Multi-domain diagnosis of end-to-end service failures in hierarchically routed networks

Małgorzata Steinder
IBM T. J. Watson Research Center
19 Skyline Dr, Hawthorne, NY 10532
E-mail: steinder@us.ibm.com

Adarshpal S. Sethi
Computer and Information Sciences
University of Delaware, Newark, DE 19716
E-mail: sethi@cis.udel.edu

Abstract

Probabilistic inference was shown effective in non-deterministic diagnosis of end-to-end service failures. Since exact probabilistic diagnosis is known to be an NP-hard problem, approximate techniques were investigated. They were shown efficient and accurate in isolating root causes of end-to-end disorder in networks composed of tens of nodes but did not scale well to bigger networks. In addition, the requirement that a centralized manager posess a global knowledge of the system structure and state made the techniques difficult to apply in real-life. This paper investigates an approach to improving the scalability and feasibility of probabilistic diagnosis by exploiting the domain semantics of computer networks. The proposed technique divides the computational effort and system knowledge among multiple, hierarchically organized managers. Each manager performs fault localization in the domain it manages and requires only the knowledge of its own domain. We show through simulation that the proposed approach increases the effectiveness of probabilistic diagnosis and makes it feasible in networks of considerable size [1].

I. Introduction

End-to-end connectivity in a given protocol layer is frequently provided through a sequence of intermediate nodes such as bridges in the data-link layer or routers in the network layer. Communication problems between a pair of these nodes, e.g., a malfunctioning interface, intermittent connectivity, etc., may disorder one or more end-to-end paths provided using the failing host-to-host link. These end-to-end problems propagate to higher system layers causing various application-level events, e.g., aborted transactions, session timeouts, abnormal delays, etc. Therefore, it is important that host-to-host problems, both availability- and performance-related ones, be identified quickly and accurately. Unfortunately, oftentimes host-to-host failures cannot be detected directly by monitoring host-to-host connectivity. This is due to the fact that certain failure conditions cannot be monitored on a host-to-host basis either because there is no appropriate monitoring mechanism or because of the associated overhead. Moreover, an end-to-end service user frequently does not have the administrative authority allowing her to monitor host-to-host connectivity. In these situations, host-to-host problems have to be identified by correlating indications of end-to-end disorder.

This paper adopts a service-oriented view of the network [16], in which end-to-end or host-to-host connectivity between two nodes in a given protocol layer is considered a service provided by this layer to higher layers. End-to-end service between nodes *a* and *b* is implemented using (i.e., depends on) a set of host-to-host services between neighboring nodes on a path from *a* to *b*.

Diagnosis of end-to-end network service failures [37], [41] is a sub-task of fault localization [17], [22], [46] that isolates host-to-host services responsible for availability or performance problems experienced by end-to-end services. In a complete, multi-layer fault localization solution, end-to-end service failures diagnosed by this sub-task may be reported by higher-layer fault localization mechanisms, which identify them as causes of disorders observed in higher layers. Similarly, host-to-host service failures identified by

M. Steinder completed this work while with the Dept. of Computer and Information Sciences, University of Delaware

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

the process of end-to-end service failure diagnosis, may be further analyzed by lower-layer fault localization techniques to perform a more detailed fault determination [35]. These higher- and lower-layer techniques are not discussed in this paper.

In the previous work [41], [36], we investigated an application of probabilistic reasoning to end-to-end service failure diagnosis. The proposed approaches rely on a probabilistic *fault propagation model* (FPM), which represents causal relationships between end-to-end and host-to-host service failures. The model has a form of a bipartite causality graph with host-to-host and end-to-end service problems at the tails and at the heads of the edges, respectively. Given the FPM, the fault localization problem is to find a set of host-to-host failures (faults) that provide the most probable explanation (MPE) of observed end-to-end service failures (symptoms). To solve this problem, in [41], an adaptation of Pearl's belief updating [30] was used, and in [36], a novel algorithm based on incremental hypothesis updating was proposed. The algorithms were shown effective in the diagnosis of end-to-end service failures in networks composed of tens of nodes. In addition, they proved to be resilient against lost and spurious symptoms, and to be insensitive to the inaccuracies of the probabilistic FPM [39], [40].

Today's networks are frequently composed of multiple domains, each with a different organization and management policy. For example, the Internet is built of tens of thousands of autonomous systems (AS) [43]. An exterior gateway protocol between ASs, e.g., BGP [33], ensures that traffic routed from one AS to another adheres to policies set-up by AS owners, which may concern privacy, security, or business objectives. A network domain may be further divided into smaller sub-domains. In the Internet, the OSPF protocol [28] divides an AS into areas, which are connected via a backbone. Networks that do not exhibit an explicit domain-semantics in their structure may be also divided into domains for management purposes. For example, a tree-shaped network topology may be divided into multiple sub-trees that are managed by separate management entities. The algorithms proposed in [36], [41], [39], [40] are difficult to apply to such big multi-domain topologies. They exhibit shortcomings typical of any centralized management scheme, which include:

- Infeasibility – When subsystems are in different administrative domains, obtaining management information, such as topology, routing, or internal state, may be impossible outside of a domain.
- Inefficiency – In large systems, the FPM's size makes fault localization prohibitively time-consuming.
- Inflexibility – The same management strategy is applied to the entire system even though particular subsystems may have different requirements.
- Single point of failure
- Vulnerability to security breaches resulting from maintaining management information of the entire system in a central location

This paper introduces a multi-domain fault-localization technique, which increases the admissible network size by an order of magnitude by taking advantage of the domain semantics of communication systems. The proposed technique divides the computational effort and system knowledge involved in end-to-end service-failure diagnosis among multiple hierarchically organized managers. Each manager is responsible for fault localization within the network domain it governs, and reports to a higher-level manager that oversees and coordinates the fault-localization process of multiple domains. With this organization, the technique is suitable for distributed diagnosis of end-to-end service failures in hierarchically routed networks.

Distributed fault localization has been recognized as an important objective of fault management systems [6], [21], [46], but few such distributed techniques have actually been proposed. A theoretical foundation for the design of such systems has been laid by Bouloutas et al. [6] and Katzela et al. [20], who investigate different schemes of non-centralized fault localization: decentralized and distributed schemes. In a distributed approach, a system is divided into domains managed by separate managers that have to cooperate to reach a solution. All managers have a partial knowledge of the system, both of its structure and current state, and are organized according to various paradigms: either form a hierarchy (decentralized scheme) or cooperate on a peer-to-peer basis (distributed scheme). The technique proposed in this paper has properties of both these schemes. Similar to the decentralized scheme [20], we envision a hierarchy of managers with a central manager making the final fault determination. Unlike the decentralized scheme, however, higher-level managers not only arbitrate among solutions proposed by lower-level managers, but

also participate in the actual fault determination by proposing their own hypotheses composed of network faults that cannot be identified by the lower-level managers.

The choice of the hierarchical paradigm is natural in fault management; even if multiple managers cooperate as peers to reach a solution, a central trusted entity is needed to verify and present the solution to a system administrator. In addition, the hierarchical paradigm is well suited to hierarchically organized networks, such as the Internet.

While distributed fault management alleviates the problems associated with the centralized approach, it is significantly more difficult to achieve due to the following reasons.

- Failure propagation among domains—Symptoms of a fault which occurred in one domain may be observed in other domains. In fact, it is possible that a fault is not at all detected in the domain in which it occurred.
- A lack of global information about the system structure and state—A symptom diagnosis is complicated because not all its possible causes are visible in a domain in which the symptom was observed. A symptom diagnosis is also hampered by the lack of information about symptoms observed or faults identified in other domains; in the absence of sufficient internal observations, a domain manager may be unable to diagnose a problem, which could be possible should the information from other domains be available to the manager.
- The necessity to coordinate the operation of multiple managers—In a distributed solution, in addition to analyzing observed symptoms, a manager has to cooperate with other managers to reach the final solution. It is therefore necessary to define a protocol for communication among managers.

Our goal in this paper is to find a solution to end-to-end service failure diagnosis in a multi-domain network while addressing the problems of failure propagation among domains and the lack of global knowledge. While recognizing the fact that various probabilistic reasoning mechanisms can be used by managers, we aim at presenting a generic technique of decomposing the problem of end-to-end service failure diagnosis into multiple smaller subproblems that complies with the domain semantics of the communication systems and may be specialized for a variety of such reasoning mechanisms. We also aim at showing two specializations of the generic technique tailored toward the iterative belief updating [41] and incremental hypothesis updating [36] as probabilistic reasoning mechanisms. Our solution does not define a communication protocol used in the distributed approach and therefore does not constitute a complete distributed technique. While we present what information has to be exchanged among managers, we do not investigate types and structures of the exchanged PDUs, nor do we decide on a communication mechanism that should be used by the managers or define specific actions taken by a manager when it receives a particular PDU.

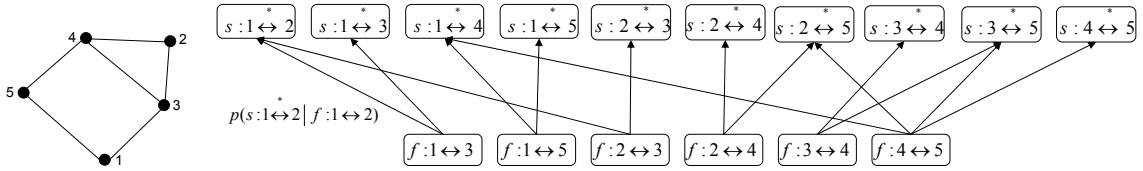
The paper is structured as follows. In Section II, the centralized probabilistic algorithms, which were introduced in [36], [41] are presented. In Section III, an outline of a multi-domain fault localization technique for hierarchically routed networks is proposed. A distributed fault propagation model is proposed in Section IV, and a multi-domain fault localization algorithm is presented in Section V. In Section VI a multi-domain algorithm based on event-driven belief updating [41] is introduced. In Section VII a multi-domain algorithm derived from incremental hypothesis updating [36] is proposed. Section VIII presents results of the simulation study conducted to verify the effectiveness of the proposed multi-domain techniques.

II. Probabilistic diagnosis of end-to-end service failures

When connectivity between nodes *a* and *b* in a given network layer is achieved through a sequence of intermediate nodes, we say that the service of end-to-end communication between hosts *a* and *b* provided by this layer to higher layers is implemented in terms of multiple services of host-to-host communication between subsequent hops on the path from node *a* to node *b*. A failure of a host-to-host service, such as excessive delay, high packet loss rate, erroneous packet transmission, or total loss of connectivity, propagates to an end-to-end service implemented using the failing host-to-host service. How a specific failure of a host-to-host service affects a dependent end-to-end service is decided by the communication protocol used in the given layer. For example, when the protocol implements an error detection mechanism, erroneous output produced by a host-to-host service results in data loss in a dependent end-to-end service. When the protocol

does not implement an error detection mechanism, erroneous output produced by a host-to-host service does not affect data loss rate of a dependent end-to-end service. Instead, erroneous output will be observed.

The problem of end-to-end service failure diagnosis is to identify the set of host-to-host service failures that are the most probable causes of the observed end-to-end disorder based on the information on causal relationships between host-to-host and end-to-end service failures provided in the form of a fault propagation model (FPM). The FPM for end-to-end service failure diagnosis is a bipartite causality graph in which parentless nodes (called *link* nodes) represent host-to-host service failures (faults) and childless nodes (called *path* nodes) represent end-to-end service failures (symptoms). Multiple *link* or *path* nodes may exist for every host-to-host or end-to-end service that correspond to different types of failures. Since causal relationships between host-to-host and end-to-end service failures are difficult to determine due to their dynamic and unpredictable nature, the FPM is a probabilistic one, in which each *link* node is labeled with the probability of the corresponding fault's independent occurrence, and causal edges between *link* nodes and *path* nodes are weighted with the probability of the causal implication between corresponding faults and symptoms (Figures 1(a) and 1(b)).



(a) Network (b) Fault propagation model

Fig. 1. The construction of an FPM for an example network. The FPM models only one failure type per path or link. It assumes that all routes are bidirectional and that end-to-end routes for paths $1 \overset{*}{\leftrightarrow} 2$, $1 \overset{*}{\leftrightarrow} 4$, $2 \overset{*}{\leftrightarrow} 5$, and $3 \overset{*}{\leftrightarrow} 5$ are $1 \leftrightarrow 3 \leftrightarrow 2$, $1 \leftrightarrow 5 \leftrightarrow 4$, $2 \leftrightarrow 4 \leftrightarrow 5$, and $3 \leftrightarrow 4 \leftrightarrow 5$, respectively.

In this paper, we denote by $\mathcal{S}$ and $\mathcal{F}$ the set of all possible end-to-end service failures (symptoms) and the set of all possible host-to-host service failures (faults), respectively. The set of all observed symptoms is denoted by $\mathcal{S}_O$. In the process of fault localization, each observed symptom is mapped into the corresponding *path* node of the FPM. End-to-end service diagnosis correlates all observed symptoms to isolate one or more responsible faults, i.e., *link* failures.

In our previous work, two approaches to solving this problem have been proposed. The first technique utilizes Pearl's belief updating [30] for polytrees as an approximation scheme in belief networks with undirected loops and adapts it to calculating the most probable explanation (MPE) of observed symptoms [41], [39]. The second approach obtains the MPE by incrementally updating a set of alternative explanation hypothesis. Hence, the technique is called *incremental hypothesis updating* [36], [40]. In this section, we present a summary of these two techniques.

A. Fault localization through iterative belief updating

The FPM for end-to-end service failure diagnosis can be interpreted as a belief network [30], in which every node represents a binary valued random variable. Symptom observation is represented by assigning 1 to the corresponding belief network node and constitutes a part of *evidence*. In this context, the fault localization problem is to find the most probable assignment of *link* nodes given the observed evidence.

Iterative belief updating proposed by Pearl [30] for singly-connected belief networks utilizes a message passing schema in which the belief network nodes exchange λ and π messages that encode various conditional probabilities [30]. Belief network node X receives messages λ and π from its children and parents, respectively. Based on these messages it calculates new messages λ and π that it sends to its parents and children, respectively. Moreover, node X calculates function $bel : \{0, 1\} \rightarrow [0, 1]$, where $bel(x)$ ($x \in \{0, 1\}$) represents the probability that $X = x$ given the observed evidence. The belief updating algorithm in polytrees starts from an evidence node and propagates the changed belief along the graph edges by computing λ and π messages. In the application to belief networks with undirected loops, several such propagations are performed to enforce the algorithm's convergence.

In the application of iterative belief updating to fault localization (Algorithm 1), one traversal of the entire belief network is performed for every observed symptom. The traversal starts from the belief network node that represents the observed symptom. The network nodes are visited in a breadth-first order while ensuring that no node is visited more than once. The event-driven analysis of all observed symptoms produces, for every fault, the probability of its occurrence given the observed evidence. Based on this information we approximately calculate the MPE using the following procedure. First, we choose a fault represented by a random variable with the highest posterior probability of being equal to 1, where the assignment of 1 indicates the fault's presence. Then we place the chosen fault in the MPE hypothesis, assign 1 to the corresponding random variable, and perform one iteration of belief updating starting from this variable's node. This step is repeated until the following conditions hold: (1) the posterior distribution contains faults whose probability is greater than 0.5, and (2) unexplained symptoms remain in S_O.

The computational complexity of the algorithm is bound by $\mathcal{O}(|\mathcal{S}_O|^2|\mathcal{F}|)$ [41]. In particular, in the application to the problem of end-to-end service failure diagnosis, it is bound by $\mathcal{O}(n^5)$, where n denotes the number of intermediate network nodes such as routers or bridges.

Algorithm 1: MPE through iterative belief updating

FUNCTION *inference*(s_i):
  let o be the breadth-first order starting from node s_i
  FOR EACH *node x such that x is not an unobserved path node, along ordering o* DO
    compute λ and π messages for all parents and children of x, respectively
  DONE
END
Initialization:
  set all λs to 1
Symptom analysis phase:
  FOR EACH *observed symptom s_i* DO *inference*(s_i) DONE
  compute $bel(x)$ for every node x, $x \in \{0, 1\}$
Fault selection phase:
  WHILE $\exists$ *link node f_j for which $bel(1) > 0.5$ and $S_O \neq \emptyset$* DO
    take f_j with the greatest $bel(1)$ and set it to 1
    inference(f_j)
    remove all s_i such that $p(s_i|f_j) > 0$ from S_O
    compute bel for every node f_i
  END

B. Fault localization through incremental hypothesis updating

Incremental hypothesis updating [36] (IHU) creates a set of most likely hypotheses and makes all of them available to a system administrator on a continuous basis. Each hypothesis is a subset of $\mathcal{F}$ that explains all symptoms in $\mathcal{S}_O$. We say that hypothesis $h_j \subseteq \mathcal{F}$ explains symptom $s_i \in \mathcal{S}_O$ if it contains at least one fault that explains s_i. After the i^{th} symptom analysis, the hypotheses are ranked using belief metric b_i. The algorithm proceeds in an event-driven and incremental fashion. The execution triggered by the i^{th} symptom, s_i, creates a set of hypotheses, $\mathcal{H}_i$, each explaining symptoms s_1 through s_i. Set $\mathcal{H}_i$ is created by updating $\mathcal{H}_{i-1}$ with an explanation of symptom s_i. We define H_{s_i} as a set $\{f_k \in \mathcal{F}\}$ such that f_k may cause s_i, i.e., the FPM contains a directed edge from f_k to s_i. Using the notation from [22], H_{s_i} is the domain of symptom s_i.

After the i^{th} symptom is processed, belief metric b_i represents the probability that (1) all faults belonging to h_j have occurred, and (2) h_j explains every observed symptom $s_k \in \mathcal{S}_{O,i} = \{s_1, \ldots, s_i\}$. Formally, $b_i(h_j)$ is defined as follows:

$$b_i(h_j) = \Big(\prod_{f_k \in h_j} p(f_k) \Big) \prod_{s_l \in \mathcal{S}_{O,i}} \Big(1 - \prod_{f_k \in h_j} (1 - p(s_l|f_k) \Big) \tag{1}$$

To incorporate an explanation of symptom s_i into the set of fault hypotheses, in the i^{th} iteration of the algorithm, we analyze each $h_j \in \mathcal{H}_{i-1}$. If h_j is able to explain symptom s_i, we put h_j into $\mathcal{H}_i$. Otherwise, h_j has to be extended by adding to it a fault from H_{s_i}. To avoid a very fast growth in the size of $\mathcal{H}_i$, the following heuristic is used. Fault $f_l \in H_{s_i}$ may be added to $h_j \in \mathcal{H}_{i-1}$ only if $b_i(h_j \cup \{f_l\})$ is bigger than $\mu(f_l)$, the maximum b_i of a hypothesis in $\mathcal{H}_{i-1}$ that contains f_l and explains s_i. While updating the set of hypothesis, $b_i(h_j)$ is approximated iteratively based on $b_{i-1}(h_j)$ using the following equations:

1) If $h_j \in \mathcal{H}_{i-1}$ and h_j explains s_i

$$b_i(h_j) = b_{i-1}(h_j)\Big(1 - \prod_{f_l \in h_j \cap H_{s_i}} (1 - p(s_i|f_l)\Big) \tag{2}$$

2) Otherwise, if f_l explains s_i

$$b_i(h_j \cup \{f_l\}) = b_{i-1}(h_j)\, p(f_l)\, p(s_i|f_l) \tag{3}$$

The upper bound on the worst case computational complexity of the resultant algorithm is $\mathcal{O}(|\mathcal{S}_O|k|\mathcal{F}|)$ [36], where k is the maximum size of the set of hypotheses and k is $\mathcal{O}(|\mathcal{F}|)$ (we use $k = 2|\mathcal{F}|$). When $|\mathcal{H}_i| = k$, a new hypothesis may be added to $\mathcal{H}_i$ only after a hypothesis with the smallest $b_i()$ is removed. In the application to end-to-end service failure diagnosis in an n-node network, the worst case computational complexity of Algorithm 2 is $\mathcal{O}(n^4)$.

Algorithm 2: Incremental hypothesis updating

FUNCTION *inference*(s_i)
  LET $\mathcal{H}_i = \emptyset$
  FOR EACH $f_l \in \mathcal{F}$ LET $\mu(F_l) = 0$ DONE
  FOR EACH $h_j \in \mathcal{H}_{i-1}$ DO
    FOR EACH $f_l \in h_j$ *such that* $f_l \in H_{S_i}$ DO $\mu(f_l) = \max(\mu(F_l), b_i(h_j))$ DONE
    add h_j *to* $\mathcal{H}_i$
  DONE
  FOR EACH $h_j \in \mathcal{H}_{i-1} - \mathcal{H}_i$ DO
    FOR EACH $f_l \in \mathcal{F} \cap H_{S_i}$ *such that* $\mu(f_l) < b_i(h_j \cup \{f_l\})$ DO
      add $h_j \cup \{f_l\}$ *to* $\mathcal{H}_i$
    DONE
  DONE
END

Initialization:
  let $\mathcal{H}_0 = \{\emptyset\}$ *and* $b_0(\emptyset) = 1$

Symptom analysis phase:
  FOR EACH *observed symptom* s_i DO *inference*(s_i) DONE

Fault selection phase:
  choose $h_j \in \mathcal{H}_{|\mathcal{S}_o|}$ *such that* $b_{|\mathcal{S}_o|}(h_j)$ *is maximum*

C. Comparison of techniques

In our previous work [36], [41], Algorithms 1 and 2 were evaluated through simulation in their application to the problem of end-to-end service failure diagnosis. They both proved almost optimally accurate when applied to a well instrumented network [36], [41]. However, Algorithm 2 proved much more efficient allowing the isolation of up to 4 simultaneous faults in a 100-node network in less than 10 seconds. Using a 10-second fault-localization time as an admissibility criterion, the admissible network size for Algorithm 1, in a similar scenario, is 35. In addition, while both algorithms analyze symptoms in an event-driven manner, Algorithm 2 is also incremental at any time offering a complete solution to the symptoms observed thus far. Algorithm 1 requires an additional computation to form an MPE and therefore it can propose a solution, without increasing the complexity bound, only at the end of the fault localization process. Moreover, while Algorithm 1 proposes a single solution, Algorithm 2 makes many solutions available by which it facilitates

an easy hypothesis replacement in case the most probable explanation turns out to be incorrect. On the other hand, Algorithm 1 applies to FPMs of arbitrary shape (and therefore may be applied to almost all fault localization problems), while Algorithm 2 is suitable for bipartite FPMs only.

Both algorithms can be easily enhanced to be resilient against lost and spurious symptoms. Such extensions were proposed in [39] and [40] for Algorithms 1 and 2, respectively. In addition, they were both shown resilient to the inaccuracies of the probabilistic FPM. In Table I, we summarize and compare the most important features of Algorithms 1 and 2.

TABLE I

SUMMARY OF THE MOST IMPORTANT FEATURES OF ALGORITHMS 1 AND 2

	Algorithm 1	Algorithm 2
Computational complexity in end-to-end service failure diagnosis	$\mathcal{O}(n^5)$	$\mathcal{O}(n^4)$
Admissible network size	35	100
Fault propagation model	arbitrary	bipartite only
Isolates multiple simultaneous faults	YES	YES
Resilient to observation noise	YES	YES
Does not require exact probabilistic model	YES	YES
Event-driven	YES	YES
Incremental	NO	YES
Multiple alternative solutions	NO	YES

III. MULTI-DOMAIN APPROACH TO END-TO-END SERVICE FAILURE DIAGNOSIS

In this section, we introduce a multi-domain approach to probabilistic diagnosis of end-to-end service failures in hierarchically organized networks. (The technique presented in this paper is a continuation of the initial study on multi-domain diagnosis presented in [38] and is free from accuracy and performance problems thereof.) The approach takes advantage of the domain semantics of real-life communication systems. The management domains considered by the technique correspond to administrative or routing network domains. We adopt the hierarchical organization of the management system, in which network domains $\mathcal{D}_i$ are managed by separate managers DM_i. If $\mathcal{D}_i$ has sub-domains, the managers of these sub-domains report to DM_i. Thus, the management hierarchy established by domain managers (DMs) is isomorphic to the domain relationship graph.

The multi-domain fault localization algorithm relies on the cooperation among DMs. Symptoms are typically observed by DMs at the lowest-level of the management hierarchy, as they are usually reported by either source or destination node of a failed end-to-end path. A DM begins the diagnosis of an end-to-end path failure only if all of nodes of the path are located in its domain. Otherwise, the corresponding symptom is delegated to the higher-level manager. While analyzing the failure of an end-to-end path that was reported to it by a DM, the higher-level manager coordinates the actions of DMs that manage domains traversed by the failed path. In particular, the higher-level manager delegates some tasks involved in the diagnosis of the path failure to DMs of domains traversed by the failed path.

Although the technique proposed in this paper may be applied in networks with multiple levels of the hierarchy, for simplicity, we focus on a two-level architecture. Consequently, we use $\mathcal{N}$ and $\mathcal{D}_i$ to denote the entire network and its sub-domain, respectively. At the root of the management hierarchy we position a network manager NM which oversees and coordinates the operation of domain managers DM_i.

We introduce the following notation.

$n_k \rightarrow n_l$	A directed link from node n_k to node n_l, where n_k and n_l are node identifiers that are unique network-wide, e.g., IP addresses
$n_{p_1} \stackrel{*}{\rightarrow} n_{p_m}$	A directed, possibly multi-hop path from node n_{p_1} to node n_{p_m} consisting of links $n_{p_1} \rightarrow n_{p_2}$, ..., $n_{p_{m-1}} \rightarrow n_{p_m}$.

$s : n_k \xrightarrow{*} n_l$	A symptom indicating a failure of path $n_k \xrightarrow{*} n_l$
$f : n_k \rightarrow n_l$	A fault associated with link $n_k \rightarrow n_l$
$i \xrightarrow{*} j$	The set of all paths that begin in domain $\mathcal{D}_i$ and end in domain $\mathcal{D}_j$, i.e., $i \xrightarrow{*} j = \{n_k \xrightarrow{*} n_l \mid n_k \in \mathcal{D}_i \text{ and } n_l \in \mathcal{D}_j\}$, where i and j are unique domain identifiers, e.g., IP subnet masks.
$s : i \xrightarrow{*} j$	A symptom associated with the set of paths $i \xrightarrow{*} j$. We say that symptom $s : i \xrightarrow{*} j$ occurred when at least one $s : n_k \xrightarrow{*} n_l$ occurred such that $n_k \in \mathcal{D}_i$ and $n_l \in \mathcal{D}_j$.
$d(n_k)$	A function mapping a node identifier into an identifier of a domain to which the node belongs. In IP networks, function $d(n_k)$ is implemented using an IP address mask.

For an end-to-end path $n_{p_1} \xrightarrow{*} n_{p_m}$ consisting of links $n_{p_1} \rightarrow n_{p_2}$, ..., $n_{p_{m-1}} \rightarrow n_{p_m}$ we define the following concepts.

Definition 1: Path $n_{p_1} \xrightarrow{*} n_{p_m}$ *traverses* $\mathcal{D}_i$ iff $\exists n_{p_j} | n_{p_j} \in \mathcal{D}_i$. Path $n_{p_1} \xrightarrow{*} n_{p_m}$ is an *intra-domain path* in $\mathcal{D}_i$ if $\forall n_{p_j} | n_{p_j} \in \mathcal{D}_i$. Path $n_{p_1} \xrightarrow{*} n_{p_m}$ that traverses $\mathcal{D}_i$ but is not an intra-domain path in $\mathcal{D}_i$ is an *inter-domain path* with respect to $\mathcal{D}_i$.

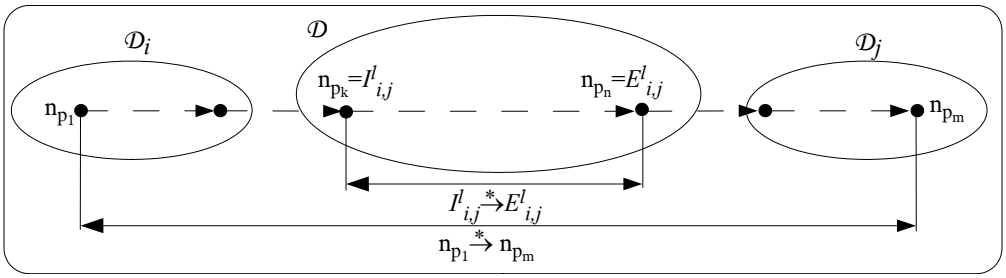


Fig. 2. Definition of a path segment, and ingress and egress gateways.

Definition 2: Let path $n_{p_1} \xrightarrow{*} n_{p_m}$ be an inter-domain path with respect to $\mathcal{D}_l$. Let $\mathcal{D}_i$ and $\mathcal{D}_j$ be domains such that $n_{p_1} \in \mathcal{D}_i$ and $n_{p_m} \in \mathcal{D}_j$. Node n_{p_k} such that $1 < k \leq m$, $n_{p_k} \in \mathcal{D}_l$, and $n_{p_{k-1}} \notin \mathcal{D}_l$ is an *ingress gateway* from $\mathcal{D}_i$ to $\mathcal{D}_j$ in $\mathcal{D}_l$ and is denoted by $I^l_{i,j}$. Similarly, node n_{p_n} such that $1 \leq n < m$, $n_{p_n} \in \mathcal{D}_l$, and $n_{p_{n+1}} \notin \mathcal{D}_l$ is an *egress gateway* from $\mathcal{D}_i$ to $\mathcal{D}_j$ in $\mathcal{D}_l$ and is denoted by $E^l_{i,j}$. (Figure 2). When routes are bidirectional, for any i, j, and l, $I^l_{i,j} = E^l_{j,i} = G^l_{i,j}$ and $I^l_{j,i} = E^l_{i,j} = G^l_{j,i}$.

Definition 3: Let path $n_{p_1} \xrightarrow{*} n_{p_m}$ such that $n_{p_1} \in \mathcal{D}_i$ and $n_{p_m} \in \mathcal{D}_j$ be an inter-domain path with respect to $\mathcal{D}_l$. Path $I^l_{i,j} \xrightarrow{*} E^l_{i,j}$ is called an *intra-$\mathcal{D}_l$ segment* of path $n_{p_1} \xrightarrow{*} n_{p_m}$ (Figure 2).

We make the following simplifying assumptions, which are usually valid in the case of hierarchically routed networks.

1) Management domains are either disjoint or all-inclusive, i.e., for any $\mathcal{D}_i$ and $\mathcal{D}_j$, either $\mathcal{D}_i \cap \mathcal{D}_j = \emptyset$ or either $\mathcal{D}_i = \mathcal{D}_j$.
2) No path enters the same domain more than once, i.e., for any path $n_{p_1} \xrightarrow{*} n_{p_m}$ consisting of links $n_{p_1} \rightarrow n_{p_2}$, ..., $n_{p_{m-1}} \rightarrow n_{p_m}$, if $d(n_{p_i}) \neq d(n_{p_{i+1}})$ then for all j, such that $m > j > i + 1$, $d(n_{p_j}) \neq d(n_{p_i})$.
3) All end-to-end paths that begin in domain $\mathcal{D}_i$, end in domain $\mathcal{D}_j$, and traverse domain $\mathcal{D}_l$ enter $\mathcal{D}_l$ through the same node, i.e., if $n_{p_1} \xrightarrow{*} n_{p_m}$ and $n_{q_1} \xrightarrow{*} n_{q_n}$ are two paths that traverse links $n_{p_1} \rightarrow n_{p_2}$, ..., $n_{p_{m-1}} \rightarrow n_{p_m}$ and $n_{q_1} \rightarrow n_{q_2}$, ..., $n'_{q_{n-1}} \rightarrow n'_{q_n}$, respectively, such that $n_{p_1}, n_{q_1} \in \mathcal{D}_i$, $n_{p_m}, n_{q_n} \in \mathcal{D}_j$, and both paths traverse domain $\mathcal{D}_l$, then for n_{p_t} and n_{q_s} such that $n_{p_{t-1}}, n_{q_{s-1}} \notin \mathcal{D}_l$ and $n_{p_t}, n_{q_s} \in \mathcal{D}_l$, $n_{p_t} = n_{q_s} = I^l_{i,j}$. In addition, $n_{p_1} \xrightarrow{*} n_{p_m}$ and $n_{q_1} \xrightarrow{*} n_{q_n}$ leave $\mathcal{D}_i$ through the same node, i.e., for n_{p_t} and n_{q_s} such that $n_{p_t}, n_{q_s} \in \mathcal{D}_l$ and $n_{p_{t+1}}, n_{q_{s+1}} \notin \mathcal{D}_l$, $n_{p_t} = n_{q_s} = E^l_{i,j}$.

The solution proposed in this paper assumes that every DM has the minimum knowledge necessary for

fault diagnosis, i.e., it is able to obtain topology and routing information only in the domain it directly manages. Thus, DM$_i$ is aware of link $n_k \rightarrow n_l$ if and only if both n_k and n_l belong to $\mathcal{D}_i$, whereas NM is aware of link $n_k \rightarrow n_l$ if and only if $n_k \rightarrow n_l$ is a link between $\mathcal{D}_i$ and $\mathcal{D}_j$, and nodes n_k and n_l are egress and ingress gateways in $\mathcal{D}_i$ and $\mathcal{D}_j$, respectively. Consequently, NM is able to transform any path $n_{p_1} \stackrel{*}{\rightarrow} n_{p_m}$ that traverses domains $\mathcal{D}_{l_1}, \ldots, \mathcal{D}_{l_k}$ into a sequence of intra-domain path segments and links $n_{p_1} \stackrel{*}{\rightarrow} E^{l_1}_{l_1,l_k}$, $E^{l_1}_{l_1,l_k} \rightarrow I^{l_2}_{l_1,l_k}$, $I^{l_2}_{l_1,l_k} \stackrel{*}{\rightarrow} E^{l_2}_{l_1,l_k}$, $\ldots$, $E^{l_{k-1}}_{l_1,l_k} \rightarrow I^{l_k}_{l_1,l_k}$, $I^{l_k}_{l_1,l_k} \stackrel{*}{\rightarrow} n_{p_m}$ (Figure 3). Moreover, DM$_i$ is able to obtain a complete route for each end-to-end path $n_k \stackrel{*}{\rightarrow} n_l$ such that $d(n_k) = d(n_l) = i$, but it cannot obtain the topology and routing information for any parts of the network located outside of $\mathcal{D}_i$. Consequently, DM$_i$ is unable to determine either a complete route or a path-segment sequence for any path that is inter-domain with respect to $\mathcal{D}_i$.

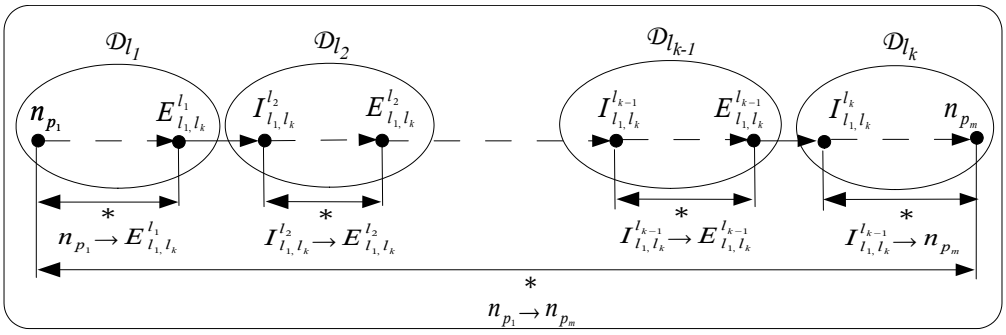


Fig. 3. Transformation of an end-to-end path into a sequence of inter-domain links and intra-domain path segments.

This property of the proposed solution affects the way a FPM for multi-domain fault diagnosis and a fault localization algorithm are designed, which are introduced in Sections IV and V, respectively.

IV. Distributed fault propagation model

Recall from Section II that a fault propagation model (FPM) for end-to-end service-failure diagnosis is a bipartite directed graph with host-to-host and end-to-end service failures at the tails and at the heads of the graph edges, respectively. In the multi-domain solution, the fault propagation model (FPM) of the entire network is distributed among DMs. Each manager maintains a part of the distributed FPM that represents the manager's knowledge of the system structure. An FPM built by DM$_i$ is a bipartite causality graph with end-to-end and host-to-host service failures at the heads and at the tails of the edges, respectively, similar to the model described in Section II. However, in the multi-domain approach, the FPM of DM$_i$ includes failures of only these end-to-end paths and host-to-host links that are entirely located in $\mathcal{D}_i$. Similarly, the FPM of NM includes failures of links that join different domains of $\mathcal{N}$. Failures of links that are completely contained in domains of $\mathcal{N}$ but which may affect end-to-end paths that span multiple domains are not explicitly included in the FPM of NM but are represented in it by proxy fault nodes, called $\mathcal{P}$-nodes. Similarly, failures located outside $\mathcal{D}_i$ that may result in an observation of a symptom corresponding to an end-to-end path located in $\mathcal{D}_i$ are represented in the FPM of DM$_i$ by proxy fault nodes, called $\tilde{\mathcal{P}}$-nodes. Thus, the FPM built by DM$_i$ has the same structure as in the centralized approach, but its scope is smaller and the interpretation of some of the nodes is different. Similar to the centralized approach, multiple failure modes can be included in the distributed FPM. In this case, an FPM of each DM$_i$ consists of multiple, possibly overlapping copies of such bipartite causality graph. However, to clarify the presentation of the technique, the description provided in this paper assumes that only one failure type is associated with every end-to-end path and host-to-host link.

A. Fault propagation model of the NM

Let us consider path $n_{p_1} \stackrel{*}{\rightarrow} n_{p_m}$ that traverses domains $\mathcal{D}_{l_1}, \ldots, \mathcal{D}_{l_k}$. Recall that NM transforms this path into a sequence of intra-domain path segments and links $n_{p_1} \stackrel{*}{\rightarrow} E^{l_1}_{l_1,l_k}$, $E^{l_1}_{l_1,l_k} \rightarrow I^{l_2}_{l_1,l_k}$, $I^{l_2}_{l_1,l_k} \stackrel{*}{\rightarrow} E^{l_2}_{l_1,l_k}$, $\ldots$, $E^{l_{k-1}}_{l_1,l_k} \rightarrow I^{l_k}_{l_1,l_k}$, $I^{l_k}_{l_1,l_k} \stackrel{*}{\rightarrow} n_{p_m}$ (Figure 3). In its FPM, NM has to represent the fact that a failure of end-to-end

path $n_{p_1} \xrightarrow{*} n_{p_m}$ may be caused by failures of one or more of these links and path segments. This can be achieved by creating a symptom node representing path $n_{p_1} \xrightarrow{*} n_{p_m}$ and fault nodes representing failures of its corresponding links and intra-domain path segments. However, we can observe that, with the exception of $n_{p_1} \xrightarrow{*} E^{l_1}_{l_1,l_k}$ and $I^{l_k}_{l_1,l_k} \xrightarrow{*} n_{p_m}$, all paths that begin in $\mathcal{D}_{l_1}$ and end in $\mathcal{D}_{l_k}$ are transformed into the same sequence of intra-domain path segments and links. (This follows from the hierarchical routing assumption.) Therefore, we can simplify the FPM by creating a single symptom node labeled $s : l_1 \xrightarrow{*} l_k$ (Figure 4) that represents all paths that begin in $\mathcal{D}_{l_1}$ and end in $\mathcal{D}_{l_k}$. For any such path $n_{p_1} \xrightarrow{*} n_{p_m}$ we say that node $s : l_1 \xrightarrow{*} l_k$ represents symptom $s : n_{p_1} \xrightarrow{*} n_{p_m}$ in the FPM of NM. Symptom $s : l_1 \xrightarrow{*} l_k$ occurs when a failure of at least one path that begins in $\mathcal{D}_{l_1}$ and ends in $\mathcal{D}_{l_k}$, e.g., $n_{p_1} \xrightarrow{*} n_{p_m}$, occurs.

The failure of $n_{p_1} \xrightarrow{*} n_{p_m}$ may be caused by a failure of one or more inter-domain links or intra-domain segments of $n_{p_1} \xrightarrow{*} n_{p_m}$. Therefore, in the FPM of NM, two types of fault nodes have to exist: (1) ordinary fault nodes, like ones in the centralized case, which represent failures of inter-domain links; these faults are directly isolated by NM, (2) proxy fault nodes, which represent failures that cannot be isolated by NM alone because they are located in domains that are not directly managed by NM. For every domain, one or more such proxy nodes are created as follows.

- For every ingress gateway node in $\mathcal{D}_i$, $I^i_{l,i}$, we create node $\mathcal{P} : I^i_{l,i} \xrightarrow{*} *$ that represents the set of intra-$\mathcal{D}_i$ paths that begin in node $I^i_{l,i}$. We write that $\mathcal{P} : I^i_{l,i} \xrightarrow{*} * = \{I^i_{l,i} \xrightarrow{*} n_r | n_r \in \mathcal{D}_i\}$.
- For every egress gateway node in $\mathcal{D}_i$, $E^i_{i,k}$, we create node $\mathcal{P} : * \xrightarrow{*} E^i_{i,k}$ that represents the set of intra-$\mathcal{D}_i$ paths that end in node $E^i_{i,k}$. We write that $\mathcal{P} : * \xrightarrow{*} E^k_{k,\cdot} i = \{n_r \xrightarrow{*} E^i_{i,k} | n_r \in \mathcal{D}_i\}$.
- Moreover, for each pair of gateway nodes $I^i_{l,k}$ and $E^i_{l,k}$, we create node $\mathcal{P} : I^i_{l,k} \xrightarrow{*} E^i_{l,k}$ that represents the intra-$\mathcal{D}_i$ path segment $I^i_{l,k} \xrightarrow{*} E^i_{l,k}$, i.e., $\mathcal{P} : I^i_{l,k} \xrightarrow{*} E^i_{l,k} = \{I^i_{l,k} \xrightarrow{*} E^i_{l,k}\}$.

In the FPM of NM, symptom node $s : l_1 \xrightarrow{*} l_k$ is connected to nodes $\mathcal{P} : * \xrightarrow{*} E^{l_1}_{l_1,l_k}$, $f : E^{l_1}_{l_1,l_k} \rightarrow I^{l_2}_{l_1,l_k}$, $\mathcal{P} : I^{l_2}_{l_1,l_k} \xrightarrow{*} E^{l_2}_{l_1,l_k}$, ..., $f : E^{l_{k-1}}_{l_1,l_k} \rightarrow I^{l_k}_{l_1,l_k}$, $\mathcal{P} : I^{l_k}_{l_1,l_k} \xrightarrow{*} *$ (Figure 4). Overall, the FPM of NM contains multiple such symptom nodes for all pairs of domains in $\mathcal{N}$. These symptom nodes are connected to overlapping sets of fault and $\mathcal{P}$-nodes. Thus, the FPM of NM is a connected bipartite graph.

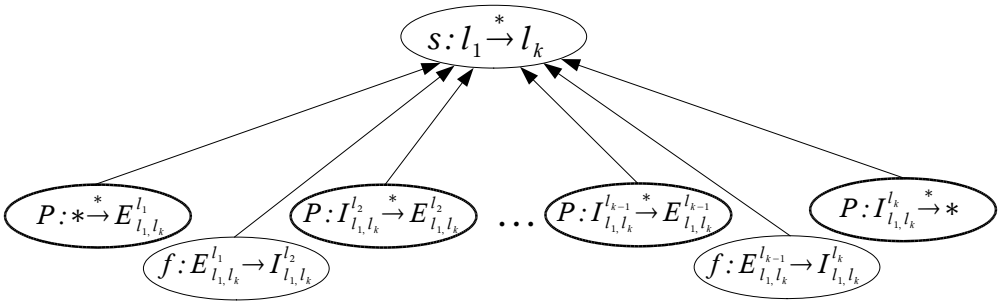


Fig. 4. Construction of FPM for NM including $\mathcal{P}$-nodes representing domains.

Example 1:

Consider the network in Figure 5. We will describe the FPM built by the NM.

Figure 6 presents the FPM build by NM while assuming that only one fault exists in the system per every end-to-end or host-to-host service, and that all routes are bidirectional. The FPM contains three symptom nodes that represent paths between domains $\mathcal{D}_1$ and $\mathcal{D}_2$, $\mathcal{D}_1$ and $\mathcal{D}_3$, and $\mathcal{D}_2$ and $\mathcal{D}_3$. Let us consider paths between domains $\mathcal{D}_2$ and $\mathcal{D}_1$. Since there is no direct route between these two domains, connectivity between them is provided via domain $\mathcal{D}_3$, in particular via gateways 3.1 and 3.5. Thus, an end-to-end path $1.n_1 \xleftrightarrow{*} 2.n_2$ between these two domains is decomposed into the following sequence of inter-domain links and intra-domain paths: $1.n_1 \xleftrightarrow{*} 1.1$, $1.1 \leftrightarrow 3.1$, $3.1 \xleftrightarrow{*} 3.5$, $3.5 \leftrightarrow 2.5$, and $2.5 \xleftrightarrow{*} 2.n_2$. Thus, in the FPM of NM, node $s : 1 \xleftrightarrow{*} 2$ is connected to link nodes $f : 1.1 \leftrightarrow 3.1$ and $f : 3.5 \leftrightarrow 2.5$, and to $\mathcal{P}$-nodes $\mathcal{P} : * \xleftrightarrow{*} 1.1$,$\mathcal{P} : 3.1 \xleftrightarrow{*} 3.5$, and $\mathcal{P} : 2.5 \xleftrightarrow{*} *$.

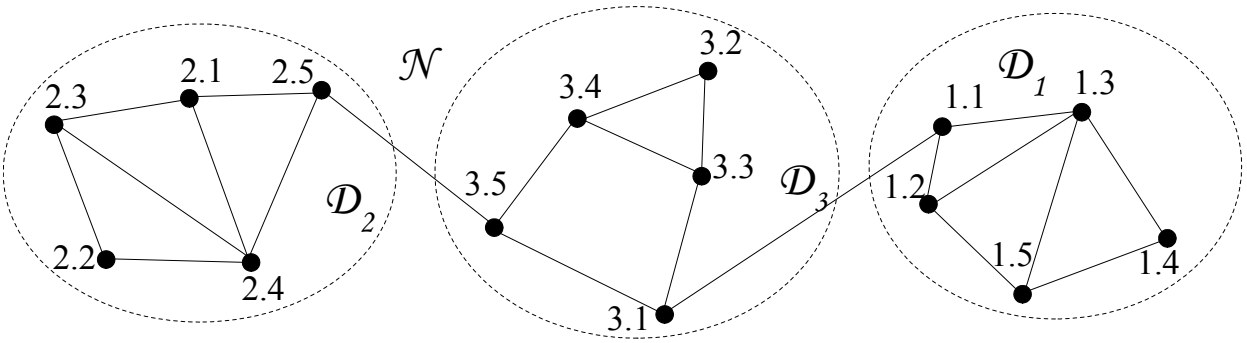


Fig. 5. Example three-domain network topology.

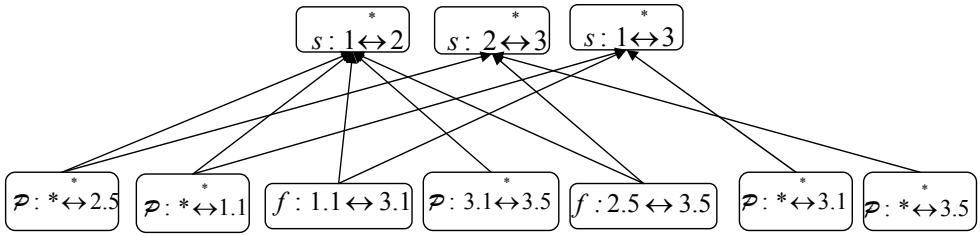


Fig. 6. Fault propagation model created by NM in Figure 5. The model assumes that all routes and path failures are bidirectional, and it allows only one failure type per path or link.

The final step in the creation of the FPM for NM is assigning prior failure probabilities to $\mathcal{P}$-nodes and conditional probabilities to causal edges between $\mathcal{P}$-nodes and symptom nodes. Conditional probabilities between $\mathcal{P}$-nodes and symptom nodes do not have any intuitive interpretation. The approach chosen in this paper assigns all conditional probabilities between $\mathcal{P}$-nodes and symptom nodes to 1. The strength with which faults located in sub-domains influence failures of paths that span multiple domains is modeled using prior failure probabilities assigned to $\mathcal{P}$-nodes. $\mathcal{P}$-nodes do not have real-life correspondents, either, since they are synthetic elements. As a result, their prior failure probabilities cannot be either assigned by an expert or learned through system observation, as it is the case with ordinary fault nodes. In addition, $\mathcal{P}$-nodes represent failures located in other domains and their prior failure probabilities change during the process of fault localization. Thus, prior failure probabilities associated with $\mathcal{P}$-nodes $\mathcal{P}: I^i_{l,i} \overset{*}{\rightarrow} *$, $\mathcal{P}: * \overset{*}{\rightarrow} E^i_{i,k}$, and $\mathcal{P}: I^i_{l,k} \overset{*}{\rightarrow} E^i_{l,k}$ in the FPM of NM must be calculated by the multi-domain technique based on the state of the fault localization process in $\mathcal{D}_i$. Since this state is not accessible to NM, the probabilities have to be calculated by DM$_i$. The process in which it is done is discussed in Section V.

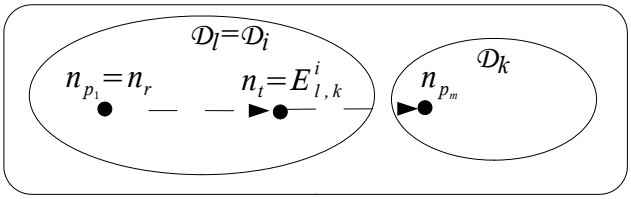
B. Fault propagation model of a DM

As it was stated at the beginning of this section, the FPM built by DM$_i$ includes all intra-$\mathcal{D}_i$ paths and links, i.e., all the information contained in the centralized model of $\mathcal{D}_i$. Such a model is sufficient for the diagnosis of symptoms observed in $\mathcal{D}_i$, but is not sufficient for the diagnosis of symptoms DM$_i$ receives from NM. In the hierarchical fault management solution presented in this paper, diagnosis of a path failure is delegated up and down the management hierarchy until managers of all domains traversed by the path are notified. In particular, NM may delegate to DM$_i$ a part of a task involved in the diagnosis of path $n_{p_1} \overset{*}{\rightarrow} n_{p_m}$ that traverses $\mathcal{D}_i$. In this case, DM$_i$ will be notified about a failure of its intra-domain path that constitutes the intra-$\mathcal{D}_i$ path segment of $n_{p_1} \overset{*}{\rightarrow} n_{p_m}$. Observe that this notification does not mean that the intra-$\mathcal{D}_i$ path has necessarily failed. It only indicates a possibility of this segment's failure, since $n_{p_1} \overset{*}{\rightarrow} n_{p_m}$ could have been caused by its path-segment or link that is not located in domain $\mathcal{D}_i$. Thus, symptoms received by DM from NM are typically associated with a high degree of uncertainty, i.e., they are likely to be spurious. In

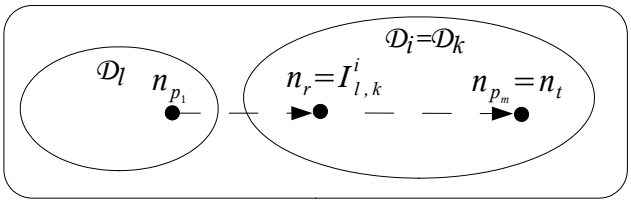
our previous work [39], we presented a solution that allows us to incorporate spurious symptoms in an FPM. In this section, we use these ideas as follows.

Let $s : n_r \xrightarrow{*} n_t$ be an intra-$\mathcal{D}_i$ symptom received by DM$_i$ from NM in the process of diagnosing a failure of inter-domain path $n_{p_1} \xrightarrow{*} n_{p_m}$. To model the possibility that $s : n_r \xrightarrow{*} n_t$ is spurious in the FPM of DM_i, we create a proxy fault node, called $\tilde{\mathcal{P}}$-node that represents all possible causes of $s : n_r \xrightarrow{*} n_t$ that are not located in $\mathcal{D}_i$. Observe that, since path $n_r \xrightarrow{*} n_t$ constitutes a segment of an inter-domain path, at least one of nodes n_r, n_t is a gateway node in $\mathcal{D}_i$. Let l and k be identifiers of domains that contain nodes n_{p_1} and n_{p_m}, respectively. Consider three possible cases.

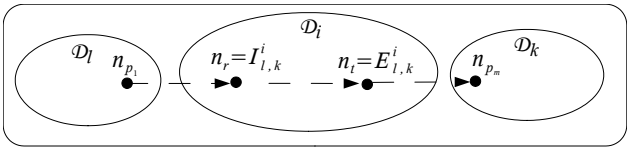
- $i = l$, and in consequence $n_t = E^i_{l,k} = E^i_{i,k}$ (Figure 7(a)). We create node $\tilde{\mathcal{P}} : * \xrightarrow{*} E^i_{i,k}$ and connect it to $s : n_r \xrightarrow{*} n_t$.
- $i = k$, and in consequence $n_r = I^i_{l,k} = I^i_{l,i}$ (Figure 7(b)). We create node $\tilde{\mathcal{P}} : I^i_{l,i} \xrightarrow{*} *$ and connect it to $s : n_r \xrightarrow{*} n_t$.
- $i \neq l$ and $i \neq k$, and in consequence $n_r = I^i_{l,k}$ and $n_t = E^i_{l,k}$ (Figure 7(c)). We create node $\tilde{\mathcal{P}} : I^i_{l,k} \xrightarrow{*} E^i_{l,k}$ and connect it to $s : n_r \xrightarrow{*} n_t$.



(a) $i = l$



(b) $i = k$



(c) $i \neq k$ and $i \neq l$

Fig. 7. Definition of proxy nodes in the FPM of DM$_i$.

Observe that each $\tilde{\mathcal{P}}$-node $\tilde{\mathcal{P}} : * \xrightarrow{*} E^i_{i,k}$, $\tilde{\mathcal{P}} : I^i_{l,i} \xrightarrow{*} *$, or $\tilde{\mathcal{P}} : I^i_{l,k} \xrightarrow{*} E^i_{l,k}$ in the FPM of DM$_i$ corresponds to $\mathcal{P}$-node $\mathcal{P} : * \xrightarrow{*} E^i_{i,k}$, $\mathcal{P} : I^i_{l,i} \xrightarrow{*} *$, or $\mathcal{P} : I^i_{l,k} \xrightarrow{*} E^i_{l,k}$, respectively, in the FPM of NM. Similar to what we did in the case of NM, conditional probabilities on edges between $\tilde{\mathcal{P}}$-nodes and symptom nodes in the FPM of DM$_i$ are set to 1, while prior failure probabilities assigned to $\tilde{\mathcal{P}}$-nodes in the FPM of DM$_i$ are calculated by NM and sent to DM$_i$ together with reported symptoms. In fact, in the FPM of DM$_i$, $\tilde{\mathcal{P}}$-nodes can be created dynamically when corresponding symptoms are reported by the NM.

Example 2:

Consider the network in Figure 5. Figure 8 presents the FPM build by DM$_3$ while assuming that only one fault exists in the system per every end-to-end or host-to-host service and that all routes are bidirectional. The FPM contains symptom nodes that represent all intra-$\mathcal{D}_3$ paths, which are connected to intra-$\mathcal{D}_3$ links that are used to provide these paths. This part of the model is identical to the centralized model in Figure 1(b), since $\mathcal{D}_3$ has the same topology and routing as the network in Figure 1(a).

In addition to the fault nodes, in the distributed FPM, the FPM built by DM$_3$ includes three $\tilde{\mathcal{P}}$-nodes. Since

there are two gateways in $\mathcal{D}_3$, i.e., nodes 3.1 and 3.5, the $\tilde{\mathcal{P}}$-nodes that need to be created are $\tilde{\mathcal{P}} : * \overset{*}{\leftrightarrow} 3.1$, $\tilde{\mathcal{P}} : * \overset{*}{\leftrightarrow} 3.5$, and $\tilde{\mathcal{P}} : 3.1 \overset{*}{\leftrightarrow} 3.5$. Node $\tilde{\mathcal{P}} : 3.1 \overset{*}{\leftrightarrow} 3.5$ is connected only to symptom node $s : 3.1 \overset{*}{\leftrightarrow} 3.5$. Node $\tilde{\mathcal{P}} : * \overset{*}{\leftrightarrow} 3.1$ is connected to symptom nodes $s : 3.1 \overset{*}{\rightarrow} 3.2$, $s : 3.1 \overset{*}{\rightarrow} 3.3$, and $s : 3.1 \overset{*}{\rightarrow} 3.4$, while node $\tilde{\mathcal{P}} : * \overset{*}{\leftrightarrow} 3.5$ is connected to nodes $3.2 \overset{*}{\leftrightarrow} 3.5$, $3.3 \overset{*}{\leftrightarrow} 3.5$, and $3.4 \overset{*}{\leftrightarrow} 3.5$. Note that other symptom nodes do not have $\tilde{\mathcal{P}}$-nodes associated with them, since their corresponding paths may not be segments on any intra-domain paths.

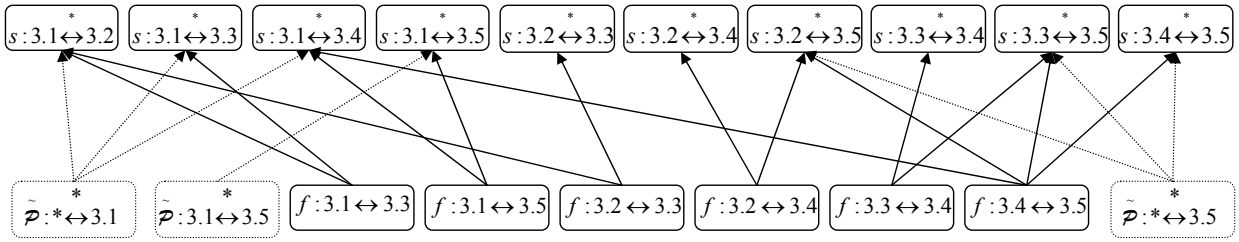


Fig. 8. Fault propagation model created by DM_3 in Figure 5. The model assumes that all routes and path failures are bidirectional, and it allows only one failure type per path or link.

V. Multi-domain fault localization algorithm

In this section, we present an outline of a multi-domain fault localization algorithm (Algorithm 3) based on the distributed fault propagation model described in Section IV. We propose a general design of the algorithm, which may be refined to create multi-domain versions of both Algorithms 1 and 2. In the pseudo-code in Section V-B, parts of the algorithm that need to be specialized for different probabilistic reasoning mechanisms are underlined. The refinements of these parts are be presented in Sections VI and VII for Algorithms 1 and 2, respectively.

Similar to the centralized algorithms, the multi-domain fault localization algorithm proceeds in three phases performed by every DM and NM: (1) model initialization, (2) symptom analysis, and (3) fault selection. In the initialization phase (see Algorithm 3), the model is reset by assigning prior failure probabilities to proxy nodes. In our implementation, these probabilities are set to 0 in the FPM of NM, while in the FPM of DM, no $\tilde{\mathcal{P}}$-nodes exist at the beginning and therefore no assignment is needed.

Symptom-analysis and fault-selection phases progress by traversing the hierarchy of managers in a bottom-up or top-down manner, where *bottom-up* and *top-down* indicate the direction of the information flow. Processing performed by DM or NM is triggered by a symptom arrival or by a message received from NM or DM, respectively. For the clarity of presentation, in the pseudo-code of Algorithm 3 we use function calls to indicate the exchange of information between managers. In the distributed implementation, the function calls should be implemented by asynchronous message exchange rather than RPC-style invocations.

A. Symptom analysis phase

The symptom analysis phase is executed for every received alarm that indicates a failure of an end-to-end path between two nodes. This alarm can be received either by the NM or a DM. A DM can start the symptom analysis only if the entire failed path belongs to its domain. If the DM is not able to diagnose the symptom it forwards it to the NM, which initiates the symptom diagnosis (function *analyze_internal*).

1) Symptom processing by NM: In the process of diagnosing symptom $s : n_{p_1} \overset{*}{\rightarrow} n_{p_m}$, the NM first maps it into a symptom node $s : l_1 \overset{*}{\rightarrow} l_k$ in its FPM, such that $n_{p_1} \in \mathcal{D}_{l_1}$ and $n_{p_m} \in \mathcal{D}_{l_k}$. Then it splits the failed path into its intra-domain path segments and inter-domain links, i.e., into a sequence $n_{p_1} \overset{*}{\rightarrow} E^{l_1}_{l_1,l_k}$, $E^{l_1}_{l_1,l_k} \rightarrow I^{l_2}_{l_1,l_k}$, $I^{l_2}_{l_1,l_k} \overset{*}{\rightarrow} E^{l_2}_{l_1,l_k}$, ..., $E^{l_{k-1}}_{l_1,l_k} \rightarrow I^{l_k}_{l_1,l_k}$, $I^{l_k}_{l_1,l_k} \overset{*}{\rightarrow} n_{p_m}$. Segments $n_{p_1} \overset{*}{\rightarrow} E^{l_1}_{l_1,l_k}$, $I^{l_2}_{l_1,l_k} \overset{*}{\rightarrow} E^{l_2}_{l_1,l_k}$,

..., $I^{l_k}_{l_1,l_k} \xrightarrow{*} n_{p_m}$ are then interpreted as symptoms $s_1 = s : n_{p_1} \xrightarrow{*} E^{l_1}_{l_1,l_k}$, $s_2 = s : I^{l_2}_{l_1,l_k} \xrightarrow{*} E^{l_2}_{l_1,l_k}$, ..., $s_k = s : I^{l_k}_{l_1,l_k} \xrightarrow{*} n_{p_m}$ that will be reported to DM$_{l_1}$, DM$_{l_2}$, ..., DM$_{l_k}$, respectively. Since high uncertainty is associated with these symptoms, the NM needs to calculate the probability with which the symptoms should be considered spurious by DMs. Note that in the FPM of DM$_{l_j}$, all causes of symptom s_j reported by NM that are not located in $\mathcal{D}_{l_j}$ are represented by a $\tilde{\mathcal{P}}$-node that is attached to node s_j. Let us label this node $\tilde{\mathcal{P}}(s_j)$. Thus, the NM needs to calculate the prior probability $p(\tilde{\mathcal{P}}(s_j))$ that should be associated with $\tilde{\mathcal{P}}(s_j)$ in the FPM of DM$_i$.

Recall that, in the FPM of NM, s_j is represented by one or more $\mathcal{P}$-nodes. Let $s_j = s : n_r \rightarrow n_t$. If n_r is an ingress gateway in $\mathcal{D}_i$ and n_t is an egress gateway in $\mathcal{D}_i$, then s_j is represented by $\mathcal{P}$-nodes $\mathcal{P} : n_r \xrightarrow{*} n_t$, $\mathcal{P} : n_r \xrightarrow{*} *$, and $\mathcal{P} : * \xrightarrow{*} n_t$. If n_r is an ingress gateway in $\mathcal{D}_i$, and n_t is not an egress gateway in $\mathcal{D}_i$, then s_j is represented by $\mathcal{P}$-node $\mathcal{P} : n_r \xrightarrow{*} *$. Finally, if n_t is an egress gateway in $\mathcal{D}_i$ and n_r is not an ingress gateway in $\mathcal{D}_i$, then s_j is represented by $\mathcal{P}$-node $\mathcal{P} : * \xrightarrow{*} n_t$. The value of $p(\tilde{\mathcal{P}}(s_j))$ is calculated using function *compute_spurious*, which is defined as follows. (The underlined parts of code need to be defined separately for Algorithms 1 and 2.)

FUNCTION *compute_spurious*($s : n_r \rightarrow n_t$)
  IF n_r *is an ingress gateway* AND n_t *is an egress gateway* THEN
    set $\mathcal{P}_{\text{set}} = \{\mathcal{P} : n_r \xrightarrow{*} n_t, \mathcal{P} : * \xrightarrow{*} n_t, \mathcal{P} : n_r \xrightarrow{*} *\}$
    RETURN $\prod_{\mathcal{P} \in \mathcal{P}_{\text{set}}}$ $\underline{not\_proxy(\mathcal{P}|\textit{evidence observed by}\ \text{NM})}$
  ELSE IF n_r *is an ingress gateway* THEN
      RETURN $\underline{not\_proxy(\mathcal{P} : n_r \xrightarrow{*} *|\textit{evidence observed by}\ \text{NM})}$
  ELSE
      RETURN $\underline{not\_proxy(\mathcal{P} : * \xrightarrow{*} n_t|\textit{evidence observed by}\ \text{NM})}$
END

After calculating $p(\tilde{\mathcal{P}}(s_j))$ the NM delegates the diagnosis of symptom s_j to DM$_{l_j}$, for $j = 1 \ldots k$ (function *analyze_external*). Along the symptom, the NM passes $p(\tilde{\mathcal{P}}(s_j))$. It also sends $\mathcal{P}_j$, a description of the $\mathcal{P}$-node connected to $s : l_1 \xrightarrow{*} l_k$ that represents $\mathcal{D}_{l_j}$ in the FPM of NM. This description may be in the form of, e.g., an IP-address mask. The last argument is needed so that DM$_{l_j}$ can calculate the value of $p(\mathcal{P}_j)$, which is required by NM to correctly diagnose symptom $s : l_1 \xrightarrow{*} l_k$. As a result of the diagnosis performed by DM$_j$, the NM obtains $p(\mathcal{P}_j)$ and updates its FPM.

Observe that the diagnosis of the same symptom s_j can be delegated to DM$_{l_j}$ multiple times in the process of fault localization if it constitutes a segment of multiple end-to-end paths whose failures are reported as symptoms. This results in duplicate observations of the same symptom by DM$_{l_j}$. Since DM$_{l_j}$ does not process a symptom more than once, duplicate symptoms unnecessarily increase the communication overhead. Thus it is desirable to prevent duplicate delegations of the same symptom to DM$_{l_j}$. Naturally, we could do this by having the NM keep a record of all symptoms delegated to DMs, but this would waste memory, and searching through the record to detect duplicates would be time consuming. Instead, we observe the following. Since all end-to-end paths between domains $\mathcal{D}_{l_1}$ and $\mathcal{D}_{l_k}$ traverse the same domains using the same ingress and egress nodes, any inter-domain symptom $s : l_1 \xrightarrow{*} l_k$ that has previously been analyzed by NM results in a duplicate symptom delegation to DM$_{l_j}$ for $j = 2 \ldots k-1$. Thus, NM maintains a marking scheme in which unobserved-symptom nodes are marked as `UNOBSERVED` in its FPM. Once a symptom is observed, it is marked `OBSERVED_INTERNAL`. While analyzing $s : n_{p_1} \xrightarrow{*} n_{p_m}$ when $s : l_1 \xrightarrow{*} l_k$ is marked `OBSERVED_INTERNAL` the NM refrains from delegating intra-domain path segments of $s : n_{p_1} \xrightarrow{*} n_{p_m}$ to DM$_{l_j}$s for $j = 2 \ldots k-1$, but it does delegate the analysis to DM$_{l_1}$ and DM$_{l_k}$, since end-to-end paths represented by $s : l_1 \xrightarrow{*} l_k$ may differ in their path segments located in $\mathcal{D}_{l_1}$ and $\mathcal{D}_{l_k}$. This method prevents NM from sending duplicate symptoms to DM$_{l_j}$s for $j = 2 \ldots k-1$ but does not aim at preventing duplicate symptoms in DM$_{l_1}$ and DM$_{l_k}$.

When DMs complete the analysis of symptoms that have been delegated to them by NM they return the values of corresponding $p(\mathcal{P}_j)$s. Then, the NM updates its FPM and incorporates the changed values of

$p(\mathcal{P}_j)$s in its state of fault localization (function *update_hypotheses*). Finally, NM analyzes symptom $s : l_1 \overset{*}{\rightarrow} l_k$ (function *inference*). This part of NMs operation depends on the probabilistic inference mechanism used by NM, e.g., Algorithms 1 or 2.

2) Symptom processing by DM: DM_i may start the processing of symptom $s : n_{p_1} \overset{*}{\rightarrow} n_{p_m}$ as a result of two events: (1) it may observe a failure of path $n_{p_1} \overset{*}{\rightarrow} n_{p_m}$ whose all nodes belong to $\mathcal{D}_i$ or (2) the symptom may be delegated to DM_i by NM. In the former case, $s : n_{p_1} \overset{*}{\rightarrow} n_{p_m}$ is an internal symptom, in the latter case it is called an external symptom. Internal symptoms are considered more significant, since they cannot be explained by faults located outside DM_i. However, in the absence of internal symptoms, the external ones help the DM_i make correct diagnoses. To distinguish between different observations of the same symptom, DM_i marks symptom nodes as either `UNOBSERVED`, `OBSERVED_INTERNAL`, and `OBSERVED_EXTERNAL` when they are not processed, processed as a result of internal observation, and processed as a result of a delegation by NM, respectively.

Internal symptoms are processed by function *analyze_internal*. First, the association between the observed symptom and its $\tilde{\mathcal{P}}$-node (if one exists) is removed, as the symptom can no longer be explained by external causes. Then, a probabilistic inference mechanism chosen for this DM is used to analyze the symptom.

The processing of external symptoms is done by function *analyze_external*. Assume that $s : n_{p_1} \overset{*}{\rightarrow} n_{p_m}$ has been delegated to DM_i as a result of a failure of an end-to-end path between domains $\mathcal{D}_l$ and $\mathcal{D}_k$. DM_i also receives two parameters from NM: $\mathcal{P}^i_{l,k}$ and p_{spurious}. Recall that $\mathcal{P}^i_{l,k}$ is a description of a $\mathcal{P}$-node that is connected to node $s : l \overset{*}{\rightarrow} k$ in the FPM of NM, and p_{spurious} is the probability with which $s : n_{p_1} \overset{*}{\rightarrow} n_{p_m}$ should be considered spurious by DM_i. In the process of analyzing $s : n_{p_1} \overset{*}{\rightarrow} n_{p_m}$, DM_i first updates its FPM by setting $p(\tilde{\mathcal{P}}(s : n_{p_1} \overset{*}{\rightarrow} n_{p_m})) = p_{\text{spurious}}$ in its FPM, where $\tilde{\mathcal{P}}(s : n_{p_1} \overset{*}{\rightarrow} n_{p_m})$ is the $\tilde{\mathcal{P}}$-node connected to symptom $s : n_{p_1} \overset{*}{\rightarrow} n_{p_m}$ in the FPM of DM_i. If the symptom has been previously analyzed, DM_i takes no further action and returns the stored value of $p(\mathcal{P}^i_{l,k})$. Otherwise, it updates the FPM by connecting $\tilde{\mathcal{P}}(s : n_{p_1} \overset{*}{\rightarrow} n_{p_m})$ to $s : n_{p_1} \overset{*}{\rightarrow} n_{p_m}$, and updates the state of fault localization to reflect the modified value of $p(\tilde{\mathcal{P}}(s : n_{p_1} \overset{*}{\rightarrow} n_{p_m}))$. Then, a probabilistic reasoning mechanism is used to analyze the symptom. Finally, function *compute_prior* is used to calculate the a value of $p(\mathcal{P}^i_{l,k})$, which is defined as follows.

DM_i: FUNCTION *compute_prior*($\mathcal{P}^i_{l,k}$)
LET $\mathcal{S}^i_{l,k} = \{n_r \overset{*}{\rightarrow} n_t \in \mathcal{P}^i_{l,k} | s : n_r \overset{*}{\rightarrow} n_t$ *is not* `UNOBSERVED`$\}$
IF $\mathcal{S}^i_{l,k} = \emptyset$ THEN RETURN 0
RETURN $\prod_{s_i \in \mathcal{S}^i_{l,k}}$ *symptom_bel*(s_i), *where*

$$\textit{symptom\_bel}(s_i) = \begin{cases} 1 & \text{if } s_i \text{ is marked } \texttt{OBSERVED\_INTERNAL} \\ 1 - \prod_{f_j \in \mathcal{F}} (1 - p(s_i|f_j)\underline{\textit{fault\_bel}(f_j)}) & \text{if } s_i \text{ is marked } \texttt{OBSERVED\_EXTERNAL} \end{cases}$$

END

Intuitively, function *compute_prior* calculates the probability that all failures of intra-$\mathcal{D}_i$ path segments represented by $\mathcal{P}$ that have been reported by the NM can be explained by DM_i. Clearly, if a failure reported by NM has also been observed by DM_i as an internal symptom, then the probability that it was caused in $\mathcal{D}_i$ is 1. Otherwise, DM_i needs to calculate the probability that a failure reported by NM but not observed as an internal symptom was caused by one or more faults in $\mathcal{D}_i$. Fault probabilities used in this case are obtained based on symptom diagnosis performed by DM_i up to this point. When no symptoms have been reported to DM_i function *compute_prior* returns 0.

Example 3:

Let us consider the symptom analysis phase performed by managers of the network in Figure 5 on page 11 after failures of paths $1.5 \overset{*}{\leftrightarrow} 2.4$, $3.1 \overset{*}{\leftrightarrow} 3.4$, $1.3 \overset{*}{\leftrightarrow} 3.4$, $1.3 \overset{*}{\leftrightarrow} 2.1$ are observed.

1) Path $1.5 \overset{*}{\leftrightarrow} 2.4$ fails. Symptom $s : 1.5 \overset{*}{\leftrightarrow} 2.4$ is observed by NM. NM splits $1.5 \overset{*}{\leftrightarrow} 2.4$ into a sequence of intra-domain path segments and links $1.5 \leftrightarrow 1.1$, $1.1 \leftrightarrow 3.1$, $3.1 \overset{*}{\leftrightarrow} 3.5$, $3.5 \overset{*}{\leftrightarrow} 2.5$, and $2.5 \overset{*}{\leftrightarrow} 2.4$. Messages containing $\{s : 1.5 \overset{*}{\leftrightarrow} 1.1, p(\tilde{\mathcal{P}} : * \overset{*}{\leftrightarrow} 1.1), \mathcal{P} : * \overset{*}{\leftrightarrow} 1.1\}$, $\{s : 3.1 \overset{*}{\leftrightarrow}$

$3.5, p(\tilde{\mathcal{P}} : 3.1 \overset{*}{\leftrightarrow} 3.5), \mathcal{P} : 3.1 \overset{*}{\leftrightarrow} 3.5\}$, and $\{s : 2.5 \overset{*}{\leftrightarrow} 2.4, p(\tilde{\mathcal{P}} : 2.5 \overset{*}{\leftrightarrow} *), \mathcal{P} : 2.5 \overset{*}{\leftrightarrow} *\}$ are sent to DM_1, DM_3, and DM_2, respectively. DM_1, DM_3, and DM_2 update their FPMs, analyze external symptoms $s : 1.5 \overset{*}{\leftrightarrow} 1.1$, $s : 3.1 \overset{*}{\leftrightarrow} 3.5$, and $s : 2.5 \overset{*}{\leftrightarrow} 2.4$, respectively, and calculate prior probabilities for $\mathcal{P}$ nodes $\mathcal{P} : * \overset{*}{\leftrightarrow} 1.1$, $\mathcal{P} : 3.1 \overset{*}{\leftrightarrow} 3.5$, and $\mathcal{P} : 2.5 \overset{*}{\leftrightarrow} *$, respectively. Finally, symptom $s : 1 \overset{*}{\leftrightarrow} 2$ is analyzed by NM and marked `OBSERVED_INTERNAL`.

2) Path $3.1 \overset{*}{\leftrightarrow} 3.4$ fails, which is an intra-domain path in $\mathcal{D}_3$. Internal symptom $s : 3.1 \overset{*}{\leftrightarrow} 3.4$ is analyzed by DM_3, and marked `OBSERVED_INTERNAL`.
3) Path $1.3 \overset{*}{\leftrightarrow} 3.4$ fails and symptom $s : 1.3 \overset{*}{\leftrightarrow} 3.4$ is analyzed by NM. The NM splits path $1.3 \overset{*}{\leftrightarrow} 3.4$ into a sequence of $1.3 \overset{*}{\leftrightarrow} 1.1$, $1.1 \leftrightarrow 3.1$ and $3.1 \overset{*}{\leftrightarrow} 3.4$. Messages containing $\{s : 1.3 \overset{*}{\leftrightarrow} 1.1, p(\tilde{\mathcal{P}} : * \overset{*}{\leftrightarrow} 1.1), \mathcal{P} : * \overset{*}{\leftrightarrow} 1.1\}$ and $\{s : 3.1 \overset{*}{\leftrightarrow} 3.4, p(\tilde{\mathcal{P}} : 3.1 \overset{*}{\leftrightarrow} *), \mathcal{P} : 3.1 \overset{*}{\leftrightarrow} *\}$ are sent to DM_1 and DM_3, respectively. DM_1 analyzes external symptom $s : 1.3 \overset{*}{\leftrightarrow} 1.1$ and computes $p(\mathcal{P} : * \overset{*}{\leftrightarrow} 1.1)$. However, DM_3 ignores symptom $s : 3.1 \overset{*}{\leftrightarrow} 3.4$ and returns the most recent value of $p(\mathcal{P} : 3.1 \overset{*}{\leftrightarrow} *)$, since the current marking of $s : 3.1 \overset{*}{\leftrightarrow} 3.4$ in DM_3 is `OBSERVED_INTERNAL`. Finally, symptom $s : 1 \overset{*}{\leftrightarrow} 3$ is analyzed by NM.
4) Path $1.3 \overset{*}{\leftrightarrow} 2.1$ fails and as a result symptom $s : 1.3 \overset{*}{\leftrightarrow} 2.1$ is analyzed by NM, which splits path $1.3 \overset{*}{\leftrightarrow} 2.1$ into sequence $1.3 \overset{*}{\leftrightarrow} 1.1$, $1.1 \leftrightarrow 3.1$, $3.1 \overset{*}{\leftrightarrow} 3.5$, $3.5 \overset{*}{\leftrightarrow} 2.5$, and $2.5 \overset{*}{\leftrightarrow} 2.1$. Messages containing $\{s : 1.3 \overset{*}{\leftrightarrow} 1.1, p(\tilde{\mathcal{P}} : * \overset{*}{\leftrightarrow} 1.1), \mathcal{P} : * \overset{*}{\leftrightarrow} 1.1\}$ and $\{s : 2.5 \overset{*}{\leftrightarrow} 2.1, p(\tilde{\mathcal{P}} : 2.5 \overset{*}{\leftrightarrow} *), \mathcal{P} : 2.5 \overset{*}{\leftrightarrow} *\}$ are sent to DM_1 and DM_2, respectively. To avoid duplicate symptoms, no message is sent to DM_3 since the current marking of symptom $s : 1 \overset{*}{\leftrightarrow} 2$ in the FPM of NM is `OBSERVED_INTERNAL`. After DM_1 and DM_2 return new values of $p(\mathcal{P} : * \overset{*}{\leftrightarrow} 1.1)$ and $p(\mathcal{P} : 2.5 \overset{*}{\leftrightarrow} *)$, respectively, the NM updates its FPM, but it does not analyze symptom $s : 1 \overset{*}{\leftrightarrow} 2$ since its marking is `OBSERVED_INTERNAL`.

B. Fault selection phase

In the fault selection phase, DMs formulate their final explanation hypotheses and report them to a system administrator. Before this can happen, DMs and NM have to synchronize their FPMs by updating prior failure probabilities associated with their proxy nodes. Although these probabilities are constantly modified during the symptom analysis phase, this is only a partial process, and more thorough synchronization of the models is needed before final hypotheses can be proposed.

Fault selection phase is a cooperative process initiated by NM, which first obtains from DMs prior failure probabilities associated with proxy nodes in its FPM, and then calculates spurious symptom probabilities that are assigned to proxy nodes in the FPMs of DMs.

Afterwards, DMs and NM can proceed independently of one another to update their fault localization state (function *update_hypotheses*) and choose the most likely hypothesis (function *select_best_hypothesis*). Both function depend on the probabilistic inference mechanism used by DM or NM. In particular, function *select_best_hypothesis* constitutes the fault selection phase of either Algorithm 1 or Algorithm 2.

Algorithm 3: Multi-domain algorithm

Initialization:
NM: FOR *every* $\mathcal{P}^i_{l,k}$ DO $p(\mathcal{P}^i_{l,k}) = 0$ DONE
Symptom analysis phase:
DM: FOR *every observed symptom* $s : n_{p_1} \overset{*}{\rightarrow} n_{p_m}$ DO
IF $d(n_{p_1}) =$ DID AND $d(n_{p_m}) =$ DID THEN
analyze_internal($s : n_{p_1} \overset{*}{\rightarrow} n_{p_m}$)
ELSE NM$\rightarrow$*analyze_internal*($s : n_{p_1} \overset{*}{\rightarrow} n_{p_m}$)
DONE

NM: FOR *every observed symptom* $s : n_{p_1} \xrightarrow{*} n_{p_m}$ DO *analyze_internal*($s : n_{p_1} \xrightarrow{*} n_{p_m}$) DONE

DM$_i$: FUNCTION *analyze_internal*($s : n_{p_1} \xrightarrow{*} n_{p_m}$)
IF $s : n_{p_1} \xrightarrow{*} n_{p_m}$ *is not marked* OBSERVED_INTERNAL THEN
set $p(s : n_{p_1} \xrightarrow{*} n_{p_m} | \tilde{\mathcal{P}}(s : n_{p_1} \xrightarrow{*} n_{p_m})) = 0$
mark $s : n_{p_1} \xrightarrow{*} n_{p_m}$ *as* OBSERVED_INTERNAL
inference($s : n_{p_1} \xrightarrow{*} n_{p_m}$)
END

NM: FUNCTION *analyze_internal*($s : n_{p_1} \xrightarrow{*} n_{p_m}$)
map $s : n_{p_1} \xrightarrow{*} n_{p_m}$ *to* $s : l_1 \xrightarrow{*} l_k$ *such that* $n_{p_1} \xrightarrow{*} n_{p_m} \in l_1 \xrightarrow{*} l_k$
transform $n_{p_1} \xrightarrow{*} n_{p_m}$ *into* $n_{p_1} \xrightarrow{*} E^{l_1}_{l_1,l_k}, E^{l_1}_{l_1,l_k} \rightarrow I^{l_2}_{l_1,l_k}, I^{l_2}_{l_1,l_k} \xrightarrow{*} E^{l_2}_{l_1,l_k}, \ldots,$
$E^{l_{k-1}}_{l_1,l_k} \rightarrow I^{l_k}_{l_1,l_k}, I^{l_k}_{l_1,l_k} \xrightarrow{*} n_{p_m}$
determine proxy nodes connected to $s : l_1 \xrightarrow{*} l_k$:
$\mathcal{P}_1 = \mathcal{P} : * \xrightarrow{*} E^{l_1}_{l_1,l_k}, \mathcal{P}_2 = \mathcal{P} : I^{l_2}_{l_1,l_k} \xrightarrow{*} E^{l_2}_{l_1,l_k}, \ldots, \mathcal{P}_k = \mathcal{P} : I^{l_k}_{l_1,l_k} \xrightarrow{*} *$
set $s_1 = s : n_{p_1} \xrightarrow{*} E^{l_1}_{l_1,l_k}, s_2 = s : I^{l_2}_{l_1,l_k} \xrightarrow{*} E^{l_2}_{l_1,l_k}, \ldots, s_k = s : I^{l_k}_{l_1,l_k} \xrightarrow{*} n_{p_m}$
FOR $1 \leq j \leq k$ DO
IF $s : l_1 \xrightarrow{*} l_k$ *is marked* UNOBSERVED OR $j = 1$ OR $j = k$ THEN
p_{spurious} =*compute_spurious*($\mathcal{P}_j$)
$p(\mathcal{P}_j) = \text{DM}_{l_j}$ →*analyze_external*(s_j, $\mathcal{P}_j$, p_{spurious})
DONE
IF $s : l_1 \xrightarrow{*} l_k$ *is not marked* OBSERVED_INTERNAL THEN
update_hypotheses()
mark $s : l_1 \xrightarrow{*} l_k$ *as* OBSERVED_INTERNAL
inference($s : l_1 \xrightarrow{*} l_k$)
END

DM$_i$: FUNCTION *analyze_external*($s : n_{p_1} \xrightarrow{*} n_{p_m}$, $\mathcal{P}^i_{l,k}$, $p(\tilde{\mathcal{P}}(s : n_{p_1} \xrightarrow{*} n_{p_m}))$
set $p(\tilde{\mathcal{P}}(s : n_{p_1} \xrightarrow{*} n_{p_m})) = p_{\text{spurious}}$
IF $s : n_{p_1} \xrightarrow{*} n_{p_m}$ *is not marked* UNOBSERVED THEN *return* $p(\mathcal{P}^i_{l,k})$
ELSE
connect $\tilde{\mathcal{P}}(s : n_{p_1} \xrightarrow{*} n_{p_m})$ *to* $s : n_{p_1} \xrightarrow{*} n_{p_m}$ *and set* $p(s : n_{p_1} \xrightarrow{*} n_{p_m} | \tilde{\mathcal{P}}(s : n_{p_1} \xrightarrow{*} n_{p_m})) = 1$
update_hypotheses()
mark $s : n_{p_1} \xrightarrow{*} n_{p_m}$ *as* OBSERVED_EXTERNAL
inference($s : n_{p_1} \xrightarrow{*} n_{p_m}$)
return compute_prior($\mathcal{P}^i_{l,k}$)
END

Fault selection phase:
NM: FOR *every* $\mathcal{P}^i_{l,k}$ DO $p(\mathcal{P}^i_{l,k}) = \text{DM}_i$ →*compute_prior*($\mathcal{P}^i_{l,k}$) DONE
FOR *every* $\mathcal{P}^i_{l,k}$ DO DM_i →*set_spurious*($\tilde{\mathcal{P}}^i_{l,k}$, *compute_spurious*($\mathcal{P}^i_{l,k}$)) DONE
FOR *every* DM_i DO DM_i →*select_faults*() DONE
select_faults()

DM/NM: FUNCTION *select_faults*()
update_hypotheses()
select_best_hypothesis()
END

C. Computational complexity

A precise bound on the computational complexity of Algorithm 3 may not be obtained without deciding on the probabilistic reasoning mechanism to be used by the algorithm, which affects the complexity of algorithm's steps presented using the underlined font. However, the general process of calculating this bound and sources of the complexity are discussed here. In Sections VI and VII, we give a tighter assessment of the algorithm's complexity when using belief updating (Algorithm 1) and the incremental technique (Algorithm 2) as reasoning mechanisms, respectively.

The complexity of Algorithm 3 results from symptom analysis performed in the symptom analysis phase, fault selection performed in fault selection phase, and the calculation of probabilities exchanged among managers in all phases of the algorithm. In the following analysis, we use $|\mathcal{N}|$ and $|\mathcal{D}_i|$ to denote the number of domains in $\mathcal{N}$ or nodes in $\mathcal{D}_i$, respectively. Recall that in the pseudo-code of Algorithm 3, parts of the code that need to be specialized for different probabilistic reasoning mechanisms are encoded as functions *not_proxy*, *fault_bel*, *inference*, *update_hypotheses*, and *select_best_hypothesis*. Let f_{not_proxy}, f_{fault_bel}, $f_{inference}$, $f_{update\_hypotheses}$, and $f_{select\_best\_hypothesis}$ be functions such that the computational complexities of *not_proxy*, *fault_bel*, *inference*, *update_hypotheses*, and *select_best_hypothesis* are $\mathcal{O}(f_{not\_proxy})$, $\mathcal{O}(f_{fault\_bel})$, $\mathcal{O}(f_{inference})$, $\mathcal{O}(f_{update\_hypotheses})$, and $\mathcal{O}(f_{select\_best\_hypothesis})$, respectively.

The computational cost incurred by the NM in the symptom analysis phase is due to (1) executing *compute_spurious* for every symptom observed by NM, and (2) analyzing previously unobserved symptoms. The former operation is invoked at least twice for each symptom observed by NM (i.e., while delegating symptoms to source and destination domains of a failed path). In addition, the operation may be invoked at most $\mathcal{O}(|\mathcal{N}|)$ times for each observed symptom whose marking is `UNOBSERVED` (i.e., while delegating symptoms to all domains traversed by a failed path). Overall, since the number of path nodes in the FPM of NM is $\mathcal{O}(|\mathcal{N}|^2)$, the number of invocations is $\mathcal{O}(\max(|\mathcal{S}_O|, |\mathcal{N}|^3))$ during the entire symptom analysis phase. Also, the computational complexity of *compute_spurious* is equal to that of not_proxy. The task of analyzing previously unobserved symptoms is executed at most $\min(|\mathcal{S}_O|, |\mathcal{N}|^2)$ times, since no symptom in the FPM of NM can be analyzed more than once. It involves functions *update_hypotheses* and *inference*. As a result, the computational complexity of the symptom analysis phase during entire fault localization process is $\mathcal{O}(|\mathcal{S}_O| f_{not\_proxy} + \min(|\mathcal{S}_O|, |\mathcal{N}|^2)(f_{inference} + f_{update\_hypotheses}))$.

The computational complexity of the symptom analysis phase performed by DM_i is due to analyzing internal symptoms in function *analyze_internal* and analyzing external symptoms in function *analyze external*. Since each symptom in the FPM of DM_i may be analyzed at most once as an internal one, the computational cost of internal symptom analysis is $\mathcal{O}(\min(|\mathcal{S}_O|, |\mathcal{D}_i|^2) f_{inference})$. The analysis of external symptoms involves three functions: *update_hypotheses*, *inference*, and *compute_prior*. The complexity of the last function depends on the number of $\mathcal{P}$-nodes that represent domain $\mathcal{D}_i$ in the FPM of NM. Observe that in the FPM of NM, $\mathcal{D}_i$ may be represented by at most $\mathcal{O}(|\mathcal{D}_i|)$ $\mathcal{P}$-nodes of type $\mathcal{P} : * \xrightarrow{*} n_t$ and $\mathcal{P} : n_r \xrightarrow{*} *$ (one for every n_r or n_t in $\mathcal{D}_i$). Each such $\mathcal{P}$-node represents $\mathcal{O}(|\mathcal{D}_i|)$ intra-$\mathcal{D}_i$ path segments, and therefore its prior probability will be calculated at most $\mathcal{O}(|\mathcal{D}_i|)$ times (once for every path segment, which is delegated as an external symptom). The computational complexity of a single invocation is, in this case, $\mathcal{O}(|\mathcal{D}_i|^2 f_{fault\_bel})$. Similarly, in the FPM of NM, $\mathcal{D}_i$ may be represented by at most $\mathcal{O}(|\mathcal{D}_i|^2)$ $\mathcal{P}$-nodes of type $\mathcal{P} : n_r \xrightarrow{*} n_t$ that each represent exactly one intra-$\mathcal{D}_i$ path segment and therefore their prior probabilities may be calculated at most once during the symptom analysis phase. In this case, the computational cost of a single invocation is $\mathcal{O}(|\mathcal{D}_i| f_{fault\_bel})$. Overall, the cost of *compute_prior* in the entire symptom analysis phase is $\mathcal{O}(|\mathcal{D}_i|^3 f_{fault\_bel})$. Also, since each symptom may be analyzed as external at most once, the number of times functions *update_hypotheses* and *inference* are executed is $\mathcal{O}(\min(|\mathcal{S}_O|, |\mathcal{D}_i|^2))$. Therefore, the computational complexity of the entire symptom analysis phase performed by DM_i is $\mathcal{O}(\min(|\mathcal{S}_O|, |\mathcal{D}_i|^2)(f_{update\_hypotheses} + f_{inference}) + |\mathcal{D}_i|^3 f_{fault\_bel})$.

The computational cost incurred by NM in the fault selection phase is due to invoking *calculate_spurious*, which is done at most $|\mathcal{N}|^3$ times. (Since each of $|\mathcal{N}|^2$ symptom nodes in the FPM of NM is connected to at most $|\mathcal{N}|$ proxy nodes, there are at most $|\mathcal{N}|^3$ probabilities to calculate). The computational cost incurred

by NM in the fault selection phase is also due to function *select_faults*, which is $\mathcal{O}(f_{update\_hypotheses} + f_{inference})$. Overall, the computational complexity of NM in the fault selection phase is $\mathcal{O}(|\mathcal{N}|^3 f_{not\_proxy} + f_{update\_hypotheses} + f_{select\_best\_hypotheses})$.

In the fault selection phase, DM_i calculates prior probabilities of $\mathcal{P}$-nodes on behalf of NM and chooses the best hypothesis. Its computational cost incurred in this phase is $\mathcal{O}(f_{update\_hypotheses}+f_{select\_best\_hypothesis})+ |\mathcal{D}_i|^3 f_{fault\_bel})$.

From the above analysis it is clear that the algorithm's performance depends on the efficiency of calculating probabilities that are exchanged between DMs and NM. In Sections VI and VII, we present the process of calculating these probabilities in detail and refine computational complexity bounds for multi-domain fault localization using belief updating and incremental hypothesis updating.

Finally, recall that one of the initial objectives of our research on multi-domain fault localization was improving the efficiency of end-to-end service failure diagnosis. While the analysis provided in this section does not allow us to quantify this improvement, we can observe that a potential algorithm's speed-up is likely to result from the following properties of the algorithm.

- DMs and NM operate on smaller FPMs than would be the case with the centralized approach. Recall that the complexities of Algorithms 1 and 2 are $\mathcal{O}(n^5)$ and $\mathcal{O}(n^4)$, respectively. If we managed to keep the complexities of managers' algorithms within these bounds when using belief-network approach and the incremental technique as reasoning mechanisms, by decomposing the fault localization problem into m smaller subproblems we could achieve complexities of $\mathcal{O}(mn^5)$ and $\mathcal{O}(mn^4)$ instead of $\mathcal{O}(m^5n^5)$ and $\mathcal{O}(m^4n^4)$, respectively.
- DMs process a smaller number of symptoms than in the centralized case. Observe that multiple end-to-end path failures map into a single symptom node in the FPM of the NM. In the centralized technique, these path failures are mapped into different symptom nodes and therefore have to be analyzed separately.
- Since DMs can execute in parallel, many symptoms may be simultaneously analyzed, thereby reducing the overall fault localization time.

D. Signaling overhead

The signaling overhead of Algorithm 3 results from the exchange of symptoms and probabilities between NM and DMs and therefore is related to the number of probability values produced in every phase of the algorithm. In the symptom analysis phase, the number of messages exchanged between NM and DMs is related to the number of observed symptoms. For every symptom observed by the NM at least two messages will be sent to DMs. In addition, symptoms that have not been previously analyzed by NM result in messages sent to at most $|\mathcal{N}|$ DMs for each such symptom. As a result the messaging overhead incurred in this phase is $\mathcal{O}(\max(|\mathcal{S}_O|, |\mathcal{N}|^3))$.

In the fault selection phase, the messaging overhead results from the exchange of probabilities needed to update the FPMs. The number of messages exchanged between NM and DMs is also $\mathcal{O}(|\mathcal{N}|^3)$. Thus, in the entire algorithm, the messaging overhead is $\mathcal{O}(\max(|\mathcal{S}_O|, |\mathcal{N}|^3))$.

VI. Multi-domain fault localization using belief networks

In this section, we introduce **Algorithm 3A**, a multi-domain version of Algorithm 1, which constitutes a refinement of the fault localization technique presented in Sections III-V. The refinement introduces definitions of functions *not_proxy*, *fault_bel*, *inference*, *update_hypotheses*, and *select_best_hypothesis*, which were left undefined in the description of the general multi-domain technique.

Function *not_proxy*$(\mathcal{P}^i_{l,k})$ calculates the conditional probability that faults represented by $\mathcal{P}^i_{l,k}$ in the FPM of NM did not occur, given the observed evidence. Using the probabilistic reasoning mechanism of Algorithm 1 this probability may be expressed using λ messages received by node $\mathcal{P}^i_{l,k}$ from its children nodes in the belief network that constitutes the FPM of NM. Let $\lambda_{\mathcal{P}^i_{l,k}}(x)$ indicate a product of messages λ received by $\mathcal{P}^i_{l,k}$ from its children. For $x = 1$, $\lambda_{\mathcal{P}^i_{l,k}}(x)$ is interpreted as the probability that the observed symptoms, children of $\mathcal{P}^i_{l,k}$, occur given the failure condition represented by $\mathcal{P}^i_{l,k}$ has occurred. Similarly, for $x = 0$,

$\lambda_{\mathcal{P}^i_{l,k}}(x)$ is interpreted as the probability that the observed symptoms, children of $\mathcal{P}^i_{l,k}$, occur given the failure condition represented by $\mathcal{P}^i_{l,k}$ has not occurred. Using this information, we can derive *not_proxy*($\mathcal{P}^i_{l,k}$) as follows. Recall that in the FPM of DM$_i$, the values of $\lambda_{s_j}(x)$, which are assigned to node s_j that represents a failure of a path in $\mathcal{P}^i_{l,k}$, are 1 and 0 for $x = 1$ and $x = 0$, respectively. This assignment indicates our total confidence in the ocurrence of s_j. To represent the fact that the symptom may be spurious, we should use assignment $\lambda_{s_j}(x) = \lambda_{\mathcal{P}^i_{l,k}}(x)$ for all such s_j. In our solution, instead of modifying the assignment of λ_{s_j}, we use a $\tilde{\mathcal{P}}$-node that represents all external causes of s_j. Thanks to this choice, every time $\lambda_{\mathcal{P}^i_{l,k}}(x)$ changes, we need to modify just one prior probability in the FPM of DM$_i$ instead of multiple values of λ_{s_j}. The assignement of the prior probability to the $\tilde{\mathcal{P}}$-node that is connected to symptom node s_j in the FPM of DM$_i$ should be such that the behavior of s_j, which is expressed by the values of λ messages it sends to its children, is the same as if $\lambda_{s_j}(x)$ was set to $\lambda_{\mathcal{P}^i_{l,k}}(x)$. This objective can be achieved by setting *not_proxy*($\mathcal{P}^i_{l,k}$) as follows.

$$\textit{not\_proxy}(\mathcal{P}^i_{l,k}) = \frac{\lambda_{\mathcal{P}^i_{l,k}}(0)}{\lambda_{\mathcal{P}^i_{l,k}}(1)}$$

Function *fault_bel*(f_k) estimates the probability that fault f_k exists in $\mathcal{D}_i$ given evidence observed in $\mathcal{D}_i$. It may be calculated using messages received by the belief network node that represents f_i in the FPM of DM$_i$. Let $\lambda_{f_i}(x)$ indicate a product of messages λ sent to node f_i by its children. The probability that f_k occured or did not occur given the observed evidence may be expressed as $\alpha\lambda_{f_i}(1)p(f_i)$ and $\alpha\lambda_{f_i}(0)(1 - p(f_i))$, respectively, where α is a normalizing constant. Therefore, we calculate *fault_bel*(f_k) as follows.

$$\textit{fault\_bel}(f_i) = \alpha\lambda_{f_i}(1)p(f_i)$$

Function *inference*(s_i) is identical to the like-named function in the centralized Algorithm 1, which was presented in Section II.

Function *update_hypotheses* updates the state of fault localization to incorporate modified values of prior failure probabilities associated with $\mathcal{P}$- and $\tilde{\mathcal{P}}$-nodes in the FPMs of the NM and a DM, respectively. In the context of iterative belief updating, this process involves recalculating messages π sent by a $\mathcal{P}$- or $\tilde{\mathcal{P}}$-node to its children, for every $\mathcal{P}$- or $\tilde{\mathcal{P}}$-node whose prior failure probability has changed. For this purpose, the message-updating equations used in function *inference* of the centralized algorithm are used [30].

Function *select_best_hypotheses* executes the fault selection phase of Algorithm 1.

Observe that the computational complexities of functions *not_proxy* and *fault_bel* are $\mathcal{O}(1)$. Function *inference* is $\mathcal{O}(n^3)$, where n is the number of network nodes. In the case of NM and DM$_i$ this is expressed as $\mathcal{O}(|\mathcal{N}|^3)$ and $\mathcal{O}(|\mathcal{D}_i|^3)$, respectively. Since function *update_hypotheses* involves the calculation of π messages for some belief network nodes, while function *inference* performes this calculation for all belief network nodes, the computational complexity of *update_hypotheses* is clearly $\mathcal{O}(f_{inference}) = \mathcal{O}(n^3)$. Finally, the computational complexity of function *select_best_hypotheses* is $\mathcal{O}(n^4)$, where $n = |\mathcal{N}|$ or $n = |\mathcal{D}_i|$ in the case of NM or DM$_i$, respectively.

After substituting the above refinements for the partial results derived in Section V-C, we conclude that the computational cost incurred by NM in the symptom analysis phase and fault selection phase is $\mathcal{O}(\min(|\mathcal{S}_O||\mathcal{N}|^3, |\mathcal{N}|^5))$ and $\mathcal{O}(|\mathcal{N}|^4)$, respectively. Therefore, we can express the computational complexity of fault localization performed by NM as $\mathcal{O}(|\mathcal{N}|^5))$.

Similarly, we can refine the computational cost of the symptom analysis phase and fault selection phase of DM$_i$ to $\mathcal{O}(\min(|\mathcal{S}_O||\mathcal{D}_i|^3, |\mathcal{D}_i|^5))$ and $\mathcal{O}(|\mathcal{D}_i|^4)$. Thus, the computational complexity of fault localization performed by DM$_i$ is $\mathcal{O}(|\mathcal{D}_i|^5)$.

Note that both for NM and DM, we managed to maintain their original computational complexity limit of $\mathcal{O}(n^5)$, which allows us to maximize scalability gains resulting from the multi-domain approach.

VII. Multi-domain incremental hypothesis updating

In this section, we introduce **Algorithm 3B**, a multi-domain version of Algorithm 2, which constitutes a refinement of the fault localization technique presented in Sections III-V. Similar to Algorithm 1, the

refinement introduces definitions of functions *not_proxy*, *fault_bel*, *inference*, *update_hypotheses*, and *select_best_hypothesis*, which were left undefined in Section V.

In the context of the incremental technique **function** *not_proxy*$(\mathcal{P}^i_{l,k})$ calculates the conditional probability that faults represented by $\mathcal{P}^i_{l,k}$ in the FPM of NM did not occur, given the observed evidence. This probability may be expressed as follows. Let $\mathcal{H}_j$ represent the current set of hypotheses produced by DM$_i$. Recall that each hypothesis is a subset of $\mathcal{F} \cup \mathcal{P}$, where $\mathcal{P}$ is the set of all $\mathcal{P}$-nodes in the FPM of NM.

$$\textit{not\_proxy}(\mathcal{P}^i_{l,k}) = 1 - \sum_{h \in \mathcal{H}_j | \mathcal{P}^i_{l,k} \in h} b_j(h)$$

Function *fault_bel*(f_k) estimates the probability that fault f_k exists in $\mathcal{D}_i$ given evidence observed in $\mathcal{D}_i$. It is caclulated by summing the belief metric associated with hypotheses that contain f_k.

$$\textit{fault\_bel}(f_k) = \sum_{h \in \mathcal{H}_j | f_k \in h} b_j(h)$$

Function *inference*(s_i) is is identical with the like-named function in the centralized Algorithm 2, which was presented in Section II.

Function *update_hypotheses* updates the state of fault localization to incorporate modified values of prior failure probabilities associated with $\mathcal{P}$- and $\tilde{\mathcal{P}}$-nodes in the FPMs of the NM and a DM, respectively. In the context of the incremental technique, this function involves replacing the old value of the prior probability of node x with the new value of this probability, in the belief metric associated with each hypothesis h that contains x. This is done by setting $b_j(h) = b_j(h)\frac{p_{\text{new}}(x)}{p_{\text{old}}(x)}$. This replacement is done for every node x, whose prior failure probability has changed.

Function *select_best_hypotheses* executes the fault selection phase of Algorithm 2.

Observe that functions *not_proxy* and *fault_bel* involve searching through the set of hypotheses and summing up the belief metric of some of the hypotheses. These calculations can be done in advance, when hypotheses are created in function *inference*, which allows us to amortize the cost of calculating *not_proxy* and *fault_bel* with the cost of running function *inference*. Thus, the computational complexities of both functions are $\mathcal{O}(1)$. Function *inference* is $\mathcal{O}(n^2)$, where n is the number of network nodes, i.e., $\mathcal{O}(|\mathcal{N}|^2)$ and $\mathcal{O}(|\mathcal{D}_i|^2)$ in the case of NM and DM, respectively. Function *update_hypotheses* scans through $|\mathcal{H}_j|$ hypotheses and updates at most n probabilities each time it is invoked. Recall that $|\mathcal{H}_j| = \mathcal{O}(n)$. Thus the computational complexity of *update_hypotheses* is $\mathcal{O}(|\mathcal{N}|^2)$ and $\mathcal{O}(|\mathcal{D}_i|^2)$ in the case of NM and DM, respectively. Finally, the computational complexity of function *select_best_hypotheses* is $\mathcal{O}(1)$.

After substituting the above refinements for the partial results derived in Section V-C, we conclude that the computational cost incurred by NM in the symptom analysis phase and fault selection phase is $\mathcal{O}(\min(|\mathcal{S}_O||\mathcal{N}|^2, |\mathcal{N}|^4))$ and $\mathcal{O}(|\mathcal{N}|^3)$, respectively. Therefore, we can express the computational complexity of fault localization performed by NM as $\mathcal{O}(|\mathcal{N}|^4)$.

Similarly, we can refine the computational cost of the symptom analysis phase and fault selection phase of DM$_i$ to $\mathcal{O}(\min(|\mathcal{S}_O||\mathcal{D}_i|^2, |\mathcal{D}_i|^4))$ and $\mathcal{O}(|\mathcal{D}_i|^3)$. Thus, the computational complexity of fault localization performed by DM$_i$ is $\mathcal{O}(|\mathcal{D}_i|^4)$.

Note that similar to the multi-domain version of Algorithm 1 we managed to maintain the computational complexity bound of the centralized Algorithm 2 in its multi-domain version for both NM and DM.

VIII. Simulation study

In this section, we evaluate the performance of Algorithms 3A and 3B through simulation. Our purpose is to assess the accuracy of both algorithms in a multi-domain communication network. The study uses sets of fault localization scenarios in which faults and symptoms are randomly generated based on the conditional probability distribution that describes non-deterministic causal relationships between faults and symptoms in a real-life system. In the distribution, for every fault f and symptom s, we set $p(s|f) = 0$ if the end-to-end service whose failure is represented by s is not provided using the host-to-host service whose failure is represented by f. Otherwise, $0 < p(s|f) \leq 1$. One may argue that a better method of evaluating the algorithms is to simulate a network injecting faults in it in a controlled manner and allowing symptoms to be

generated by the simulated system itself. Unfortunately, network simulators available today do not facilitate creating such a study as they do not provide functionality that is needed to inject performance problems, monitor system performance, or obtain configuration information necessary to build the fault propagation model. The amount of effort necessary to provide this functionality makes it impractical for us to design such a simulation. In addition, our approach relies on the existence of solutions to several problems that are still a subject of an active research activity, e.g., building and updating an FPM or optimal probe placement. When using a network simulator to create an experimental study, we first would have to solve these open problems. Moreover, using synthetic tests rather than a network simulator, makes the study independent of particular network configurations, routing protocols, and instrumentation mechanisms.

In this section, we first describe the design of the simulation experiments and then present and explain the results of the study.

A. Simulation design

The simulation study presented in this section uses network topologies similar to those of the Internet. The generation of random graphs resembling the topology of real-life networks has been a widely studied research area [1], [2], [4], [9], [18], [26]. Out of several topology generators available, we choose one based on Barabasi-Albert power-law model [4], because its implementation is available in public domain, and because topologies built based on this model have been shown to be reasonably representative of the Internet topology [8]. We use an implementation of the Barabasi-Albert model provided by BRITE generator [25], which is capable of generating hierarchical network topologies: AS-level and router-level ones.

The simulation model of the study created two level hierarchical topologies using N and n to denote the number of domains and the number of routers in every domain, respectively. To investigate the impact of network topology, we use $N = 10$ and $N = 50$, and we vary n from 5 to 75. Typically, we choose a maximum domain size such that the fault localization time of a single scenario does not exceed 10s. Our experiments ignore positive, lost, and spurious symptoms. Consequently, we assume that the observation of the system state is accurate.

Using the topology generator we create a random network composed of N domains and n nodes in each domain. We determine routes between any source and destination using the shortest-path policy for intra-domain routes. We choose inter-domain routes such that the number of visited domains is minimized. Then, we generate prior failure probabilities for inter-domain and intra-domain links, which are uniformly distributed over the range $[0.0001, 0.001]$. For each intra-domain link l and path p, we randomly choose the probability that p fails if l fails from set $\{0.25, 0.5, 0.75\}$. For each inter-domain path p, we assume that if any path segment or link involved in p fails then p fails as well. Consequently, in the FPM of the NM, the conditional probabilities are all equal to 1. Furthermore, we randomly generate a subset of symptoms observable in every domain to include 50% of all intra-domain paths. The observability ratio for inter-domain paths is 2%. The observability ratio [39], [40] is a measure of the system instrumentation degree. By using it, we recognize that only some failure conditions are monitored by the management systems. As a result, a manager can see only a fraction of failures that exist in the system it manages.

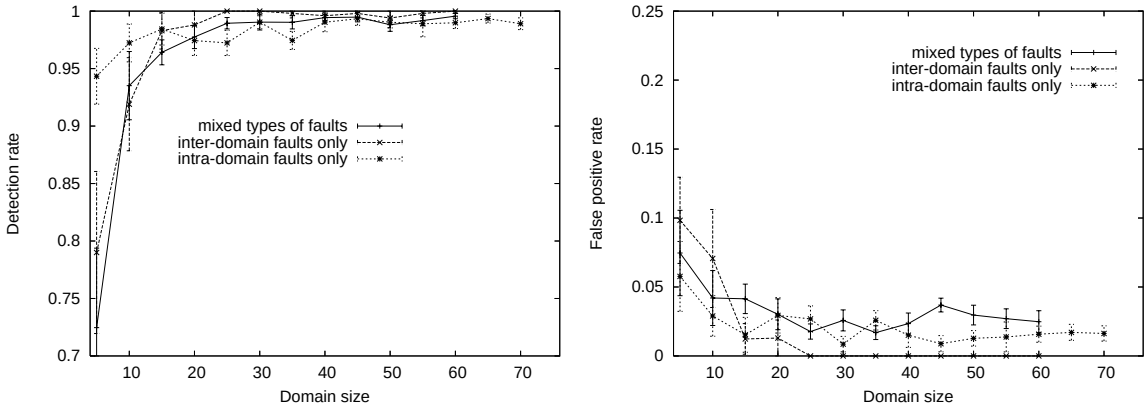
Test scenarios are generated using the same conditional probability distribution that is used by the managers in their FPMs. This technique of generating scenarios assumes that the fault propagation model accurately represents relationships among faults and symptoms. However, from our previous studies, we know that the fault localization techniques considered in this paper are accurate even if an FPM they are executed on is approximate.

We distinguish three types of experiments: those involving only intra-domain link failures, those involving only inter-domain link failures, and those involving both types of failures. In every study, two performance metrics are calculated: detection rate, DR, defined as a percentage of faults occurring in the network which are isolated by the technique, and false positive rate, FPR, defined as a percentage of faults reported by the technique that are not occurring in the network [41], [39].

B. Experimental results

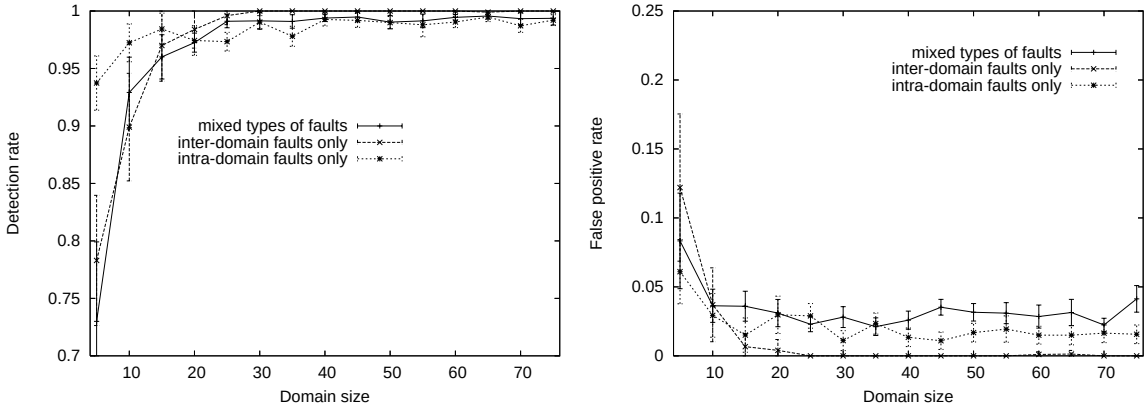
In Figures 9(a)-9(b), we show the accuracy of Algorithm 3A applied to fault localization in a ten-domain network, in which each domain is composed of up to 70 nodes. Thus the entire network consists of up to

700 nodes. Figures 10(a)-10(b) present the results of the same experiment executed using Algorithm 3B.



(a) Detection rate (b) False positive rate

Fig. 9. Accuracy of Algorithm 3A in a ten-domain network.



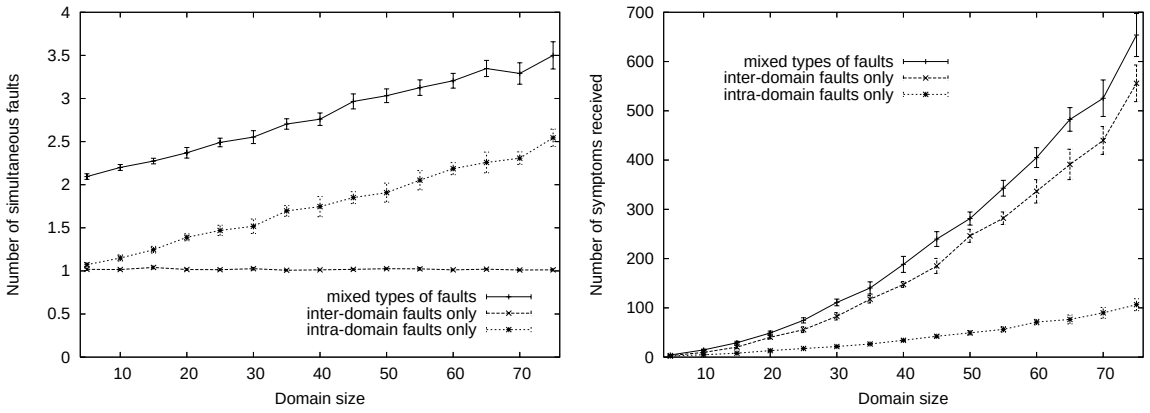
(a) Detection rate (b) False positive rate

Fig. 10. Accuracy of Algorithm 3B in a ten-domain network.

The figures compare the accuracy achievable in scenarios involving only inter-domain, only intra-domain, and both types of faults. Clearly, the mixed-failure scenarios are the most difficult to diagnose since they always involve at least two concurrent faults and the interpretation of their symptoms, which may overlap, leads to ambiguity. This difficulty results in a lower fault-localization accuracy of mixed-fault scenarios compared to that of other types of scenarios, which is conspicuous in networks of small size. Scenarios involving only inter-domain symptoms are the easiest to solve as the number of suspect faults is usually small compared to the amount of evidence available even with the very small observability ratio we have chosen. In addition, in our two-level set-up, the NM does not receive any ambiguous information (from a higher-level manager). Henceforth, it knows that all symptoms have to be explained in its domain. Intra-domain scenarios are similar to mixed scenarios, because both inter-and intra-domain symptoms may be generated as a result of intra-domain faults. Thus, in intra-domain scenarios, domain managers have to deal with the same level of ambiguity as is the case with mixed-fault scenarios.

To understand the difference among these three types of experiments it is useful to compare the numbers of simultaneous faults and symptoms generated in each experiment, which are presented in Figures 11(a)-11(b). These figures show that in inter-domain scenarios, the number of faults existing in the network is small (in most experiments only one fault was present), and does not change as the domain size increases, while the number of symptoms observed grows fast with the growing domain size. When the number of observed symptoms is big and the number of faults to isolate is small, fault localization may be performed with very high accuracy. Naturally, a big number of symptoms to diagnose increases the fault localization

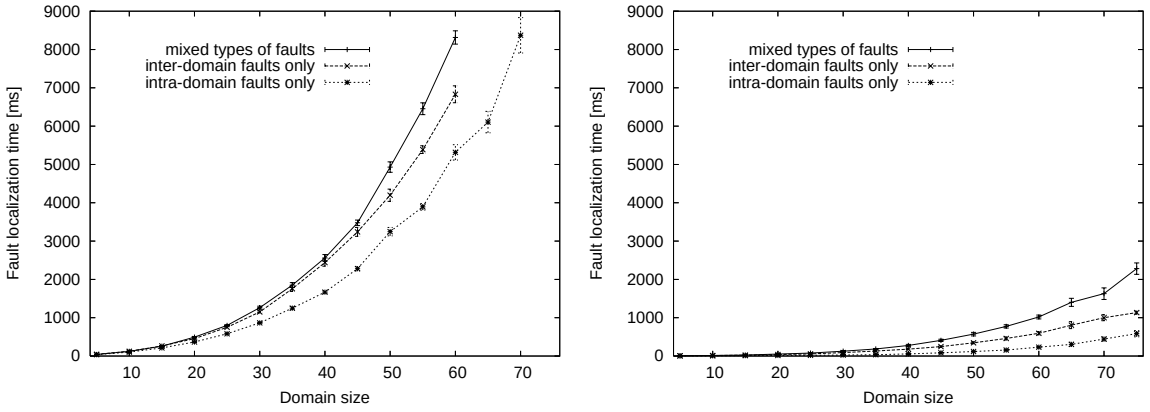
time. In intra-domain- and mixed-fault scenarios, increasing the domain size also increases the frequency of multi-fault scenarios.



(a) Average number of simultaneous faults (b) Average number of observed symptoms

Fig. 11. Average number of faults and symptoms generated in experiment scenarios for a ten-domain network

Figures 12(a)-12(b) compare the fault localization times of Algorithms 3A and 3B. The fault localization time is defined as the time needed to analyze all symptoms received in the considered fault localization scenario and to propose the most probable hypothesis. It is measured under the assumption that each symptom is available to the fault localization process as soon as the analysis of the previous symptom has completed. Thus, this measurement ignores the impact of symptom latencies. As expected, Algorithm 3B offers a much better performance than Algorithm 3A, which is due to its lower computational complexity. The difference in performance among mixed-, intra-domain-, and inter-domain-fault scenarios results from their different complexities expressed by the number of simultaneous faults and the number of received symptoms.

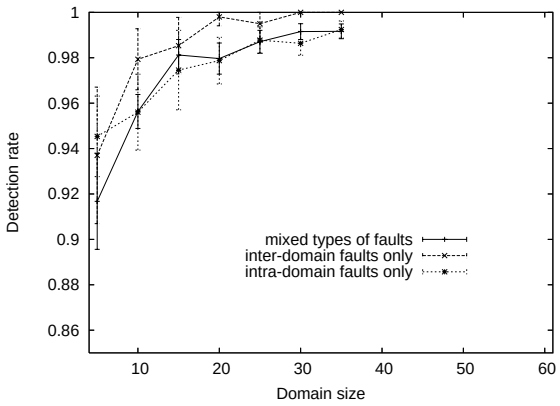


(a) Algorithm 3A (b) Algorithm 3B

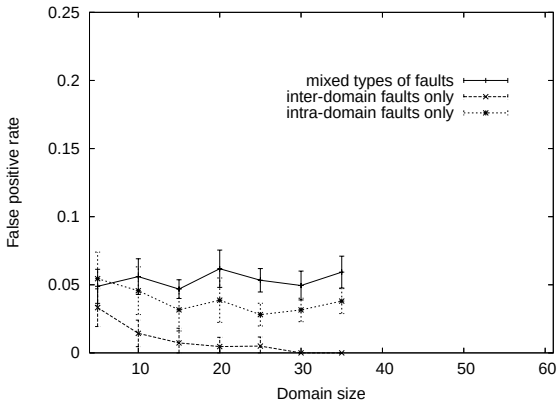
Fig. 12. Fault localization time in a network composed of ten domains.

We repeat the same set of experiments using networks composed of 50 domains. The results of the study are presented in Figures 13(a)-13(b) for Algorithm 3A and in Figures 14(a)-14(b) for Algorithm 3B, respectively. Note that, in the case of Algorithm 3B, we now work with networks composed of as many as 3000 nodes. For completeness, we also include Figures 15(a)-15(b), which show the average numbers of faults and symptoms generated in the considered fault scenarios. Figures 16(a)-16(b) compare the fault localization times of Algorithms 3A and 3B. The study performed on a fifty-domain network confirms the results discussed previously. However, note that in a bigger network, the complexity of scenarios is much higher: in a fifty-domain network, our fault localization techniques are required to accurately diagnose scenarios that involve more than 6 simultaneous faults (Figure 15(a)) and more than 2500 symptoms (Figure 15(b)).

In Figures 17(a)-17(c) we present the comparison of detection rate, false positive rate, and fault localiza-

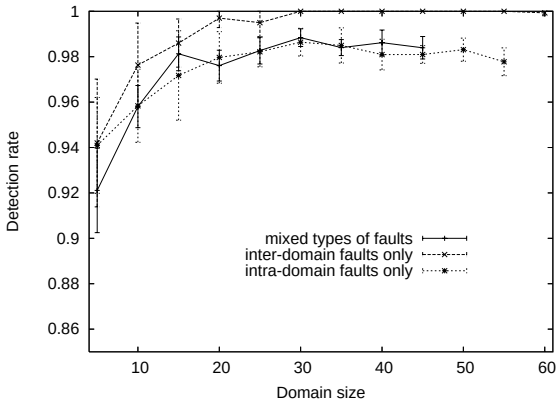


(a) Detection rate

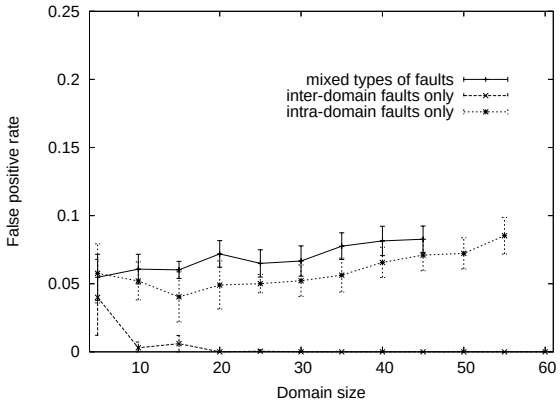


(b) False positive rate

Fig. 13. Accuracy of Algorithm 3A in a fifty-domain network.



(a) Detection rate



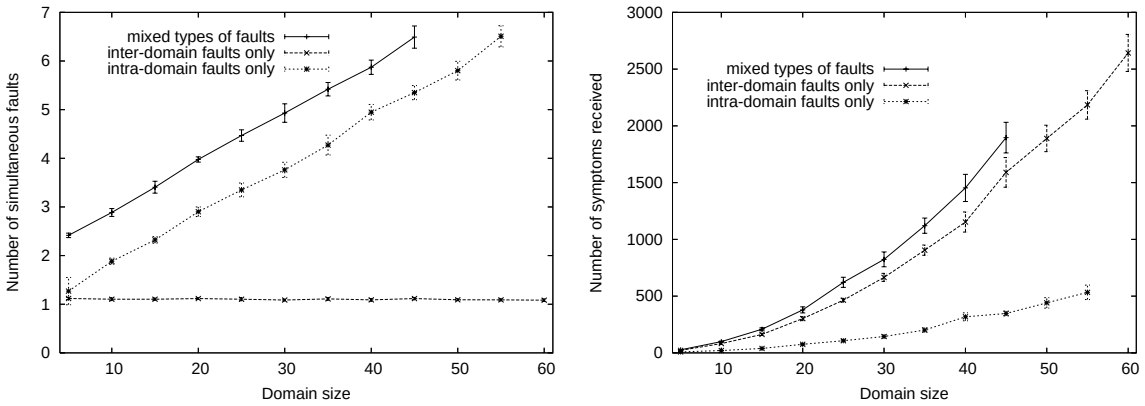
(b) False positive rate

Fig. 14. Accuracy of Algorithm 3B in a fifty-domain network.

tion time for centralized and multi-domain versions of Algorithms 1 and 2. Due to the excesive computation time of centralized algorithms we had to significantly limit the scope of the experiments, which were executed in a five-domain network in which the domain size varied between 5 and 15. The observability ratios of intra-domain and inter-domain symptoms were 0.5 and 0.1, respectively. The figures show that distributed fault localization performed according to the framework defined by Algorithm 3 may be as accurate as centralized fault localization, while offering much better scalability. In fact, in smaller networks, multi-domain fault localization may be even more accurate as centralized one, because it takes advantage of the hierarchical composition of network paths. Multi-domain fault localization proves much more efficient than centralized one, decreasing the fault localization time by a order of magnitude. (Since the algorithms are implemented in JAVA, some of the computational cost incurred by centralized algorithms is due to the large size of the fault propagation model, which reduces the amount of memory available to the fault localization process thereby increasing garbage-collection overhead.) Further improvement of the algorithms' efficiency is possible by executing managers in parallel.

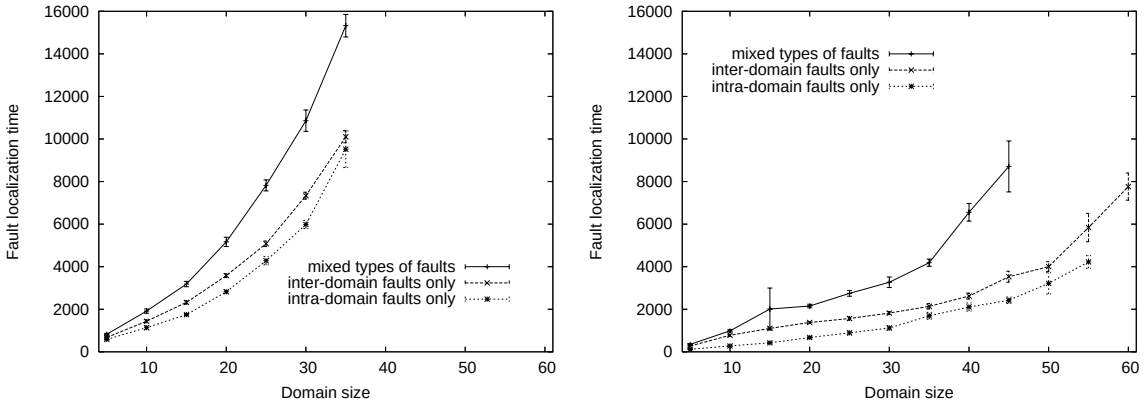
IX. Discussion

In this section, we discuss practical aspects of Algorithm 3 that are important to its applicability in real-life systems. These issues concern the creation of the fault propagation model, the validity of the distributed technique's assumptions, multi-layer fault localization, and the incremental property of the distributed version of Algorithm 2.



(a) Average number of simultaneous faults (b) Average number of observed symptoms

Fig. 15. Average number of faults and symptoms generated in experiment scenarios for a fifty-domain network



(a) Algorithm 3A (b) Algorithm 3B

Fig. 16. Fault localization time in a network composed of fifty domains.

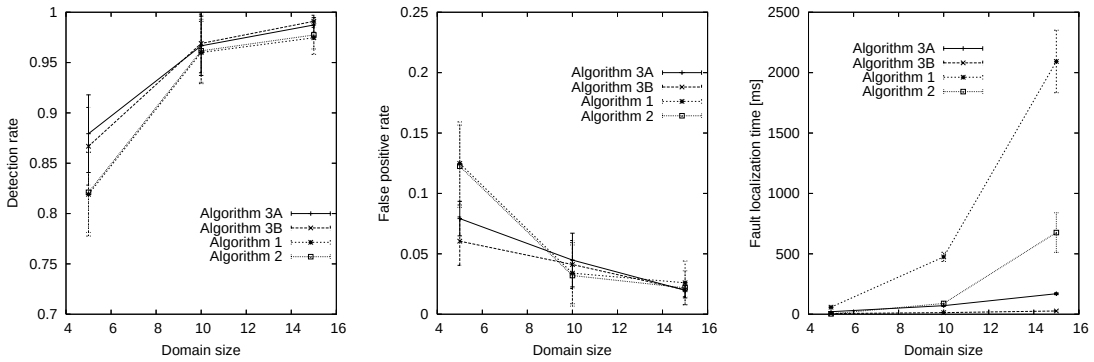
A. Obtaining the fault propagation model

Clearly, the biggest challenge in applying the fault localization technique proposed in this paper to real-life problems is obtaining the probabilistic fault propagation model. To build an FPM for end-to-end service failure diagnosis a knowledge of network logical topology and communication protocols is needed. The problem of building FPMs is beyond the scope of this paper. However, in this section we would like to emphasize that although it is not an easy task, building such models is possible with the information that is typically available through widely-deployed management protocols.

The network topology may be obtained automatically through various network topology detection mechanisms, which are built into some commercially available network management systems [44]. Other management systems implement proprietary techniques that allow the discovery of hardware configuration changes such as addition or removal of a network adapter or host [14]. The IETF has recently recognized a need for a standardized means of representing the physical network connections by proposing the `Physical Topology MIB` [5], which can be used to obtain topology information if it is implemented in the managed domain.

Obtaining dynamic dependencies is significantly more difficult since the dependency model has to be continuously updated while the modeled system is running. In spite of that, many tools exist that facilitate the process of model building. These tools are usually specific to particular network services or functions. For example, all active TCP connections on a host may be retrieved using the *netstat* application [42]. A current route between a host and any other host may be obtained using program *traceroute* [42].

Network management protocols such as SNMP [10] provide a means to determine dependencies established using configuration or real-time routing protocols. For example, the management system may obtain



(a) Detection rate (b) False positive rate (c) Fault localization time

Fig. 17. Comparison of centralized and multi-domain fault localization.

the topology, which was dynamically established in the data-link layer by the Spanning Tree Protocol [31] using the data contained in `dot1dBase` Group of *Bridge MIB* [12]. Updates of the spanning tree may be triggered by `newRoot` and `topologyChange` traps [12]. In the network layer of the Internet, current routes may be calculated from `ipRoutingTable` of *TCP/IP MIB-II* [24]. In source routing protocols (e.g., Source-Directed Relay of the military protocol MIL-STD 188-220 [13] and Dynamic Source Routing [19] proposed for wireless mobile networks), the current route between two hosts is embedded in the header of every transmitted packet. When BGP [33] is used as an inter-domain routing protocol, the NM can use the SNMP protocol [11] to query the BGP MIB [45] on exterior gateways. The BGP MIB contains all information that is needed by the NM to build its FPM, i.e., the list of exterior gateways (BGP peers [33]) and the information on interconnections among the gateways. Similarly, intra-domain routing information can be obtained via the SNMP from the intra-domain routers. For example, the OSPF MIB [3] can be queried in a domain using the OSPF [28] as a routing protocol.

Other techniques of obtaining network topology have been also investigated. To monitor hierarchical network topology, Novaes [29] uses IP multicast. Siamwalla et al. [34] propose several heuristics that exploit SNMP, DNS, ping, and traceroute facilities to discover the network level topology of the Internet on both intra-domain and backbone levels. Govindan et al. [15] infer the Internet map using hop-limited probes. Reddy et al. [32] investigate a distributed topology discovery technique for the Internet. Breitbart et al. [7] present an algorithm that obtains both network and data-link layer topology using *TCP/IP MIB-II* [24] and *Bridge MIB* [12]. The algorithm of Lowekamp et al. [23] discovers the topology of a large Ethernet network allowing incomplete data in *Bridge MIB* tables.

To build a probabilistic FPM, in addition to the information on the structure of dependencies in the considered system, we also need an assessment of confidence that a failure of an antecedent function or service causes a failure of a dependent function or service. When such information is not available, the same value may be assigned to all edges in the probabilistic FPM. As more information becomes available that allows us to say that causal relationships between certain link and path failures are stronger than causal relationships between other link and path failures, this knowledge can be used to modify the FPM by changing the conditional probabilities accordingly. Thus, the probabilistic FPM does not require the precise knowledge of the conditional probability distribution. In fact, discrete confidence levels can be used instead of continuous probability values. In our previous work [39], [40], we have shown that, in a well instrumented network, as few as three confidence levels allow fault localization to be almost as accurate as with the precise knowledge of the conditional probability distribution. The value of the probabilistic fault localization technique proposed in this paper is that it can improve the accuracy of fault localization, as compared to deterministic techniques, by taking advantage of available information on non-deterministic causal relationships among failure conditions.

B. Hierarchical routing assumption

The are a number of real-life situations, where the assumptions introduced in Section III do not hold. This concerns particularly the single ingress- and egress-gateway assumption, which implies single-path routing. In reality, multiple paths may exist between a source and a destination. A router decides which of available paths it should choose based on various criteria [27], which include, for example, the parameters of a TCP connection to which a packet belongs (to ensure all packets in the same TCP session use the same route), (2) the TOS field in an IP datagram (in OSPF [28] used to differentiate the provided service based on an objective specified in the TOS fields), or the *options* field of an IP datagram. The easiest way to incorporate multiple routes between a source and a destination in the FPM is by including all links in these routes as possible causes of an end-to-end disorder between the source and destination. Then, conditional probabilities may be used to account for the frequency with which one route is used with respect to other routes. Investigating more refined techniques of modeling multi-path routing is a future research problem.

C. Multi-level fault localization

In this paper, we presented a multi-domain fault localization solution that organizes domain and network managers into a two-level hierarchy. This technique may be easily extended to a multi-level hierarchical solution. In the extended technique, managers at intermediate levels perform functions of both a DM and the NM and their FPMs contain features of FPMs of both a DM and the NM, i.e., they contain both $\mathcal{P}$- and $\tilde{\mathcal{P}}$-nodes. Combining these models and functionalities is rather straightforward. The only issue requiring an explanation is the calculation of the prior failure probabilities assigned to $\mathcal{P}$-nodes in an FPM of a DM that is not a leaf-level manager. Recall from Section V that this probability is calculated by a DM using function *compute_prior* based on posterior fault probabilities represented by *fault_bel*. In a two-level hierarchy, each fault considered by a DM in the process of calculating *compute_prior* belongs to its managed domain (it represents a failure of one of the domain's host-to-host links), and therefore the values of *fault_bel* can be obtained easily. In a multi-level hierarchy, a DM$_i$ that is not a leaf-level manager may need to obtain *fault_bel*(f_j), where f_j represents a failure of a path segment $n_{p_1} \stackrel{*}{\rightarrow} n_{p_m}$ that spans multiple sub-domains of the $\mathcal{D}_i$. In this case, f_j may have no representation in the FPM of DM$_i$. Thus, to calculate *fault_bel*(f_j) exactly, DM$_i$ would have to split $n_{p_1} \stackrel{*}{\rightarrow} n_{p_m}$ into path-segments and links $n_{p_1} \stackrel{*}{\rightarrow} E^{l_1}_{l_1,l_k}$, $E^{l_1}_{l_1,l_k} \rightarrow I^{l_2}_{l_1,l_k}$, $I^{l_2}_{l_1,l_k} \stackrel{*}{\rightarrow} E^{l_2}_{l_1,l_k}$, ..., $E^{l_{k-1}}_{l_1,l_k} \rightarrow I^{l_k}_{l_1,l_k}$, $I^{l_k}_{l_1,l_k} \stackrel{*}{\rightarrow} n_{p_m}$, where $\mathcal{D}_{l_1}$, ..., $\mathcal{D}_{l_k}$ are sub-domains of $\mathcal{D}_i$ that are traversed by $n_{p_1} \stackrel{*}{\rightarrow} n_{p_m}$. Then, DM$_i$ would request DM$_{l_1}$, ..., DM$_{l_k}$ to estimate the failure probabilities of segments $n_{p_1} \stackrel{*}{\rightarrow} E^{l_1}_{l_1,l_k}$, $I^{l_2}_{l_1,l_k} \stackrel{*}{\rightarrow} E^{l_2}_{l_1,l_k}$, ..., $I^{l_k}_{l_1,l_k} \stackrel{*}{\rightarrow} n_{p_m}$, respectively, using function *compute_prior*. Thus, assuming s_n represents a path segment corresponding to domain $\mathcal{D}_{l_n}$, *fault_bel*(f_j) would be obtained using the following formula:

$$\textit{fault\_bel}(f_j) = 1 - \prod_{n=1}^{k}(1 - \textit{compute\_prior}(\{s_n\})) \prod_{n=1}^{k-1}(1 - \textit{fault\_bel}(f : E^{l_n}_{l_1,l_k} \rightarrow I^{l_{n+1}}_{l_1,l_k}))$$

It may be shown that, in general, such an exact approach would require the number of steps that is exponential with the depth of the management hierarchy. A reasonable approach to avoiding this complexity is to map every segment s_n into $\mathcal{P}_n$, the closest-matching $\mathcal{P}$-node in the FPM of DM$_i$, and use function *fault_bel*($\mathcal{P}_n$) instead of *compute_prior*($\{s_n\}$). Note that for $n = 2 \ldots k-1$, this mapping is exact because the FPM of DM$_i$ contains $\mathcal{P}$-nodes $\mathcal{P} : I^{l_n}_{l_1,l_k} \stackrel{*}{\rightarrow} E^{l_{n+1}}_{l_1,l_k} = \{s_n\}$. Thus, the approximation concerns only the first and last segments of path $n_{p_1} \stackrel{*}{\rightarrow} n_{p_m}$.

D. Incremental algorithm

Recall from Section II that one of the positive features of Algorithm 2 is its ability to formulate an explanation hypothesis in an incremental manner. Thanks to this feature, the algorithm continuously provides a system administrator with information about which faults are likely to exist in the system given symptoms observed thus far. The distributed version of Algorithm 2, Algorithm 3B, is not incremental as it requires model synchronization each time an explanation is to be proposed. In our future research, we would like

to restore the incremental property in Algorithm 3B. In one approach, we could eliminate the model synchronization from the fault selection phase altogether. This would make the algorithm incremental. Unfortunately, eliminating model synchronization from the fault selection phase slightly deteriorates the fault localization accuracy. Thus, to maintain the high accuracy, a more intelligent approach to model synchronization is needed that would update prior probabilities associated with $\mathcal{P}$- and $\tilde{\mathcal{P}}$-nodes on a continuous basis during the symptom analysis phase without increasing its computational complexity. Once complete model synchronization is incorporated in the fault selection phase, it will no longer be required in the fault selection phase thereby allowing the algorithm to propose an explanation in an incremental manner.

X. Conclusion

The paper introduces a multi-domain fault localization approach to end-to-end service failure diagnosis in hierarchically routed networks. This approach divides the computational effort and system knowledge involved in end-to-end service-failure diagnosis among multiple, hierarchically organized managers. Each manager is responsible for fault localization within the network domain it governs, and reports to a higher-level manager that oversees and coordinates the fault-localization process of multiple domains. The paper identifies two main difficulties of fault management in multi-domain networks: failure propagation among domains and a lack of global information about the system structure and state. To address these challenges, the paper first proposes an algorithmic framework for the design of probabilistic hierarchical multi-domain fault localization techniques. It then introduces two refinements that expand on the centralized algorithms introduced in our previous work: iterative belief updating [41] and incremental hypothesis updating [36]. The multi-domain approach is shown to provide high accuracy while increasing the admissible network size by an order of magnitude. [2]

References

[1] W. Aiello, F. Chung, and L. Lu. A random graph model for massive graphs. In *Proc. of 32$^{\text{nd}}$ Annual ACM symposium on Theory of Computing*, pages 171–180, Portland, OR, May 2000.

[2] R. Albert and A. Barabasi. Topology of evolving network:local events and universality. *Physica Review Letters*, pages 5234–5137, 2000.

[3] F. Baker and R. Coltun. *OSPF Version 2 Management Information Base*. IETF Network Working Group, 1995. RFC 1850.

[4] A. Barabasi and R. Albert. Emergence of scaling in random networks. *Science*, pages 509–512, Oct. 1999.

[5] A. Bierman and K. Jones. *Physical topology MIB.* IETF Network Working Group, 2000. RFC 2922.

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

[7] Y. Breitbart, M. Garofalakis, C. Martin, R. Rastogi, S. Seshadri, and A. Silberschatz. Topology discovery in heterogeneous IP networks. In *Proc. of IEEE INFOCOM*, pages 265–274, 2000.

[8] T. Bu and D. Towsley. On distringuishing between Internet power law topology generators. In *Proc. of IEEE INFOCOM*, New York, NY, Jun. 2002.

[9] K. Calvert, M. Doar, and E. Zegura. Modeling Internet topology. *IEEE Transactions on Communications*, pages 160–163, Dec. 1997.

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

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

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

[13] DoD. *Military Standard—Interoperability Standard for Digital Message Device Subsystems (MIL-STD 188-220B)*, Jan. 1998.

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

[15] R. Govindan and H. Tangmunarunkit. Heuristics for Internet map discovery. In *Proc. of IEEE INFOCOM*, pages 1371–1380, 2000.

[16] P. Hasselmeyer. An infrastructure for the management of dynamic service networks. *IEEE Communications Magazine*, 41(4):120–126, 2003.

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

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

[18] C. Jin, Q. Chen, and S. Jamin. Inet: Internet topology generator. Technical Report CSE-TR443-00, EECS, University of Michigan, 2000.

[19] D. Johnson and D. Maltz. *Dynamic Source Routing in Ad Hoc Wireless Networks*, chapter 5, pages 153–181. Kluwer Academic Publishers, 1996.

[20] I. Katzela, A. T. Bouloutas, and S. Calo. Comparison of distributed fault identification schemes in communication networks. Technical Report RC 19630 (87058), T. J. Watson Research Center, IBM Corp., Sep. 1993.

[21] I. Katzela, A. T. Bouloutas, and S. B. Calo. Centralized vs distributed fault localization. In A. S. Sethi, F. Faure-Vincent, and Y. Raynaud, editors, *Integrated Network Management IV*, pages 250–263. Chapman and Hall, May 1995.

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

[23] B. Lowecamp, D. R. O'Hallaron, and T. R. Gross. Topology discovery for large Ethernet networks. In *Proc. of ACM SIGCOMM*, pages 239–248, 2001.

[24] K. McCloghrie and M. Rose. *Management Information Base for Network Mangement of TCP/IP-based internets: MIB-II*. IETF Network Working Group, 1991. RFC 1213.

[25] A. Medina, A. Lakhina, I. Matta, and J. Byers. BRITE:universal topology generation from a user's perspective. Technical Report BUCS-TR-2001-003, Computer Science Department, Boston University, Apr. 2001.

[26] A. Medina, I. Matta, and J. Byers. On the origin of power laws in Internet topologies. *ACM Computer Communications Review*, 30(2):18–28, Apr. 2000.

[27] J. T. Moy. *OSPF: Anatomy of an Internet Routing Protocol*. Addison Wesley Longman, 1998.

[28] J. T. Moy. *OSPF Version 2*. IETF Network Working Group, 1998. STD 54.

[29] M. Novaes. Beacon: A hierarchical network topology monitoring system based in IP multicast. In A. Ambler, S. B. Calo, and G. Kar, editors, *Services Management in Intelligent Networks*, number 1960 in Lecture Notes in Computer Science, pages 169–180. Springer-Verlag, 2000.

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

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

[32] A. Reddy, D. Estrin, and R. Govindan. Large-scale fault isolation. *Journal on Selected Areas in Communications*, 18(5):723–732, 2000.

[33] Y. Rekhter and T. Li. *A Border Gateway Protocol 4 (BGP-4)*. IETF Network Working Group, 1995. RFC 1771.

[34] R. Siamwalla, R. Sharma, and S. Keshav. Discovering Internet topology. Technical report, Cornell University, 1998.

[35] M. Steinder. *Probabilistic inference for diagnosing service failures in communication systems*. PhD thesis, University of Delaware, 2003.

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

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

[38] M. Steinder and A. S. Sethi. Distributed fault localization in hierarchically routed networks. In M. Feridun, P. Kropf, and G. Babin, editors, *13th Int'l Workshop on Distributed Systems: Operations and Management*, number 2506 in Lecture Notes in Computer Science, pages 195–207, Montréal, Canada, Oct. 2002. Springer.

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

[40] M. Steinder and A. S. Sethi. Non-deterministic event-driven fault diagnosis through incremental hypothesis updating. In G. Goldszmidt and J. Schoenwaelder, editors, *Integrated Network Management VIII*, Colorado Springs, CO, Mar. 2003.

[41] M. Steinder and A.S. Sethi. End-to-end service failure diagnosis using belief networks. In R. Stadler and M. Ulema, editors, *Proc. Network Operation and Management Symposium*, pages 375–390, Florence, Italy, Apr. 2002.

[42] W. R. Stevens. *TCP/IP Illustrated*, volume I. Addison Wesley, first edition, 1995.

[43] A. S. Tanenbaum. *Computer Networks*. Prentice Hall, second edition, 1996.

[44] Tivoli. *Netview for Unix: Administrator's Guide, Version 6.0*, Jan. 2000.

[45] S. Willis, J. Burruss, and J. Chu. *Definitions of Managed Objects for the Fourth Version of the Border Gateway Protocol (BGP-4) using SMIv2*. IETF Network Working Group, 1994. RFC 1657.

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