Recent Advances in Fault Localization in Computer Networks

Ayush Dusia, *Student Member, IEEE*, and Adarshpal S. Sethi, *Member, IEEE*

***Abstract*—Fault localization, a core element in network fault management, is the process of inferring the exact failure in a network from the set of observed symptoms. Since faults in network systems can be unavoidable, their quick and accurate detection and diagnosis is important for the stability, consistency, and performance of a communication system. In this paper, we discuss the challenges of fault localization in complex communication systems and present an overview of recent techniques proposed in the literature along with their advantages and limitations. We start by briefly surveying passive monitoring techniques which were previously reviewed in a survey by Steinder [1]. We then describe more recent fault localization research in five categories: active monitoring techniques, techniques for overlay and virtual networks, decentralized probabilistic management techniques, temporal correlation techniques, and learning techniques.**

Keywords*—*Fault Localization, Fault Detection, Network Management, Survey, Fault Diagnosis, Passive Monitoring, Active Monitoring, Overlay and Virtual Networks.

I. Introduction

COMMUNICATION networks have dramatically increased in size and complexity over the years. The increase in complexity presents serious challenges to the operations of network management systems. One of the central issues faced by a network management system is fault management. Failures can be unavoidable in large communication networks, so the detection and identification of failures in a timely manner is important for the reliable operation of the networks. Fault management systems attempt to detect, identify, localize, and initiate recovery mechanisms for correcting any condition that degrades the performance of networks.

Faults are network events that are the root cause of problems that can occur in the network. A failure refers to an inability of a device or service to function correctly. A failure of one component in a network may cause other components to fail. Faults and failures may occur in components that are hardware devices (e.g., routers, links), or software (e.g., malfunction in a routing table, failed web service). Symptoms are external manifestations of the failures in a network. Symptoms can be observed in various ways, such as alarms in managed networks, by human observations, or by using monitoring mechanisms like probing. Networking devices often provide notifications in the form of alarms when they sense any malfunction in the network. These notifications may be generated by management agents via management protocol messages, such as SNMP trap and CMIP EVENT-REPORT. An alarm only indicates that a managed network component has observed a deviation from a normal operation or an ideal behavior of the component, e.g., packets loss, failed connection, or link failure. Thus alarms are considered to be symptoms of possible faults in the network. The challenge associated with the use of these alarms for diagnosing a network for faults is that the actual faults could be anywhere in the network.

Let us explain a fault scenario with the help of an example shown in Figure 1. The client is trying to use ssh to connect to a remote server, Server 2, over the Internet. The end router has multiple entries for Server 2 in its Forwarding Table with different forwarding ports. Always the first entry in the Forwarding Table is selected and the ssh request for destination Server 2 is erroneously forwarded to Server 1 where it is dropped. The client does not receive a response to the ssh request which eventually times out. The failed connection is reported as an alarm to the network administrator. This example shows how a fault can cause a service disruption at a distant location.

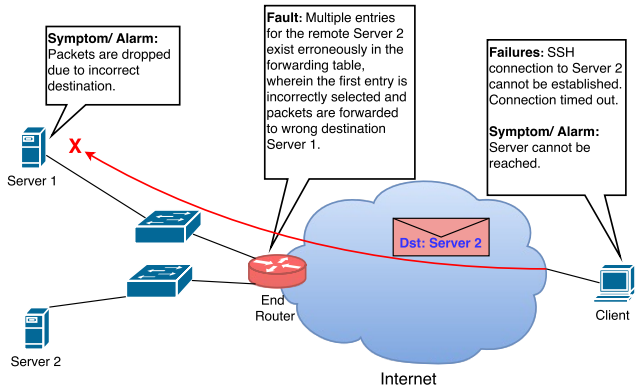


Fig. 1. Example to illustrate faults, failures and symptoms.

The fault management process usually includes two activities: fault diagnosis and fault recovery. Fault diagnosis consists of three steps: fault detection, fault localization, and testing. Fault detection is the process of observing network symptoms to determine that one or more failures may have occurred. Fault localization uses the set of observed symptoms to infer the location of the exact failures in a network. Testing is the process of verifying the inferred failures by checking the identified components. Fault recovery is the process of fixing the fault after it has been identified. Among the three phases of fault diagnosis, fault localization is the most challenging

A. Dusia and A. S. Sethi are with the Department of Computer and Information Sciences, Univesity of Delaware, Newark, DE 19716, USA (e-mail: adusia@udel.edu; sethi@udel.edu).

task because of its complexity. The complexity increases with the increase in size, unreliability, and non-determinism of a network. In addition, fault localization techniques may face the challenge of partial and incorrect input from the fault detection phase. A variety of fault localization techniques have been proposed in the literature. Many of these techniques were described in a comprehensive survey by Steinder [1]. In the decade since Steinder's survey, many new fault localization techniques have emerged, and this field has advanced in many new directions.

In this paper we address some of these recent advances and present an overview by discussing the advantages and limitations of each approach. The paper is structured as follows. Section II summarizes the work presented in the previous survey [1]. In Section III, we discuss the active monitoring techniques which mostly use a probing approach to detect and localize faults in a network. In Section IV, techniques that can be applied to an overlay network or a virtual network are described. Section V describes the distributed and decentralized probabilistic network management techniques, which are based on collaboration and information sharing among nodes. In Section VI, techniques which consider the temporal aspect of a network model are discussed. Fault localization techniques based on learning algorithms are described in Section VII. Section VIII presents some of the open research problems and possible future work in the area. Finally, Section IX includes conclusions of this paper.

II. Passive Monitoring Techniques

Passive monitoring techniques monitor a network by deploying monitoring agents on the networking devices. Any failure condition in the network could generate multiple alarms by monitoring agents. The alarms are then used as symptoms by the Network Management System (NMS) to analyze the exact failure condition in the network. These techniques are passive because the Network Management System waits passively for alarms to be sent by the agents. Figure 2 shows an example of an NMS which is using passive monitoring. Many fault localization techniques based on a variety of paradigms have been proposed in the past. These paradigms derive from different areas such as artificial intelligence, graph theory, neural networks, information theory, and automata theory. The classification of these fault localization techniques has been provided by Steinder [1] and is shown in Figure 3.

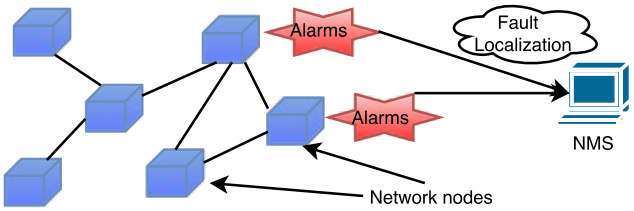


Fig. 2. Passive monitoring.

Passive monitoring techniques are broadly classified into three categories: 1) Artificial intelligence (AI) techniques, 2) Model traversing techniques, and 3) Graph-theoretic techniques. Some of the AI techniques are rule-based, model-based, and case-based. These techniques use an expert system with a knowledge base to do the analysis by imitating human knowledge resulting from experience and deep understanding of network behavior. The approaches that rely solely on expert knowledge expressed as a set of predefined directive, i.e. rules, to diagnose faults are called rule-based approaches [2], [3]. The rules are typically formulated as a set of if-then statements. The research on rule-based fault localization focuses on efficiently structuring the knowledge base and designing the rule-definition language. Use of rule-based approaches do not require profound understanding of the underlying system architecture and operational principles. Rule-based approaches possess disadvantages such as inability to learn from experience, inability to deal with unseen problems, and difficulty in updating the knowledge base [4]. The disadvantages limit the usability of rule-based approaches for isolating faults in complex systems.

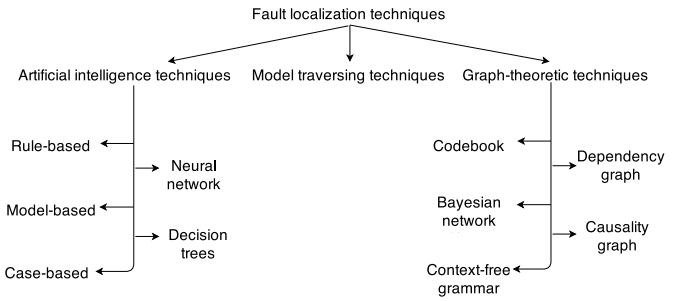


Fig. 3. Classification of fault localization techniques [1].

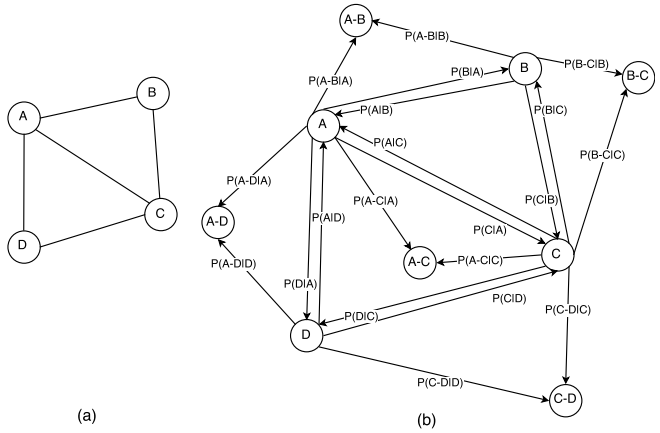


Fig. 4. (a) A simple network topology and (b) its corresponding dependency graph.

Model-based approaches use an expert knowledge that describes the behavior of a system as a mathematical model. The observed symptoms are compared to the behavior predicted by the model. Faults are detected when the observed symptoms fail to conform to the predicted behavior. Deep knowledge of

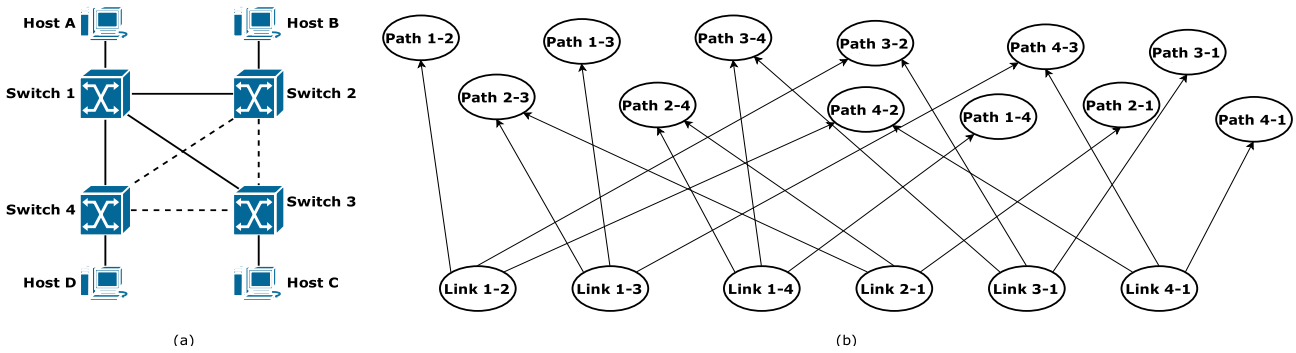


Fig. 5. (a) A simple network topology and (b) its corresponding Bayesian network.

the network connectivity and operations is required by model-based approaches, and they also have the potential to solve novel problems. The knowledge base used in the model-based approaches is organized in an expandable, upgradeable and modular fashion. However, the models may be difficult to obtain and keep up-to-date [5]. Case-based approaches diagnose a system based on an expert knowledge gained from experience and past situations. The knowledge is acquired from relevant past cases and the solutions provided to the previous problems [4]. Neural network approaches use an expert knowledge that tries to mimic operation of a human brain. The knowledge is in the form of interconnected nodes called neurons. The neural network is capable of learning, and is resilient to noise and inconsistencies in the input data. However, the training periods required are long and any behavior outside the training data is difficult to predict [6]–[8]. Decision tree approaches use an expert knowledge that guides the symptoms of failure towards locating the root cause of the problem [9]. The expert knowledge is a simple and expressive representation of the behavior of a system. However, the applicability of decision tree based fault localization approaches is limited to specific applications, and the accuracy of these approaches can be degraded in the presence of noise.

The second category, model traversing techniques, uses a formal representation of the network and relationships among the network entities to create an object-oriented representation of the network [10]–[12]. The model is used to correlate events for locating the faulty network elements. Model traversing techniques seem reasonable when relationships between network components are graph-like and easy to obtain [13]. Event correlation in model-traversing techniques is usually event-driven, and for every observed event the model is searched recursively. The advantage is robustness against frequent network configuration changes. Also, the modular relationship encourages to design the distributed fault localization algorithms. However, these techniques are inflexible to model propagating fault patterns. In particular, to model situations in which failure of a device depends on a logical combination of other device failures [14].

The third category, graph-theoretic techniques, exploits a graphical model called a Fault Propagation Model (FPM). An FPM describes the relationship between all possible faults and symptoms which can occur in a network. Graph-theoretic techniques require a-priori knowledge of how a failure condition or alarm in a network component is related to failure conditions or alarms in other components. To create such a model, an accurate knowledge of dependencies among network components is required. The efficiency and accuracy of the fault localization algorithm depends on the accuracy of a-priori knowledge. Some possible graph-theoretic representations of an FPM are dependency graphs and Bayesian networks.

A dependency graph is a directed graph in which nodes represent the events, and edges represent the relationships between events (e.g. an edge from node A to node B if event 'A' causes event 'B'). Every directed edge is labeled with a conditional probability that the node at the end of an edge fails, provided that the node at the beginning of an edge fails. Figure 4 shows an example of a simple network topology and its dependency graph. Nodes in the dependency graph of this figure represent either the failures of a network node (e.g., A), or the failures of a network link (e.g., A-B). Some of the work on fault diagnosis that uses dependency graphs is described in [15]–[17].

A Bayesian network, also called a belief network, is a directed acyclic graph (DAG), in which each node represents a random variable over a multivalued domain. The directed edges represent causal relationships among the variables and the strengths of these influences are specified by conditional probabilities. Figure 5 shows an example of a Bayesian network for a simple network topology with four switches. End-to-end paths in the network follow the links in the spanning tree shown by solid lines. The Bayesian network depicts the dependency between nodes that represent end-to-end paths (e.g., path 4-2) and nodes that represent the links traversed by these paths (e.g., link 4-1 and link 1-2). This example only considers a simple failure scenario in which each link is either working properly or has failed. However, it is possible to design Bayesian networks to capture more general failure scenarios. For instance, each path or link in the network may experience multiple types of problems such as delay, excessive packet loss, or complete loss of connectivity. This situation can be represented in the fault propagation model by having

TABLE I. COMPARISON OF APPROACHES IN ARTIFICIAL INTELLIGENCE AND MODEL TRAVERSING CATEGORIES

Passive Monitoring	Rule-based	Model-based	Case-based	Neural networks	Decision trees	Model traversing
Expert System	Yes	Yes	Yes	Yes	Yes	No
Require profound understanding of the underlying system behavior	No	Yes	Yes, require application specific model	No	Yes, require application specific model	Yes, require application specific model
Ability to diagnose unforeseen problems	No	Yes	No	No	No	No
Applicability to deal with system with complex dependencies	No	No	No	No	No	No
Ability to learn from experience	No	Yes, but difficult to keep up-to-date	Yes, can learn correlation patterns	Yes, but require training data	Yes, but require training data	No
Deal with inaccurate or incomplete expert knowledge	No	No	No	Yes	No	No
Reusability	No	Yes	Yes	Yes	Yes	No
Time-window or Event-driven based	Time-window based	Time-window based	Time-window based	Time-window based	Time-window based	Event-driven
Resilient to network changes	No	Yes	Yes	No	No	Yes

multiple path or link nodes for each path or link, where each node corresponds to a different failure type.

Dependency graphs are not required to be acyclic which makes them easier to learn from a complete set of data as compared to Bayesian networks [18]. Standard classification or regression algorithms can be used for dependency graphs, to learn the conditional probability distribution for each node, and to select parents for a node. Furthermore, there are many classification and regression learning algorithms that scale up for large data sets. These algorithms can be used to produce scalable algorithms for learning dependency graphs. However, these algorithms cannot be easily modified to respect the graphical constraints imposed by Bayesian networks. Bayesian networks also have some advantages over dependency networks. Many well-established approximate inference techniques such as loopy propagation [19] and variational methods [20] can be used for complex Bayesian network structures. The factored form of their joint distribution also leads to efficient inference algorithms, whereas in dependency graphs, the probabilistic inference can be slow and complicated [21].

Most of the FPM based fault localization techniques stand on the use of: 1) Divide and conquer algorithms, 2) Context-free grammar, 3) Codebook techniques, 4) Bayesian networks, and 5) Bipartite causality graphs. The divide and conquer approach [15] uses a dependency graph as an FPM assuming that only one failure type is allowed per network component. The approach is window-based and is tailored towards identifying dependent faults. The algorithm first identifies the symptom domain, which is the set of all faults that may have caused the observed symptoms within a time frame. The set is then repeatedly partitioned into subsets such that nodes belonging to the edge with the highest probability stay in the same subset. The recursion continues until a set becomes singleton. This algorithm does not handle lost or spurious symptoms.

In the context-free grammar approach, the feature of building expressions using sub-expressions is used to represent an FPM. This approach allows a network with complex dependencies between network components to be easily modeled. Information cost is assigned to each fault in the network. Two fault localization algorithms are described in [22] that use context-free grammar to represent fault propagation patterns. The first algorithm selects a minimum set of faults that best explains all the observed symptoms. The algorithm starts by selecting a fault that has largest number of symptoms. The symptoms explained by the selected fault are removed from the symptom set and the algorithm continues until all symptoms are explained. The second algorithm takes into account lost and spurious symptoms, and tries to find a subset of faults that best explains all symptoms with the minimal information cost. A tree structure is used where each node stores a set of identified faults, a set of observed symptoms which remains to be explained, and a cost associated with that node. A minimum cost node of the tree is selected as the best solution.

In the codebook approach, the fault propagation model is represented by a matrix of problem (event) codes. Problems are distinguished from one another based on their code [23], [24]. In a deterministic model, a code is a sequence of $\{0, 1\}$ values. The value of 1 at the i^{th} position of a code generated for problem P_j indicates a cause-effect implication between problem P_j and symptom S_i. In a probabilistic model, codes contain values $\in$ [0, 1] that represent likelihood of the cause-effect implication between a given problem and a given symptom. In this interpretation, the event correlation is equivalent to decoding a received output symbol to one of the valid input symbols. Spurious and lost symptoms are considered as channel errors. The number of errors that may be detected or corrected depends on the codebook and the decoding scheme.

When a Bayesian network (BN) is used as a fault propagation model, a random variable represents the state of a network component or the occurrence of a network event. Bayesian networks are used to make four basic queries given evidence set e: 1) belief assessment, 2) most probable explanation, 3) maximum a posteriori hypothesis, and 4) maximum expected utility [25]. The first two queries are of particular interest in fault localization. The belief assessment task is to compute $bel(V_i = v_i) = P(V_i = v_i|e)$ for one or more variables V_i. The most probable explanation (MPE) task is to find a complete assignment of values to random variables in V that best explains the observed evidence e. The fault localization

TABLE II. COMPARISON OF GRAPH-THEORETIC TECHNIQUES

Graph theoretic techniques	Divide and conquer	Context-free grammar	Code-based	Bayesian network	Causal graphs
Require profound understanding of the underlying system behavior	A priori information of symptoms and failure	A priori information of symptoms and failure	A priori information of symptoms and failure	A priori information of symptoms and failure	A priori information of symptoms and failure
Graphical model used	Dependency graph	Tree structure	Correlation matrix	Bayesian network	Bipartite causal graph
Time-window or Event-driven based	Time-window based	Time-window based	Event-driven	Event-driven	Event-driven
Applicability to deal with system with complex dependencies	Yes	Yes	No	Yes	Yes
Ability to deal with spurious and lost symptoms	No	Yes	No, considered errors	Yes	Yes
Multiple faults detection	No	No	No	Yes	Yes
Resilient to network changes	No	No	No	No	No
Algorithmic approach	Divide and conquer	Tree traversal	Hamming distance	Incremental	Incremental

problem is often formulated as the problem of calculating the most probable explanation (MPE) of the observed evidence (symptoms) in a BN. However, this problem is NP-hard in general. A belief updating algorithm for calculating MPE is available for polytrees which is polynomial with respect to the number of random variables. However, in unconstrained polytrees, the MPE algorithm still has exponential bound. The polynomial algorithm for calculating MPE [26] utilizes a message passing scheme in which BN nodes exchange messages that encode certain conditional probabilities. An approximation of the polynomial algorithm is used for performing fault localization using BN [27]. In this approach, a simplified model of Bayesian networks called noisy-OR gates [26] is considered to represent the conditional probabilities. The simplified model contains binary-valued random variables. The model assumes that all the faults are independent; this assumption of independence is ubiquitous in all the probabilistic fault localization approaches in the literature [15] [23]. This simplification helps to avoid exponential time and memory otherwise needed to store and process conditional probability matrices associated with random variables in the belief network. The approximation algorithm proceeds in an event driven manner, where an ordered iteration of message passing is done in the graph after each symptom is observed. When all symptoms are analyzed in this manner, the algorithm yields the probability of each fault given the observed symptoms.

To represent causality an FPM can also be modeled as a bipartite graph. Although relationships between faults and symptoms in real-life networks are usually more complex than a bipartite graph representation, many fault localization techniques use bipartite FPM [15], [24], [28]. The use of a bipartite model is justified for the following reasons: 1) Performing fault localization with complex representations is difficult, NP-hard in general, 2) Detailed models are often reduced to bipartite ones through a sequence of graph reduction operations, 3) Building complex models requires a profound knowledge of the underlying network, while symptom-fault maps may be obtained through external observations. In many real-life problems, only bipartite symptom-fault model solutions are feasible. One such approach tailored towards a bipartite FPM is Incremental Hypothesis Updating (IHU) [29], 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, which expresses the confidence associated with a given hypothesis relative to other hypotheses. The algorithm proceeds in an event-driven and incremental fashion. So, instead of waiting for a specific period of time before presenting a solution, the approach makes the hypotheses available on a continuous basis, and constantly upgrades them with information learned from new symptoms.

In order to reduce the computation intractability associated with the use of Bayesian networks for large number of nodes, a hybrid approach is proposed by Bennacer *et al.* in [30] that combines the use of Bayesian networks (BN) and case-based reasoning (CBR). The basic idea of this approach is to use CBR for simplifying and optimizing the fault diagnosis process by reducing the inherent complexity associated with the BN-based fault diagnosis, and also retain the advantages associated with the use of BN. During the diagnosis, a subset of the BN structure is identified by CBR, and in this subset only those nodes for which a variation is observed are considered for reasoning. The use of CBR also helps to utilize previously learned situations to solve current and future problems. The knowledge base of CBR is continuously updated with the solutions and that helps to improve the efficiency of diagnosis with time. This combined approach accelerates the fault diagnosis process and also reduces the computational complexity associated with the reasoning.

Tables I and II provide a comparison of the various passive monitoring techniques and list the important features of each technique. Table I compares approaches in the Artificial Intelligence and Model Traversing categories, whereas Table II compares the Graph-theoretic techniques.

III. ACTIVE MONITORING TECHNIQUES

A probing station is a node in the network that transmits one or more packets called probes for the purpose of monitoring the state of the network. Examples of probes may be ping or traceroute; probes may also be more complex and may be handled by any protocol layer. The use of probes to determine the network behavior or measure the quality of network performance is called probing. Active monitoring techniques use probing for a variety of network management applications. The use of probes helps the NMS to respond more quickly and accurately to the large number of network events, as opposed to the traditional passive event correlation approach. The probes

are typically transmitted to obtain end-to-end statistics such as latency, loss, and throughput. These statistics are then used to infer the health of network components. Network parameters and conditions can also be inferred from probe results. Active monitoring techniques have the potential to provide effective solutions for network monitoring applications due to their fundamental end-to-end nature and flexibility in responding to events. However, drawbacks of active monitoring techniques are the invasive character of probes and potentially large overhead.

Diagnosing a network using probes requires probing stations to be placed at one or more locations in the network. Then probes are selected to target different network nodes and links. The configuration of both the probing stations and probes impose a cost to network management; probing stations because the probing code must be installed, operated, and maintained, and probes because their use entails additional network traffic overhead and also due to the collection, storage and analysis of probe results. A trade-off exists between these costs, as higher number of probing stations allows selection of fewer probes. Various techniques have been proposed to select probing stations and probes where the goal is to obtain an optimal number of both probing stations and probes while minimizing the probing costs, and yet provide a wide coverage to locate problems anywhere in the network.

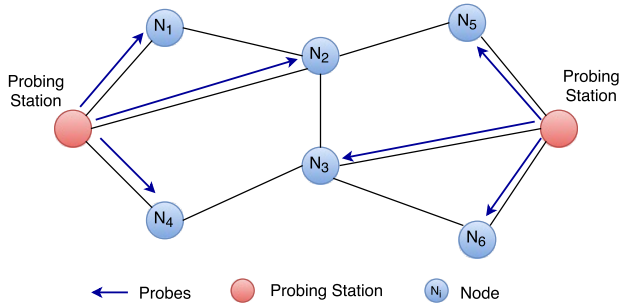


Fig. 6. Preplanned probing strategy example.

The process of selecting probes can be classified into two categories: 1) Preplanned, and 2) Adaptive[1]. Preplanned probing involves offline selection of a set of probes based on the network topology. These probes are periodically transmitted through the probing stations and the probe results are analyzed by a passive data mining phase to infer the network state. Figure 6 shows an example of a preplanned probing strategy. In this strategy, the probes are selected so that the probing results can uniquely identify all possible faults, and hence localize any single fault. In adaptive probing, the probing strategy is adapted according to the observed network state. So, instead of sending probes for locating all potential faults in the network, a minimal number of probes are sent initially for fault detection and then the probe set is adapted according to the observed network state to provide fault localization. Adaptive probing is explained in detail later in this section.

[1]Some authors use the term Active probing to refer to techniques that other authors call Adaptive probing. We prefer the term Adaptive probing here as it avoids confusing these techniques with the more general term Active monitoring.

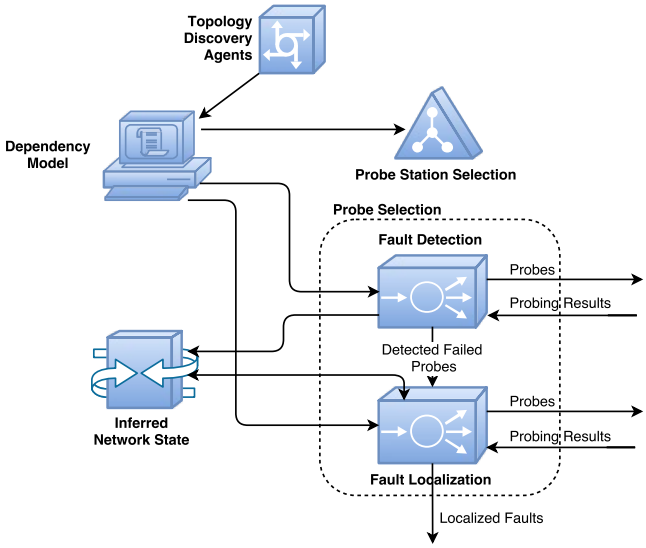


Fig. 7. System architecture for fault diagnosis using active monitoring.

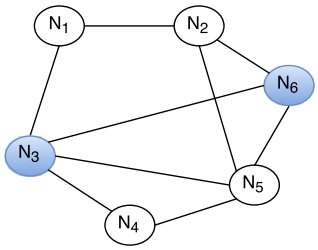
A. Architecture

Many system architectures that facilitate probing have been proposed in [31]–[33]. One of the architectures is shown in Figure 7 [32]. All the necessary tasks required to diagnose a network are addressed in the architecture and are described below.

- Probing Station Selection: The task of selecting locations in a network where probing stations should be placed. A minimum requirement of placement is the ability to probe the entire network from the selected probing stations.
- Probe Selection: The task of selecting the optimal set of probes after the probing stations have been selected. This task can be divided into two sub-tasks:
  - Fault Detection: The process of selecting the probes only to detect presence of failures in a network. These probes are few in number and they might not be able to exactly localize faults.
  - Fault Localization: Once a failure is detected, additional probes are selected that can provide maximum information about the suspected area of the network. The probing results are analyzed to localize the exact cause of failure.
- Topology Discovery: Exact network topology is required to select probing stations and probes. The network topology can be learned through commands such as ping and traceroute or by using any network discovery agents.
- Dependency Model: The dependency relationships between probes and nodes is modeled using the network topology. The model maintains the relationship between nodes and probes and is used in the process of probe

selection. The model is also required to analyze probe results.

- Inferred Network State: The fault detection process stores the intermediate diagnosis results as an inferred network state. The fault localization process refines the results by sending and analyzing additional probes.



(a)

	N_1	N_2	N_3	N_4	N_5	N_6	N_7
P_{6-1}	1	1	0	0	0	1	0
P_{6-2}	0	1	0	0	0	1	0
P_{6-3}	0	0	1	0	0	1	0
P_{6-4}	0	0	0	1	1	1	0
P_{6-5}	0	0	0	0	1	1	0
P_{3-1}	1	0	1	0	0	0	0
P_{3-2}	0	1	1	0	1	0	0
P_{3-4}	0	0	1	1	0	0	0
P_{3-5}	0	0	1	1	1	0	0

(b)

Fig. 8. (a) A network with N_3 and N_6 as probing station and (b) its dependency matrix.

B. Preplanned Probing

The first application of probing techniques for network fault diagnosis was proposed by Brodie *et al.* [31] who explained the problem of selecting probes for preplanned probing by using a dependency matrix. A dependency matrix D, is an r-by-n matrix, where r is the number of probes and n is the number of nodes in a network. The matrix element $D(i, j) = 1$ if probe P_i passes through node N_j, otherwise $D(i, j) = 0$. Assumption of static or shortest path routing is made because a deterministic dependency matrix is used. A simple network and its dependency matrix are shown in Figure 8, with N_3 and N_6 as probing stations, and each link of weight 1. Each row is a probe e.g., the probe P_{6-4} passes through $N_6 \rightarrow N_5 \rightarrow N_4$. The column N_7 represents no failure anywhere in the network and does not actually exist as a node. Each row in the matrix is a candidate probe to compute a minimal set of probes, such that any single node failure among the 6 nodes can be uniquely diagnosed.

Three algorithms are proposed for finding a minimal set of probes: an exponential time exhaustive search algorithm, and two approximation algorithms. The algorithms exploit the interactions among the paths traversed by the probes to compute a smallest set of probes which can diagnose all faults in the network. This fault detection approach is flexible because of the control that can be exercised in the process of probe selection. However, there are certain limitations with the use of this approach. The algorithms assume only single node failure. Also the effects of lost probes, spurious symptoms of node failures, and dynamic probe paths are not considered. But the results of this approach were promising which encouraged further research to look for more flexible solutions.

C. Adaptive Probing

The earlier work of fault diagnosis using active monitoring is based on a preplanned probing strategy. A major drawback with this strategy is the large amount of management traffic sent out over the network, a majority of which is wasteful as not all failures occur simultaneously. Some of the other drawbacks are: the delay involved in sending a large probe set, the difficulty involved in predicting an optimal probe set, and collecting the probe results. These drawbacks may lead to inaccurate results and increased localization time. The use of adaptive probing strategies, although defined in the earlier work [31], was explored and described in detail by Natu *et al.* [34]–[38]. In this approach, instead of sending probes to localize all the potential failures, a small set of probes is first sent to monitor the health of all the network components. These probes can detect the presence of failures, but are not comprehensive enough to identify the exact location of the failures. Based on the probe results, suspected areas of failure are identified, and additional probes are then used to localize the exact cause of the failure.

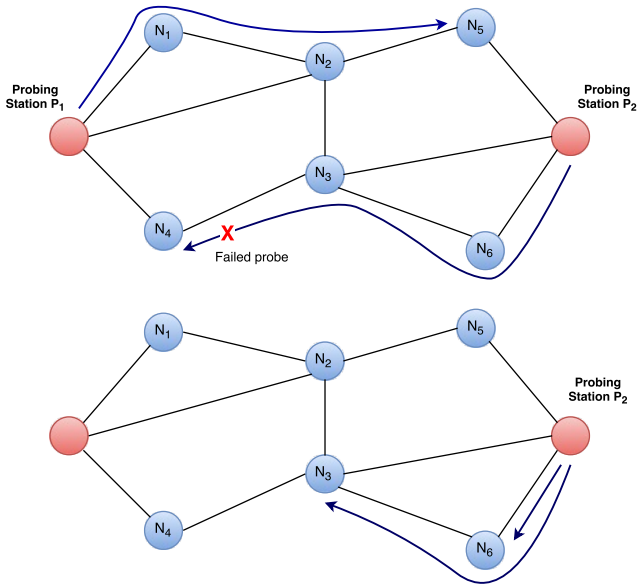


Fig. 9. Adaptive probing strategy for active monitoring.

An example of adaptive probing strategy is shown in Figure 9. In this example, on observing a failed probe from the probing station P_2 to the node N_4, additional probes are sent over the path for further diagnosis. A key goal of adaptive probing is to obtain accurate reliable estimates by using only a small number of probes. Various adaptive probing algorithms have been proposed by Natu considering a deterministic model of a network as well as for a non-deterministic model.

Another adaptive method based on the ideas presented in [34] is proposed by Lu *et al.* in [39], where they try to reduce the network traffic overhead induced by active probing schemes. The whole fault detection process is divided into a series of stages, wherein at each stage, a small set of probes is selected to check a few network nodes until all the nodes in the network have been covered. There are three main steps which are repeated in a loop: 1) a set of nodes is selected as the target set of nodes; 2) a set of probes, which has the maximum

detection ability, is selected to detect the target nodes; and 3) next set of undetected target nodes are selected until all the target nodes have been probed.

Many other problems associated with network fault diagnosis, such as probing station placement and probing station failures are addressed by Natu and solutions are proposed in [37]. An incorrect placement of probing stations can make some nodes unreachable and thus non-diagnosable in certain failure scenarios. Furthermore, placement of probing stations to monitor the non-probing stations fails to consider the probing station failures. A heuristic based approach to these problems has been proposed which incrementally selects nodes that provide suitable locations to instantiate probing stations. The algorithm considers the possibility of node failures as well as probing station failures. The problem of selecting minimum number of stations for fault localization is proved to be NP-Complete. Approximation algorithms are proposed based on the constraint of detecting *k* non-probing station failures and *l* probing station failures. Many other heuristic based adaptive algorithms for fault detection and fault localization are also proposed based on greedy approaches such as:

- Max Search: A probe that covers maximum number of suspected nodes is selected and added to the probe set.
- Min Search: A probe for each suspected node, which goes through a minimum number of other suspected nodes, is selected and added to the probe set.
- Binary Search: The probes are sent in binary search fashion till one failure on that path is located.

D. Probabilistic Approximations

Rish *et al.* [33], [40], [41] describe a cost efficient and adaptive diagnostic probing technique which uses an information-theoretic approach to select probes. The technique uses a fast online approach to infer the current network state via adaptive selection of only a small set of most-informative probes. The use of probabilistic inference yields an adaptive diagnostic solution of selecting probes that provide maximum information gain about the network state. The approach consists of three steps: 1) Initial probe set selection for fault detection: a small subset of probes that "cover" all nodes is preselected and run on a scheduled basis. So, when a fault occurs in a network, it can be immediately detected, 2) Adaptive probing for fault diagnosis: the most-informative next probe T_k is selected that maximizes the information gain $I(X, T_k|T_1, ...T_{k-1})$ given the previous probe observations $(T_1, ..., T_{k-1})$, and 3) Analysis of probe results (Inference): the probe results are integrated and analyzed to infer the current state of the network. The state is represented by a belief metric of Bayesian networks. Simple approximation algorithms for inference are also described which can be used when the exact probabilistic inference in BN is intractable. The incremental adaptive probing approach is well-suited for real-time monitoring and diagnosis, since probes are selected and sent as required in response to problems that actually occur. The results demonstrate an improvement of over 60% in reducing the number of probes required for diagnosis. The time required for localizing the faults also shows improvement when compared with non-adaptive probing techniques.

Significant improvement was demonstrated by Zheng *et al.* [42], in usage of probabilistic models for selecting probes using entropy approximation. The approximate approach proposes the use of loopy belief propagation infrastructure to compute approximations of marginal and conditional entropy. The entropy is used as the cost function to select a set of probes that provide maximum information, or minimum conditional entropy, about the network state.

The optimal probe set selection techniques that depend upon a probabilistic model have a high computational complexity even with approximation methods. Therefore, most of the recent work concentrates on reducing the time complexity associated with probabilistic inference. Various techniques have been proposed to reduce the computing time of a probe set, which exploit the conditional independence property of variables in Bayesian networks [43]–[45]. The information gain of a candidate probe T_j is non-increasing given the previous set of selected probes D_i during an iterative selection of probes. For instance, the information gain $I(T_j|D_i)$ in the i^{th} iteration will decrease or remain the same as compared to the information gain $I(T_j|D_{i-1})$ of $(i-1)^{th}$ iteration. Hence, for two candidate probes T_a and T_b, if in the previous iteration of evaluation $I(T_a|D_{i-1}) \geq I(T_b|D_{i-1})$ then in the current iteration also $I(T_a|D_i) \geq I(T_b|D_i)$. So, reevaluation of T_b and also all probes with information gain less than T_b can be avoided. Another property of conditional independence is also exploited to reduce the evaluations in each iteration. If probes T_i and T_j are conditionally independent, and T_i is the probe selected in the i^{th} iteration, then in the $(i+1)^{th}$ iteration of evaluation, the information gain of T_j will remain the same as in the i^{th} iteration.

E. Active Integrated Reasoning

A novel fault localization technique called Active Integrated fault Reasoning or AIR was proposed by Tang *et al.* [46]. This technique integrates the advantages of both passive and active monitoring into one framework, by incorporating actions into the traditional Symptom-Fault model. A Symptom-Fault-Action (SFA) model is shown in Figure 10, where f_i is a node for fault i, s_j is a node for symptom j that can be caused by one or more faults, and a_k is a node for action k that can be used to verify existence of symptoms. Actions can simply include commonly used network utilities like ping and traceroute. Table III contains a list of the notations used by the AIR framework.

The AIR framework is shown in Figure 11, and it consists of three modules: fault reasoning, fidelity evaluation, and action selection. The fault reasoning module passively analyzes the observed symptoms S_O and generates a fault hypothesis set $\mathcal{H}$. The fault hypothesis set $\mathcal{H}$ includes a set of hypotheses $(h_1, h_2, ..., h_l)$, where each hypothesis contains a set of faults that explains all the observed symptoms so far. The fault hypotheses set is then sent to the fidelity evaluation module to verify if the fidelity value of any hypothesis $h_i (h_i \in \mathcal{H})$ is satisfactory. If the correlated symptoms necessary to explain

TABLE III. NOTATIONS FOR AIR FRAMEWORK AND SFA MODEL

Notation	Definition
F	The set of all possible faults $\{f_1, f_2, ..., f_n\}$
S	The set of all possible symptoms $\{s_1, s_2, ..., s_m\}$
A	The set to denote the list of actions $\{a_1, a_2, ..., a_p\}$
f_i	A fault $f_i \in F$
s_j	A symptom s_j that can be caused by one or more faults in F
a_k	An action a_k to verify existence of one or more symptoms in S
S_O	The set of all observed symptoms so far
$\mathcal{H}$	The set of fault hypotheses
h_i	A set of faults that constitutes a possible hypothesis to explain S_O
S_N	The set of correlated but unobserved symptoms
S_V	A subset of S_N for symptoms whose existence is confirmed
S_U	A subset of S_N for symptoms whose existence is not confirmed

the fault hypothesis are observed then the fault reasoning process terminates. Otherwise, a list of most likely unobserved symptoms S_N that can contribute to the fidelity of h_i are sent to the action selection module. The action selection module then selects actions (probes) to verify symptoms that have occurred but not observed (i.e. lost) and accordingly adjusts the fidelity value of h_i. The actions return the test results along with a set of existing symptoms S_V and non-existing symptoms S_U. The set of symptoms $< S_U, S_V >$ are sent to the fidelity evaluation module and if the new fidelity value is satisfactory (i.e. above a threshold) then the reasoning process terminates; otherwise, the new symptom evidence $< S_O, S_U >$ is fed into the fault reasoning module to create a new hypothesis. This process is recursively invoked until a highly credible hypothesis is found.

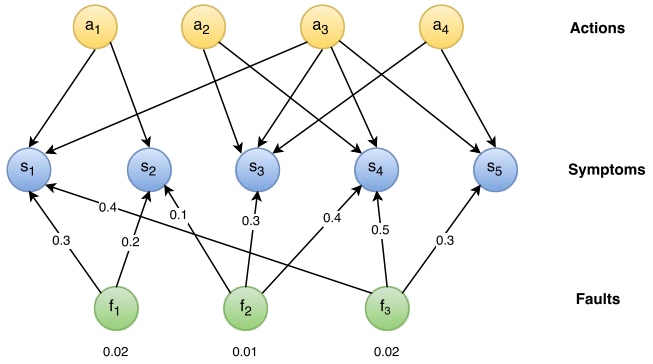


Fig. 10. Example of a Symptom-Fault-Action (SFA) model in AIR framework.

The AIR technique is the first to seamlessly integrate passive and active fault reasoning in order to reduce fault detection time as well as to improve accuracy of fault diagnosis. AIR is designed to minimize the intrusiveness of probing while enhancing the accuracy of fault hypothesis and optimizing the action selection process. The use of the action selection verification module decreases the impact of lost or spurious symptoms that may be caused by malfunctioning agents or devices.

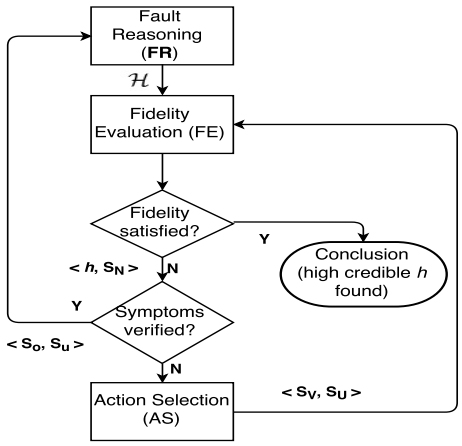


Fig. 11. AIR framework [46].

IV. TECHNIQUES FOR OVERLAY AND VIRTUAL NETWORKS

Overlay networks have emerged as a powerful and flexible platform for today's Internet. An example of an overlay network is shown in Figure 12. Overlay networks have provided ways to improve the Internet's routing by guaranteeing quality of service. But the flexible characteristics of overlay networks bring new challenges for the fault diagnosis process. These challenges include inaccessible substrate network fault information, incomplete and inaccurate network observations, dynamic symptom-fault causal relationships, and multi-layer complexity. To tackle these challenges, various overlay fault diagnosis techniques have been proposed in the past few years [47]–[55]. The novel AIR framework [46] described in Section III-E has been extended by Tang *et al.*, and a new Overlay-AIR framework has been proposed in [47]. Some of the other approaches are based on end-user observations [51]–[53], multi-layer diagnosis [49], [50], [54], and efficient placement of overlay nodes [55]. Instead of probing the networks or using alarms from monitoring agents for analysing the state of the network, some of the approaches just use the end-user observations as symptoms to perform fault diagnosis. The multi-layer monitoring approaches focus on selecting an optimal set of native (physical) and overlay links to monitor the entire overlay network. The overlay node placement approaches look to place the overlay nodes in an efficient manner in order to reduce the fault diagnosis time.

A. O-AIR

Similar to the previous AIR framework shown in Figure 11, the overlay fault diagnosis framework O-AIR [47] integrates the use of active monitoring with passive fault reasoning. This approach assumes hosting of overlay agents on each overlay node. An agent monitors the links and collects information regarding the fault probability of the network components, and the conditional probabilities of observing the corresponding overlay symptoms. The collected information is used to construct a centralized query-based knowledge system called

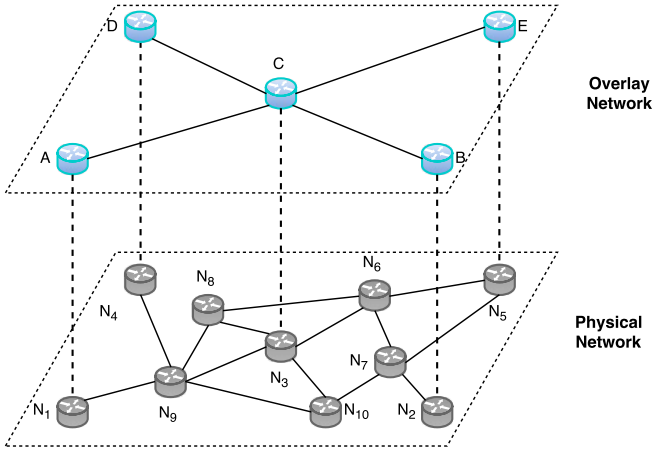


Fig. 12. Example overlay network to show a mapping between the overlay network layer and the physical network layer.

Overlay Network Profile (ONP). The knowledge-based system is then used to build an Overlay Symptom-Fault model by correlating end-to-end symptoms and faults. However, the observed symptoms may not be sufficient to identify a network fault, so a set of monitoring actions are taken to determine the root cause. The actions are then integrated with symptoms to build an Overlay Symptom-Fault-Action (O-SFA) model to verify faults. Table IV describes the notations used for the O-SFA model.

TABLE IV. NOTATIONS FOR OVERLAY SFA MODEL

Notation	Definition
L	The set of overlay links
l_{ij}	An overlay link between overlay nodes i and j
P_{ij}	An overlay path between overlay nodes i and j
N_{ij}	The underlying network between overlay nodes i and j
O	The set of overlay path symptoms $\{O_1, O_2, ..., O_q\}$
O_{ij}	An overlay symptom between overlay nodes i and j
S	The set of overlay link symptoms $\{S_1, S_2, ..., S_m\}$
F	The set of faults in overlay nodes $\{F_1, F_2, ..., F_n\}$
A	The set of overlay actions $\{a_1, a_2, ..., a_p\}$

The overlay fault diagnosis (O-AIR) process includes three functional modules: symptom mining, fault reasoning, and action selection. The symptom mining module uses observed overlay symptoms to dynamically create an O-SFA model based on an ONP. The fault reasoning module takes O-SFA as input and returns a fault hypothesis *h* as output. The fault hypothesis *h* contains a set of faulty components that explains all the observed symptoms. The corresponding overlay actions are selected to verify the hypothesis. If all faults in *h* are verifiable, the overlay fault diagnosis process terminates and faults are reported. Otherwise, the action results are used to update previously constructed O-SFA by removing unexplained overlay symptoms, as well as irrelevant components, and adding new symptoms and related components. This process repeats until a verifiable hypothesis is found.

Figure 13 shows the overlay network in correspondence to the network shown in Figure 12. The overlay network has two overlay paths: P_{DE} between the overlay nodes D and E; and P_{AB} between the overlay nodes A and B. The overlay path P_{DE} consists of two overlay links l_{DC} and l_{CE}. Similarly, the overlay path P_{AB} consists of two overlay links l_{AC} and l_{CB}. N_{AC} represents the underlying network components between overlay nodes A and C. Similarly, N_{CE}, N_{DC} and N_{CB} represent their respective underlying network components. The Overlay-SFA model of the overlay network is shown in Figure 14. The O-SFA model has nodes: O_i for every overlay path related symptom; S_j for every overlay link related symptom; F_k to represent a fault in the overlay component or its underlying network; and a_l is an action node to verify the faults.

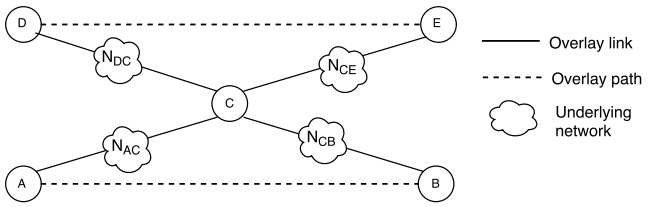


Fig. 13. An overlay network showing overlay paths and overlay links.

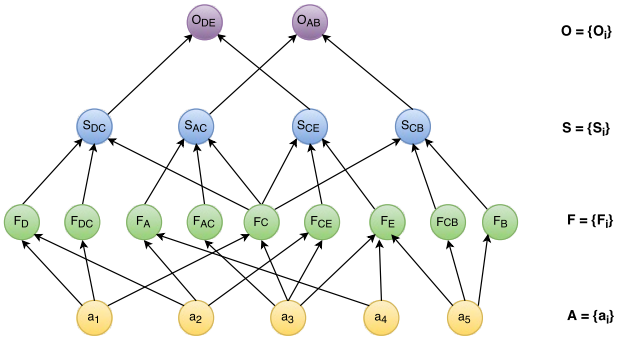


Fig. 14. Overlay Symptom-Fault-Action model for overlay network in Figure 13.

B. End-user observations

Most of the fault diagnosis solutions either require massive information collected passively by using the network monitoring agents, or by actively monitoring the network using probes. Both these approaches have their limitations. Passive monitoring approaches require extensive network knowledge regarding posterior or aprior fault probabilities of the underlying network components which may limit their effectiveness in real practice. Whereas the additional synthetic traffic generated for probing the network may burden the underlying network infrastructure. So, various solutions have been proposed based on the usage of only the end-user observations as negative symptoms to develop hypotheses of potential faulty network components [51]–[53].

An overlay fault diagnosis framework called DigOver has been proposed by Tang *et al.* in [51]. In this framework, the end-user observations are associated with the related overlay

path, and represented by the set of components along the overlay path. The evidences are used as negative symptoms to develop hypotheses of potential faulty network components. Each hypothesis is evaluated using the Dempster-Shafer theory [56] to quantify the fault probability along with the evaluation uncertainty. The use of Dempster-Shafer theory (DST) for fault diagnosis makes this framework different than other probabilistic fault diagnosis frameworks which use Bayesian reasoning. Adequate a-priori information regarding the fault probabilities collected by active monitoring or network monitoring agents is required for Bayesian reasoning. However, the DigOver framework requires no such information and assumes little or no knowledge about the network topology. In particular, the faults are reasoned at real-time based on the user observations, where the end-user views are used as evidence to compute a belief for most plausible root causes. However, the reasoning of all the end-user evidences using DST can be time consuming, especially in the case of complex mapping of substrate and overlay nodes. Improvements to the DigOver framework are proposed by Wang *et al.* in their DiaEO framework [52], wherein the end-user observations are screened and broken down into subsets before the evidence reasoning analysis, in order to improve the reasoning time.

Another fault diagnosis approach based on end-user observations has been proposed by Gillani *et al.* in [53]. The end-users share their network performance information which is considered as negative symptoms, and is used to localize performance anomalies and determine the packet loss contribution of each network component. The problem is formulated as a constraint-satisfaction problem, wherein the model localizes packet loss in large-scale networks accurately without requiring information collected from active probing or network monitoring agents. This model exploits the correlation in the end-user negative symptoms due to shared symptoms and path segments, in conjunction it also identifies network loss invariants, which are all then encoded as constraints and solved using satisfiability modulo theories (SMT). Advanced SMT solvers are available that can solve large number of constraints and variables which makes this approach scalable to a large number of symptoms and network components. The model used in this approach develops hypotheses of potential root causes in the network which satisfy the reported end-user observations and network loss invariants. However, this approach may inherit the problem of insufficient observations by end-users which is solved by adding more observations through selectively asking appropriate end-users to generate network probes. These probes are targeted to only the potential root cause network components which may help to minimize the probing traffic in the network. Also, the problem of spurious observations by malicious end-users is addressed by performing an in-depth analysis of each end-user to identify any malicious users and remove them from the system, and also the SMT solvers provide features to remove spurious symptoms from the evidence set.

C. Multi-layer fault diagnosis in Overlay Networks

Monitoring all links in a multi-layer overlay network is important; however, a full monitoring operation in a large network can constitute a significant overhead in terms of bandwidth and processing. A flexible technique has been proposed by Demirci [49] that monitors end-to-end overlay measurements along with certain native physical links. A suitable combination of native layers and overlay links is then used to infer the state of the overlay network. This approach is named multi-layer monitoring. Along similar lines, Chen *et al.* [50] has proposed a linear algebraic approach to find a minimal basis set of just the overlay links to monitor and infer the loss rates of the remaining links. This multi-layer monitoring strategy is a general approach to offer significant flexibility in monitoring overlay networks. The main idea is to minimize monitoring cost by monitoring the optimal mix of overlay and native links. The problem of selecting the mix of native and overlay links is formulated as a linear programming problem. Constraints such as cost to monitor each link (overlay and native), and overlapping relationship between native and overlay links are represented as linear mathematical relationships. The constraints are used to determine the optimal mix of native and overlay links using linear programming.

Most of the passive fault diagnosis approaches developed for overlay networks use the network model which represents the relationship between substrate network, virtual network, and symptoms. Then the fault diagnosis problem is formulated as a probabilistic inference problem to identify the faults at the substrate layer given the observed symptoms as evidence. However, these methods ignore the existence of independent faults at the virtual layer. Yan *et al.* in [54] provides a clear distinction between the three types of faults that can occur in a virtual network environment. The faults at the substrate layer are called substrate faults (SFs), which can lead to the failure of corresponding virtual nodes and links, which are called correlative virtual faults (CVFs). The faults that occur because of software errors in the virtual layers are called independent virtual faults (IVFs). Yan *et al.* have described a method to identify all virtual faults (CVFs and IVFs) based on the symptom fault relationship and the observed symptoms. The substrate faults (SFs) are located based on the mapping relationship between CVFs and SFs. The layer-by-layer fault diagnostic approach can solve the problem which is caused by the inaccessible substrate network information. A filtering mechanism is used to distinguish CVFs from IVFs, which improves the quality of fault diagnosis and also help in efficient recovery in case of failure.

D. Efficient Overlay Network Placement

Along with the multi-layer monitoring approach, Demirci has also proposed the idea of efficient node placement that can accelerate fault diagnosis and optimize overlay monitoring [55]. A new term diagnosability is defined as the overlay network's property to allow accurate diagnosis of potential faults with minimum overhead. A large part of diagnosability depends on the topology of the substrate network. However, the placement of overlay nodes can also play a significant role in determining an overlay's diagnosability. Figure 15 shows an example of different placements of three-node overlay on a substrate network. For overlay ABC, a failure of substrate

node X could be observed on path BC through end-to-end monitoring of that path. However, the fault cannot be pinned down to a single component because there are two substrate links and a substrate node that can be the source of the fault. On the other hand, with the overlay placement of A'BC, the failure of node X could be observed on all three end-to-end paths and will be uniquely identified. Usually a trade-off exists between diagnosability and the amount of stress on substrate components, and higher diagnosability comes at the expense of stressing out the substrate network. The overlay placement approach is based on the following ideas:

1) Increase sharing between overlay paths on average: The more end-to-end paths that share a substrate component, the higher the number of symptoms that can observe a fault at that component. This is helpful in localizing the fault.
2) Limit the maximum stress on a component: A substrate node/link shared by too many overlay nodes/links has more risk of congestion and resource exhaustion. An upper limit on stress is required for substrate components.
3) Limit the number of overlay paths that overlap almost completely: Two complete overlapping end-to-end paths observing a fault provide no more valuable information for diagnosis than one.

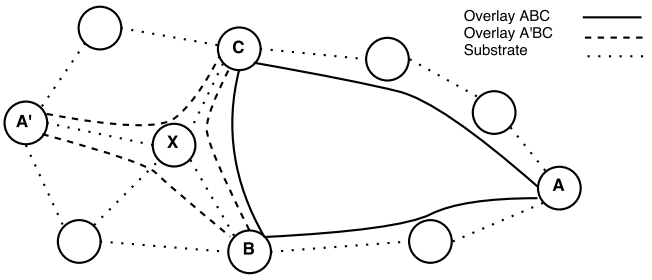


Fig. 15. Sample topology with two possible placements of an overlay network, ABC and A'BC.

The three ideas are combined into a single metric q that represents suitability of placement of overlay nodes. The goal of the placement algorithm is to maximize the placement quality metric q. The algorithm starts with a random placement of the overlay onto the substrate, and gradually improves q by making small adjustments. At each step, the overlay path whose placement may reduce q is changed by moving its end nodes to other substrate nodes. This approach leverages the flexibility in overlay design to maximize the diagnosability of the overlay network.

V. Decentralized Probabilistic Management Techniques

Most probabilistic network management solutions typically follow a paradigm in which a few dedicated network nodes autonomously execute management tasks in order to manage a set of network nodes. The dedicated nodes then communicate with a centralized system to report failures and anomalies in the network. Yu *et al.* [57] have identified one of the main risks of centralized approaches to be increased link load at the central collection node. Further, the authors have also elicited that coordination between neighbors for fault detection can reduce communication overhead and detect network faults with higher certainty. Different strategies for fault detection and fault localization have been investigated by Zhuang *et al.* [58], in which nodes can either independently or collaboratively decide the status of a neighboring node. Their results show a decrease in detection time of fault localization but an increase in control overhead with the use of algorithms that employ neighbor-collaboration and information-sharing among nodes. Figure 16 shows a comparison between centralized and decentralized management approaches. The network nodes in centralized management approach communicate with a centralized system where all the analysis is done. On the other hand, in decentralized management approach, each network node is embedded with a management process. The network nodes communicate with each other and the final analysis is reported to the network administrator.

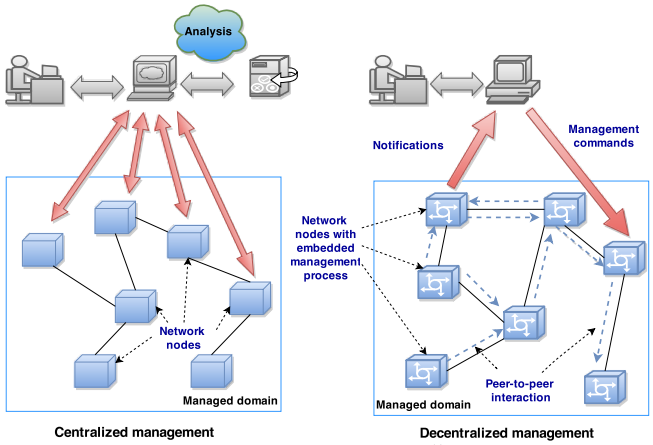


Fig. 16. Centralized vs decentralized management.

More recently, Prieto *et al.* [59], [60] and Steinert [61], [62] have also proposed decentralized probabilistic fault management techniques. Prieto described a network monitoring protocol that provides a management station with a continuous estimate of a global metric for given performance objectives [63]. An example of a global metric in the context of the Internet could be a list of subscribers with the longest end-to-end delay. The global aggregate metric is computed from the local metrics of nodes across a network system. Key parts of this model are the distributed monitoring process and the nodes communication using spanning-tree based approach. In order to incrementally compute a global metric, each node holds information about its children in the tree structure and push-based updates are used to send information to management stations along the spanning tree.

Another distributed and adaptive algorithm for fault detection and fault localization has been proposed by Steinert [62]. This probing based approach is based on the requirements of

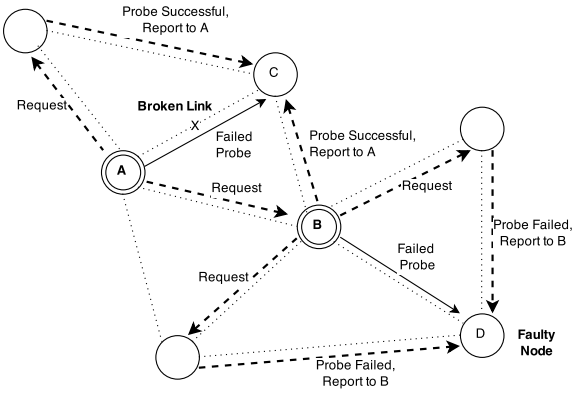


Fig. 17. Example of distributed fault localization algorithm.

reducing the manual efforts of configuring the algorithms to run on networks of different sizes and types, and to improve bandwidth usage compared to conventional probing fault localization techniques. The algorithm autonomously adapts to local network conditions and parameters of network resources to continuously monitor link disturbances. The data collected by monitoring neighbor nodes is processed locally to allow for faster detection of network failures. The aim of this approach is to observe network behavior and look for patterns that deviate from normal observations, rather than finding any particular type of fault. For each link in the network, parameter estimation of the observed response delays, drop rate, etc. is performed to model their probability distribution. Based on the probability distribution, the expected response delay and drop rate are computed. Probes are sent in certain intervals to test the availability of adjacent nodes. The intervals autonomously adapt to the observed behavior of individual connections. If a probe on a connection fails, a symptom is marked and the fault-localization process is initiated. The fault localization process involves collaboration between nodes in order to localize the origin of the abnormal network behavior. The algorithm is designed to distinguish node faults from link faults. A distributed fault localization algorithm is shown in Figure 17. In this algorithm, when a probe test from A to C fails, A initiates the fault localization process by asking neighbors of C to test their connection with C and report back to A. In this case, the neighbors of C were successfully able to probe C, so a broken link was concluded and reported by A. Similarly, when a probe test from B to D fails, B initiates the fault localization process, by asking neighbors of D to test their connection with D and report back to B. However, in this case none of the neighbors receive a probe response from D, so a faulty node is concluded and reported by B.

"NATO - Not All at Once" is another probabilistic algorithm proposed by Cohen *et al.* [64]. The algorithm estimates the ideal size of a group of nodes that should report their collaborative group metrics. The bandwidth and resources are efficiently utilized, and excessive traffic is avoided at the ingress channel of the management station by not having each node report its local metric independently. Aggregating local metrics of nodes to compute a group metric can significantly reduce node-collaboration traffic at the expense of marginal computational load on each node. Examples of such metrics are number of nodes with high packet rate, emerging congestion, etc. This approach is an alternative to Prieto's metric aggregation approach described previously. NATO! is an example of a family of algorithms that implement grouping of nodes to avoid congestion at the ingress channel of the management station. The approach aims to be scalable to networks of any size, and provide a flexibility to control the trade-off between accuracy, timeliness, and overhead in the network.

VI. Temporal Correlation Techniques

Some of the newer fault localization techniques have explored temporal and dynamic aspects of a network model. Many existing fault localization techniques assume a static model of a network, i.e. the network state does not change during the diagnostic process. Such an assumption may not be valid in dynamically changing networks. For instance, in a mobile ad-hoc network, mobility of nodes results in a high degree of non-determinism in the network model because of dynamically changing topology and lack of accurate and timely information [65]. While "hard" failures on network components are relatively rare, "soft" failures such as performance degradation, e.g. response time exceeding a certain threshold, may happen more frequently in a dynamic network. So, there is a need to have more sophisticated models that can account for network dynamics and provide more accurate diagnosis.

An approach to use a temporal model is to consider the general framework of Dynamic Bayesian Networks (DBNs) for representing symptom-fault relationship of a network. This approach can handle a wide range of dynamic systems, but suffers from the same computational complexity problem as the basic BN framework (i.e., inference in DBNs is NP-hard) [40], [66]. Steinert *et al.* [67] have also developed an approach that considers a spatio-temporal correlation between events that are spread across multiple layers in a multi-layer virtual network. This fully distributed approach is based on the assumption that events in one layer may arise from a series of events in lower layers. Natu *et al.* [65] have also developed an approach that considers a temporal correlation between dependency models of a network over a period of time and tries to capture the changing dependencies. The dependency model used in the approach is in the form of a symptom-fault relationship.

A. Dynamic Bayesian Network

One usage of dynamic Bayesian networks to model situations where the state of a network component changes over time has been shown by Rish *et al.* in [40] and by Li *et al.* in [66]. Dynamic BN model extends the static BN model by introducing the notion of time-slices and specifying transition probabilities $P(X^t|X^{t-1})$ between these slices, where $X^t = (x_1^t, \ldots, x_n^t)$ is the vector of node states at time slice t. However, the key thing here is that the DBN model is time-invariant; the term "Dynamic" represents a dynamic system,

not a system that changes over time. DBN uses the Markov property that the future network state is independent of its past states given the present state. The intra-slice dependencies are described by a static BN, and inter-slice dependencies are described by the transition probabilities. A DBN model describes a stationary stochastic process (i.e., the transition probabilities and intra-slice dependencies do not change with time). Figure 18(a) shows a static BN, which is extended by a dynamic BN in Figure 18(b), by adding inter-slice dependencies that encode the transition probabilities.

Once a dynamic BN is specified, any standard BN inference algorithm can be applied to compute $P(X^t|X^{t-1}, Y^t)$, given the prior distribution $P(X^{t-1})$ and the observations Y^t at time-slice t, where $Y \subseteq T$ is a subset of symptoms observed at time t. The computational complexity of exact inference in DBN is high, but some efficient approximation algorithms are available in [68]–[70]. The most recent work of using DBN to model a dynamic network describes the use of clusters for reducing the time complexity of inference. Li *et al.* [66] have used DBN model to represent a large-scale IP network. In their work they have divided the fault nodes into independent clusters, to reduce the diagnosis time and make inference feasible. Clustering of fault nodes is valid because of the ubiquitous assumption of conditional independence among fault nodes to model symptom-fault relationship. However, the use of clusters reduces the accuracy of fault diagnosis.

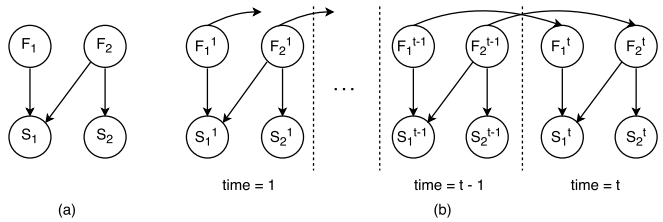


Fig. 18. (a) Bayesian network, and (b) Dynamic Bayesian network.

B. Spatio-Temporal Event Correlation for Multi-layer Virtual Network

Steinert *et al.* [67] have considered spatio-temporal correlation among network events in multi-layer virtual networks. In this work, the problems addressed are related to asynchronous clocks among stacked overlay networks and network equipment, which complicates the problem of event correlation. As event correlation is highly dependent on the temporal order in which the events appear, the collection of events from monitoring nodes requires some kind of synchronization. In wired networks, nodes are often synchronized through the Network Time Protocol (NTP). However, in wireless ad-hoc networks with some degree of heterogeneity, synchronization can be a problem. Therefore, a protocol to compensate for asynchronous timestamps can be useful. A cross-layer protocol design is proposed by Steinert *et al.* [67] that operates in a fully distributed manner and takes asynchronous timestamps into account.

In a multi-layered virtualized network, a virtual overlay can be assumed to have inherited the characteristics from the underlay layers in the hierarchical stack. So, a failure in an overlay is likely to be caused by events in the underlay. An example of event correlation in a multi-layer network is shown in Figure 19. The events can be related to malfunctioning equipment, malicious activity, reconfiguration, performance degradation, and changes in load patterns in shared resources. In order to accurately localize the root cause of events, the information from the underlying layers needs to be considered, in addition to observed relevant events of the layer in which the alarm was first reported.

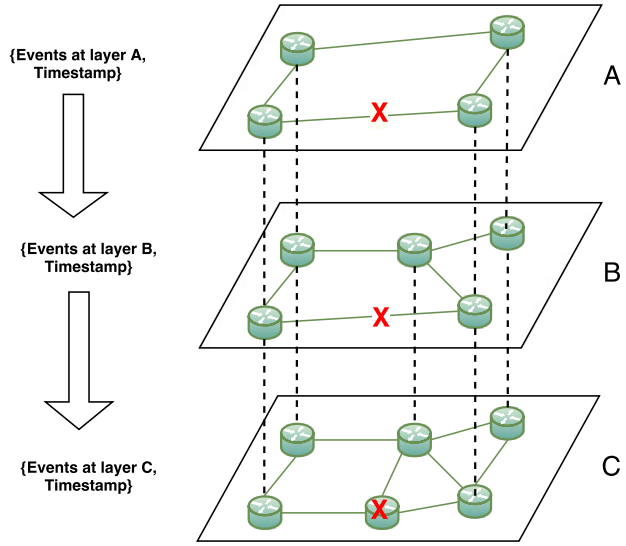


Fig. 19. An example showing spatio-temporal event correlation.

The algorithms used for fault detection and localization in a particular virtual layer are from the authors' previous work [60], [61] and are described in Section V. For correlating the events across multiple layers, the detection time t_f is recorded, of an event $E^{(l)}$ in a layer l with topology $T^{(l)}$, and the time t_w is also recorded when everything was known to be functional in the layer. All events that are relevant in accordance with the topology $T^{(l)}$ and the timestamps $[t_w, t_f]$ are then aggregated and disseminated to the underlay. The set of aggregated events constitute a root cause at level l which is also disseminated to the underlay. As the underlay $l-1$ knows what the topology $T^{(l)}$ corresponds to in terms of its own topology $T^{(l-1)}$, the events that have occurred within the time interval $[t_w, t_f]$ are searched in $T^{(l-1)}$. Again, the set of reported events within the time interval constitutes a root cause at level $l-1$, and is disseminated to the next underlay layer. The dissemination of events to the underlay is terminated when no events in the current underlay are found, or if the protocol has reached the physical layer. Aggregated events are temporally ordered and reported for further analysis.

C. Temporal Correlations in Network Topology

An architecture has been presented by Natu *et al.* [65] that considers temporal correlation in the network topologies over a period of time. A dynamic dependency model is introduced to capture changing dependencies of the network. A fault hypotheses search space is used to store different hypotheses generated over a period of time. A timestamp is assigned to each dependency model and also to each reported symptom. Timestamp information is used to calculate the relevancy of a dependency model used for processing a particular symptom. The fault correlation algorithm processes the observed symptoms incrementally as they arrive, and a fault hypothesis set is generated. The hypothesis set is modified after receiving a change in topology information. A set of hypotheses is reported as possible causes of failure symptoms and are ranked based on the degree of confidence. The ranks (beliefs) are computed by using the IHU algorithm [71] and also according to the temporal information present in the dependency model and reported symptoms. The temporal information is weighted based on the difference in timestamp of symptoms and dependency models being considered. The smaller the difference, the higher is the relevance. As the individual symptom-fault dependencies are stored in the hypotheses search space, the belief computation is not required to be done on arrival of each new symptom. Instead, to improve the performance, the beliefs are computed after collecting certain number of symptoms or after a period of time has elapsed.

VII. Learning Techniques

All the techniques described in the previous sections require knowledge of a dependency model of the network. But for a real-world complex network, obtaining a full dependency model is challenging. So, fault localization based on learning techniques can be useful as it would not require complete dependency model of a network. A network management system that monitors complex networks can generate large log files. The empirical data in these log files can be diagnosed using statistical and machine learning techniques. Statistical techniques can summarize and interpret empirical data using approaches such as correlation, histogram comparison and dimension-reduction techniques. The statistical techniques are data-centric and require little expert knowledge or detailed model of the system. Machine learning is a scientific discipline that is concerned with the design and development of algorithms that can learn based on a sample of data. Machine learning techniques rely on training and cross-validating which involves partitioning a sample of data into subsets, performing the analysis on one subset called the training set, and validating the analysis on the other subset called the validation set. Cross-validation helps in providing an estimate of accuracy of the model.

A. Statistical Techniques

Most of the passive monitoring techniques discussed earlier rely on statistical techniques, such as correlation and regression, in conjunction with deep knowledge of the system's behavior to diagnose problems[2]. By contrast, the statistical techniques discussed in this section make fewer assumptions about the system's behavior. Statistical techniques can be classified as parametric and non-parametric techniques. Parametric techniques assume that the data is drawn from a known distribution and makes inferences about the parameters of the distribution, e.g., normal distribution. Non-parametric techniques do not rely on data belonging to a particular distribution but rather estimate the underlying distribution. Non-parametric methods make fewer assumptions than parametric methods, which make them more robust and give them wider applicability. The difference between parametric model and non-parametric model is that the former has a fixed number of parameters, while the latter grows the number of parameters with the amount of training data.

1) Parametric techniques: Parametric techniques assume that data is drawn from a known distribution. Normal distributions are commonly used for anomaly detection and diagnosis because of their tractability, and also because normality can sometimes be justified by the central-limit theorem which explains why many distributions tend to be close to the normal distribution. These techniques typically detect anomalous behavior by identifying significant deviations from the mean of performance counters, which they assume to follow a normal distribution.

In statistical analysis, change point detection techniques [72] can be used to identify the times, when the probability distribution of a stochastic process or time series changes. In general, the problem concerns both detecting whether or not a change has occurred, or whether several changes might have occurred, and identifying the times of any such changes. Agarwal *et al.* [73] use change point detection technique, to detect changes in time series of performance metrics of individual nodes in the system and then identify a failure condition based on its signature. Abrupt changes in system behavior are detected by monitoring changes to the mean value of performance counters over consecutive windows of time. The change point detection technique works better than the static or dynamic thresholds set for individual metrics, as they require an in-depth knowledge of the system architecture and design. Change point detection techniques operate on a window of samples rather than individual sample points to find change points. In this way, only statistically important events are generated, and the false positives and false negatives are considerably reduced. However, this technique does not scale well with large number of nodes and metrics.

Kandula *et al.* [74] have developed a system NetMedic that can diagnose faults in enterprise networks. The goal is to design and implement a comprehensive system that can diagnose and provide solutions to a wide variety of faults in the system. The propagating faults are diagnosed by analyzing dependencies between nodes, and correlating the state perturbations to localize them. Figure 20 shows the workflow of the NetMedic system. The three main modules of the workflow

[2]The literature on learning techniques generally uses the term "problem" to refer to a fault. To be consistent with conventions in the mainstream fault management community, we use the term "fault" in this paper, including this section.

are to capture the state of network components, generate the dependency graph, and diagnose based on component states and the dependency graph. During diagnosis, the first step is to determine the extent to which various components and variables are statistically abnormal. Then the weight for edges in the dependency graph are computed. The weights are used to compute path weights and produce a ranked list of likely culprits. The state for each system component is represented as a vector that indicates whether each metric was anomalous or normal, by assuming that each metric obeys a normal distribution and flagging anomalies based on deviation from the mean. If two components which depend on each other are anomalous, the system searches for time periods where the source component's state is similar to its current state, and searches for destination states that have experienced significant changes in the same period. The destination states are the likely culprits. The system is based on an intuitive technique that uses the joint behavior of two components in the past to estimate the likelihood of them impacting one another in the present.

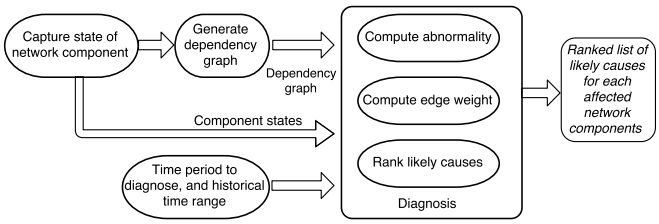


Fig. 20. The workflow of NetMedic.

Kavulya *et al.* [75] have discussed the challenges associated with discovering and diagnosing chronics, i.e., recurrent faults that fly under the radar and do not trigger alarm thresholds. A statistical approach is used for diagnosis of never-before seen chronics that does not rely on models of system behavior, or historical data to localize problems. The approach uses a scalable Bayesian distribution learner coupled with an information theoretic measure of distance, to identify the sets of attributes that best distinguish failed requests from successful requests. Draco, a statistical fault diagnosis tool that implements this approach, is presented in [76]. Draco performs statistical diagnosis of faults in large Voice-over-IP (VoIP) systems by comparing differences in the distributions of attributes, such as hostnames and customer IP addresses, in successful and failed calls. Draco assumes that these attributes are drawn from a Beta distribution and localizes faults by identifying attributes that are most correlated with failed calls. Due to its statistical nature, Draco does not require extensive domain expertise in the form of rules or models, thus making it easily portable to multiple applications. By comparing successes and failures over the same window of time, Draco avoids the need for separate learning passes, and can thus diagnose faults that have never been seen before.

2) Non-Parametric techniques: Non-parametric techniques assume that data is drawn from an unknown distribution. Non-parametric techniques estimate the underlying data distribution using kernel density estimation, or make generalizations about the populations from which the samples were drawn, e.g., using correlation. The term non-parametric is not meant to imply that such models completely lack parameters, but that the number and nature of the parameters are flexible and not fixed in advance.

Correlation-based techniques can be used to analyze data and automatically discover causal relationships among pairs of metrics [24], [77]–[79]. Perturbation in the learned correlations may indicate faults. Several solutions have been proposed for event correlation and alarm filtering with applications in network management. The codebook approach by Yemini *et al.* [24] discussed earlier in Section II correlates each known fault with a specific set of events, which is essentially used as the signature in fault determination. Gruschke's use of dependency graphs to represent correlation knowledge in [78], and alarm filtering approach by Mas *et al.* discussed in [80], are correlation approaches that are essentially also used to determine fault signatures. These approaches assume that the prior knowledge of system dependency is available. However, as discussed earlier, it is difficult to extract system dependencies or define fault signatures in complex network systems.

Mahimkar *et al.* [81] have developed a tool Giza, which uses correlation techniques for characterizing and troubleshooting faults and performance impairments in IPTV distribution networks. The tool uses hierarchical heavy hitter detection to identify the spatial locations where the symptom events are dominant in the network. The hierarchy for spatial locations is created using the IPTV multicast tree structure. This greatly reduces the amount of data for subsequent processing. Then statistical event correlation analysis at heavy hitter locations are applied to identify those event-series that are strongly correlated with the heavy hitter symptom. The list of strongly correlated event-series includes both potential root causes and impacts of the symptom events. Then, the statistical lag correlation and l^1 norm minimization techniques are applied to discover the causal dependencies between events. A causal dependency graph for each symptom event-series is constructed. The graph generated by Giza is sparse, and helps network operators to effectively and automatically diagnose symptom events. The discovery process requires minimal domain knowledge. Figure 21 shows the architecture of the tool Giza.

The usage of statistical techniques for diagnosis have certain limitations. The diagnosis techniques need to rely on well established statistical theories to ground their algorithms, and to prove their results are statistically significant. However, to build a statistical profile of behavior, sufficient data samples are required, and valid assumptions on data distribution need to be considered. Incorrectly assuming that the data is drawn from a normal distribution can lead to a high error rate. Also, the statistical techniques do not integrate the semantic knowledge about semantic behavior of the system, so certain difficulties can be experienced to distinguish legitimate changes in behavior, such as workload changes from performance problems.

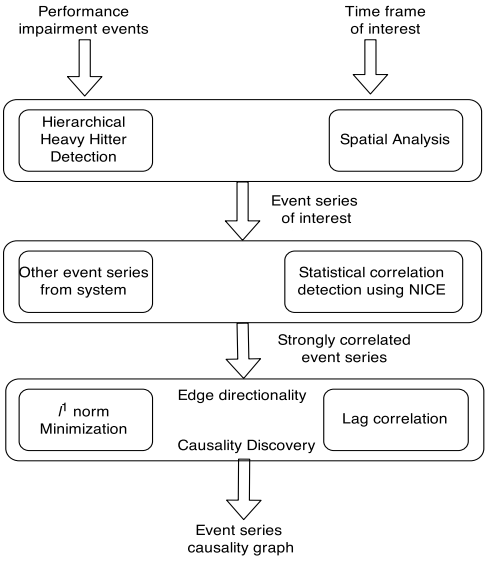


Fig. 21. The architecture of Giza.

B. Machine learning techniques

Machine learning techniques borrow heavily from statistical techniques, e.g., data distributions and probability theory. Machine learning techniques rely on learning algorithms, to study the correlation of network events from a limited set of data, called the training data. The training data consists of observed network events and corresponding network faults. This information is obtained from historical data recorded in logs and diagnosis reports. The training data is used to develop a hypothesis about the relationships of network events. The learned hypothesis is then applied to new observations to predict the state of the network. During the process of developing a hypothesis and predicting network outcome, the prior probability distribution of faults or the conditional probabilities of observing symptoms are not required. Diagnosis algorithms that rely on machine learning can be categorized into two broad categories namely: 1) Supervised learning which infer a function that best classifies successful and failed states from labeled data, and 2) Unsupervised learning which identifies patterns in unlabeled data typically through clustering.

1) Supervised learning: Supervised learning algorithms infer a function from labeled training data. The training data consist of a set of training examples, where each example is a pair consisting of an input object, typically a vector, and a desired output value. A supervised learning algorithm analyzes the training data and produces an inferred function, which can be used for mapping new examples. Symptom attribution approaches localize faults by identifying network components that are highly correlated with failed states. This can allow network operators to filter thousands of symptoms in their system, and narrow down the handful of symptoms that can yield insight to the cause of the fault and its location. This approach guides the network operators to perform more detailed root-cause analysis. After the root-cause analysis, the output of symptom attribution can be annotated with the root-cause and the database of known failures can be built. This database can be used by the knowledge based techniques described in Section II.

Chen *et al.* have developed frameworks Pinpoint [82] and MinEntropy [83] that use data clustering and decision trees, respectively, to diagnose faults in large network systems. The frameworks do root cause analysis, and do not require application-level knowledge of the systems being monitored. The frameworks are developed to automatically detect faults in large and dynamic systems. In these approaches, the real client requests are tagged and traced as they travel through the system. Each request is recorded as believed success or failure, and the set of components used to service it.

In Pinpoint approach, the data clustering and statistical techniques are performed to correlate the believed failures and successes of the tagged requests to determine which components are most likely to be at fault. In MinEntropy approach, decision trees are trained on the traces of requests from time periods in which failures are present. Paths through the tree are ranked according to their degree of correlation with failure, and nodes are merged according to the observed partial order of system components. Tracing real requests through the system enables them to support fault determination in dynamic systems where using dependency models is not possible. Tracing also allows to distinguish between multiple instances of what would be a single logical component in a dependency model. These approaches claim to achieve high accuracy and low false positives, as they analyze the components that are used in the failed requests, but are not used in the successful requests. The analysis detects individual faulty components, as well as faults occurring due to interactions among multiple components. The approach is well suited for large and dynamic Internet services as live tracing of client requests allows to analyze both the logical and physical behavior of the system. Tracing does not require human intervention to adapt to system changes, so these approaches can scale to constantly evolving Internet services. However, some limitations of these approaches are that they assume the requests fail independently and they do not fail because of the activities of other requests. These approaches also cannot distinguish between failure of network components that are tightly coupled and are always used together.

Cohen el al [84] have also proposed the use of learning techniques for automating performance diagnosis and performance management of computer networks. They have used Tree-Augmented Bayesian Networks or TANs [85], to identify combinations of resource-usage metrics and threshold values that are most correlated with anomalous periods. They basically automate the analysis of instrumentation data from network services in order to forecast, diagnose, and repair failure conditions. TANs are less powerful than generalized Bayesian networks, but they are simple, compact and efficient. An extension to this work has also been proposed that uses ensembles of Bayesian models to adapt to changing workloads and infrastructure [86].

2) Unsupervised learning: Unsupervised learning identifies patterns in unlabeled data typically through clustering, and

TABLE V. COMPARISON OF ALL THE FAULT LOCALIZATION TECHNIQUES IN SECTIONS III TO VII

	Active probing	**Techniques for overlay and virtual networks**	**Decentralized probabilistic management**	**Temporal models**	**Learning techniques**
Approach	Monitor and diagnose the network nodes and links using probes	Overlay symptom-fault relationship models, end-user observations, multi-layer diagnosis, and efficient placement of overlay nodes	Distribute the network diagnosis to reduce the communication overload at the central node	Use of temporal models to capture the temporal and dynamic aspects of networks	Use of learning algorithms and correlate the network events to diagnose the network for faults
Strategy	• Adaptive probing • Probabilistic inference for probe selection • Integrate active and passive monitoring	• AIR framework • Linear programming • Placement quality metric • Dempster-Shafer theory • Constraint-satisfaction problem	• All nodes participate in diagnosing all other nodes • Global aggregate metrics are used	• Dynamic Bayesian network • Spatio-temporal event techniques correlation in multilayer virtual network • Temporal correlation in network topology	• Statistical — learning parametric and non-parametric techniques • Machine — learning supervised, unsupervised and particle filtering techniques
Network overhead	Increase in network traffic	No increase in network traffic	Increase in network traffic and control overhead	No increase in network traffic	No increase in network traffic
Challenges	Selection and placement of probing-stations, selection of a smallest probe set	Monitoring agents need to be installed on all overlay nodes	All networking devices need to have embedded management process	Large space complexity to do the analysis	Large space complexity, overfitting, lengthy retraining period, insufficient data samples
Response time	Event-driven	Event-driven	Event-driven	Time-window based	After the hypothesis has been learned it can be instantly applied to set of symptoms
Resilient to change in network topology	No, probe set will need to be modified based on new network topology	No, overlay placement will need to be modified. A new O-SFA model will need to be created	Yes, algorithms will adapt to collect global aggregate metrics and also monitor nodes according to new network topology	No, new model will need to be created based on the new network topology	No, new hypothesis will need to be learned based on the new symptoms-fault relationship
Advantages	Quick response to large number of network events as compared to passive event correlation techniques	Reduces the cost to monitor multilayer overlay networks. Use of end user observations instead of network symptoms	Reduces the communication load at central collection node. Also, results show decrease in detection time of network faults	Better monitoring in dynamically changing networks	Useful when complete dependency model (symptoms-fault relationship) of a network is not available

detects unexpected anomaly in data points that might be indicators of failures.

Kiciman *et al.* [87] have proposed a similar approach as Chen [82] [83] to model the causal path in the system by capturing the runtime path of each request serviced by the system. From these paths, they extract two specific low-level behaviors likely to reflect high-level functionality: component interactions and path shapes. A reference model of the normal behavior of the system with respect to component interactions and path shapes is built as a probabilistic context-free grammar (PCFG) [88]. The current behavior of the system is to analyze, and search for anomalies with respect to the learned reference model. The primary differentiator between this work and the fault localization approach described by Chen *et al.* in [82] [83] is that these systems assume the existence of prelabeled data, e.g., failed or successful requests, and attempt to localize the fault to part of the system. In contrast, Kiciman's work with PCFG assumes no prior knowledge of faults and starts with fault detection.

Another novel learning technique of using particle filtering, to compute a probabilistic model of the network links, is proposed by Johnsson *et al.* in [89], [90]. The approach is autonomous in nature and provides the network administrator with a probability mass function that indicates the location of faulty components in the network. The algorithm performs online computation without storing results from previous measurements. The particle filtering analysis is lightweight from a computational perspective and does not need detailed knowledge about the network components being measured [91].

The algorithm starts by measuring end-to-end metrics such as one-way delay, round-trip time, loss, and jitter for each link in the network. The measurements are then fed into the Network Management System (NMS) where the network fault localization algorithm operates. The algorithm maps network segment identifiers to probability mass distribution values (weights). Each measurement result is compared to an agreed SLA for the path between the two nodes. Weights are assigned to the edges of a model that represents the network. The algorithm assigns equal weight to all paths in the network model. The weights of edges on a path are increased when an SLA violation is detected for that path, whereas all other edge weights are decreased. The weights in the vector must sum up to one, in order to make the weights correspond to probabilities. If there is an edge in the network not meeting the SLA requirement, the probability associated with that edge will increase over time. The measurements used as sample data for computing probability mass distribution are represented by a vector $< m_i, m_e, P, b >$, where m_i and m_e are start and end nodes, P is the path between m_i and m_e and b is a value

which is either true or false according to the SLA agreement.

The basic principle of the particle filtering algorithm is to construct a discrete sample based representation of the probability function for the tracked network state. The state space is discretized by assigning each particle to exactly one discrete point in the state space (i.e. one edge from the set of edges). The weight of each particle is updated according to the measurement model, and the system state estimate is obtained by calculating the weighted average of all particles. A threshold value is selected for the particle weights. If the weight of a specific particle crosses the threshold and stays there for a defined period of time, that particle is selected as the fault location. The algorithm finds the exact location of an anomaly or performance degradation in the network links, and is suitable for multi-layer and multiple fault localization.

Machine learning techniques can be used to automatically learn profiles of system behavior by using clustering to identify fault signatures. Machine learning can also be used to localize problems by identifying network components that are highly correlated with faults. However, for large systems with large number of symptoms, these techniques can suffer because of their space complexity, and approximation algorithms that try to reduce space requirement compromising their accuracy. Also, these techniques are susceptible to overfitting, a phenomenon in which the learner learns features of the evidence that are circumstantial rather than those that actually define the relationship between the faults and their symptoms. Over-fitted models can generalize poorly, and can fail when presented with an evidence that is only slightly different from the one on which the model was trained. Finally, lengthy retraining may be required whenever the system behavior changes significantly, because machine learning techniques learn a direct mapping between the symptoms and underlying root causes, without an intermediate structural model of the system. Furthermore, previously learned models often have to be thrown away during the period of retraining, leaving the systems vulnerable to any failures. Therefore, machine learning techniques may not be appropriate for systems that are frequently upgraded.

VIII. Open Research Problems and Future Research Directions

Despite several years of active research in fault localization many problems remain to be solved. Also, some of the recent developments in the industry have introduced new challenges for fault localization and to overall network management. In this section we elaborate on some of the existing and new research problems.

A. Obtaining network models

The fault localization techniques that are based on a model representation of the network are faced with the challenges of obtaining and maintaining accurate information [15], [29], [36]. The accuracy of a technique directly depends upon the accuracy of the information in the model. However, little research has been done to standardize and automate the process of obtaining the information and building a network model. The task of obtaining the information is considered to be a separate process because of the complexity associated with it. In fact, most of the techniques assume prior existence of the model.

The information used in the model may be static or dynamic in nature depending upon the type of network and model. Use of dynamic information is significantly more challenging since the information has to be incorporated in the model while the system is running. Furthermore, partial, incomplete or inaccurate information may delay the diagnosis process or remarkably affect the accuracy of the diagnosis. Some of the researchers have considered utilizing non-deterministic models to incorporate dynamic dependencies [34], [71]. In these models, the uncertainty is represented by assigning probabilities to the dependencies. However, the assignment of feasible probabilities and the use of non-deterministic models are also challenging tasks.

B. Intractable computation

The intrinsic high time and space complexity associated with probabilistic inference makes the computation intractable for large and complex networks. Furthermore, these computations need to be repeatedly performed in order to continuously monitor the network. Over the years, several approximation solutions have been proposed but they degrade the accuracy of fault diagnosis [27], [40], [42], [66]. In spite of the significant work done, several research opportunities exist. Some of these opportunities are: 1) design approximation algorithms that reduce the time and space complexity of probabilistic inference without degrading the diagnostic accuracy, 2) develop optimization strategies for the information theoretic approaches used in selecting optimal probe sets, 3) utilize pruning strategies in some of the graphical approaches, 4) create online algorithms to continuously diagnose the network symptoms instead of using event-driven or time-window based strategies.

C. Decentralized fault localization techniques

The decentralized techniques help to reduce the fault detection time and bottleneck at the centralized management node [59], [62]. However, this increases the control overhead in the network. In order to reduce the control overhead, network partitioning strategies can be applied. The partitioning strategies can be based on clustering algorithms or graph partitioning to select an optimal number of diagnostic nodes in the network. Having an optimal number of diagnostic nodes can help to reduce the communication overhead as well as reduce the control overhead.

Furthermore, changing the monitoring objective of a network management system may require a certain amount of manual configuration in the nodes. This configuration procedure further complicates the decentralized techniques. Therefore, flexible ways to modify monitoring objectives should be added to the design constructs for future solutions.

D. Fault localization in Software Defined Networks

Software defined networking (SDN) has emerged as a new network paradigm which transforms the network by abstracting

the physical network devices from the control logic. The SDN architecture provides a centralized programmability of the network devices, which leads to separation of the control plane from the forwarding plane. This separation may help to develop generic network management and monitoring solutions, which are indifferent to the vendor-specific physical devices and proprietary protocols. Another benefit of this separation and centralized control is to enable a more global view of the overall network which encourages intelligent management decisions. However, the centralized design faces some of the existing problems of traditional networks. The SDN controller placement problem can be mapped to the problem of placing a centralized network management system in a traditional network. The centralized control can help to provide better diagnostic and monitoring capabilities. However, it introduces a single point of failure for the entire network operation. The SDN environment may also present some new categories of faults for the network management systems.

E. Network management in highly dynamic environments

The frequency of infrastructure changes in the highly dynamic environments composed of SDN-enabled components and NFV-based components brings new challenges to the network management. Some of the challenges could be with respect to the frequency and scale of configuration changes enabled by SDN, dynamic addition and removal of the NFV-based components depending on the business and user requirements, tracking the inventory fluctuations, and monitoring resource utilization. One example is monitoring of an interface for a specific utilization threshold, which can change dramatically by addition or removal of services. Another problem is to keep track of all the frequent changes to network services and consider them in making management decisions.

The deployment of next-generation network management solutions should be based on agile methodologies to keep up with the pace of frequent infrastructure changes. New functionalities should be included in the management tasks where the system not only detects and locates the problems but also helps to prevent them by capturing and considering the most recent view of the state of the network components. In order to provide a possibility of better network stability and service delivery, different aspects of management such as provisioning, monitoring, analytics, and notification should be coupled into a single solution.

F. Heterogeneity in network environments

As new network environments are deployed, the traditional and legacy network environments are still expected to coexist with the newer technologies for a long time. During this transition period, the network management solutions should have the ability to manage this heterogeneity in the network. Moreover, the network-based services will typically traverse both the traditional and new networks as well as physical and virtual components. Therefore, the coexistence is required to be managed seamlessly considering the end-to-end view of the composite network environment.

IX. Conclusions

Networks are dramatically scaling and various new paradigms are being rapidly developed. These changes demand development of a new generation of network fault management tools. Faults in network systems will remain unavoidable, so their quick and accurate detection and diagnosis is important for the stability, consistency, and performance of network systems. Therefore, fault localization will continue to play an essential role in the overall network management operations.

This paper presents a comprehensive survey of recent fault localization techniques proposed for computer networks. These techniques are designed considering the evolution of computer networks over the past decade. We discuss the benefits and limitations associated with each of the techniques and categorized them based on their applicability and key design principles. In order to highlight the key ideas, challenges, and advantages of each of the techniques, a summary of the comparisons is included in Table V. We believe that description and comparisons of the fault localization techniques presented in this paper will be helpful for developing new generic network management solutions for various network environments.

References

[1] M. Steinder and A. S. Sethi, "A survey of fault localization techniques in computer networks," *Science of Computer Programming*, vol. 53, no. 2, pp. 165–194, November 2004.

[2] G. Liu, A. Mok, and E. Yang, "Composite events for network event correlation," in *Proceedings of the Sixth IFIP/IEEE International Symposium on Integrated Network Management, IM 1999*, Boston, MA, May 1999, pp. 247–260.

[3] K. E. Lor, "A network diagnostic expert system for acculink multiplexers based on a general network diagnostic scheme," in *Proceedings of the Third IFIP/IEEE International Symposium on Integrated Network Management, IM 1993*, San Franciso, CA, April 1993, pp. 659–669.

[4] L. Lewis, "A case-based reasoning approach to the management of faults in communication networks," in *Proceedings of the IEEE Computer and Communications Societies, INFOCOM 1993*, vol. 3, San Francisco, CA, March 1993, pp. 1422–1429.

[5] K. Appleby, G. Goldszmidt, and M. Steinder, "Yemanja - a layered fault localization system for multi-domain computing utilities," *Journal of Network and Systems Management*, vol. 10, no. 2, pp. 171–194, June 2002.

[6] R. Gardner and D. Harle, "Alarm correlation and network fault resolution using the Kohonen self-organising map," in *Proceedings of the IEEE Global Telecommunications Conference, GLOBECOM 1997*, vol. 3, Phoenix, AZ, November 1997, pp. 1398–1402.

[7] ——, "Pattern discovery and specification techniques for alarm correlation," in *Proceedings of the IEEE/IFIP Network Operations and Management Symposium, NOMS 1998*, vol. 3, New Orleans, LA, February 1998, pp. 713–722.

[8] H. Wietgrefe, "Investigation and practical assessment of alarm correlation methods for the use in GSM access networks," in *Proceedings of the IEEE/IFIP Network Operations and Management Symposium, NOMS 2002*, Florence, Italy, April 2002, pp. 391–403.

[9] G. D. Rodosek and T. Kaiser, "Intelligent assistant: User-guided fault localization," in *Proceedings of Ninth International Workshop on Distributed Systems, DSOM 1998*, Newark, DE, October 1998, pp. 119–129.

[10] A. Dupuy, E. Dupuy, S. Sengupta, O. Wolfson, and Y. Yemini, "Design of the Netmate network management system," in *Proceedings of the Second IFIP/IEEE International Symposium on Integrated Network Management, IM 1991*, Washington, DC, April 1991, pp. 639–650.

[11] G. Forman, M. Jain, M. Mansouri-Samani, J. Martinka, and A. Snoeren, "Automated end-to-end system diagnosis of networked printing services using model based reasoning," in *Proceedings of Ninth International Workshop on Distributed Systems, DSOM 1998*, Newark, DE, October 1998, pp. 142–154.

[12] G. Jakobson and M. Weissman, "Alarm correlation," *IEEE Network*, vol. 7, no. 6, pp. 52–59, November 1993.

[13] S. Katker and M. Paterok, "Fault isolation and event correlation for integrated fault management," in *Proceedings of the Fifth IFIP/IEEE International Symposium on Integrated Network Management, IM 1997*, San Diego, California, May 1997, pp. 583–596.

[14] K. Houck, S. Calo, and A. Finkel, "Towards a practical alarm correlation system," in *Proceedings of the Fourth IFIP/IEEE International Symposium on Integrated Network Management, IM 1995*, Santa Barbara, CA, May 1995, pp. 226–237.

[15] I. Katzela and M. Schwartz, "Schemes for fault identification in communication networks," *IEEE/ACM Transactions on Networking*, vol. 3, no. 6, pp. 753–764, December 1995.

[16] M. K. Agarwal, K. Appleby, M. Gupta, G. Kar, A. Neogi, and A. Sailer, "Problem determination using dependency graphs and run-time behavior models," in *Proceedings of the Fifteenth IFIP/IEEE International Workshop on Distributed Systems, DSOM 2004*, Davis, CA, November 2004, pp. 171–182.

[17] K. Appleby, J. Faik, G. Kar, A. Sailer, M. K. Agarwal, and A. Neogi, "Threshold management for problem determination in transaction based e-commerce systems," in *Proceedings of the Ninth IFIP/IEEE International Symposium on Integrated Network Management, IM 2005*, Nice, France, May 2005, pp. 733–746.

[18] D. Heckerman, D. M. Chickering, C. Meek, R. Rounthwaite, and C. Kadie, "Dependency networks for inference, collaborative filtering, and data visualization," *Journal of Machine Learning Research*, vol. 1, pp. 49–75, September 2000.

[19] K. P. Murphy, Y. Weiss, and M. I. Jordan, "Loopy belief propagation for approximate inference: An empirical study," in *Proceedings of the Uncertainty in Artificial Intelligence, UAI 1999*, Stockholm, Sweden, July 1999, pp. 467–475.

[20] M. I. Jordan, Z. Ghahramani, T. S. Jaakkola, and L. K. Saul, "An introduction to variational methods for graphical models," *Machine Learning*, vol. 37, no. 2, pp. 183–233, November 1999.

[21] G. Hulten, D. M. Chickering, and D. Heckerman, "Learning Bayesian networks from dependency networks: A preliminary study," in *Proceedings of the Ninth International Workshop on Artificial Intelligence and Statistics*, Key West, FL, January 2003.

[22] A. Bouloutas, S. Calo, and A. Finkel, "Alarm correlation and fault identification in communication networks," *IEEE Transactions on Communications*, vol. 42, no. 234, pp. 523–533, February 1994.

[23] S. Klinger, S. Yemini, Y. Yemini, D. Ohsie, and S. Stolfo, "A coding approach to event correlation," in *Proceedings of the Fourth IFIP/IEEE International Symposium on Integrated Network Management, IM 1995*, Santa Barbara, CA, May 1995, pp. 266–277.

[24] S. Yemini, S. Kliger, E. Mozes, Y. Yemini, and D. Ohsie, "High speed and robust event correlation," *IEEE Communications Magazine*, vol. 34, no. 5, pp. 82–90, May 1996.

[25] R. Dechter, "Bucket elimination: A unifying framework for reasoning," *Artificial Intelligence*, vol. 113, no. 1-2, pp. 41–85, November 1999.

[26] J. Pearl, *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 1988.

[27] M. Steinder and A. S. Sethi, "Probabilistic fault localization in communication systems using belief networks," *IEEE/ACM Transactions on Networking*, vol. 12, no. 5, pp. 809–822, October 2004.

[28] ——, "End-to-end service failure diagnosis using belief networks," in *Proceedings of the IEEE/IFIP Network Operations and Management Symposium, NOMS 2002*, Florence, Italy, April 2002, pp. 375–390.

[29] ——, "Probabilistic fault diagnosis in communication systems through incremental hypothesis updating," *Computer Networks*, vol. 45, no. 4, pp. 537–562, July 2004.

[30] L. Bennacer, Y. Amirat, A. Chibani, A. Mellouk, and L. Ciavaglia, "Self - diagnosis technique for virtual private networks combining Bayesian networks and case - based reasoning," *IEEE Transactions on Automation Science and Engineering*, vol. 12, no. 1, pp. 354–366, January 2015.

[31] M. Brodie, I. Rish, and S. Ma, "Optimizing probe selection for fault localization," in *Proceedings of Twelfth International Workshop on Distributed Systems, DSOM 2001*, Nancy, France, October 2001, pp. 88–98.

[32] M. Natu and A. S. Sethi, "Application of adaptive probing for fault diagnosis in computer networks," in *Proceedings of the IEEE/IFIP Network Operations and Management Symposium, NOMS 2008*, April 2008, pp. 1055–1060.

[33] I. Rish, M. Brodie, N. Odintsova, S. Ma, and G. Grabarnik, "Real-time problem determination in distributed systems using active probing," in *Proceedings of the IEEE/IFIP Network Operations and Management Symposium, NOMS 2004*, vol. 1, Seoul, Korea, April 2004, pp. 133–146.

[34] M. Natu and A. S. Sethi, "Probabilistic fault diagnosis using adaptive probing," in *Proceedings of the Eighteenth IFIP/IEEE International Workshop on Distributed Systems, DSOM 2007*, San Jose, CA, October 2007, pp. 38–49.

[35] ——, "Active probing approach for fault localization in computer networks," in *Proceedings of the Fourth IEEE/IFIP Workshop on End-to-End Monitoring Techniques and Services*, Vancouver, Canada, April 2006, pp. 25–33.

[36] M. Natu, A. S. Sethi, and E. Lloyd, "Efficient probe selection algorithms for fault diagnosis," *Telecommunication Systems*, vol. 37, no. 1-3, pp. 109–125, April 2008.

[37] M. Natu and A. S. Sethi, "Probe station placement for fault diagnosis," in *Proceedings of the IEEE Global Telecommunications Conference, GLOBECOM 2007*, Washington D.C., November 2007, pp. 113–117.

[38] ——, "Efficient probing techniques for fault diagnosis," in *Proceedings of the Second International Conference on Internet Monitoring and Protection, ICIMP 2007*, Silicon Valley, CA, July 2007.

[39] L. Lu, Z. Xu, W. Wang, and Y. Sun, "A new fault detection method for computer networks," *Reliability Engineering and System Safety*, vol. 114, pp. 45–51, June 2013.

[40] I. Rish, M. Brodie, S. Ma, N. Odintsova, A. Beygelzimer, G. Grabarnik, and K. Hernandez, "Adaptive diagnosis in distributed systems," *IEEE Transactions on Neural Networks*, vol. 16, no. 5, pp. 1088–1109, September 2005.

[41] M. Brodie, I. Rish, S. Ma, and N. Odintsova, "Active probing strategies for problem diagnosis in distributed systems," in *Proceedings of the Eighteenth International Joint Conference on Artificial Intelligence, IJCAI 2003*, Acapulco, Mexico, August 2003, pp. 1337–1338.

[42] A. Zheng, I. Rish, and A. Beygelzimer, "Efficient test selection in active diagnosis via entropy approximation," in *Proceedings of the Conference on Uncertainty in Artificial Intelligence, UAI 2005*, Edinburgh, Scotland, July 2005.

[43] L. Yu, L. Cheng, Y. Qiao, Y. Yuan, and X. Chen, "An efficient active probing approach based on the combination of online and offline strategies," in *Proceedings of the Sixth International Conference on Network and Service Management, CNSM 2010*, Niagara Falls, ON, October 2010, pp. 298–301.

[44] L. Cheng, X. Qiu, L. Meng, Y. Qiao, and R. Boutaba, "Efficient active probing for fault diagnosis in large scale and noisy networks," in *Proceedings of the IEEE Computer and Communications Societies, INFOCOM 2010*, San Diego, CA, March 2010, pp. 1–9.

[45] L. Guan, Y. Wang, W. Li, and C. Yan, "Efficient probing method for active diagnosis in large scale network," in *Proceedings of the Ninth International Conference on Network and Service Management, CNSM 2013*, Zurich, Switzerland, October 2013, pp. 198–202.

[46] Y. Tang, E. Al-Shaer, and R. Boutaba, "Active integrated fault localization in communication networks," in *Proceedings of the Ninth IFIP/IEEE International Symposium on Integrated Network Management, IM 2005*, Ottawa, Canada, May 2005, pp. 543–556.

[47] ——, "Efficient fault diagnosis using incremental alarm correlation and active investigation for Internet and overlay networks," *IEEE Transactions on Network and Service Management*, vol. 5, no. 1, pp. 36–49, March 2008.

[48] Y. Pan, X. Qiu, and S. Zhang, "Fault diagnosis in network virtualization environment," in *Proceedings of the Eighteenth International Conference on Telecommunications, ICT 2011*, Ayia Napa, Cyprus, May 2011, pp. 517–522.

[49] M. Demirci, S. Lo, S. Seetharaman, and M. Ammar, "Multi-layer monitoring of overlay networks," in *Proceedings of the Tenth International Conference on Passive and Active Network Measurement, PAM 2009*, Seoul, Korea, April 2009, pp. 77–86.

[50] Y. Chen, D. Bindel, H. H. Song, and R. Katz, "Algebra-based scalable overlay network monitoring: Algorithms, evaluation, and applications," *IEEE/ACM Transactions on Networking*, vol. 15, no. 5, pp. 1084–1097, October 2007.

[51] Y. Tang, E. Al-Shaer, and K. Joshi, "Reasoning under uncertainty for overlay fault diagnosis," *IEEE Transactions on Network and Service Management*, vol. 9, no. 1, pp. 34–47, March 2012.

[52] H. Wang, Y. Wang, X. Qiu, W. Li, and A. Xiao, "Fault diagnosis based on evidences screening in virtual network," in *Proceedings of the IFIP/IEEE International Symposium on Integrated Network Management, IM 2015*, Ottawa, Canada, May 2015, pp. 802–805.

[53] S. Gillani, M. Demirci, E. Al-Shaer, and M. Ammar, "Problem localization and quantification using formal evidential reasoning for virtual networks," *IEEE Transactions on Network and Service Management*, vol. 11, no. 3, pp. 307–320, September 2014.

[54] C. Yan, Y. Wang, X. Qiu, W. Li, and L. Guan, "Multi-layer fault diagnosis method in the Network Virtualization Environment," in *Proceedings of the Sixteenth Asia-Pacific Network Operations and Management Symposium, APNOMS 2014*, Hsinchu, Taiwan, September 2014, pp. 1–6.

[55] M. Demirci, F. Gillani, M. Ammar, and E. Al-Shaer, "Overlay network placement for diagnosability," in *Proceedings of the IEEE Global Communications Conference, GLOBECOM 2013*, Atlanta, GA, December 2013, pp. 2236–2242.

[56] A. P. Dempster, "Upper and lower probabilities induced by a multivalued mapping," *The Annals of Mathematical Statistics*, vol. 38, no. 2, pp. 325–339, April 1967.

[57] M. Yu, H. Mokhtar, and M. Merabti, "Fault management in wireless sensor networks," *IEEE Wireless Communications*, vol. 14, no. 6, pp. 13–19, December 2007.

[58] S. Zhuang, D. Geels, I. Stoica, and R. Katz, "On failure detection algorithms in overlay networks," in *Proceedings of the IEEE Computer and Communications Societies, INFOCOM 2005*, vol. 3, Miami, FL, March 2005, pp. 2112–2123.

[59] A. Prieto, D. Dudkowski, C. Meirosu, C. Mingardi, G. Nunzi, M. Brunner, and R. Stadler, "Decentralized in-network management for the Future Internet," in *Proceedings of the IEEE International Conference on Communications Workshops, ICC 2009*, Dresden, Germany, June 2009, pp. 1–5.

[60] A. Prieto, D. Gillblad, R. Steinert, and A. Miron, "Toward decentralized probabilistic management," *IEEE Communications Magazine*, vol. 49, no. 7, pp. 80–86, July 2011.

[61] R. Steinert and D. Gillblad, "Long-term adaptation and distributed detection of local network changes," in *Proceedings of the IEEE Global Telecommunications Conference, GLOBECOM 2010*, Miami, FL, December 2010, pp. 1–5.

[62] ——, "Towards distributed and adaptive detection and localisation of network faults," in *Proceedings of the Sixth Advanced International Conference on Telecommunications, AICT 2010*, Barcelona, Spain, May 2010, pp. 384–389.

[63] A. Prieto and R. Stadler, "Adaptive real-time monitoring for large-scale networked systems," in *Proceedings of the Eleventh IFIP/IEEE International Symposium on Integrated Network Management, IM 2009*, Long Island, NY, June 2009, pp. 790–795.

[64] R. Cohen and A. Landau, ""Not All At Once!" - A generic scheme for estimating the number of affected nodes while avoiding feedback implosion," in *Proceedings of the IEEE Computer and Communications Societies, INFOCOM 2009*, Rio de Janeiro, Brazil, April 2009, pp. 2641–2645.

[65] M. Natu and A. S. Sethi, "Using temporal correlation for fault localization in dynamically changing networks," *International Journal of Network Management*, vol. 18, no. 4, pp. 303–316, August 2008.

[66] Z. Li, L. Cheng, X. Qiu, and Y. Zeng, "Fault diagnosis for large-scale IP networks based on dynamic Bayesian model," in *Proceedings of the Fifth International Conference on Natural Computation, ICNC 2009*, vol. 6, Tianjin, China, August 2009, pp. 67–71.

[67] R. Steinert, S. Gestrelius, and D. Gillblad, "A distributed spatio-temporal event correlation protocol for multi-layer virtual networks," in *Proceedings of the IEEE Global Telecommunications Conference, GLOBECOM 2011*, Houston, TX, December 2011, pp. 1–5.

[68] K. Murphy, "Dynamic Bayesian networks: Representation, inference and learning," Ph.D. dissertation, UC Berkeley, Computer Science Division, July 2002.

[69] K. Murphy and Y. Weiss, "The factored frontier algorithm for approximate inference in DBNs," in *Proceedings of the Conference Uncertainty in Artificial Intelligence, UAI 2001*, Seattle, Washington, August 2001, pp. 378–385.

[70] X. Boyen and D. Koller, "Tractable inference for complex stochastic processes," in *Proceedings of the Uncertainty in Artificial Intelligence, UAI 1998*, Madison, Wisconsin, July 1998, pp. 33–42.

[71] M. Steinder and A. S. Sethi, "Probabilistic event-driven fault diagnosis through incremental hypothesis updating," in *Proceedings of the Eighth IFIP/IEEE Eighth International Symposium on Integrated Network Management, IM 2003*, Colorado Springs, CO, March 2003, pp. 635–648.

[72] M. Basseville and I. V. Nikiforov, *Detection of Abrupt Changes: Theory and Application*. Upper Saddle River, NJ: Prentice-Hall, Inc., 1993.

[73] M. Agarwal, M. Gupta, V. Mann, N. Sachindran, N. Anerousis, and L. Mummert, "Problem determination in enterprise middleware systems using change point correlation of time series data," in *Proceedings of the IEEE/IFIP Network Operations and Management Symposium, NOMS 2006*, Vancouver, Canada, April 2006, pp. 471–482.

[74] S. Kandula, R. Mahajan, P. Verkaik, S. Agarwal, J. Padhye, and P. Bahl, "Detailed diagnosis in enterprise networks," in *Proceedings of the ACM Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications, SIGCOMM 2009*, Barcelona, Spain, August 2009, pp. 243–254.

[75] S. Kavulya, K. R. Joshi, M. A. Hiltunen, S. Daniels, R. Gandhi, and P. Narasimhan, "Practical experiences with chronics discovery in large telecommunications systems," *ACM Special Interest Group on Operating Systems Operating Systems Review, OSR 2011*, vol. 45, no. 3, pp. 23–30, December 2011.

[76] S. P. Kavulya, S. Daniels, K. Joshi, M. Hiltunen, R. Gandhi, and P. Narasimhan, "Draco: Statistical diagnosis of chronic problems in large distributed systems," in *Proceedings of the 42nd Annual IEEE/IFIP International Conference on Dependable Systems and Networks DSN 2012*, Boston, Massachusetts, June 2012, pp. 1–12.

[77] G. Jiang, H. Chen, K. Yoshihira, and A. Saxena, "Ranking the importance of alerts for problem determination in large computer systems," in *Proceedings of the Sixth International Conference on Autonomic Computing, ICAC 2009*, Barcelona, Spain, June 2009, pp. 3–12.

[78] B. Gruschke, "Integrated event management: Event correlation using dependency graphs," in *Proceedings of Ninth International Workshop on Distributed Systems, DSOM 1998*, Newark, DE, October 1998.

[79] Z. Guo, G. Jiang, H. Chen, and K. Yoshihira, "Tracking probabilistic correlation of monitoring data for fault detection in complex systems,"

in *Proceedings of the International Conference on Dependable Systems and Networks, DSN 2006*, Philadelphia, PA, June 2006, pp. 259–268.

[80] C. Mas and J.-Y. L. Boudec, "An alarm filtering algorithm for optical communication networks," in *Proceedings of the IEEE/IFIP International Conference on Management of Multimedia Networks and Services*, 1998, pp. 205–218.

[81] A. A. Mahimkar, Z. Ge, A. Shaikh, J. Wang, J. Yates, Y. Zhang, and Q. Zhao, "Towards automated performance diagnosis in a large IPTV network," in *Proceedings of the ACM Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications, SIGCOMM 2009*, Barcelona, Spain, August 2009, pp. 231–242.

[82] M. Chen, E. Kiciman, E. Fratkin, A. Fox, and E. Brewer, "Pinpoint: problem determination in large, dynamic Internet services," in *Proceedings of the International Conference on Dependable Systems and Networks, DSN 2002*, Bethesda, MD, June 2002, pp. 595–604.

[83] M. Chen, A. Zheng, J. Lloyd, M. Jordan, and E. Brewer, "Failure diagnosis using decision trees," in *Proceedings of the International Conference on Autonomic Computing*, New York, NY, May 2004, pp. 36–43.

[84] I. Cohen, M. Goldszmidt, T. Kelly, J. Symons, and J. S. Chase, "Correlating instrumentation data to system states: A building block for automated diagnosis and control," in *Proceedings of the Sixth Conference on Symposium on Opearting Systems Design and Implementation*, San Francisco, CA, October 2004, pp. 231–244.

[85] J. Cerquides and R. L. de Mantaras, "Tractable Bayesian learning of tree augmented naive Bayes models," in *Proceedings of the Twentieth International Conference on Machine Learning, ICML 2003*, Washington, DC, August 2003, pp. 75–82.

[86] S. Zhang, I. Cohen, M. Goldszmidt, J. Symons, and A. Fox, "Ensembles of models for automated diagnosis of system performance problems," in *Proceedings of the International Conference on Dependable Systems and Networks, DSN 2005*, Yokohama, Japan, June 2005, pp. 644–653.

[87] E. Kiciman and A. Fox, "Detecting application - level failures in component - based Internet services," *IEEE Transactions on Neural Networks*, vol. 16, no. 5, pp. 1027–1041, September 2005.

[88] C. D. Manning and H. Schutze, *Foundations of Statistical Natural Language Processing*. Cambridge, MA, USA: MIT Press, 1999.

[89] A. Johnsson and C. Meirosu, "Towards automatic network fault localization in real time using probabilistic inference," in *Proceedings of the Thirteenth IFIP/IEEE International Symposium on Integrated Network Management, IM 2013*, Ghent, Belgium, May 2013, pp. 1393–1398.

[90] A. Johnsson, C. Meirosu, and C. Flinta, "Online network performance degradation localization using probabilistic inference and change detection," in *Proceedings of the IEEE/IFIP Network Operations and Management Symposium, NOMS 2014*, Krakow, Poland, May 2014, pp. 1–8.

[91] M. Arulampalam, S. Maskell, N. Gordon, and T. Clapp, "A tutorial on particle filters for online nonlinear/non-gaussian Bayesian tracking," *IEEE Transactions on Signal Processing*, vol. 50, no. 2, pp. 174–188, February 2002.

Ayush Dusia received an M.S. degree in Computer & Information Sciences from the University of Delaware, Newark, Delaware, USA. He is currently pursuing a Ph.D. degree in Computer & Information Sciences from the University of Delaware. His research interests include network fault localization, network diagnosis, network monitoring and management. His current focus is on designing an architecture for software defined networks and analyzing its impact on various network environments.

Dr. Adarshpal Sethi is a Professor in the Department of Computer & Information Sciences at the University of Delaware, Newark, Delaware, USA. He has an MTech degree in Electrical Engineering and a PhD in Computer Science, both from the Indian Institute of Technology, Kanpur, India. He has served on the faculty at IIT Kanpur, was a visiting faculty at Washington State University, Pullman, WA, and Visiting Scientist at IBM Research Laboratories, Zurich, Switzerland, and at the US Army Research Laboratory, Aberdeen, MD. He is on the editorial board of the International Journal of Network Management and is an Associate Editor for the Electronic Commerce Research Journal. He is also active on the program committees of numerous conferences. He is co-author of the book The Practical OPNET User Guide for Computer Network Simulation. His research interests include architectures and protocols for network management, fault management and diagnosis, and management of wireless networks.