Available online at www.sciencedirect.com

SCIENCE DIRECT®

Science of Computer Programming 53 (2004) 165–194

Science of Computer Programming

www.elsevier.com/locate/scico

A survey of fault localization techniques in computer networks[☆]

Małgorzata Steinder[a,*], Adarshpal S. Sethi[b]

[a] *IBM T. J. Watson Research Center, Hawthorne, NY 10532, USA*

[b] *Computer and Information Sciences, University of Delaware, Newark, DE, USA*

Received 3 October 2003; accepted 7 January 2004

Available online 1 July 2004

Abstract

Fault localization, a central aspect of network fault management, is a process of deducing the exact source of a failure from a set of observed failure indications. It has been a focus of research activity since the advent of modern communication systems, which produced numerous fault localization techniques. However, as communication systems evolved becoming more complex and offering new capabilities, the requirements imposed on fault localization techniques have changed as well. It is fair to say that despite this research effort, fault localization in complex communication systems remains an open research problem. This paper discusses the challenges of fault localization in complex communication systems and presents an overview of solutions proposed in the course of the last ten years, while discussing their advantages and shortcomings. The survey is followed by the presentation of potential directions for future research in this area.

Keywords: Fault localization; Event correlation; Root cause analysis

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

[*] Corresponding author. Tel.: +1-914-7847014; fax: +1-914-7846807.
E-mail address: steinder@us.ibm.com (M. Steinder).

0167-6423/$ - see front matter © 2004 Elsevier B.V. All rights reserved.
doi:10.1016/j.scico.2004.01.010

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 the robustness, reliability, and accessibility of a system. In large and complex communication networks, automating fault diagnosis is critical.

Let us first introduce basic concepts in the field of fault management.

Event, defined as an exceptional condition occurring in the operation of hardware or software of a managed network, is a central concept pertaining to fault diagnosis [45,63,104].

Faults (also referred to as *problems* or *root causes*) constitute a class of network events that can cause other events but are not themselves caused by other events [45,63,104]. Faults may be classified according to their duration time as: (1) permanent, (2) intermittent, and (3) transient [99]. A permanent fault exists in a network until a repair action is taken. Intermittent faults occur on a discontinuous and periodic basis causing a 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 automatically repaired by error recovery procedures [99].

Error is defined as a discrepancy between a computed, observed, or measured value or condition and a true, specified, or theoretically correct value or condition [99]. Error is a consequence of a fault. Faults may or may not cause one or more errors. Errors may cause deviation of a delivered service from the specified service, which is visible to the outside world. The term *failure* is used to denote this type of an error. Errors do not need to be directly corrected, and in many cases they are not visible externally. However, an error in a network device or software may cause a malfunctioning of dependent network devices or software. Thus, errors may propagate within the network causing failures of faultless hardware or software [99].

Symptoms are external manifestations of failures [45]. They are observed as *alarms*—notifications of a potential failure [45,51,63,104]. These notifications may originate from management agents via management protocol messages (e.g., SNMP trap [10] and CMIP EVENT-REPORT [44]), from management systems that monitor the network status, e.g., using command *ping* [96], system log-files or character streams sent by external equipment [84].

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 a management system that destroy evidence of fault occurrence, or (3) the lack of management functionality necessary to provide indications of fault existence. Some faults may be partially-observable—the management system provides indications of a fault occurrence, but the indications are not sufficient to precisely locate the fault.

Let us illustrate the concepts described thus far with an example [104]. In Fig. 1, a simple communication network is presented in which a client accesses a remote database server. An interface of one of the routers between the client and server gets intermittently out of sync causing bursts of bit errors in transmitted IP datagrams. As a result, many IP

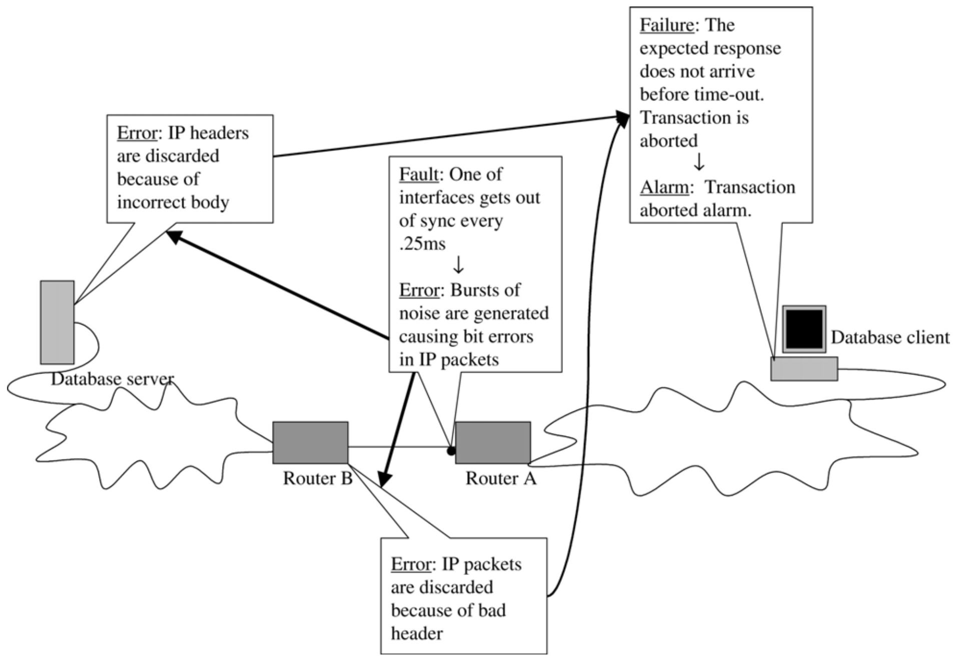

Failure: The expected response does not arrive before time-out. Transaction is aborted
↓
Alarm: Transaction aborted alarm.
Error: IP headers are discarded because of incorrect body
Fault: One of interfaces gets out of sync every .25ms
↓
Error: Bursts of noise are generated causing bit errors in IP packets
Database server
Database client
Router B
Router A
Error: IP packets are discarded because of bad header

Fig. 1. Distinction between fault, error, failure, and symptom.

datagrams passing through the router are rejected by the next router because of header errors, or by the server because of the corrupted datagram body. The client does not receive any response to its query and times out. This example illustrates how a seemingly invisible fault manifests itself through a failure at a location distant from the location of the fault.

Since most faults are not directly observable, the management system has to infer their existence from information provided by the received alarms. The information carried within reported alarms may include the following: the identity of the object that generated the alarm, type of failure condition, timestamp, alarm identifier, measure of severity of the failure condition, a textual description of the failure, etc. [41,84].

In a communication 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) generating multiple alarms by a device for a single fault, (4) detection of and issuing a notification about the same network fault by many devices simultaneously, and (5) error propagation to other network devices causing them to fail and, as a result, generate additional alarms [41]. It may be argued that typical networked systems provide plenty of information necessary to infer existence of faults [104].

The process of fault diagnosis usually involves three steps:

- Fault detection [4]—a process of capturing on-line indications of network disorder provided by malfunctioning devices in the form of alarms.

- Fault localization [4,5,53] (also referred to as fault isolation, alarm/event correlation, and root cause analysis)—a set of observed fault indications is analyzed to find an explanation of the alarms.
- Testing [5,53]—a process that, given a number of possible hypotheses, determines the actual faults.

This survey focuses on the second component of fault diagnosis—fault localization—as a process of deducing the exact source of the failure (a root cause) from the set of observed failure indications. The most popular fault localization technique is alarm correlation—a process of grouping alarms related by having the same root cause.

Fault localization is subject to complications resulting from complexity, unreliability, and non-determinism of communication systems. The following paragraphs present common problems that have to be addressed by a fault localization technique.

Fault evidence may be ambiguous, inconsistent, and incomplete [17,38,55].

Ambiguity in the observed set of alarms stems from the fact that the same alarm may be generated as an indication of many different faults. Inconsistency results from a disagreement among devices with regard to the facts related to network operation; one device may have perception that a component is operating correctly, while another may consider the component faulty [17]. Incompleteness is a consequence of alarm loss or delay [38]. It is essential that a fault management system be able to create a consistent view of network operation even in the presence of ambiguous, inconsistent, or incomplete information [17].

A fault management system should provide means to represent and interpret uncertain data within the system knowledge and fault evidence [15,17,38,58].

A set of alarms generated by a fault may depend on many factors such as dependencies among network devices, current configurations, services in use since fault occurrence, presence of other faults, values of other network parameters, etc. Due to this non-determinism the system knowledge may be subject to inaccuracy and inconsistency. Fault evidence may also be inaccurate because of spurious alarms, which are generated by transient problems or as a result of overly sensitive fault detection mechanisms. When spurious symptoms may occur, the management system may not be sure which observed alarms should be taken into account in the fault localization process.

An event management system should be able to isolate multiple simultaneous related or unrelated root causes [17].

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

A fault localization process should try to find the optimal solution according to some accepted optimality criteria [53].

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 this paper combine fault localization with testing to enable resolving 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 [17]. Therefore, some 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 each formulated hypothesis based on some measure of goodness [53]. This measure may be a probability that a hypothesis is valid, its information cost, etc. The optimality criteria may include the minimum size of the hypotheses set, the lowest cost solution, the lowest error probability, etc.

Fault localization process in large networks should be performed in a distributed fashion [5,54,104].

Communication networks become more and more advanced in terms of their size, complexity, speed, and the level of heterogeneity. Processing large volumes of information necessary to perform fault localization in such systems would be computationally prohibiting. It is also impractical to assume that the fault localization process has access to the information on the entire system. Many researchers [5,54,104] 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 the managers governs a subset of network hardware and/or software components within boundaries marked by protocol layers or network domains. Errors propagate horizontally—between peer associated devices within the same layer—and/or vertically—from upper layers to lower layers and vice versa between related services [99]. They may 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 [54,99]. Therefore, distributed fault localization schemes are necessary that would allow the management nodes to reach the solution collectively.

A fault localization process has to take into account temporal relationships among events [46,53,62].

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

In the past, numerous paradigms were proposed upon which fault localization techniques were based. These paradigms derive from different areas of computer science, including artificial intelligence, graph theory, neural networks, information theory, and automata theory. In Fig. 2, a classification of the existing solutions is presented. These solutions include techniques derived from the field of artificial intelligence (rule-, model-, and case-based reasoning tools as well as decision trees, and neural networks),

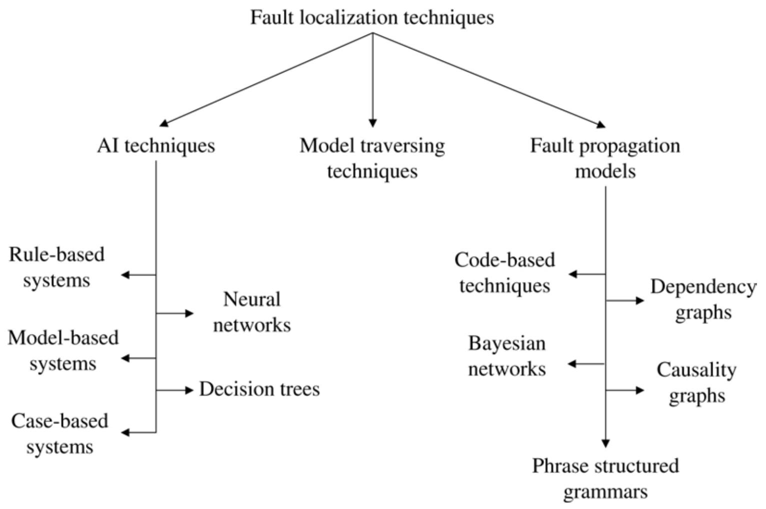


Fig. 2. Classification of fault localization techniques.

model-traversing techniques, graph-theoretic techniques, and the codebook approach. Model-, case-, and rule-based reasoning tools will be described in Section 2. Section 3 is an introduction to model traversing techniques. Section 4 discusses graph-theoretic techniques. Finally, Section 5 presents open research issues in the field of fault localization in communication systems.

2. Expert-system techniques for fault localization

The most widely used techniques in the field of fault localization and diagnosis are expert systems [74]. 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. Most expert systems use rule-based representation of their knowledge-base. In the domain of fault localization, the inference engine usually uses a forward-chaining inferencing mechanism, which executes in a sequence of rule-firing cycles. In each cycle the system chooses rules for execution, whose antecedents (conditions) match the content of the working memory.

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. The research on rule-based fault localization systems addresses the structure of the knowledge base and the design of the rule-definition language. Lor et al. [64] organize the system of rules by distinguishing between core and customized knowledge. The core knowledge may be understood as a generic or reusable knowledge. It is useful to identify an approximate location of a fault in a large network. Customized knowledge allows us to precisely isolate a fault from the selected group of system entities. In JECTOR [62] correlation rules are organized as *composite event* definitions. In this approach, the distinction is made between primitive events, i.e., alarms, and composite events, which are composed of primitive and other composite events, and include a set of conditions that have to be verified before a composite event can be asserted.

Rule-based systems, which rely solely on surface knowledge, do not require profound understanding of the underlying system architectural and operational principles and, for small systems, may provide a powerful tool for eliminating the least likely hypotheses [53]. However, rule-based systems possess a number of disadvantages that limit their usability for fault isolation in more complex systems. The downsides of rule-based systems include inability to learn from experience, inability to deal with unseen problems, and difficulty in updating the system knowledge [61]. 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 observation of statistical data [57], it is still necessary to regenerate the large portion of correlation rules when the system configuration changes. Rule-based systems are also inefficient and unable to deal with inaccurate information [42]. The lack of structure in the system of rules typically makes it very difficult to allow reusability of rules that seems so intuitive in hierarchically built distributed systems. Another problem is that rule-based systems 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 [102].

Given the difficulties of techniques that rely only on surface knowledge, expert systems used in the field of fault localization usually include some form of deep knowledge that represents relationships among system entities [24,28,48,97,103]. The system model may describe the system structure (static knowledge) and its functional behavior (dynamic knowledge) [38,97]. In model-based expert systems, conditions associated with the rules usually include predicates referring to the system model. The predicates test the existence of a relationship among system components. The model is usually defined using an object-oriented paradigm [16,22,38,45,97] and frequently has the form of a graph of dependencies among system components. A different model is proposed in SINERGIA [8], which represents structural knowledge as a set of network topology templates selected in such a way that any network topology may be expressed as instances of these templates. For each template all possible alarm patterns are listed along with fault diagnosis functions. ECXpert [72] uses *correlation tree skeletons* describing cause-effect relationships between network events. The root of the tree is always a symptom, the leaves are possible symptom causes. In the process of alarm correlation, *correlation tree instances* are created. An observed alarm may be added to an existing correlation tree instance or placed in a new tree instance. It may also cause combining a number of tree instances or splitting an instance into two or more trees.

Thanks to representing a deep knowledge of the network connectivity and operation, model-based approaches do not possess the disadvantages that characterize rule-based systems. They 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. The approach presented in [1] avoids maintaining an explicit network model by providing scenario templates organized on a hierarchically based network structure, which are instantiated with the data obtained from the arriving event attributes or from the configuration database. The scenarios communicate using internal composite events. The internal event publishers need not be aware which components consume the events that they forward; therefore, a change to a higher-level scenario does not require changes to any of the lower-level scenarios. One of the problems that the approach in [1] does not solve is dealing with complex network topologies. The solution shows how to propagate events between layers gradually increasing their level of abstraction. It does not however show how the reasoning should be performed within a layer if the network topology in this layer is complex.

Regardless of the type of knowledge used by expert systems, the fault localization process is driven by an inference engine according to event-correlation rules, which are usually defined using a high-level language. The design of an expressive yet simple language has been a goal of many publications on this subject. Rule conditions are frequently expressed as patterns, which test alarm frequency and origin as well as values of alarm attributes [1,102]. In some approaches, tests on temporal relationships among correlated events may be also defined. These tests may simply check whether the correlated alarms were received within a certain time-window [1,62] or be represented by more complex predicates encoding various tests on start, end, and duration times of correlated events [46]. Rule actions typically allow alarm filtering, replacement, suppression, escalation, clustering, and generalization [46]. They can also involve more complex repair or testing actions.

Case-based systems are a special class of expert systems that base their decisions on experience and past situations. They try to acquire relevant knowledge of past cases and previously used solutions to propose solutions for new problems [61]. They are well suited to learning correlation patterns [25]. When a problem is successfully solved, the solution (or its parts) may be used in dealing with subsequent problems. They are resilient to changes in network configuration. However, case-based systems require an application-specific model for the resolution process [102]. Also, time inefficiency may make them unusable in real-time alarm correlation [25].

In addition to the techniques already described in this section, decision trees [80], and neural networks [26,27,101] have been used for the purpose of fault localization. Neural networks, which are systems composed of interconnected nodes called neurons, try to mimic operation of a human brain. They are capable of learning [69] and resilient to noise or inconsistencies in the input data. The disadvantage of neural network systems is that they require long training periods [25,102], and that their behavior outside their area of training is difficult to predict [102].

Decision trees are used as a representation of an expert knowledge to guide a user observing symptoms of failure toward locating the root cause of the problem [80]. They allow a simple and expressive representation of the expert knowledge. However, their applicability is limited by the dependence on specific applications and the degraded accuracy in the presence of noise [82].

3. Model traversing techniques

Model traversing techniques [32,41,47,50–52] use formal representation of a communication system with clearly marked relationships among 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 reported in the literature use object-oriented representation of the system [41]. One approach [47] exploits the OSI management framework. The approach described in [50] uses Guidelines for the Definition of Managed Objects (GDMO) [43] (with non-standard extensions) to model services and dependencies between services in a distributed system. The proposed refinements of this model include the possibility of testing operational and quality of service states of managed services [51].

Event correlation in model-traversal techniques is usually event-driven. For every observed event, which is typically mapped into a model node, the model is searched recursively following the relationship links between managed objects. The actual fault localization algorithms may be quite different. Jordan et al. [47] consider all observed events as singleton equivalence classes. In the process of fault localization classes are merged whenever the events they include are traced down to the same managed object. To facilitate the search certain event properties can be explored, e.g., (1) level at which event occurred—primary failure is the most likely at the lowest level, (2) event type—some events are more likely to indicate primary failures than others, (3) severity of an event—more severe events are likely to have caused other events in the equivalence class, and (4) the origin of an event, which may provide explicit information if the event indicates primary or secondary fault. Other techniques [51] combine fault localization with testing. During the model traversal managed objects are tested to determine their operational status. The root cause is found when the currently explored malfunctioning object does not depend on any other malfunctioning object. In multi-layer models, first a horizontal search is performed in the layer in which failure has been reported [52]. When a failing component is located, a vertical search carries the fault localization process to the next lower layer. In the lower layer the horizontal search is started again. In NetFACT [41], the fault localization process is performed in two phases. First, in horizontal search, votes are assigned to potentially faulty elements based on the number of symptoms pertaining to these elements. The root cause is determined in the second phase—tree search, which determines if the device that received the most votes in the first step was at fault or if it failed because one of the components it depends upon was faulty.

Model traversing techniques are robust against frequent network configuration changes [52]. 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. These models naturally enable design of distributed fault localization algorithms. However, they are inflexible in modeling fault propagation patterns. In particular, they are unable to model situations in which failure of a device may depend on a logical combination of other device failures [41].

4. Graph-theoretic techniques

Graph-theoretic techniques rely on a graphical model of the system, called a fault propagation model (FPM), which describes which symptoms may be observed if a specific

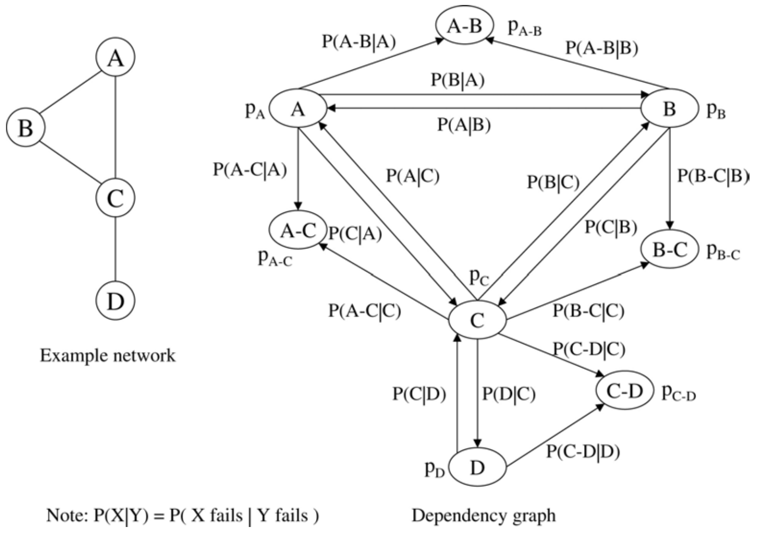


Fig. 3. Simple network and a corresponding dependency graph.

fault occurs [52]. The FPM includes the representation of all faults and symptoms that occur in the system. Observed symptoms are mapped into FPM nodes. The fault localization algorithm analyzes the FPM to identify the best explanation of the observed symptoms.

Graph-theoretic techniques require a priori specification of how a failure condition or alarm in one component is related to failure conditions or alarms in other components [41]. To create such a model, an accurate knowledge of current dependencies among abstract and physical system components is required. The efficiency and accuracy of the fault localization algorithm are dependent on the accuracy of this a priori specification.

Fault propagation models take the form of causality or dependency graphs. 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 $e_i \rightarrow e_j$ [33]. Nodes of a causality graph may be marked as problems or symptoms. Some nodes are neither problems nor symptoms, while others may be marked as problems and symptoms at the same time. Causality graph edges may be labeled with a probability of the causal implication. Similarly, it is possible to assign a probability of independent occurrence to all nodes labeled as problems.

A *dependency graph* is a directed graph $G = (O, D)$, where O is a finite, non-empty set of objects and D is a set of edges between the objects. With each object a probability of its failure independent of other objects may be associated. 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. Every directed edge is labeled with a conditional probability that the object at the end of an edge fails, provided that the object at the beginning of an edge fails [53,55]. A dependency graph for an example network is presented in Fig. 3. Note that the presented dependency graph models all possible dependencies between managed objects in the example network. In reality the graph could be reduced based on information on currently open connections.

Many approaches using dependency graph models assume that an object may fail in only one way. If this is the case in a real system, then a failure of an object may be represented as an event. In this case, the two representations, causality graph and dependency graph, are equivalent. When multiple failures may be associated with a single object, they can typically be enumerated into a small set of failure modes, such as *complete failure*, *abnormal transmission delay*, *high packet loss rate*, etc. [92]. The dependency graph then associates multiple failure modes with each object, whereas dependency edges between objects are weighted with probability matrices rather than with single probability values, where each matrix cell indicates the probability with which a particular failure of an antecedent object causes a particular failure of a dependent object [92]. The dependency graph may still be mapped into a causality graph by creating a separate causality graph node for each object and each of its failure modes [92], and then connecting the nodes accordingly.

While there are clear benefits of using a dependency graph as a system model (e.g., it is more natural and easier to build), causality graphs are better suited to the task of fault localization as they provide it with a more detailed view of the system and allow it to deal with a simple notion of an event rather than with potentially multi-state system objects.

Most graph-theoretic techniques reported in the literature allow the FPM to be non-deterministic by modeling prior and conditional failure probabilities. However, many of these techniques require the probability models to be restricted to canonical models such as OR and AND models [55,58,92,93]. An OR model combines possible causes of an event using logical operator OR, meaning that at least one of the possible causes has to exist for the considered event to occur. An AND model uses logical operator AND, instead. Some techniques may be extended to work with hybrid models that allow the relationship between an event and its possible causes to have the form of an arbitrary logical expression [93,94].

A fault localization algorithm, based on the provided FPM, should return a number of fault hypotheses that best explain the set of observed symptoms. It has been shown that this problem is NP-hard, in general [4,53,55].

Given the complexity of the problem, the fault localization techniques proposed in the literature seldom allow arbitrary probabilistic FPMs. Frequently the shape of the model is restricted, e.g., to a bipartite graph [58,93], or the model is deterministic [33,104]. The problem may be also simplified by assuming that only one fault exists in the system at a time [58] or by restricting the number of simultaneous faults to a certain number [4]. Most techniques assume that the observation of system state is accurate and therefore do not attempt to address the problem of lost and spurious symptoms [33,55].

In the following sections, we present some graph-theoretic techniques described in the literature. They may be divided into those that are tailored toward the isolation of dependent faults [55], and those that assume that faults are independent of one another [94]. They also adopt different measures of hypothesis goodness: a minimum number of faults [33] or a probabilistic measure of confidence [55,93,94]. Symptom processing is usually window-based, i.e., an algorithm works with a group of symptoms observed over a certain time-window [55,58]. A more flexible approach adopts event-driven processing, which allows symptoms to be analyzed when they arrive thereby reducing the algorithm's latency, allowing fault localization to be interleaved with testing, and increasing its resilience to the changes of the FPM [33,93,94].

4.1. Divide and conquer algorithm

The *divide and conquer* algorithm [55] uses a dependency graph as an FPM assuming that one failure type is allowed per object. It is a window-based technique tailored toward identifying dependent faults. The algorithm first identifies the alarm domain, a set of all faults that may have caused the observed alarms. In the first phase, the alarm cluster domain is partitioned into two subsets characterized by the 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. Intuitively, a maximum mutual dependency set groups together all objects that are the most dependent on one another. In the second phase, the algorithm decides which of the subsets should be used to recursively invoke the fault localization procedure. If the subset with higher probability that one of its members was a primary source of a failure is able to explain all alarms, the recursive procedure is invoked using the entire alarm cluster passed to this recursion step and the higher-probability subset as parameters. Otherwise, the alarm cluster domain is divided into two subclusters. The first subcluster contains all alarms that may be explained using the subset with the higher probability that one of its members was a primary source of a failure. The second subcluster contains the remaining alarms from the original alarm cluster. Its domain is the other of the two subsets. The recursive procedure is then invoked twice, taking the two subclusters and their respective domains as parameters. The recursion continues until the input alarm cluster domain is a singleton. If this is the case, the element of the singleton set is added to the final solution. To avoid the computational cost of calculating the mutual dependency subsets in every recursive invocation of the algorithm, the mutual dependency subsets are precomputed at the beginning using the bottom–up technique. While building maximum mutual dependency sets, the procedure has to determine the probability that at least one member of a subset is faulty. For subset S this probability is approximated using the following formula [55]:

$$P(S) = \sum_{o_i \in S} \left(P(o_i) + \sum_{o_j \in S, j \neq i} P(o_j) P(o_i \mid o_j) \right).$$

Let us illustrate the divide and conquer algorithm with an example. Fig. 4(a) shows an example dependency graph. Suppose that alarms A_1, A_2 and A_3 have been generated indicating failures in vertices $A - B$, $B - C$, and $C - D$, respectively. The alarm cluster domain is presented in Fig. 4(b), and the steps of the algorithm are depicted in Fig. 4(c). In the presented scenario, the algorithm associates fault at device C with alarms A_2 and A_3, and fault at device B with alarm A_1.

The divide and conquer algorithm always explains all the observed alarms, but may fail to give their best explanation [55]. The most accurate solution will be found in a system whose objects have the same probability of independent failure. The algorithm does not

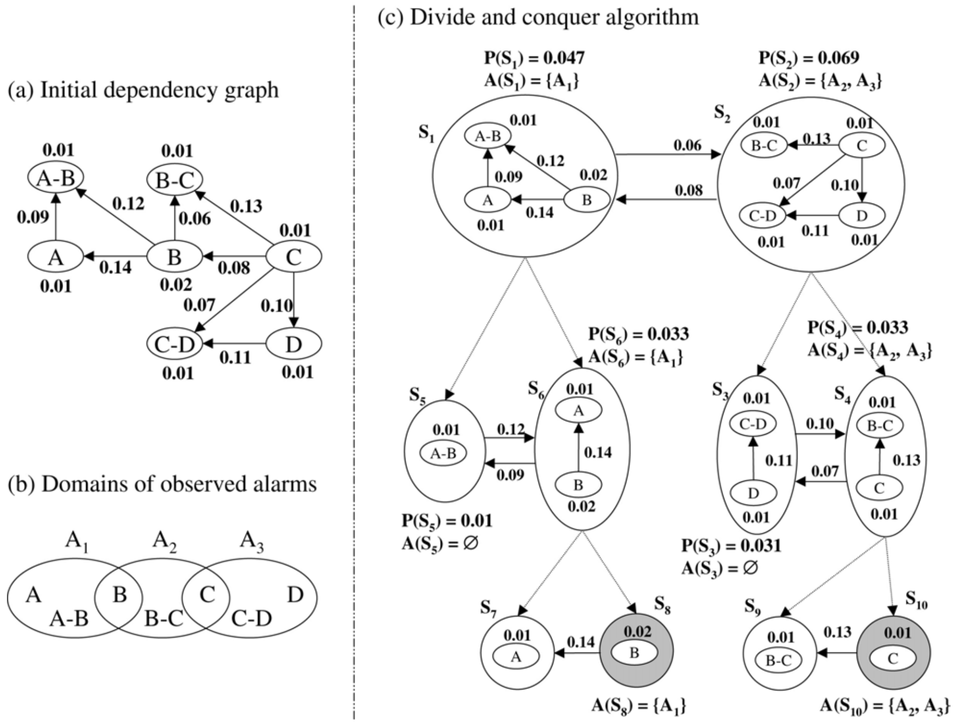


Fig. 4. Divide and conquer algorithm.

handle lost or spurious symptoms. The computational complexity is $\mathcal{O}(N^3)$, where N is the number of the dependency graph nodes.

The divide and conquer technique described above may be used only if there are failure dependencies among objects. For the case of a system in which all objects fail independently of one another, a different algorithm was proposed [55]. This procedure involves partitioning the initial set of objects that may have caused the observed alarm cluster. The partitioning is performed around a randomly chosen element in such a way that one subset contains all the objects whose failure probability is less than or equal to the failure probability of the randomly chosen element, and the other contains all objects whose failure probability is greater than the failure probability of the randomly chosen element. The second phase is very similar to the second phase of the divide and conquer algorithm. The algorithm has a polynomial worst case complexity [55].

4.2. 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 [4]. In this model, terminals correspond to the indivisible network components—terminal objects. Productions are used to build compound network objects

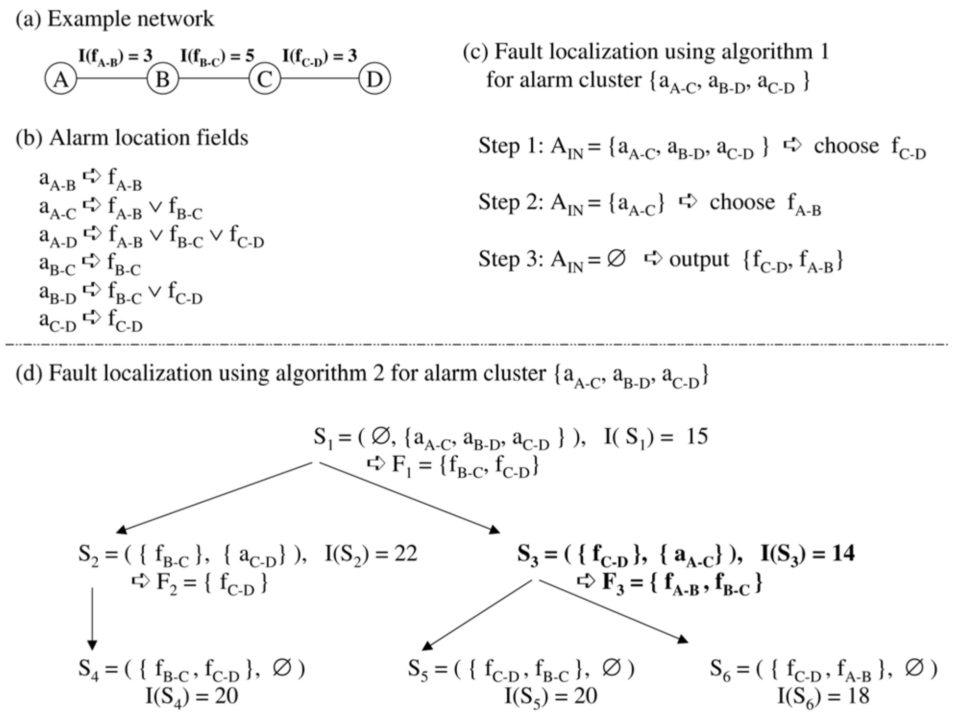

(a) Example network
I(f_A-B) = 3
I(f_B-C) = 5
I(f_C-D) = 3
A
B
C
D
(b) Alarm location fields
a_A-B ⇨ f_A-B
a_A-C ⇨ f_A-B ∨ f_B-C
a_A-D ⇨ f_A-B ∨ f_B-C ∨ f_C-D
a_B-C ⇨ f_B-C
a_B-D ⇨ f_B-C ∨ f_C-D
a_C-D ⇨ f_C-D
(c) Fault localization using algorithm 1 for alarm cluster {a_A-C, a_B-D, a_C-D }
Step 1: A_IN = {a_A-C, a_B-D, a_C-D } ⇨ choose f_C-D
Step 2: A_IN = {a_A-C} ⇨ choose f_A-B
Step 3: A_IN = ∅ ⇨ output {f_C-D, f_A-B}
(d) Fault localization using algorithm 2 for alarm cluster {a_A-C, a_B-D, a_C-D}
S_1 = (∅, {a_A-C, a_B-D, a_C-D }), I(S_1) = 15
⇨ F_1 = {f_B-C, f_C-D}
S_2 = ({ f_B-C }, { a_C-D}), I(S_2) = 22
⇨ F_2 = { f_C-D }
S_3 = ({ f_C-D }, { a_A-C}), I(S_3) = 14
⇨ F_3 = { f_A-B , f_B-C }
S_4 = ({ f_B-C , f_C-D }, ∅)
I(S_4) = 20
S_5 = ({ f_C-D , f_B-C }, ∅)
I(S_5) = 20
S_6 = ({ f_C-D , f_A-B }, ∅)
I(S_6) = 18

Fig. 5. Fault localization using context-free grammar algorithms.

from the already defined objects. To illustrate how context-free grammars may be used to represent dependencies between network objects let us present an example. A path P_{A-D} between source A and destination D in a network presented in Fig. 3 is modeled by the following productions:

$$P_{A-D} \rightarrow P_{A-C}.L_{C-D}$$
$$P_{A-C} \rightarrow L_{A-B}.L_{B-C}|L_{A-C}.$$

A context-free grammar allows systems with complex dependencies between managed objects to be modeled [4]. In particular, one can easily represent a situation in which a component depends on an arbitrary logical combination of other components. A simple example of such a situation was presented above. A context-free grammar may be considered an extension of dependency graph.

This section presents two fault localization algorithms that use context-free grammar to represent fault propagation patterns [4]. To illustrate them, the example of a tandem network presented in Fig. 5(a) will be used. In this network three faults are possible: f_{A-B}, f_{B-C}, and f_{C-D}, associated with links A–B, B–C, and C–D, respectively. With these faults we associate information cost as presented in Fig. 5(a). The following alarms are possible: a_{A-B}, a_{A-C}, a_{A-D}, a_{B-C}, a_{B-D}, and a_{C-D}. Alarm a_{X-Y} may, e.g., indicate failure of a connection established between hosts X and Y. Domains of alarms considered in the presented example, called by the technique fault localization fields, are presented in Fig. 5(b).

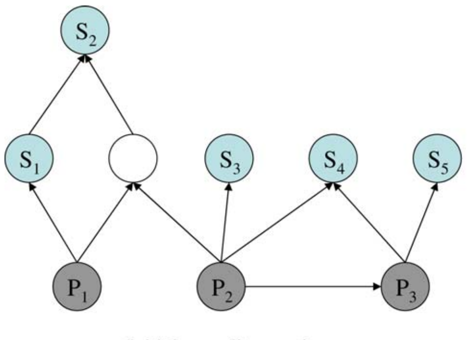
The first algorithm presented in [4] chooses the minimum set of faults that explains all observed alarms. First, the algorithm identifies a fault that belongs to the greatest number of alarm domains. If there are more than one such faults, the one with the highest probability of occurrence (or lowest information cost) is chosen and placed in the alarm explanation fault set. For example, if a set of observed alarms is $\{a_{A-C}, a_{B-D}, a_{C-D}\}$ (see Fig. 5(c)), there are two faults belonging to the greatest number of alarm domains, f_{B-C} and f_{C-D}; fault f_{C-D} is chosen because its information cost is the lowest. Next, all alarms explained by the chosen fault are removed from the set of observed alarms and the algorithm continues with the remaining alarms until all of them are explained. Thus, in the first step of the example presented in Fig. 5(c), alarms $\{a_{B-D}, a_{C-D}\}$ are removed from the set of unexplained alarms and in the second step the set of alarms $\{a_{A-C}\}$ is explained using fault f_{A-B}.

The second algorithm proposed in [4] takes into account lost and spurious symptoms. Suppose that $\mathcal{A}_f$ is a set of alarms that should be generated if fault f occurs. We can use f to explain the set of observed alarms $\mathcal{A}_o$ only if $\mathcal{A}_o = (\mathcal{A}_f - \mathcal{A}_l) \cup \mathcal{A}_s$, where $\mathcal{A}_l \subseteq \mathcal{A}_f$ is a set of alarms that were lost and $\mathcal{A}_s$ is a set of spurious alarms. Whenever alarm loss or spurious alarm generation is used as a part of an explanation, the information cost assigned to this explanation is increased. The algorithm tries to find such a subset of faults that explains all alarms with the minimal explanation cost. In the first phase of the algorithm, a tree is created that represents all possible solutions. Every node S_i of the tree is labeled $(\mathcal{F}_i, \mathcal{A}_i)$, where $\mathcal{F}_i$ is a set of faults identified so far, and $\mathcal{A}_i$ is a set of observed alarms that remain to be explained. In the root node $\mathcal{F}_1 = \emptyset$ and $\mathcal{A}_1 = \mathcal{A}_o$, where $\mathcal{A}_o$ is a set of all observed alarms. The recursive procedure of building such a tree descends from the root creating children until the leaves—nodes with empty unexplained alarm list—are reached. At every node S_i, the procedure finds a subset of faults $\mathcal{F}_k$ that explains the most alarms in $\mathcal{A}_i$. Let $\mathcal{A}_{k_j} \subseteq \mathcal{A}_k$ be the set of alarms explained by fault $f_j \in \mathcal{F}_k$. For every fault $f_j \in \mathcal{F}_k$, a new child of the node $(\mathcal{F}_i, \mathcal{A}_i)$ is created labeled with $(\mathcal{F}_i \cup \{f_j\}, \mathcal{A}_i - \mathcal{A}_{k_j})$. This procedure is illustrated in Fig. 5(d).

The cost associated with every node S_i of the tree is computed as follows [4]: $I(S_i) = I(\mathcal{F}_i) + I(EXPECTED_i) + I(ADD_i) + I(DELETE_i)$, where $I(\mathcal{F}_i)$, $I(EXPECTED_i)$, $I(ADD_i)$, and $I(DELETE_i)$ denote the cost of generating all faults in $\mathcal{F}_i$, the cost of generating all observed alarms that were expected for faults in $\mathcal{F}_i$, the cost of generating all spurious alarms, and the cost of losing alarms that are expected for faults in $\mathcal{F}_i$, but do not belong to $\mathcal{A}_i$, respectively. In Fig. 5(d), $I(EXPECTED_i) = 0$, $I(ADD_i) = 5$, and $I(DELETE_i) = 6$, for all alarms and faults.

In the second phase of the algorithm, a minimum cost node of the entire tree is chosen as the best solution. Since the algorithm chooses any node (not necessarily a leaf), some alarms may be left unexplained. These alarms are assumed to be spurious. In the example in Fig. 5(d), node S_3 is chosen because its cost is minimum. Note that, in this solution one alarm is assumed to have been lost and one alarm is assumed to be spurious.

The presented algorithm is rather complex and, as the authors point out, should be considered a guideline for designing fault localization algorithms rather than a practical solution.



Initial causality graph

	P_1	P_2	P_3
S_2	1	1	0
S_3	0	1	0
S_4	0	1	1
S_5	0	1	1

Correlation matrix that allows us to detect one lost or spurious symptom (min. Hamming distance = 2)

Fig. 6. Correlation matrix derived from an example causality graph.

Fault localization problem using context-free grammar as a fault propagation model has also been addressed in [53]. The problem of locating the source of failure for a limited class of context-free grammar models, semantically equivalent to dependency graphs, is transformed into a zero–one integer linear programming problem. Based on available solutions to the integer linear programming problem, the optimal (but complex) and suboptimal algorithms for solving the zero–one subproblem are proposed. The fault localization problem using an unrestricted context-free grammar model is formulated as an integer non-linear programming problem. The algorithm is proposed for converting this integer non-linear programming problem into an integer linear programming problem.

4.3. Codebook technique

Fault propagation patterns in the codebook technique are represented by a matrix of problem codes that distinguish problems from one another [58,104]. In the deterministic technique, a code is a sequence of values from $\{0, 1\}$. The value of 1 at the ith position of a code generated for problem p_j indicates cause-effect implication between problem p_j and symptom s_i. In the probabilistic model, codes contain values $\in [0, 1]$ that represent likelihood of the cause-effect implication between a given problem and a given symptom. The codebook is basically a matrix representation of a bipartite causality graph that has been optimized to minimize the number of symptoms that have to be analyzed while still ensuring that the symptom patterns corresponding to problems allow us to distinguish between the problems. A correlation matrix derived from an example initial causality graph is presented in Fig. 6.

Conceptually, the codebook approach considers event propagation as data transmission over a discrete memoryless lossy channel, whose input alphabet is a set of optimal codes and output alphabet is a set of all possible symptom combinations [81]. With such an interpretation, event correlation is equivalent to decoding a received output symbol to one of the valid input symbols. The received output symbol in the event correlation problem 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. Spurious and lost symptoms

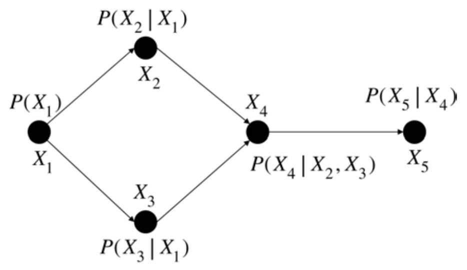


Fig. 7. Simple Bayesian network representing distribution $P(X_1)$, $P(X_2|X_1)$, $P(X_3|X_1)$, $P(X_4|X_2, X_3)$, and $P(X_5|X_4)$.

correspond to channel errors. The number of errors that may be detected or corrected depends on the codebook and the decoding scheme.

The codebook approach uses minimum symbol distance as a decision making scheme. For the deterministic model Hamming distance is used [81]. In the probabilistic model, $d(a, b)$—the distance between two probabilities $a, b \in [0, 1]$—is computed as $\log(a/b)$ with $\log(0/0) = 0$ and $\log(a/0) = a$. This log-likelihood measure generalizes Hamming distance used in the deterministic model [58]. Since the coding phase is performed only once, the codebook 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 [58]. On the other hand, the accuracy of the codebook technique is hard to predict when more than one faults occur with overlapping sets of symptoms. In addition, since each system configuration change requires regenerating the codebook, which is a time consuming process, the technique is not suitable for environments with dynamically changing dependencies.

4.4. Belief-network approach

Belief network is a DAG whose nodes represent random variables, the edges denote existence of direct causal influences between the linked variables and the strengths of these influences are expressed by forward conditional probabilities (Fig. 7). In the context of fault diagnosis, the random variables represent states of network objects or the occurrence of network events.

The fault localization problem may be formulated as a problem of calculating the most probable explanation of the observer evidence in a belief network used as an FPM. Although the problem is known to be NP-hard in general [13], a polynomial time algorithm was proposed for singly-connected belief networks [75]. The polynomial algorithm [75] utilizes a message passing schema in which BN nodes exchange messages that encode certain conditional probabilities. The message that node X sends to its parent V_j for every valid V_j's value v_j is denoted by $\lambda_X(v_j)$. The message that node X sends to its child U_i for every valid value of X, x, is denoted by $\pi_{U_i}(x)$. Messages $\lambda_X(v_j)$ and $\pi_{U_i}(x)$ are calculated by node X based on messages it receives from its neighbors using the following equations (where β is any constant and α is a normalizing constant).

$$\lambda_X(v_j) = \beta\left(\lambda(1) - (1 - p(X|V_j))^{v_j}(\lambda(1) - \lambda(0))\prod_{k\neq j}(1 - p(X|V_k)\pi_{V_k}(1))\right) \quad (1)$$

$$\pi_{U_i}(x) = \alpha\prod_{k\neq i}\lambda_{U_k}(x)\pi(x) \quad (2)$$

$$\lambda(x) = \prod_{i=1}^{n}\lambda_{U_i}(x) \quad (3)$$

$$\pi(x) = \begin{cases}\alpha\prod_{j=1}^{m}(1 - p(X|V_j))\pi_{V_j}(1) & \text{if } x = 0,\\ \alpha(1 - \prod_{j=1}^{m}(1 - p(X|V_j))\pi_{Vj}(1)) & \text{if } x = 1.\end{cases} \quad (4)$$

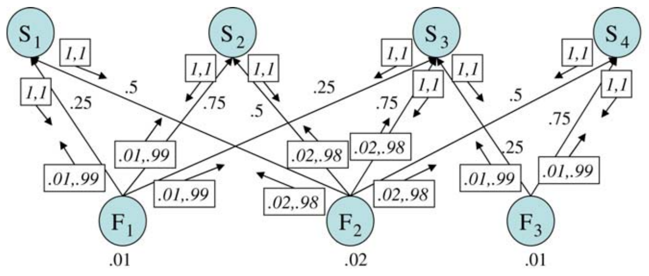
Based on messages received from its parents and children, node X computes $bel(x) = \alpha\lambda(x)\pi(x)$, the probability that the event represented by node X exists ($x = 1$) or does not exist ($x = 0$).

In [94] this algorithm was adapted as an approximation scheme for performing fault localization in FPMs of arbitrary shape. The adapted algorithm proceeds in an event-driven manner, after every symptom observation applying one iteration of message passing traversing the graph according to some order. For every symptom a different ordering is used that is equivalent to the breadth-first order started from the node representing the observed symptom. The graph exploration ends when an unobserved symptom node is visited. In Fig. 8, a simple belief network is shown that encodes causal relationships between faults F_1, F_2, and F_3, and symptoms S_1, S_2, S_3, and S_4. As depicted in Fig. 8(a), initially, no symptoms are observed, and therefore messages $\lambda_X(v_j)$ sent by node X to node V_j are equal to 1, for all X and V_j and for both $v_j = 0$ and $v_j = 1$. Consequently, messages $\pi_{U_i}(x)$ sent by node X to node U_i are equal to $p(x)$, where $p(x)$ is a prior probability associated with X.

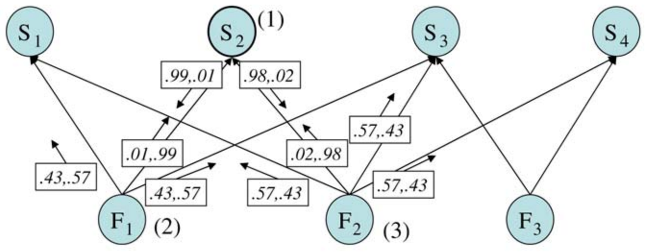
Parts (b), and (c) show message propagation in this network triggered by the observation of symptoms S_2 and S_4, respectively.

When all symptoms are analyzed in this manner, the algorithm produces, for every fault, the probability of its existence given the observed symptoms. The final hypothesis is chosen using the following heuristic: (1) a most-likely fault is chosen and placed in the final hypothesis, (2) the chosen fault node is considered a part of evidence, and (4) one iteration of message passing starting from the chosen fault node is performed. Steps (1)–(3) are repeated as long as (1) the posterior distribution contains fault nodes whose probability is greater than 0.5, and (2) unexplained negative symptoms remain. An inherent property of the adapted algorithm is the capability to isolate multiple simultaneous faults even if their symptoms overlap.

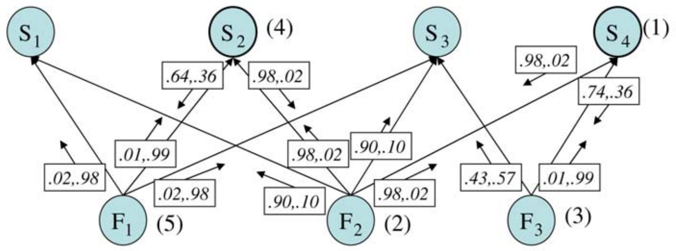
Many other fault localization techniques based on belief-network theory were reported in the literature. However, they are limited to rather specific fault diagnosis problems, which use simplified belief network models. In fault diagnosis in linear light-wave networks [15] and in diagnosing connectivity problems in communication systems [100] conditional probabilities are 0,1-valued. Bayesian reasoning used by Smyth [90] to monitor and diagnose the state of an antenna is applicable to the management of a system with a small number of possible states. As a result, it cannot be used in a system with a big number of (possibly multi-state) components. To trouble-shoot printing services a



(a) Fault propagation model and initial values λ of π and messages.



(b) Message-passing traversal after observation of symptom S_2; (1), (2), (3) denotes the order in which the nodes are visited. Note that messages sent by S_2 in this phase are calculated based on messages sent by its parents F_1 and F_2 in the previous phase (Fig. (a)). Messages sent by F_1 in this phase are calculated based on messages calculated by S_2 in this phase and those calculated by S_1 and S_3 in the previous phase.



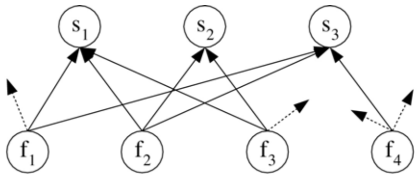
(c) Graph traversal after symptom S_4 is observed.

Fig. 8. Example execution of message-passing algorithm.

tree-shaped belief network is used [35]. Hood et al. [40] reported an application of Bayesian network theory to proactive fault detection. The belief network used there is also tree-shaped based on the structure of SNMP [10] MIBs [67].

4.5. Bipartite causality graphs

A bipartite causality graph is a special form of a fault propagation model that encodes direct causal relationships among faults and symptoms. Although relationships between faults and symptoms in real-life systems are usually more complex than may be represented by a bipartite graph (in particular, they are frequently indirect), many fault localization



$$\mathcal{H}_0 = \emptyset$$
$$s_1 : \mu(f_1) = \mu(f_2) = \mu(f_3) = \mu(f_4) = \infty \quad \rightarrow \mathcal{H}_1 = \{\{f_1\}, \{f_2\}, \{f_3\}\}$$
$$s_3 : \mu(f_1) = \mu(f_2) = \mu(f_3) = 1, \mu(f_4) = \infty \rightarrow \mathcal{H}_2 = \{\{f_1\}, \{f_2\}, \{f_3, f_4\}\}$$
$$s_2 : \mu(f_1) = \mu(f_2) = 1, \mu(f_3) = \mu(f_4) = 2 \quad \rightarrow \mathcal{H}_3 = \{\{f_1, f_3\}, \{f_2\}, \{f_3, f_4\}\}$$

Fig. 9. Incremental hypothesis updating after observation of symptoms s_1, s_3, and s_2.

techniques proposed in the literature [11,55,92,104] use bipartite FPMs. The focus on this type of model is justified by the following arguments: (1) Performing fault localization with more complex representations is difficult. (In general, the problem is NP-hard [55].) To avoid this complexity, more detailed models are frequently reduced to bipartite ones through a sequence of graph reduction operations [104]. (2) Building more complex models requires a profound knowledge of the underlying system, while symptom-fault maps may be obtained through external observation. In many real-life problems, only bipartite symptom-fault models are feasible [11]. (3) Some fault localization subproblems may be accurately represented by bipartite symptom-fault maps [92], thereby necessitating fault localization algorithms suitable for bipartite FPMs.

One of the techniques tailored toward a bipartite FPM is *incremental hypothesis updating* [93] (IHU), which works with a set of hypotheses, each of which is a complete explanation of the observed symptoms. The hypotheses are ranked using a belief metric, b. The algorithm proceeds in an event-driven and incremental fashion. The execution triggered by the ith symptom, s_i, creates a set of hypotheses, $\mathcal{H}_i$, each explaining symptoms s_1 through s_i. Set $\mathcal{H}_i$ is created by updating $\mathcal{H}_{i-1}$ with an explanation of symptom s_i. After the ith symptom is processed, belief metric b_i associated with hypothesis $h_j \in \mathcal{H}_i$ represents the probability that (1) all faults belonging to h_j have occurred, and (2) h_j explains every observed symptom $s_k \in \mathcal{S}_{O,i} = \{s_1, \ldots, s_i\}$. To incorporate an explanation of symptom s_i into the set of fault hypotheses, in the ith iteration of the algorithm, we analyze each $h_j \in \mathcal{H}_{i-1}$. If h_j is able to explain symptom s_i, we put h_j into $\mathcal{H}_i$. Otherwise, h_j has to be extended by adding to it a fault from H_{s_i}, where H_{s_i} is a domain of symptom s_i. To avoid a very fast growth in the size of $\mathcal{H}_i$, the following heuristic is used. 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 $\mu(f_l)$, the minimum size of a hypothesis in $\mathcal{H}_{i-1}$ that contains f_l and explains s_i. The usage of this heuristic is derived from the fact that the probability of multiple simultaneous faults is small. While updating the set of hypothesis, $b_i(h_j)$ is calculated iteratively based on $b_{i-1}(h_j)$. In Fig. 9, the algorithm is illustrated using a simple example.

Incremental hypothesis updating creates a set of the most likely hypotheses, which may all be presented to the system administrator. Rather than wait for a specific period of time before presenting a solution, the technique makes all these hypotheses available on a continuous basis, and constantly upgrades them with information learned from arriving symptoms. Thus, the algorithm is incremental. This feature allows the system administrator to initiate recovery actions sooner, and it allows additional testing procedures to be performed. The fact that multiple alternative hypotheses are available makes it easier to replace a solution when the most probable one proves to be incorrect.

A bipartite belief network is also used in [11] to pinpoint LAN segments suspected of having a particular fault. The reasoning mechanism used in [11] is not able to precisely identify the fault (e.g., the malfunctioning node). Moreover, it is based on observations that are particular to the LAN environment and are not applicable in general. Fecko et al. [20] use statistical methods to identify root causes of symptoms in bipartite FPMs.

5. Open research problems

Although fault localization has been an active research area for many years, many problems remain unsolved. In this section we describe some of them.

5.1. Multi-layer fault localization

In modern services such as e-commerce, telecommuting, virtual private networks [85], application service provisioning [12], grid services [23], etc., fault localization techniques capable of isolating faults in application and service layers are needed. Since upper layers depend on lower layers, it is not uncommon for a minor low-level problem to heavily affect a service delivered to a user. Therefore, a fault management system has to integrate fault diagnosis across multiple protocol layers. This goal is difficult to achieve since faults, as they propagate between system layers, change their semantics, and thus a correlation between an original fault and an observed system disorder is not obvious. In addition, multi-layer diagnosis is hampered by the difficulty of building a system model and the complexity involved in its analysis.

Despite some work on this problem reported in the literature [1,92], multi-layer fault localization remains a challenging and open research area. So far, the research has focused on defining multi-layer fault propagation models that clearly identify services provided by neighboring network layers [29,73,92]. Appleby et al. [1] propose a rule-based solution that allows events from multiple layers to be correlated by defining correlation scenarios that describe system behavior in various layers and communicate using internal composite events. As events propagate up the scenario hierarchy, their semantics changes gradually increasing the level of abstraction.

It is clear that none of the fault localization techniques surveyed in this paper is adequate in multi-layer environments. In fact, we believe that a comprehensive solution to the problem of multi-layer fault localization may require an application of several different techniques. For example, the rule-based approach may be the most suitable for symptom filtering. Model-based expert systems might facilitate correlation of alarms originating from neighboring network devices or layers. Graph theoretic non-deterministic approaches are needed in problems which are complex due to big network topology or convoluted system structure, e.g., to diagnose end-to-end problems in a network with complex topology [94]. Combining these different paradigms into a coherent multi-layer solution remains an open problem.

5.2. *Temporal correlation among events*

Many researchers have concluded that temporal relationships between events provide valuable information that could be used in the alarm correlation process [33,46,53,62]. However, few publications on fault localization address this issue.

Temporal correlation recognizes that alarms caused by the same fault are likely to be observed within a short time after the fault occurrence. The simplest method of temporal correlation relies on time-windows [1,46,55,104]: only alarms occurring within a time-window may be correlated. A more elaborate set of temporal correlation predicates has been proposed in [46,66]. These approaches are deterministic, which limits their applicability.

A natural approach to achieving probabilistic fault localization with temporal correlation is to use dynamic belief networks [83] (DBNs). A DBN can be built from a belief network by creating a set of random variables $\{V_{i,t}, V_{i,t+1}, \ldots\}$ for every random variable V_i in the original belief network. Random variable $V_{i,t}$ represents random variable V_i at time t. If, in the original belief network, V_i may be caused by V_j, in a DBN $V_{i,t}$ may be caused by $V_{j,t}, V_{j,t-1}, V_{j,t-2}, \ldots$ and $V_{i,t-1}, V_{i,t-2}, \ldots$. The resultant DBN consists of a number of interconnected copies of the original belief network. It is likely to be much bigger and more convoluted than the original belief network, and therefore more efficient fault localization techniques are needed to use DBNs in practice.

5.3. *Distributed fault localization techniques*

In large and complex communication networks large volumes of information have to be processed by the management system. As many researchers and practitioners agree, management of these large networks has to be performed in a distributed fashion. Distributed management provides natural partitioning of management information and function among management system components.

Distributed management seems to be well suited for most networks, which are divided into hierarchically organized domains or layers. A distributed management system may be designed to match the natural network architecture. By doing so, a single management center may be shielded from information that is not useful for its operation.

Dividing network data among event managers may force them to make decisions based on incomplete information. Due to fault propagation event managers may receive notifications of problems that appeared in domains/layers managed by other event managers. Similarly, symptoms associated with a problem in a certain domain/layer may be unobservable in this domain. To locate faults in such a complex scenario a distributed fault localization algorithm is required.

Surprisingly, very little work has been done in the area of distributed fault localization. Theoretical foundations for the design of such techniques have been laid by Katzela et al. [54], who compare three schemes of fault localization: centralized, decentralized, and distributed while assessing the theoretical bound on the relative accuracy of these schemes. In [95], a distributed algorithm is proposed suitable for diagnosis of end-to-end connectivity problems in hierarchically routed networks.

5.4. Fault localization in service-oriented environment

Technologies developed in recent years make it possible to automate business interactions, both B2B and B2C ones. As a result of research and standardization initiatives such as Web services [31], and the propensity of businesses toward downsizing and outsourcing, a new trend has emerged in a way these interactions are being structured: a business is built from both organizationally and geographically distributed units, business partners can locate each other using the Internet, connect and perform transactions automatically, and terminate their relationship when no longer needed. This dynamic services infrastructure is referred to as dynamic e-business [70] or virtual enterprise [37].

As a result of the involvement of multiple organizations in the provisioning of a service offered to an end-user, fault localization in dynamic-services environments becomes even more difficult. Not only do the organizations have to cooperate in order to identify root causes, they also have to preserve their privacy and security in the process. It is necessary to integrate data from different organizations, which may come in different formats and contain information with various degrees of detail. Fault localization objectives are likely to be affected by priorities resulting from business goals. The fault management task needs to be able to assess a (business) impact of a fault and prioritize fault isolation, testing, and recovery actions to maximize a business utility. These fault management subtasks: fault detection, fault isolation, testing, and recovery are likely to be much more integrated than they used to be in the past.

5.5. Fault localization in mobile networks

One of the biggest challenges facing the fault management community is related to the management of wireless networks. The well known issues of environment-related device failures, inherently unreliable communication medium, power supply problems, and restricted bandwidth, which are typically associated with a wireless environment, also affect the fault management function. They result in new failures to consider, higher fault frequencies, higher symptom loss rates, increased number of transient faults, less computing resources available, and severely restricted amount of management information that may be exchanged between network nodes. In ad hoc and mobile networks, the problem is further complicated by the dynamically changing topology.

When provided with an accurate and constantly updated fault propagation model, some of the techniques described in this survey may still be applied to the mobile wireless environment. For example, in event-driven techniques, every symptom may be always analyzed with the most up-to-date model. When the fault propagation model changes during the fault localization process, other symptoms may be analyzed using the modified dependency information.

Thus the main obstacle in applying the techniques surveyed in this paper to fault localization in a mobile environment is the difficulty of building and keeping the model up-to-date. In fact, it is fair to say that the goal of providing the fault localization process with exact information on the topology and routing in a mobile environment is not realistic. Root cause analysis in a mobile environment may take one or more of the following approaches: creating approximate models based on the information on nodes’ locations and mobility patterns, multi-stage fault localization aiming first at identifying affected network domains or clusters and then focusing on the identified areas of interest, distributed fault localization, geographic approach aiming first at identifying an approximate location of a failing node, and usage of on-demand probing to make the fault localization process faster and less bandwidth consuming.

5.6. Obtaining fault localization models

The major drawback of model-based techniques is the necessity of obtaining and maintaining accurate information on error propagation patterns and relationships among network events. In the literature, there is little work addressing the issue of obtaining fault models. Most approaches anticipate the prior existence of such a model.

Since in communication systems errors propagate between related system components, models used for fault localization are usually derived from a model of relationships among system components. Such models usually record two types of dependencies between services and functions in neighboring protocol layers: *static* and *dynamic* dependencies. Static dependencies result from, e.g., standardized definition of functions provided by different layers of the protocol stack, or from static network topologies. Dynamic dependencies result from, e.g., run-time addition and deletion of services (such as establishment and termination of TCP sessions), dynamic routing, etc. The network topology may be obtained automatically through various network topology detection mechanisms, which are built into some commercially available network management systems [98]. Other management systems implement proprietary techniques that allow the discovery of hardware configuration changes such as addition or removal of a network adapter or host [19]. The IETF has recently recognized a need for a standardized means of representing the physical network connections by proposing the `Physical Topology MIB` [3], which can be used to obtain topology information if it is implemented in the managed domain.

Obtaining dynamic dependencies is significantly more difficult since the dependency model has to be continuously updated while the modeled system is running. In spite of that, many tools exist that facilitate the process of model building. These tools are usually specific to particular network services or functions. For example, all active TCP connections on a host may be retrieved using the *netstat* application [96]. A current route between a host and any other host may be obtained using program *traceroute* [96].

Network management protocols such as SNMP [9] provide a means to determine dependencies established using configuration or real-time routing protocols. For example, the management system may obtain the topology, which was dynamically established in the data-link layer by the Spanning Tree Protocol [76] using the data contained in `dot1dBase` Group of *Bridge MIB* [14]. Updates of the spanning tree may be triggered by `newRoot` and `topologyChange` traps [14]. In the network layer of the Internet, current routes may be calculated from `ipRoutingTable` of *TCP/IP MIB-II* [67].

Other techniques of obtaining network topology have been also investigated. To monitor hierarchical network topology, Novaes [71] uses IP multicast. Siamwalla et al. [88] propose several heuristics that exploit SNMP, DNS, ping, and traceroute facilities to discover the network level topology of the Internet on both intra-domain and backbone levels.

Govindan et al. [30] infer the Internet map using hop-limited probes. Reddy et al. [79] investigate a distributed topology discovery technique for the Internet. Breitbart et al. [6] present an algorithm that obtains both network and data-link layer topology using *TCP/IP MIB-II* [67] and *Bridge MIB* [14]. The algorithm of Lowekamp et al. [65] discovers the topology of a large Ethernet network allowing incomplete data in *Bridge MIB* tables.

Several techniques have been proposed that are tailored toward discovering relationships among services and software components. An automated technique of detecting static dependencies within software components on a single machine is proposed in [49]. Certain dependencies among software components may be obtained from system information repositories maintained by many operating systems [56]. Ramanathan et al. [78] propose techniques that obtain relationships among services within ISP systems. Brown et al. [7] and Bagchi et al. [2] describe an active approach based on controlled system perturbations designed for discovering dependencies among components of a typical e-commerce application. Ensel [18] applies neural networks to obtain relationships among the defined set of system components based on measurable values describing the components' activities and performance (e.g., CPU usage, bandwidth consumption, etc.). Hasselmeyer [34] argues that the dependencies among distributed cooperating components should be maintained and published by services themselves, and proposes a schema that allows these dependencies to be obtained.

Despite all the methods cited in this section, it has to be observed that obtaining dependency information in an automatic fashion is still an open research problem. The complexity of the task is related to the fact that obtaining dependency information is a problem that has to be solved separately for every system, layer, or type of device using techniques available or the most suitable given the circumstances. We hope that this section helps illustrate the plethora of approaches that may be adopted while solving this problem. Nevertheless, the amount of effort currently invested in this research topic lets us anticipate that more flexible, more comprehensive, and more general techniques will be available in the future.

6. Conclusions

Fault localization, a central aspect of network fault management, is a process of deducing the exact source of a failure from a set of observed failure indications. It has been a focus of research activity since the advent of modern communication systems, which produced numerous fault localization techniques. Fault localization is subject to complications resulting from complexity, unreliability, and non-determinism of communication systems.

This paper presents a comprehensive survey of fault localization techniques in communication systems. These techniques derive from different areas of computer science, including artificial intelligence, graph theory, neural networks, information theory, and automata theory, and include model-based reasoning tools, model traversing techniques, case-based reasoning tools, graph-theoretic techniques, and the codebook approach.

Despite this research effort, fault localization in complex communication systems remains an open research problem. The most challenging issues concern multi-layer fault localization, distributed diagnosis, temporal correlation, fault localization in mobile ad hoc networks, and root cause analysis in a service-oriented environment.[1]

References

[1] K. Appleby, G. Goldszmidt, M. Steinder, Yemanja—a layered event correlation system for multi-domain computing utilities, Journal of Network and Systems Management 10 (2) (2002) 171–194.

[2] S. Bagchi, G. Kar, J. Hellerstein, Dependency analysis in distributed systems using fault injection: Application to problem determination in an e-commerce environment, in: O. Festor, A. Pras (Eds.), Twelfth Internat. Workshop on Distributed Systems: Operations and Management, Nancy, France, October 2001 [21].

[3] A. Bierman, K. Jones, Physical Topology MIB, IETF Network Working Group, 2000, RFC 2922.

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

[5] A.T. Bouloutas, S.B. Calo, A. Finkel, I. Katzela, Distributed fault identification in telecommunication networks, Journal of Network and Systems Management 3 (3) (1995).

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

[7] A. Brown, G. Kar, A. Keller, An active approach to characterizing dynamic dependencies for problem determination in a distributed application environment, in: G. Pavlou, N. Anerousis, A. Liotta (Eds.), Integrated Network Management VII, Seattle, WA, May 2001.

[8] S. Brugnoni, R. Manione, E. Montariolo, E. Paschetta, L. Sisto, An expert system for real time diagnosis of the Italian telecommunications network, in: H.G. Hegering, Y. Yemini (Eds.), Integrated Network Management III, North-Holland, Amsterdam, 1993 [36].

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

[10] J.D. Case, K. McCloghrie, M.T. Rose, S. Waldbusser, Protocol Operations for Version 2 of the Simple Network Management Protocol (SNMPv2), IETF Network Working Group, 1996, RFC 1905.

[11] C.S. Chao, D.L. Yang, A.C. Liu, An automated fault diagnosis system using hierarchical reasoning and alarm correlation, Journal of Network and Systems Management 9 (2) (2001) 183–202.

[12] R. Comerford, The new software paladins, IEEE Spectrum 37 (6) (2000).

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

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

[15] R.H. Deng, A.A. Lazar, W. Wang, A probabilistic approach to fault diagnosis in linear lightwave networks, in: H.G. Hegering, Y. Yemini (Eds.), Integrated Network Management III, North-Holland, Amsterdam, 1993, pp. 697–708 [36].

[16] A. Dupuy, S. Sengupta, O. Wolfson, Y. Yemini, Design of the Netmate network management system, in: I. Krishnan, W. Zimmer (Eds.), Integrated Network Management II, North-Holland, Amsterdam, 1991, pp. 639–650 [59].

[17] A. Dupuy, J. Schwartz, Y. Yemini, G. Barzilai, A. Cahana, Network fault management: A user's view, in: B. Meandzija, J. Westcott (Eds.), Integrated Network Management I, North-Holland, Amsterdam, 1989, pp. 101–107 [68].

[18] C. Ensel, Automated generation of dependency models for service management, in: Workshop of the OpenView University Association, OVUA, Bologna, Italy, June 1999.

[19] S. Fakhouri, G. Goldszmidt, I. Gupta, M. Kalantar, J. Pershing, GulfStream—a system for dynamic topology management in multi-domain server farms, in: IEEE International Conference on Cluster Computing, 2001.

[1] The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied of the Army Research Lab or the US Government.

[20] M. Fecko, M. Steinder, Combinatorial designs in multiple faults localization for battlefield networks, in: IEEE Military Commun. Conf., MILCOM, McLean, VA, 2001.

[21] O. Festor, A. Pras (Eds.), Twelfth Internat. Workshop on Distributed Systems: Operations and Management, Nancy, France, October 2001.

[22] G. Forman, M. Jain, J. Martinka, M. Mansouri-Samani, A. Snoeren, Automated end-to-end system diagnosis of networked printing services using model based reasoning, in: A.S. Sethi (Ed.), Ninth Internat. Workshop on Distributed Systems: Operations and Management, University of Delaware, Newark, DE, October 1998, pp. 142–154 [87].

[23] I. Foster, C. Kesselman (Eds.), "The Grid": Blueprint for a New Computing Infrastructure, Morgan Kaufmann Publishers, San Francisco, CA, 1998.

[24] M. Frontini, J. Griffin, S. Towers, A knowledge-based system for fault localization in wide area network, in: I. Krishnan, W. Zimmer (Eds.), Integrated Network Management II, North-Holland, Amsterdam, 1991, pp. 519–530 [59].

[25] R.D. Gardner, D.A. Harle, Methods and systems for alarm correlation, in: Proc. of GLOBECOM, London, UK, November 1996, pp. 136–140.

[26] R.D. Gardner, D.A. Harle, Alarm correlation and network fault resolution using the Kohonen self-organizing map, in: Proc. of IEEE GLOBECOM, Toronto, Canada, September 1997.

[27] R.D. Gardner, D.A. Harle, Pattern discovery and specification techniques for alarm correlation, in: NOMS'98, Proc. Network Operation and Management Symposium, New Orleans, LA, 1998, pp. 713–722 [77].

[28] J. Goldman, P. Hong, C. Jeromnion, G. Louit, J. Min, P. Sen, Integrated fault management in interconnected networks, in: B. Meandzija, J. Westcott (Eds.), Integrated Network Management I, North-Holland, Amsterdam, 1989, pp. 333–344 [68].

[29] R. Gopal, Layered model for supporting fault isolation and recovery, in: J.W. Hong, R. Weihmayer (Eds.), Proc. Network Operation and Management Symposium, Honolulu, Hawaii, April 2000, pp. 729–742 [39].

[30] R. Govindan, H. Tangmunarunkit, Heuristics for Internet map discovery, in: Proc. of IEEE INFOCOM, 2000, pp. 1371–1380.

[31] S. Graham, S. Simeonov, T. Boubez, D. Davis, G. Daniels, Y. Nakamura, R. Neyama, Building Web Services with "Java", SAMS Publishing, 2002.

[32] B. Gruschke, Integrated event management: Event correlation using dependency graphs, in: A.S. Sethi (Ed.), Ninth Internat. Workshop on Distributed Systems: Operations and Management, University of Delaware, Newark, DE, October 1998, pp. 130–141 [87].

[33] M. Hasan, B. Sugla, R. Viswanathan, A conceptual framework for network management event correlation and filtering systems, in: M. Sloman, S. Mazumdar, E. Lupu (Eds.), Integrated Network Management VI, IEEE, 1999, pp. 233–246 [89].

[34] P. Hasselmeyer, Managing dynamic service dependencies, in: O. Festor, A. Pras (Eds.), Twelfth Internat. Workshop on Distributed Systems: Operations and Management, Nancy, France, October 2001 [21].

[35] D. Heckerman, M.P. Wellman, Bayesian networks, Communications of the ACM 38 (3) (1995) 27–30.

[36] H.G. Hegering, Y. Yemini (Eds.), Integrated Network Management III, North-Holland, Amsterdam, 1993.

[37] Y. Hoffner, S. Field, P. Grefen, H. Ludvig, Contract-driven creation and operation of virtual enterprises, Computer Networks 37 (2) (2001) 111–136.

[38] P. Hong, P. Sen, Incorporating non-deterministic reasoning in managing heterogeneous network faults, in: I. Krishnan, W. Zimmer (Eds.), Integrated Network Management II, North-Holland, Amsterdam, 1991, pp. 481–492 [59].

[39] J.W. Hong, R. Weihmayer (Eds.), Proc. Network Operation and Management Symposium, Honolulu, Hawaii, April 2000.

[40] C.S. Hood, C. Ji, Proactive network management. In Proc. of IEEE INFOCOM, Kobe, Japan, 1997, pp. 1147–1155.

[41] K. Houck, S. Calo, A. Finkel, Towards a practical alarm correlation system, in: A.S. Sethi, F. Faure-Vincent, Y. Raynaud (Eds.), Integrated Network Management IV, Chapman and Hall, London, 1995, pp. 226–237 [86].

[42] Automating root cause analysis: codebook correlation technology vs. rule based analysis, White Paper, SMARTS, 2001, http://www.smarts.com/.

[43] ISO, Information Technology–OSI, ISO/IEC 10165-4 standard: Management Information Services: Structure of Management Information, Part 4: Guidelines for the Definition of Managed Objects, 1991.
[44] ISO, Information Processing Systems—OSI, ISO Standard 9596-1: Common Management Information Protocol, Part 1: Specification.
[45] G. Jakobson, M.D. Weissman, Alarm correlation, IEEE Network 7 (6) (1993) 52–59.
[46] G. Jakobson, M.D. Weissman, Real-time telecommunication network management: Extending event correlation with temporal constraints, in: A.S. Sethi, F. Faure-Vincent, Y. Raynaud (Eds.), Integrated Network Management IV, Chapman and Hall, London, 1995, pp. 290–302 [86].
[47] J.F. Jordaan, M.E. Paterok, Event correlation in heterogeneous networks using the OSI management framework, in: H.G. Hegering, Y. Yemini (Eds.), Integrated Network Management III, North-Holland, Amsterdam, 1993, pp. 683–695 [36].
[48] C. Joseph, J. Kindrick, K. Muralidhar, T. Toth-Fejel, MAP fault management expert system, in: B. Meandzija, J. Westcott (Eds.), Integrated Network Management I, North-Holland, Amsterdam, 1989, pp. 627–636 [68].
[49] G. Kar, A. Keller, S. Calo, Managing application services over service provider networks, in: J.W. Hong, R. Weihmayer (Eds.), Proc. Network Operation and Management Symposium, Honolulu, Hawaii, April 2000 [39].
[50] S. Kätker, A modeling framework for integrated distributed systems fault management, in: C. Popien (Ed.), Proc. IFIP/IEEE Internat. Conference on Distributed Platforms, pp. 187–198, Dresden, Germany, 1996.
[51] S. Kätker, K. Geihs, A generic model for fault isolation in integrated management systems, Journal of Network and Systems Management 5 (2) (1997) 109–130.
[52] S. Kätker, M. Paterok, Fault isolation and event correlation for integrated fault management, in: A. Lazar, R. Sarauo, R. Stadler (Eds.), Integrated Network Management V, Chapman and Hall, London, 1997, pp. 583–596 [60].
[53] I. Katzela, Fault diagnosis in telecommunications networks, Ph.D. Thesis, School of Arts and Sciences, Columbia University, New York, 1996.
[54] I. Katzela, A.T. Bouloutas, S.B. Calo, Centralized vs distributed fault localization, in: A.S. Sethi, F. Faure-Vincent, Y. Raynaud (Eds.), Integrated Network Management IV, Chapman and Hall, London, 1995, pp. 250–263 [86].
[55] I. Katzela, M. Schwartz, Schemes for fault identification in communication networks, IEEE/ACM Transactions on Networking 3 (6) (1995) 733–764.
[56] A. Keller, U. Blumenthal, G. Kar, Classification and computation of dependencies for distributed management, in: Proc. of the Fifth IEEE Symposium on Computers and Communications, ISCC, Antibes–Juan-les-Pins, July 2000.
[57] M. Klemettinen, H. Mannila, H. Toivonen, Rule discovery in telecommunication alarm data, Journal of Network and Systems Management 7 (4) (1999) 395–423.
[58] S. Kliger, S. Yemini, Y. Yemini, D. Ohsie, S. Stolfo, A coding approach to event correlation, in: A.S. Sethi, F. Faure-Vincent, Y. Raynaud (Eds.), Integrated Network Management IV, Chapman and Hall, London, 1995, pp. 266–277 [86].
[59] I. Krishnan, W. Zimmer (Eds.), Integrated Network Management II, North-Holland, Amsterdam, 1991.
[60] A. Lazar, R. Saracco, R. Stadler (Eds.), Integrated Network Management V, Chapman and Hall, London, 1997.
[61] L. Lewis, A case-based reasoning approach to the resolution of faults in communications networks, in: H.G. Hegering, Y. Yemini (Eds.), Integrated Network Management III, North-Holland, Amsterdam, 1993, pp. 671–681 [36].
[62] G. Liu, A.K. Mok, E.J. Yang, Composite events for network event correlation, in: M. Sloman, S. Mazumdar, E. Lupu (Eds.), Integrated Network Management VI, IEEE, 1999, pp. 247–260 [89].
[63] C.-C. Lo, S.-H. Chen, A scheduling-based event correlation scheme for fault identification in communications network, Computer Communications 22 (5) (1999) 432–438.
[64] K.-W.E. Lor, A network diagnostic expert system for Acculink™ multiplexers based on a general network diagnostic scheme, in: H.G. Hegering, Y. Yemini (Eds.), Integrated Network Management III, North-Holland, Amsterdam, 1993, pp. 659–669 [36].
[65] B. Lowecamp, D.R. O'Hallaron, T.R. Gross, Topology discovery for large Ethernet networks, in: Proc. of ACM SIGCOMM, 2001, pp. 239–248.

[66] M. Mansouri-Samani, M. Sloman, GEM—a generalised event monitoring language for distributed systems, IEE/IOP/BCS Distributed Systems Engineering Journal 4 (2) (1997).
[67] K. McCloghrie, M. Rose, Management Information Base for Network Management of TCP/IP-based internets: MIB-II, IETF Network Working Group, 1991, RFC 1213.
[68] B. Meandzija, J. Westcott (Eds.), Integrated Network Management I, North-Holland, 1989.
[69] J.-A. Meyer, Artificial life and the animat approach to artificial intelligence, in: M.A. Boden (Ed.), Artificial Intelligence, second ed., Handbook of Perception and Cognition, Academic Press, New York, 1996, pp. 325–354 (Chapter 11).
[70] C. Mohan, Dynamic e-business: Trends in web services, in: Proc. 3rd VLDB Workshop on Technologies for E-Services, TES2002, Hong Kong, August 2002.
[71] M. Novaes, Beacon: A hierarchical network topology monitoring system based in IP multicast, in: A. Ambler, S.B. Calo, G. Kar (Eds.), Services Management in Intelligent Networks, Lecture Notes in Computer Science, no. 1960, Springer-Verlag, Berlin, 2000, pp. 169–180.
[72] Y.A. Nygate, Event correlation using rule and object based techniques, in: A.S. Sethi, F. Faure-Vincent, Y. Raynaud (Eds.), Integrated Network Management IV, Chapman and Hall, London, 1995, pp. 278–289 [86].
[73] D. Ohsie, A. Mayer, S. Kliger, S. Yemini, Event modeling with the MODEL language, in: A. Lazar, R. Sarauo, R. Stadler (Eds.), Integrated Network Management V, Chapman and Hall, London, 1997, pp. 625–637 [60].
[74] A. Patel, G. McDermott, C. Mulvihill, Integrating network management and artificial intelligence, in: B. Meandzija, J. Westcott (Eds.), Integrated Network Management I, North-Holland, Amsterdam, 1989, pp. 647–660 [68].
[75] J. Pearl, Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference, Morgan Kaufmann Publishers, San Mateo, CA, 1988.
[76] R. Perlman, Interconnections, Second Edition: Bridges, Routers, Switches, and Internetworking Protocols, Addison Wesley, Reading, MA, 1999.
[77] Proc. IEEE/IFIP Network Operation and Management Symposium, New Orleans, LA, 1998.
[78] S. Ramanathan, D. Caswell, S. Neal, Auto-discovery capabilities for service management: An ISP case study, Journal of Network and Systems Management 8 (4) (2000) 457–482.
[79] A. Reddy, D. Estrin, R. Govindan, Large-scale fault isolation, IEEE Journal on Selected Areas in Communications 18 (5) (2000) 723–732.
[80] G.D. Rodosek, T. Kaiser, Intelligent assistant: User-guided fault localization, in: A.S. Sethi (Ed.), Ninth Int'l Workshop on Distributed Systems: Operations and Management, University of Delaware, Newark, DE, October 1998, pp. 119–129 [87].
[81] S. Roman, Coding and Information Theory, first ed., Graduate Texts in Mathematics, no. 134, Springer-Verlag, Berlin, 1992.
[82] S. Russell, Machine learning, in: M.A. Boden (Ed.), Artificial Intelligence, second ed., Handbook of Perception and Cognition, Academic Press, New York, 1996, pp. 89–133 (Chapter 4).
[83] S. Russell, P. Norvig, Artificial Intelligence: Modern Approach, Prentice Hall, Englewood Cliffs, NJ, 1995.
[84] P.H. Schow, The alarm information base: A repository for enterprise management, in: Proc. Second IEEE Internat. Workshop on Systems Management, Los Alamitos, CA, 1996, pp. 142–147.
[85] C. Scott, P. Wolfe, M. Erwin, Virtual private networks, second ed., O'Reilly, 1999.
[86] A.S. Sethi, F. Faure-Vincent, Y. Raynaud (Eds.), Integrated Network Management IV, Chapman and Hall, 1995.
[87] A.S. Sethi (Ed.), Ninth Int'l Workshop on Distributed Systems: Operations and Management, University of Delaware, Newark, DE, October 1998.
[88] R. Siamwalla, R. Sharma, S. Keshav, Discovering Internet topology, Technical Report, Cornell University, Ithaca, NY, 1998.
[89] M. Sloman, S. Mazumdar, E. Lupu (Eds.), Integrated Network Management VI, IEEE, 1999.
[90] P. Smyth, Markov monitoring with unknown states, IEEE Journal on Selected Areas in Communications 12 (9) (1994) 1600–1612.
[91] R. Stadler, M. Ulema (Eds.), Proc. Network Operation and Management Symposium, Florence, Italy, April 2002.

[92] M. Steinder, A.S. Sethi, End-to-end service failure diagnosis using belief networks, in: R. Stadler, M. Ulema (Eds.), Proc. Network Operation and Management Symposium, Florence, Italy, April 2002, pp. 375–390 [91].

[93] M. Steinder, A.S. Sethi, Probabilistic fault diagnosis in communication systems through incremental hypothesis updating, Comput. Networks 45 (2004) 537–562.

[94] M. Steinder, A.S. Sethi, Non-deterministic fault localization in communication systems using belief networks, IEEE/ACM Transactions on Networking (2004), in press.

[95] M. Steinder, A.S. Sethi, Multi-domain diagnosis of end-to-end service failures in hierarchically routed networks, in: IFIP Networking, Athens, Greece, May 2004.

[96] W.R. Stevens, TCP/IP Illustrated, vol. I., first ed., Addison Wesley, Reading, MA, 1995.

[97] M.T. Sutter, P.E. Zeldin, Designing expert systems for real time diagnosis of self-correcting networks, IEEE Network (1988) 43–51.

[98] Tivoli, Netview for Unix: Administrator's Guide, Version 6.0, January 2000.

[99] Z. Wang, Model of network faults, in: B. Meandzija, J. Westcott (Eds.), Integrated Network Management I, North-Holland, Amsterdam, 1989, pp. 345–352 [68].

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

[101] H. Wietgrefe, Investigation and practical assessment of alarm correlation methods for the use in GSM access networks, in: R. Stadler, M. Ulema (Eds.), Proc. Network Operation and Management Symposium, Florence, Italy, April 2002, pp. 391–404 [91].

[102] P. Wu, R. Bhatnagar, L. Epshtein, M. Bhandaru, Z. Shi, Alarm correlation engine (ACE). In Proc. Network Operation and Management Symposium, NOMS'98, New Orleans, LA, 1998, pp. 733–742 [77].

[103] T. Yamahira, Y. Kiriha, S. Sakata, Unified fault management scheme for network troubleshooting expert system, in: B. Meandzija, J. Westcott (Eds.), Integrated Network Management I, North-Holland, Amsterdam, 1989, pp. 637–646 [68].

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