spurious symptoms.

Context-free grammar: The natural feature of context-free grammars, which allows expressions to be built from subexpressions, may be effectively used to represent a hierarchically organized communication system [3]. In this model, terminals correspond to the indivisible network components. Productions are used to build compound network objects from the already defined objects. A context-free grammar allows the modeling of systems with complex dependencies between managed objects [3]. The fault localization technique proposed in [3] chooses the minimum set of faults that explains all observed alarms. It looks for a fault that belongs to the greatest number of alarm domains. If there are more than one such faults, the one with the highest probability of occurrence (or lowest information cost) is chosen.

Code-based techniques: Code-based techniques use information-theory to facilitate the process of fault localization. They were first used in SMARTS InCharge system [41].

Fault propagation patterns in code-based techniques are represented by a codebook [30, 41]. For every problem, a code is generated that makes it possible to distinguish this problem from other problems. In the deterministic code-based technique, a code is a sequence of values from $\{0, 1\}$. Problem codes are generated based on the available system model and a fault information model. InCharge uses a causality graph as an intermediate fault propagation model from which the codebook is generated. The causality graph is pruned to remove cycles, unobservable events, and indirect symptoms (symptoms that cause other symptoms, thus they do not provide any additional information).

As a result of pruning, the causality graph contains direct cause-effect relationships between problems and symptoms. It is then represented as a matrix whose columns are indexed by problems and rows are indexed by symptoms. The matrix cell indexed by (s_i, p_j) contains a probability that problem p_j causes symptom s_i. In the deterministic model the probability is either 0 or 1. The correlation matrix is then optimized to minimize the number of symptoms that have to be analyzed but still ensure that the symptom patterns corresponding to different problems allow the problems to be distinguished. The optimized correlation matrix constitutes a codebook whose columns are problem codes. The observed code is a sequence of symbols from $\{0, 1\}$, with 1 denoting the appearance of a particular symptom, and 0 meaning that the symptom has not been observed. Fault localization is performed by finding a fault in the correlation matrix whose code is the closest match to the observed coded. Since the coding phase is performed only once, the InCharge correlation algorithm is very efficient. Its computational complexity is bounded by $(k + 1) \log(p)$, where k is the number of errors that the decoding phase may correct, and p is the number of problems [30].

4 CURRENT CHALLENGES IN FAULT LOCALIZATION

In the past, fault diagnosis efforts concentrated mostly on detecting, isolating, and correcting faults related to network connectivity. The diagnosis focused on lower layers of the protocol stack (typically physical and data-link layers), and its major goal was to isolate faults related to the availability of network resources, such as broken cable, inactive interface, etc. Since these types of problems are relatively rare, most event correlation techniques presented in Section 3 assume that only one fault may exist in the system at any time, and do not attempt to detect multiple simultaneous faults. In addition, these techniques frequently use a deterministic model, which implies that all dependencies and causal relationships are known with a 100% certainty. The prominent example of a diagnostic system suited for low-level deterministic fault diagnosis is InCharge [41], which utilizes the codebook technique described in Section 3. InCharge belongs to the most popular fault management systems on the market today. To the best of our knowledge none of the techniques available today solve all the fault localization problems described in Section 2. It may be argued that in the past such solutions were acceptable in practical applications.

Popularization of modern enterprise services such as e-commerce, telecommuting, virtual private networks, and application service provision made it necessary to manage them effectively. To manage enterprise services it is necessary to address some of the problems presented in Section 2.

Multi-layer fault localization
In multi-layer communication systems, errors propagate up the protocol stack causing malfunctioning of network objects (both hardware and software ones) in various layers. Frequently, inability to connect to a Web server may be traced down to a faulty physical connection between neighboring network devices, the lack of resources on the Web server, network layer buffer overflow, etc. To be able to isolate these types of faults based on the "Web server inaccessible" observation, fault diagnosis may no longer be constrained to the lowest layers of the protocol stack. On the contrary, it has to reach through transport and application layers into the service layer, and integrate fault diagnosis across multiple protocol layers.

To perform fault localization in a multi-layer environment it is useful to create a system model, which shows dependencies between objects in neighboring protocol layers. Unfortunately, building and maintaining such a model is not easy because of a huge number of dependencies in real-life systems and their frequent changes. This makes the manual creation of the model infeasible. In large multi-layer systems the model should be built automatically and dynamically adapted to the system configuration changes. In lower layers, current dependencies may be obtained with the use of management agents on network devices and/or management applications, e.g., *ping* or *traceroute*. In higher layers, due to the lack of the appropriate management information bases, other methods have to be proposed. Several reports on the automatic obtaining of the system dependencies have been published, e.g., [4].

Performance problems' diagnosis
E-business customers increasingly demand support for quality of service (QoS) guarantees. QoS parameters are negotiated between the customer and the e-business as a part of Service Level Agreements [13] (SLAs), which also specify pricing rules for the offered services and a penalty schema to be used if the quality of the offered service violates the agreed upon SLA contract. Various techniques have been investigated to supervise execution of the SLA contract [1], and to notify the management application about any QoS violations. In addition to dealing with resource availability problems, fault diagnosis has to isolate the causes of these performance/QoS related notifications. Support for performance problems' diagnosis is needed in both the system model and fault localization algorithm. The system model should represent different types of network object failures (availability and performance related ones), which can frequently be enumerated, e.g., total failure, data loss, data delay, erroneous output, etc. Performance related problems are more frequent than availability related ones; in large systems, it is likely for two unrelated performance problems to occur simultaneously. Therefore, fault diagnosis has to be able to isolate multiple unrelated root causes.

Uncertainty
Most of the approaches to fault localization presented in Section 3 use a deterministic fault model, i.e., they assume that the dependency link from **a** to **b** implies that if **a** fails **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, deterministic fault localization techniques are rather difficult to apply when faults are related to service performance. In the transport and application layers, frequent reconfigurations of service dependencies make it impossible to keep such a deterministic model up-to-date. Because of complex dependencies between network objects, it is possible for an object failure to induce a failure of a dependent object at one time and have no effect on the dependent object at another time.

Uncertainty about dependencies between communication system entities is represented by assigning probability to the links in the dependency or causality graph [28, 30]. 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** (OR relationship between alternative causes of the same event), and (3) faults are independent of one another. When performance problems are taken into account by introducing multiple failure modes in every dependency graph object, instead of

a single probability values, probability matrices should rather be assigned to the dependency links, which for every object and its every possible failure describe the probability that it results in a particular dependent object's failure.

In nondeterministic fault model, alarm correlation aims at finding the most probable explanation of the observed alarms. In the past, some research has been performed on finding appropriate heuristics to solve the problem in polynomial time, including the *divide and conquer* algorithm described in Section 3. Another approach to dealing with uncertainty is based on belief networks [36], which were used to diagnose failures in linear light-wave networks [7] and break faults in dynamically routed networks [39]. Both techniques are tailored towards specific applications and focus on particular types of faults. Their uncertainty model is restricted by not allowing the modeling of nondeterminism within relationships between objects. Therefore, more research is needed in this area.

Temporal correlation
Many researchers have pointed out that temporal relationships between events should be explored in the alarm correlation process [12, 23, 32]. So far, the temporal correlation has been performed by specifying a *time window*, i.e., a period of time over which events are observed; at the end of each time window, the correlation algorithm exploring causal correlation is run [28, 30]. Usage of time windows has several drawbacks: (1) it does not allow testing to be interleaved with alarm collection, (2) it is difficult to apply to alarm correlation across multiple protocol layers, because different time windows may apply to different layers, and (3) it is inefficient because the entire computational effort is performed at the end of a time-window instead of being distributed over time. It is believed that event-driven fault localization [2, 12] is a better approach to the problem. Preferably, the final solution should be created incrementally based on the solutions available after previous symptom observations.

Real-time fault diagnosis
Substantial loss of revenue may result if faults within the enterprise communication system are not detected, isolated, and corrected soon enough. The amount of time available to perform fault localization depends on the latency of fault detection mechanisms used in the system. Performance monitoring mechanisms are usually invasive, consuming network and/or device resources in order to collect data. In order not to interfere with the services provided over the monitored network, the problem detection mechanisms sample the network state in certain time intervals, typically ranging from tens of seconds to several minutes. Frequently, a consistent pattern has to be observed over multiple sampling intervals in order to notify about the detected abnormality. Although fault localization should be performed as fast as possible, fault localization time comparable to the time necessary to detect problems is usually acceptable. When iterative techniques are used, fault localization may be performed simultaneously with problem detection for as long as it is necessary to observe symptoms. Thus, in many practical applications, fault localization should take no more than a few minutes. In large networks, achieving this goal requires fault localization techniques with low computational complexity.

5 A NEED FOR END-TO-END SERVICE FAULT LOCALIZATION

Recent publications on fault localization recognize the need for multi-layer fault localization and try to address some of the issues described in Section 4.

Yemanja [2], a model-based reasoning system, tries to overcome the difficulties involved in managing system models by defining a set of entity models, which correspond to physical or conceptual network entities, e.g., network layers. It avoids maintaining an explicit network model by providing scenario templates organized in a hierarchical structure, which are instantiated with the data obtained from the arriving event attributes or from the configuration database. The scenario templates embedded into entity models are reusable and communicate using internal *composite events* [32]. In addition, the internal event publishers need not be aware which components consume the events that they forward; therefore, a change to a higher-level entity does not require changes to any of the lower-level entities. As events propagate up the entity model hierarchy, their semantics becomes more and more abstract so that higher-level scenarios do not need to know the low-level details of the network state to correlate low-level faults with the symptoms observed in the higher layers.

Gopal et al. [10] proposed a model for multi-layer fault diagnosis. Although it was originally used to enhance the presentation of the system state on the management console for the purpose of manual fault localization, the model is also suitable for automatic fault localization with the use of model traversal techniques or algorithms based on fault propagation models. In the layered fault model, the definition of entity dependencies is based on real-life relationships between layers on a single host and between network nodes communicating within a single protocol layer. The fault model components may be generally divided into *services*, *protocols*, and *functions*. 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.

Both Yemanja [2] and the layered system model [10] address the issue of vertical fault propagation. However, the fault propagation techniques proposed in [2, 10] are based on the assumption that, when higher-level entity fails or experiences performance problems, it is easy to choose a small set of lower-level entities which may be responsible for the higher-level entity failure, which is not the case in complex communication systems. End-to-end connectivity in a given network layer is frequently achieved through a sequence of intermediate nodes invisible to the layers above. For example, in the data-link layer, end-to-end connectivity is provided by a network of bridges; in the network layer, end-to-end connectivity is realized by a network of routers. Similar scenarios exist in the application layer. We say that the end-to-end service is realized by a network of hop-to-hop services. End-to-end performance problems in the network layer may be caused by, e.g., bit errors introduced by a serial link between two routers over which the end-to-end communication is provided. In this case, in order to explain the performance problems experienced by the end-to-end service, one has to find the hop-to-hop service responsible for the end-to-end service failure. Clearly, the number of such hop-to-hop services may be huge in a large communication system. We believe that solving the problem of end-to-end service failure diagnosis is critical to perform fault management in a multi-layer enterprise environment.

Research currently performed in Network Management Lab at University of Delaware attempts to solve the problem of end-to-end service failure diagnosis addressing the issues presented in Section 4. It makes contributions in the following areas [38].

Building the system model suitable for end-to-end service failure diagnosis: The system model is based upon the layered model template proposed in [10], which was refined to expose the model for the end-to-end service failure diagnosis. An end-to-end service offered by layer L is represented as a subgraph showing all the end-to-end service hop-to-hop components. Thus, an end-to-end service offered in layer L between hosts **a** and **b** is built of (depends on) hop-to-hop services offered in layer L between subsequent hosts on the path of the layer L packet from node **a** to node **b**. A set of possible failure modes are associated

with every node of the refined layered dependency graph. Uncertainty is modeled by assigning a probability matrix rather than a single probability value to every dependency graph edge. So far, the refined system model has been designed for the situation when the set of hop-to-hop links in a given layer used to provide an end-to-end service in this layer may be determined based on the information available through management protocols.

Modeling failure propagation patterns in a multi-layer system: Failure propagation is modeled as a belief network [36]. The mapping of the layered dependency graph into a belief network is proposed. The fault localization problem is mapped into the problem of computing queries in belief network.

Designing probabilistic fault localization algorithms for end-to-end service failure diagnosis: An application of two iterative algorithms for computing queries in belief networks [36] to bipartite belief networks representing fault propagation within the end-to-end service model has been investigated. In addition, a novel technique that allows to build a fault hypothesis in iterative and incremental fashion is proposed.

Evaluating the proposed algorithms through analytical study and through simulation: The proposed algorithms have been shown to have a polynomial computational complexity bound. They were implemented in Java and evaluated through an extensive simulation study on a large set of randomly generated tree-shaped network topologies. The algorithms' accuracy and performance is compared with that of an optimal, but exponential, algorithm for computing queries in belief networks. The initial results of the study are very encouraging.

Currently, our research focuses on extending the model to represent symptom loss and/or spurious generation, dealing with imprecise specification of conditional probability distribution, and evaluating the algorithms on arbitrary real-life network topologies. We plan to investigate the problem of end-to-end service diagnosis when the set of hop-to-hop links in a given layer used to provide an end-to-end service may not be easily obtained.

6 CONCLUSIONS

For the past ten years, fault localization has been a very vivid research area. Fault localization solutions available today, which were described in Section 3, focus on low-level faults related to resource availability. Modern enterprise environments require that fault diagnosis be performed in integrated fashion in multiple layers of the protocol stack. This demands solutions related to modeling multi-layer, dynamically changing communication systems, and representing uncertainty involved in dependencies between network objects. Another goal is to find iterative real-time fault localization techniques capable of isolating multiple root causes of the observed problems in the presence of uncertain information in the system model and the set of observed symptoms. The entirely new challenge is in the area of performance problem diagnosis.

End-to-end service fault localization is a critical step towards multi-layer fault diagnosis. Research performed in Network Management Lab at University of Delaware focuses on finding efficient event correlation techniques by building a suitable system model, modeling failure propagation using belief networks, and designing probabilistic fault localization algorithms.

7 REFERENCES

[1] K. Appleby, S. Fakhouri, L. Fong, G. Goldszmidt, M. Kalantar, S. Krishnakumar, D. Pazel, J. Pershing, and B. Rochwerger. Océano – SLA-based management of computing utility. In IM'01 [16]. (to appear).

[2] K. Appleby, G. Goldszmidt, and M. Steinder. Yemanja – a layered event correlation engine for multi-domain server farms. In IM'01 [16].

[3] 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.

[4] A. Brown, G. Kar, and A. Keller. An active approach to characterizing dynamic dependencies for problem determination in a distributed application environment. In IM'01 [16]. (to appear).

[5] S. Brugnoni, R. Manione, E. Montariolo, E. Paschetta, and L. Sisto. An expert system for real time diagnosis of the Italian telecommunications network. In IM'93 [19].

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

[7] R. H. Deng, A. A. Lazar, and W. Wang. A probabilistic approach to fault diagnosis in linear lightwave networks. In IM'93 [19], pp. 697–708.

[8] A. Dupuy, J. Schwartz, Y. Yemini, G. Barzilai, and A. Cahana. Network fault management: A user's view. In IM'89 [17], pp. 101–107.

[9] M. Frontini, J. Griffin, and S. Towers. A knowledge-based system for fault localization in wide area network. In IM'91 [18], pp. 519–530.

[10] R. Gopal. Layered model for supporting fault isolation and recovery. In NOMS'00 [34].

[11] B. Gruschke. Integrated event management: Event correlation using dependency graphs. In *DSOM'98*, pp. 130–141, 1998.

[12] M. Hasan, B. Sugla, and R. Viswanathan. A conceptual framework for network management event correlation and filtering systems. In IM'99 [21], pp. 233–246.

[13] A. Hiles. *Service Level Agreements : Managing Cost and Quality in Service Relationships*. Chapman and Hall, 1993.

[14] P. Hong and P. Sen. Incorporating non-deterministic reasoning in managing heterogeneous network faults. In IM'91 [18], pp. 481–492.

[15] K. Houck, S. Calo, and A. Finkel. Towards a practical alarm correlation system. In IM'95 [20], pp. 226–237.

[16] *Proc. of IFIP/IEEE International Symposium on Integrated Network Management*, 2001.

[17] *Proc. of IFIP/IEEE International Symposium on Integrated Network Management*, 1989.

[18] *Proc. of IFIP/IEEE International Symposium on Integrated Network Management*, 1991.

[19] *Proc. of IFIP/IEEE International Symposium on Integrated Network Management*, 1993.

[20] *Proc. of IFIP/IEEE International Symposium on Integrated Network Management*, 1995.

[21] *Proc. of IFIP/IEEE International Symposium on Integrated Network Management*, 1999.

[22] G. Jakobson and M. D. Weissman. Alarm correlation. *IEEE Network*, pp. 52–59, Nov. 1993.

[23] G. Jakobson and M. D. Weissman. Real-time telecommunication network management: Extending event correlation with temporal constraints. In IM'95 [20], pp. 290–302.

[24] J. F. Jordaan and M. E. Paterok. Event correlation in heterogeneous networks using the OSI management framework. In IM'93 [19], pp. 683–695.

[25] S. Kätker and K. Geihs. A generic model for fault isolation in integrated management systems. *Journal of Network and Systems Management*, 5(2):109–130, 1997.

[26] S. Kätker and M. Paterok. Fault isolation and event correlation for integrated fault management. In *IM'97*, pp. 583–596, 1997.

[27] I. Katzela, A. T. Bouloutas, and S. B. Calo. Centralized vs distributed fault localization. In IM'95 [20], pp. 250–263.

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

[29] M. Klemettinen, H. Mannila, and H. Toivonen. Rule discovery in telecommunication alarm data. *Journal of Network and Systems Management*, 7(4):395–423, Dec. 1999.

[30] S. Kliger, S. Yemini, Y. Yemini, D. Ohsie, and S. Stolfo. A coding approach to event correlation. In IM'95 [20], pp. 266–277.

[31] L. Lewis. A case-based reasoning approach to the resolution of faults in communications networks. In IM'93 [19], pp. 671–681.

[32] G. Liu, A. K. Mok, and E. J. Yang. Composite events for network event correlation. In IM'99 [21], pp. 247–260.

[33] K.-W. E. Lor. A network diagnostic expert system for Acculink^{TM} multiplexers based on a general network diagnostic scheme. In IM'93 [19], pp. 659–669.

[34] *Proc. of IEEE Network Operation and Management Symposium*, 2000.

[35] Y. A. Nygate. Event correlation using rule and object based techniques. In IM'95 [20], pp. 278–289.

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

[37] S. H. Schwartz and D. Zager. Value-oriented network management. In NOMS'00 [34].

[38] 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.

[39] 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.

[40] Z. Wang. Model of network faults. In IM'89 [17], pp. 345–352.

[41] 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.