Efficient Fault Diagnosis Using Incremental Alarm Correlation and Active Investigation for Internet and Overlay Networks

Yongning Tang, *Member, IEEE,* Ehab Al-Shaer, *Member, IEEE,* and Raouf Boutaba, *Senior Member, IEEE*

***Abstract*—Fault localization is the core element in fault management. Symptom-Fault map is commonly used to describe the Symptom-Fault causality in fault reasoning. For Internet service networks, a well-designed monitoring system can effectively correlate the observable symptoms (i.e., alarms) with the critical network faults (e.g., link failure). However, the lost and spurious symptoms can significantly degrade the performance and accuracy of a passive fault localization system. For overlay networks, due to limited underlying network accessibility, as well as the overlay scalability and dynamics, it is impractical to build a static overlay Symptom-Fault map. In this paper, we firstly propose a novel Active Integrated fault Reasoning (*AIR*) framework to incrementally incorporate active investigation actions into the passive fault reasoning process based on an extended Symptom-Fault-Action (*SFA*) model. Secondly, we propose an Overlay Network Profile (*ONP*) to facilitate the dynamic creation of an Overlay Symptom-Fault-Action (called *O-SFA*) model, such that the *AIR* framework can be applied seamlessly to overlay networks (called *O-AIR*). As a result, the corresponding fault reasoning and action selection algorithms are elaborated. Extensive simulations and Internet experiments show that *AIR* and *O-AIR* can significantly improve both accuracy and performance in the fault reasoning for *Internet* and *Overlay* service networks, especially when the ratio of the lost and spurious symptoms is high.**

***Index Terms*—Fault localization, symptom-fault map, fault reasoning, overlay networks.**

I. Introduction

FAULT localization is the core element in the fault management system, because it identifies the fault causes that can best explain the observed network symptoms (i.e. alarms). Most fault reasoning algorithms use a bipartite directed acyclic graph to describe the Symptom-Fault correlation, which represents the causal relationship between each fault f_i and a set of its observed symptoms S_{f_i} ([9]). The Symptom-Fault causality graph provides a vector of correlation likelihood measure $p(s_i|f_i)$, to bind a fault f_i to a set of its symptoms S_{f_i}.

Manuscript received February 5, 2007; revised July 1, 2007; approved by IEEE Transactions on Network and Service Management Editor Joseph Hellerstein.

Y. Tang is with the School of Information Technology, Illinois State University, Normal, IL USA (e-mail: ytang@ilstu.edu).

E. Al-Shaer is with the School of Computer Science, Telecommunications and Information Systems, DePaul University, Chicago, IL, USA (e-mail: ehab@cs.depaul.edu).

R. Boutaba is with the David R. Cheriton School of Computer Science, University of Waterloo, Waterloo Canada (e-mail: rboutaba@uwaterloo.ca).

Digital Object Identifier 10.1109/TNSM.2008.080104.

For an Internet Service Provider (e.g., AT&T), a well-designed network monitoring system can be deployed to continuously monitor the critical network elements (e.g., routers) for certain network behaviors (e.g., link connectivity) and raise alarms in the event of a failure. The corresponding Symptom-Fault map can be used to reflect such an intrinsic relationship between the symptoms (i.e., alarms) and the corresponding faults (e.g., link failure). Two approaches are commonly used in fault reasoning and localization: passive diagnosis ([5], [8], [9], [12] and active investigation ([6], [10], [11], [15]). In the passive approach, all symptoms are passively collected and then processed to infer the root faults. In the active approach, faults are detected by conducting a set of investigation actions. The passive approach causes less network intrusiveness; however, it may take a long time to discover the root faults, particularly if the symptom loss ratio (SLR) is high. Here, $SLR = (S - S_O)/S$, where S_O is the observed symptoms and S is the total generated symptoms. Although the active investigation approach is more efficient to identify faults quickly, active investigation (e.g., probing) might cause significant overhead particularly in large-scale networks.

For an Overlay Service Provider (e.g., Akamai), because of commercial reasons, critical information (e.g., network fault statistics) from ISPs is not shareable to an OSP. Moreover, due to the dynamics and scalability of overlay networks, it is impractical for an OSP to construct a static Symptom-Fault map to facilitate fault localization in overlay networks.

In this paper, we address the fault localization problem for two common but different networks: Internet and Overlay service networks. Firstly, we propose a novel fault localization technique that integrates the advantage of both passive and active fault reasoning into one framework, called *Active Integrated fault Reasoning* or *AIR*. The goal of AIR is to balance the use of active and passive measurements. Active investigation actions are properly selected only if the passive reasoning is not sufficient. If there are too many passive symptoms that might explain most of the faults, then the use of active measurements will be automatically reduced. Secondly, to tackle the new challenges (e.g., inaccessible underlying network information and overlay network dynamics) in overlay fault localization, we propose to build an Overlay Network Profile (*ONP*) to facilitate overlay fault localization. Based on the ONP, we introduce a dynamic Overlay Symptom-Fault-Action (*O-SFA*) model for overlay networks to incorporate the *AIR* framework into the overlay fault localization system (called *O-AIR*). Our approach significantly improves the performance of fault localization while minimizing the intrusiveness of the active fault reasoning.

The paper is organized as follows. After discussing the related work in Section II, we introduce our research motivation and the problem formalization in section III. In section IV, we describe the components and algorithms of *AIR*. In Section V, we introduce the Overlay Network Profile and the dynamic Overlay Symptom-Fault-Action model. Then we discuss the overlay fault localization framework called *O-AIR*. In Section VI, we present our simulations and Internet experiments to evaluate *AIR* and *O-AIR* performance and accuracy for both Internet and Overlay service networks. In section VII, we conclude the paper and discuss our future work.

II. Related Work

There is a significant amount of work has been done in the area of fault localization, for both Internet and Overlay networks. Accordingly, we classify the corresponding solutions into Administrator-Level and User-Level fault localization.

A. Administrator-Level Fault Localization.

For a single administratively controlled network such as an ISP network, the Symptom-Fault causality model is commonly used to infer the root faults based on the observation of network disorders. In the following, we classify this type of related work into two categories:

Passive Approach: Passive fault management techniques typically depend on monitoring agents to detect and report network abnormalities using alarms or symptom events. These events are then analyzed and correlated in order to find the root faults. Various event correlation models were proposed including rule-based analyzing systems [17], case-based diagnosing systems, and model-based systems [18]. Different techniques were also introduced to improve the performance, accuracy and resilience of fault localization. In [12], a model-based event correlation engine is designed for multi-layer fault diagnosis. In [5], the coding approach is applied to a deterministic model to reduce the reasoning time and improve the system resilience. A novel incremental event-driven fault reasoning technique is presented in [8] and [9] to improve the robustness of a fault localization system by analyzing lost, positive and spurious symptoms.

The techniques above were developed based on the passively received symptoms. If the symptoms are collected correctly, the fault reasoning results can be accurate. However, in real systems, the symptom loss or spurious symptoms (called observation noise) are unavoidable. Even with a good strategy ([9]) to deal with the observation noise, those techniques still only have limited resilience to such observation noise because of the passive fault analysis, which might also increase the fault detection time.

Active Approach: Recently, researchers have proposed several active fault localization approaches. In [11], an active probing fault localization system is introduced, in which preplanned active probes are associated with system status using a dependency matrix. An on-line action selection algorithm is studied in [10] to optimize action selection. In [15], a fault detection and resolution system is proposed for large distributed transaction processing systems. In [3], a Computing Utility profit maximization problem is formulated as a Markov Decision Process (MDP), which shows an useful model to formalize comprehensive action selection problems.

The active approach is more efficient in locating faults in a timely fashion and more resilient to the observation noise. However, this approach lacks a scalable technique that can deal with multiple simultaneous faults. It also cannot easily isolate intermittent network faults and performance-related faults because it solely depends on the active probing model. In this approach, the number of required probes might be increased exponentially to the number of possible faults ([10]).

Both passive and active approaches have their own good features and limitations. Thus, incorporating active investigation actions into the passive fault reasoning approach is an ideal framework. *AIR* combines the good features of both passive and active approaches and overcomes their limitations by optimizing the fault reasoning result and the action selection process.

B. User-Level Fault Localization.

Recently, more research has focused on user-level network measurement and fault diagnosis tools/approaches, particularly for overlay networks. In the following, we classify them as diagnosis tools and frameworks.

Diagnosis Tools: Many end-to-end traffic measurement tools were proposed for monitoring packet loss and other path properties for problem diagnosis such as *Sting*, *Cing* and *Tulip*. These tools are good for diagnosing a specific network property but are not adequate as a general problem diagnosis in overlay networks. Most of these techniques cause extensive intrusiveness due to active probings. Moreover, they may not discover intermittent problems.

Diagnosis Framework: One of the most interesting recent works is the Tomography-based approach that estimates network performance parameters based on traffic measurement on a limited subset of overlay nodes [21], [22]. However, similar to previous tools, it is still a purely active approach that usually requires extensive probings in order to achieve accurate results no matter whether the problems exist or not. On the other hand, PlanetSeer([23]), taking the Multiple Vantage Point Approach locates Internet faults by selectively and periodically invoking "traceroute" from multiple vantage points. The measurement model is manually managed and only matches the application domain directions of data flow.

To the best of our knowledge, our proposed *O-AIR* is the first comprehensive user-level probabilistic overlay fault diagnosis framework that integrates active monitoring with passive fault reasoning based on the dynamic generated Overlay Symptom-Fault-Action model.

III. Motivation and Problem Formulation

For traditional Internet service networks, active fault management does not scale well when the number of nodes or faults grows significantly in the network. In fact, some faults, such as the intermittent reachability problem, may

TABLE I
ACTIVE INTEGRATED FAULT REASONING NOTATION

Notation	Definition
S_{f_i}	a set of all symptoms caused by the fault f_i
F_{s_i}	a set of all faults that might cause symptom s_i
S_O	a set of all observed symptoms so far
S_{O_i}	a set of *observed* symptoms caused by fault f_i
S_{U_i}	a set of *not-yet-observed* (lost) symptoms caused by the fault f_i
h_i	a set of faults that constitute a possible hypothesis that can explain S_O
Φ	a set of all different fault hypotheses, h_i, that can explain S_O
S_N	a set of correlated but not-yet-observed symptoms associated with any fault in a hypothesis
S_V	a subset of S_N, which includes symptoms the *existence* of which is confirmed
S_U	a subset of S_N, which includes symptoms the *non-existence* of which is confirmed

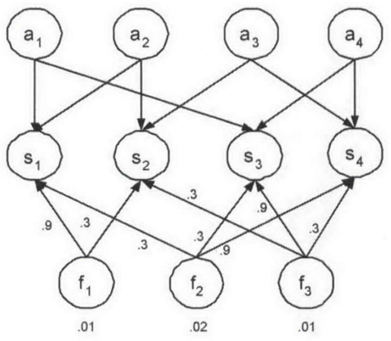

Fig. 1. Symptom-Fault-Action Model.

not even be identified if only active fault management is used. However, this can easily be reported using passive fault management systems because agents are configured to report abnormal system conditions or symptoms, such as a high average packet drop ratio. On the other hand, symptoms can be lost due to noisy or unreliable communications channels, or they might be corrupted due to spurious (untrue) symptoms generated as a result of malfunctioning agents or devices. This significantly reduces the accuracy and the performance of passive fault localization. Only the integration of active and passive reasoning can provide efficient fault localization solutions.

To incorporate actions into a traditional Symptom-Fault model, we propose an extended Symptom-Fault-Action (*SFA*) model as shown in Fig. 1. In our model, actions are properly selected probes or test transactions that are used to detect or verify the existence of observable symptoms. Actions can simply include commonly used network utilities, like ping and traceroute; or some proprietary fault management system (e.g., [6]. We assume that symptoms are verifiable, which means that if the symptom ever occurred, we could verify the symptom existence by executing some investigation actions (e.g., probing) or checking the system status through, for example, system logs.

In this paper, we use $F = \{f_1, f_2, \ldots, f_n\}$ to denote the *fault set*, and $S = \{s_1, s_2, \ldots, s_m\}$ to denote the *symptom set* that can be caused by one or multiple faults in F. The causality matrix $P_{F\times S} = \{p(s_i|f_j)\}$ is used to define causal certainty between fault $f_i (f_i \in F)$ and symptom $s_i (s_i \in S)$. If $p(s_i|f_j) = 0$ or 1 for all (i, j), we call such a causality model a deterministic model; otherwise, we call it a probabilistic model. We also use $A = \{a_1, \ldots, a_k\}$ to denote the list of actions that can be used to verify symptom existence. We describe the relation between actions and symptoms using *Action Codebook* represented as a bipartite graph as shown in Fig. 1. For example, the symptom s_1 can be verified using action a_1 or a_2. The Action Codebook can be defined by network managers based on symptom type, the network topology, and the available fault diagnostic tools. The extended Symptom-Fault-Action (*SFA*) graph is viewed as a 5-tuple (S, F, A, E_1, E_2), where fault set F, symptom set S, and action set A are three independent vertex sets. Every edge in E_1 connects a vertex in S and another vertex in F to indicate a causality relationship between symptoms and faults. Every edge in E_2 connects a vertex in A and another vertex in S to indicate the Action Codebook. For convenience, in Table I, we introduce the notations used in our discussion on Active Integrated Fault Reasoning. The basic Symptom-Fault-Action model can be described as follows:

- For every action, associate an action vertex a_i, $a_i \in A$;
- For every symptom, associate a symptom vertex s_i, $s_i \in S$;
- For every fault, associate a fault vertex f_i, $f_i \in F$;
- For every fault f_i, associate an edge to each s_i caused by this fault with a weight equal to $p(s_i|f_i)$;
- For every action a_i, associate an edge of weight equal to the action cost to each symptom verifiable by this action.

Performance and accuracy are the two most important factors for evaluating fault localization techniques. Performance is measured by fault detection time T, which is the time between receiving the fault symptoms and identifying the root faults. The fault diagnostic accuracy depends on two factors: (1) the detection ratio (α), which is the ratio of the number of *true* detected root faults (F_d is the total detected fault set) to the number of *actual* occurred faults F_h, formally $\alpha = \frac{|F_d \cap F_h|}{|F_h|}$; and (2) the false positive ratio (β), which is the ratio of the number of *false* reported faults to the total number of detected faults, formally $\beta = \frac{|F_d - F_d \cap F_h|}{|F_d|}$ [9]. Therefore, the goal of any fault management system is to increase α and reduce β in order to achieve highly accurate fault reasoning results.

The task of fault reasoning is to search for root faults in F based on the observed symptoms S_O. Our objective is to improve fault reasoning by minimizing the detection time, T and the false positive ratio, β, and by maximizing the detection ratio, α.

In order to develop this system, we have to address the following three problems: (1) Given the Fault-Symptom correlation matrix and the set of observed symptoms (S_O), construct a set of the most possible hypotheses, $\Phi = \{h_1, h_2, \ldots, h_p\}, h_i \subseteq F$, that can explain the current observed symptoms; (2) Given a set of possible hypotheses, find the most credible hypothesis h, that can give the best explanation for the current observed symptoms; (3) If the selected hypothesis does not satisfy the fidelity requirement, then given the unobserved symptoms S_N to select the minimum-cost actions to search for an acceptable hypothesis.

Compared to Internet service networks, overlay networks have emerged as a powerful and flexible platform for developing new disruptive network applications. We believe the following new characteristics, challenges, and service objectives suggest that overlay fault localization has to adopt a new approach:

- *Inaccessible underlying network information and incomplete network status observation:* Overlay network infrastructure is owned and controlled by Internet service providers (ISPs); however, overlay services are provisioned, operated and monitored by overlay service providers (OSPs). Overlay networks are formed by coordinated overlay nodes on top of opaque underlying networks. It is infeasible to know the prior fault probability of these underlying components and encode a probabilistic relationship between underlying components and user-level observations. An overlay fault diagnosis technique must be developed based on incomplete and insufficient user observations without relying on an underlying network fault probabilistic model as in [7][4][9].
- *Multi-layer complexity and dynamic Symptom-Fault causality:* The flexibility and dynamics of overlay nodes and links make the interaction (also correlation) between overlay and underlying networks unpredictable. In overlay networks, observed symptoms are usually not designed and collected for monitoring specific faults (e.g., malfunctioning network router interfaces). Symptom-Fault causality relationship is dynamic and unpredictable in overlay networks.
- *Fault reasoning goal and granularity:* The goal in traditional fault reasoning is to accurately locate the faulty components and fix them. However, in overlay network, it becomes more effective for overlay applications to avoid detected faulty components instead of fixing them. Thus, for overlay fault reasoning, coarse-grained fault reasoning is acceptable and may be more preferable.

In the following, we will first discuss the solution for the above three issues in the fault localization of Internet Service Networks. Then, we will propose the solution to address the new challenges in the fault localization of Overlay Service Networks.

IV. Active Integrated Fault Reasoning

The Active Integrated Fault Reasoning (*AIR*) process (Fig. 2) includes three functional modules: Fault Reasoning (FR), Fidelity Evaluation (FE), and Action Selection (AS). The Fault Reasoning module takes passively observed symptoms S_O as input and returns the fault hypothesis set Φ as output. The fault hypothesis set Φ might include a set of hypotheses $(h_1, h_2, \ldots, h_n)$, where each one contains a set of faults that explains all observed symptoms up to that point. Then, Φ is sent to the Fidelity Evaluation module to check if any hypothesis h_i ($h_i \in \Phi$) is satisfactory. If most correlated symptoms necessary to explain the fault hypothesis h_i are observed (i.e., high fidelity), then the Fault Reasoning process terminates. Otherwise, a list of unobserved symptoms, S_N, that contribute to explain the fault hypothesis h_i of the highest fidelity is sent to the Action Selection module to determine which symptoms have occurred. As a result, the fidelity value of hypothesis h_i is adjusted accordingly. The conducted actions return the test result with a set of existing symptoms S_V and non-existing symptoms S_U. The corresponding fidelity value might be increased or decreased based on the action return results. If the newly calculated fidelity is satisfactory, then the reasoning process terminates; otherwise, S_O, S_U are sent as new input to the Fault Reasoning module to create a new hypothesis. This process is repeated until a hypothesis with high fidelity is found. Fidelity calculation is explained later in this section. In the next section, we describe the three modules in detail, then discuss the complete Active Integrated Fault Reasoning algorithm.

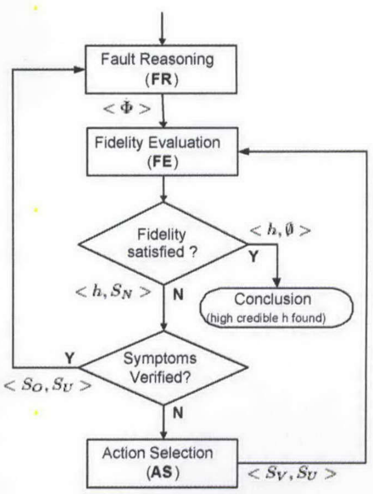

Fig. 2. **Active Action Integrated Fault Reasoning**

A. Heuristic Algorithm for Fault Reasoning

In the Fault Reasoning module, we use a *contribution function*, $C(f_i)$, as a criterion to find faults that have the maximal contribution of the observed symptoms. In the probabilistic model, symptom s_i can be caused by a set of faults $f_i, (f_i \in F_{s_i})$ with different possibilities $p(s_i|f_i) \in (0, 1]$. We assume that the Symptom-Fault correlation model is sufficient enough to neglect other undocumented faults and symptoms (i.e., prior fault/symptom probability is very low). Thus, we can also assume that symptom s_i will not occur if none of the faults in F_{s_i} happened. In other words, if s_i occurred, at least one f_i ($f_i \in F_{s_i}$) must have occurred. However, conditional probability $p(s_i|f_i)$ itself may not truly reflect the chance of fault f_i occurrence by observing symptom s_i. For example, in Fig. 1, there are three possible scenarios that can result from observing s_1: f_1 can happen, f_2 can happen or both can happen. Based on the common heuristic assumption that the possibility of multiple faults happening simultaneously is low [9], one of the faults (f_1 or f_2) should explain the occurrence of s_1. In order to measure the contribution of each fault f_i to the creation of s_i, we normalize the conditional probability $p(s_i|f_i)$ to the normalized conditional probability $\tau(s_i|f_i)$ to reflect the relative contribution of each fault f_i to the observation of s_i.

$$\tau(s_i|f_i) = \frac{p(s_i|f_i)}{\sum_{f_i \in F_{s_i}} p(s_i|f_i)} \quad (1)$$

With $\tau(s_i|f_i)$, we can compute normalized posterior probability $\mu(f_i|s_i)$ as follows.

$$\mu(f_i|s_i) = \frac{\tau(s_i|f_i)p(f_i)}{\sum_{f_i \in F_{s_i}} \tau(s_i|f_i)p(f_i)} \quad (2)$$

$\mu(f_i|s_i)$ shows the relative probability of f_i happening by observing s_i. For example, in Fig. 1, assuming all faults have the same prior probability, then $\mu(f_1|s_1) = 0.9/(0.9+0.3) = 0.75$ and $\mu(f_2|s_1) = 0.3/(0.9+0.3) = 0.25$. The following contribution function $C(f_i)$ evaluates all contribution factors $\mu(f_i|s_i)$, $s_i \in S_{O_i}$ with the observation S_{O_i}, and decides which f_i is the best candidate with maximum contribution value $C(f_i)$ to the currently not yet explained symptoms.

$$C(f_i) = \frac{\sum_{s_i \in S_{O_i}} \mu(f_i|s_i)}{\sum_{s_i \in S_{f_i}} \mu(f_i|s_i)} \quad (3)$$

Therefore, fault reasoning becomes a process of searching for the fault (f_i) with maximum $C(f_i)$. This process continues until all observed symptoms are explained. The contribution function $C(f_i)$ can be used for both the deterministic and probabilistic models.

In the deterministic model, the higher the number of symptoms observed, the stronger the indication that the corresponding fault has occurred. Meanwhile, we should not ignore the influence of prior fault probability $p(f_i)$, which represents long-term statistical observation. Since $p(s_i|f_j) = 0$ or 1 in the deterministic model, the normalized conditional probability reflects the influence of prior probability of fault f_i. Thus, the same contribution function can seamlessly combine the effect of $p(f_i)$ and the ratio of $\frac{|S_{O_i}|}{|S_{f_i}|}$ together.

The fault reasoning algorithm first finds the fault candidate set F_C, including all faults that can explain at least one symptom s_i ($s_i \in S_O$); then it calls the function $HU()$ to generate and update the hypothesis set Φ until all observed symptoms S_O can be explained. According to the contribution $C(f_i)$ of each fault f_i ($f_i \in F_C$), Algorithm 1 iteratively searches for the best explanation (i.e. the fault with the highest contribution) (lines 5-6) of S_K, which are currently observed symptoms not yet explained by the hypothesis h_i (lines 4-12). Here $S_K = S_O - \cup_{f_i \in h_i} S_{O_i}$ and initially $S_K = S_O$. If multiple faults have the same contribution, multiple hypotheses will be generated (lines 13-17). The searching process (HU) will recursively run until all observed symptoms (i.e., S_{K_i}) are explained (i.e., $S_{K_i} = \emptyset$) (lines 18-24). Notice that only those hypotheses with a minimum number of faults that cover all observed symptoms are included into Φ (lines 23-24).

Algorithm 1 Hypothesis Updating Algorithm **HU**(h, S_K, F_C)

Input: hypothesis h, observed but uncovered symptom set S_K, fault candidate set F_C
Output: fault hypothesis set Φ

```
1: $c_{max} = 0$
2: for all $f_i \in F_C$ do
3:   if $C(f_i) > c_{max}$ then
4:     $c_{max} \leftarrow C(f_i)$
5:     $F_S \leftarrow \emptyset$
6:     $F_S \leftarrow F_S \cup \{f_i\}$
7:   else
8:     if $C(f_i) = c_{max}$ then
9:       $F_S \leftarrow F_S \cup \{f_i\}$
10:    end if
11:  end if
12: end for
13: for all $f_i \in F_S$ do
14:   $h_i \leftarrow h \cup \{f_i\}$
15:   $S_{K_i} \leftarrow S_K - S_{O_i}$
16:   $F_{C_i} \leftarrow F_C - \{f_i\}$
17: end for
18: for all $S_{K_i}$ do
19:   if $S_{K_i} = \emptyset$ then
20:     $\Phi \leftarrow \Phi \cup \{h_i\}$
21:   end if
22: end for
23: if $\Phi \neq \emptyset$ then
24:   return < $\Phi$ >
25: else
26:   /* No $h_i$ can explain all $S_O$*/
27:   for all $h_i$ do
28:     $HU(h_i, S_{K_i}, F_{C_i})$
29:   end for
30: end if
```

The above Fault Reasoning algorithm can be applied to both deterministic and probabilistic models with the same contribution function $C(f_i)$ but different conditional probability $p(s_i|f_i)$.

B. Fidelity Evaluation of Fault Hypotheses

The fault hypotheses created by the Fault Reasoning algorithm may not accurately determine the root faults because of lost or spurious symptoms. The task of the Fidelity Evaluation is to measure the credibility of the hypothesis created in the reasoning phase given the corresponding observed symptoms. Objectively evaluating the reasoning result is crucial in fault localization systems.

We use the fidelity function $FD(h)$ to measure the credibility of hypothesis h given the symptom observation S_O. We assume that the occurrence of each fault is independent.

- For the deterministic model:

$$FD(h) = \frac{\sum_{f_i \in h} |S_{O_i}|/|S_{f_i}|}{|h|} \quad (4)$$

- For the probabilistic model:

$$FD(h) = \frac{\prod_{s_i \in \cup_{f_i \in h} S_{f_i}} (1 - \prod_{f_i \in h}(1 - p(s_i|f_i)))}{\prod_{s_i \in S_O} (1 - \prod_{f_i \in h}(1 - p(s_i|f_i)))} \quad (5)$$

Obviously in the deterministic model, if the hypothesis h is correct, $FD(h)$ must be equal to 1 because the corresponding symptoms can be either observed or verified. In the probabilistic model, if related symptoms are observed or verified,

$FD(h)$ of a credible hypothesis can still be less than 1 because some symptoms may not happen even when the hypotheses are correct. In either case, our fidelity algorithm takes into consideration a target Fidelity Threshold, FD_{TH}, that the user can configure to accept the hypothesis. System administrators can define the threshold based on long-term observation and previous experience. If the threshold is set too high, even correct hypothesis will be ignored; but if the threshold is too low, then a less credible hypothesis might be selected.

The fidelity evaluation function is used to evaluate each hypothesis and decide if the result is satisfactory by comparing them to the pre-defined threshold value. If an acceptable hypothesis that matches the fidelity threshold exists, the fault localization process can terminate. Otherwise, the best available hypothesis and a non-empty set of symptoms (S_N) would be verified in order to reach a satisfactory hypothesis in the next iteration.

Algorithm 2 Active Integrated Fault Reasoning S_O

```
Input: $S_O$
Output: fault hypothesis $h$
1: $S_N \leftarrow S_O$
2: while $S_N \neq \emptyset$ do
3:     $\Phi = FR(S_O)$
4:     $< h, S_N >= FE(\Phi)$
5:     if $S_N = \emptyset$ then
6:         return $< h >$
7:     else
8:         if IPP expired then
9:             /*used to schedule active fault localization periodically*/
10:            $< S_V, S_U >= AS(S_N)$
11:        end if
12:    end if
13:    $S_O \leftarrow S_O \cup S_V$
14:    $< h, S_N >= FE(\{h\})$
15:    if $S_N = \emptyset \parallel S_V = \emptyset$ then
16:        return $< h >$
17:    end if
18: end while
```

C. Action Selection Heuristic Algorithm

The main reason to verify the existence of symptoms rather than faults is that symptoms are noticeable/visible consequences of faults and thus they are easier to track and verify. The task of Action Selection is to find the least-cost actions to verify S_N (unobserved symptoms) of the hypothesis that has the highest fidelity. As the size of S_N grows very large, the process of selecting the minimal cost action that verifies S_N becomes non-trivial. The Action-Symptoms correlation graph can be represented as a 3-tuple (A, S, E) graph such that A and S are two independent vertex sets representing Actions and Symptoms respectively, and every edge e in E connects a vertex $a_j \in A$ with a vertex $s_i \in S$ with a corresponding cost (t_{ij}) to denote that a_j can verify s_i with cost $t_{ij} = t(s_i, a_j) > 0$. If there is no association between s_i and a_j, then $t_{ij} = 0$. Because a set of actions might be required to verify one symptom, we use a composite action vertex, v_j, to represent this case. The composite action vertex v_j is used to associate a set of conjunctive actions to the corresponding symptom(s). However, if multiple actions are directly connected to a symptom, then this means any of these actions can be used disjunctively to verify this symptom. To represent the relationship between the composite actions and the corresponding symptoms using a bipartite graph, we (1) set the cost of v_j, $t(s_i, v_j)$, to the total cost of the conjunctive action set; (2) then eliminate the associated conjunctive set to the v_j; and (3) associate v_j with all symptoms that can be verified by any action in the conjunctive action set.

The goal of the Action Selection algorithm is to select the actions that cover all symptoms S_N with a minimal action cost. With the representation of the Symptom-Action bipartite graph, we can model this problem as a weighted set-covering problem. Thus, the Action Selection algorithm searches for A_i such that A_i includes the set of actions that cover all the symptoms in the Symptoms-Action correlation graph with total minimum cost. We can formally define A_i as the covering set that satisfies the following conditions: (1) $\forall s_i \in S$, $\exists a_j \in A_i$ s.t. $t_{ij} > 0$, and (2) $\sum_{a_i \in A_i, s_j \in S_N} t_{ij}$ is the *minimum*.

The weighted set-covering is an NP-complete problem. Thus, we developed a heuristic greedy set-covering approximation algorithm to solve this problem. The main idea of the Algorithm is simply to first select the action (a_i or v_i) that has the maximum *relative covering ratio*, $R_i = \frac{|S_{a_i}|}{\sum_{s_j \in S_{a_i}} t_{ij}}$, where this action is added to the final set A_f and removed from the candidate set A_c that includes all actions. Here, S_{a_i} is the set of symptoms that action a_i can verify, $S_{a_i} \subseteq S_N$. Then, we remove all symptoms that are covered by this selected action from the unobserved symptom set S_N. This search continues to find the next action a_i ($a_i \in A_c$) that has the maximum ratio R_i until all symptoms are covered (i.e., S_N is empty). Thus, intuitively, this algorithm appreciates actions that have more symptom correlation or aggregation. If multiple actions have the same relative covering ratio, the action with more covered symptoms (i.e., larger $|S_{a_i}|$ size) will be selected. If multiple actions have the same ratio, R_i, and same $|S_{a_i}|$, then each action is considered independently to compute the final selected sets for each action and the set that has the minimum cost is selected. Finally, it is important to notice that each single action in the A_f set is necessary for the fault-determination process, because each one covers unique symptoms.

D. Algorithm for Active Integrated Fault Reasoning

The main contribution of this work is to incorporate active actions into fault reasoning. Passive fault reasoning could work well if enough symptoms can be observed correctly. However, in most cases, we need to deal with interference from symptom loss and spurious symptoms, which could mislead fault localization analysis. As a result of fault reasoning, the generated hypothesis suggests a set of selected symptoms S_N that are unobserved but expected to happen based on the highest fidelity hypothesis. If fidelity evaluation of such hypothesis is not acceptable, optimal actions are selected to verify S_N. Action results will either increase fidelity evaluation of the previous hypothesis or bring new evidence to generate a new hypothesis. By taking actions selectively, the system can evaluate fault hypotheses progressively and reach to root faults.

Algorithm 2 illustrates the complete process of the *AIR* technique. Initially, the system takes observed symptom S_O as input. Fault Reasoning is used to search the best hypothesis Φ (Line 3). Fidelity is the key to associate passive reasoning to active investigation actions. Fidelity Evaluation is used to measure the correctness of corresponding hypothesis h ($h \in \Phi$) and produce expected missing symptoms S_N (Line 3). If the result h is satisfied, the process terminates with the current hypothesis as output (Line 5 - 6). Otherwise, *AIR* waits until the Initial Passive Period (IPP) has expired (Line 8) to initiate actions to collect more evidence of verified symptoms S_V and not-occurred symptoms S_U (Line 10). New evidence will be added to re-evaluate the previous hypothesis (Line 13). If fidelity evaluation is still not satisfied, the new evidence from previous observation is used to search for another hypothesis (Line 3) until the fidelity evaluation is satisfied. At any point, if either the fidelity evaluation does not find symptoms to verify (S_N is $\emptyset$), or none of the verified symptom had occurred (S_V is $\emptyset$), the program will terminate and return the current selected hypothesis. In either case, this is an indication that the current selected hypothesis is credible.

V. Active Overlay Fault Localization

In contrast to an Internet Service Provider (ISP) that has full control over its own networks, An Overlay Service Provider (OSP) provides overlay services via a logical overlay network infrastructure, which is built on top of underlying networks controlled by different ISPs. As a service provider, an OSP can monitor its overlay networks by collecting end-to-end symptoms either passively from overlay users or via end-to-end monitoring actions. It is impractical for an OSP to associate all observed overlay symptoms with the underlying network components, and further, to encode the conditional probabilistic causality relationship between them. However, please note that from an overlay application standpoint, there are two types of faults: overlay faults (F^o) and underlay faults (F^u). In this paper, we call the overlay nodes related problems overlay faults and the underlying network related problems underlay faults. An overlay application usually needs to take different countermeasures to tackle the above different types of faults. For example, if an overlay multicast application detects that the cause of performance degradation is due to an overloaded overlay node, it can simply replace that faulty overlay node with another one in the same network; however, if the problem is due to the underlying network components (e.g., malfunctioned network routers), the overlay application has to choose another overlay node from different networks so that it can reroute the application traffic to avoid faulty underlying network components. Thus, one important design objective of the overlay fault diagnosis system is to effectively distinguish overlay faults from underlay faults.

Considering the above objective as well as the challenges and requirements in overlay fault diagnosis discussed in Section III, we believe an overlay fault diagnosis system has to be designed accordingly to aim at the following objectives. It should:

- diagnose faults across multiple layer abstraction and effectively distinguish overlay faults from underlay faults;
- dynamically create Symptom-Fault correlation;
- deal with insufficient user observations.

TABLE II
Overlay Active Integrated Fault Reasoning Notation

Notation	Definition
L	a set of overlay links
l^o_{ij}	an overlay link between overlay node i and j. $l^o_{ij} \in L$
P^o	a set of overlay path
N_{ij}	the underlying network between the overlay nodes i and j
O	a set of overlay path symptoms $\{o\}$,
o_{ij}	an overlay symptom between the overlay nodes i and j
S	a set of overlay link symptoms
A	a set of overlay actions

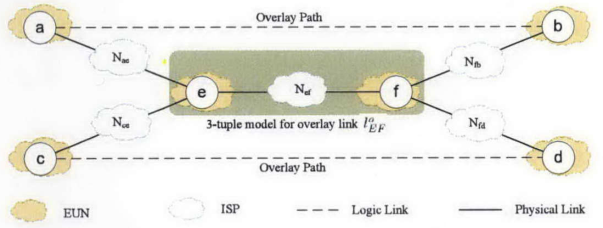

Fig. 3. Overlay Multi-layer Illustration and 3-tuple Model

In the following, we first propose an approach to construct an Overlay Network Profile (*ONP*) to facilitate overlay fault diagnosis. Then, we introduce how to dynamically build an Overlay Symptom-Fault-Action (*O-SFA*) model. Finally, we discuss the process of Overlay Active Integrated fault Reasoning (*O-AIR*). We list in Table II the relevant notations used in O-AIR discussion.

A. Overlay Network Profile

There are two types of networks or Autonomous Systems (ASes) in the Internet: Internet Service Provider (ISP) networks and End-User Networks (EUN). An overlay network consists of a set of overlay nodes (N^o) and a set of logical overlay paths (P^o). Each overlay path p^o_i ($p^o_i \in P^o$) consists of one or multiple overlay links l^o_i ($l^o_i \in L^o$, L^o is the set of overlay links). Each overlay link (l^o_i) is a direct network path between a pair of overlay nodes. An overlay link, the basic element in overlay networks, involves at least three components (source and destination overlay nodes, the underlying network between them), and can be abstracted as a *3-tuple model*. For example, in Fig.3, there are two overlay paths: p^o_{ab} between overlay nodes (a, b); and p^o_{cd} between overlay nodes (c, d). The overlay path p^o_{ab} consists of three overlay links as follows: l^o_{ae} between overlay nodes (a, e); l^o_{ef} between (e, f); and l^o_{fb} between (f, b). Each overlay link (e.g., l^o_{ae}) can be represented as a *3-tuple model* (a, N_{ae}, e). Here, a and e are overlay nodes and N_{ae} represents the logical underlay network component between a and e. Please note, the granularity of the abstracted *3-tuple model* of an overlay link is adjustable and depends on the availability information from different ISPs. In this paper, we use *3-tuple model* and assume no collaboration from ISPs.

In this overlay fault diagnosis framework (*O-AIR*), we assume that every overlay node can have multiple slices (e.g., Planet-Lab [2]) for hosting multiple overlay applications simultaneously. One of the slices on each overlay node is called the administration slice and is controlled by the Overlay Network Operation Center (*OvNOC*). Multiple overlay applications can run independently and simultaneously. Each overlay

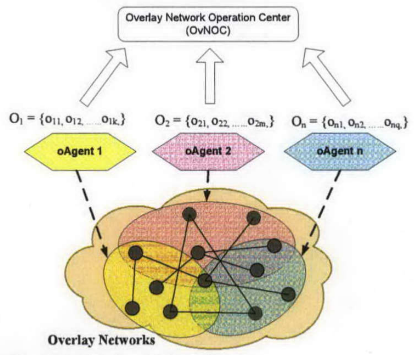
Overlay Network Operation Center (OvNOC)
O1 = {o11, o12, ..., o1k}
O2 = {o21, o22, ..., o2m}
On = {on1, on2, ..., onq}
oAgent 1
oAgent 2
oAgent n
Overlay Networks

Fig. 4. Overlay Monitoring Infrastructure

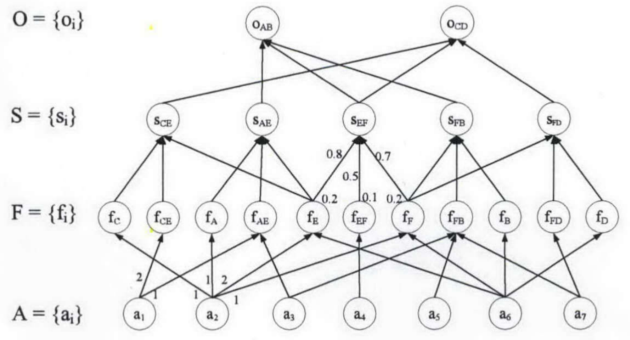

Fig. 5. Overlay Symptom-Fault-Action Model

application may have a dedicated monitoring agent (denoted as *oAgent*) to monitor various problems (e.g., unreachability or performance degradation) and send the corresponding end-to-end overlay path symptom (denoted as o_i) with the information of corresponding overlay links as a notification to OvNOC (Fig.4). We assume each *oAgent* has the knowledge of its overlay application topology. For example, as shown in Fig.3, if the *oAgent* observes an abnormal network behavior between overlay nodes a and b, it sends $o_{ab} = (s_{ae}, s_{ef}, s_{fb})$ to *OvNOC*. Here, s_{ij} is an overlay link symptom related to l^o_{ij}. If s_{ij} is negative, it means that at least one component in the corresponding *3-tuple model* (i, j, N_{ij}) is faulty. Understanding the overlay profile is challenging; However, we believe that an overlay service provider can easily obtain statistical information for overlay node faults and observe end-to-end (i.e., overlay link) network symptoms. If we monitor the given overlay link (e.g., l_{ae}) for a long enough period, we could obtain the prior fault probabilities of each component of *3-tuple model*: $p(f_a), p(f_{ae}), p(f_e)$; and the corresponding conditional probabilities: $p(s_{ae}|f_a), p(s_{ae}|f_{ae}), p(s_{ae}|f_e)$. Such information could be aggregated into a central or distributed overlay knowledge system such that a query regarding the prior/conditional fault probabilities for a specific *3-tuple model* could be answered. We call such an overlay knowledge system *Overlay Network Profile* (*ONP*).

B. Dynamic Overlay Symptom-Fault-Action Model

As shown in Fig. 4, the task of the *OvNOC* is to correlate received end-to-end symptoms ($\{o_i\}$) and identify the root causes ($\{f_i\}$). However, because the observed symptoms may not necessarily be sufficient to identify the existing network problem, the *OvNOC* may have to conduct a set of monitoring actions to determine the problems real-time with minimum cost. Overlay actions (also denoted as A) are used to verify the existence of the corresponding faults.

The observed symptoms could be positive or negative. Positive symptoms can be used to quickly narrow down the search space of root faults by removing all components explained by positive symptoms. Thus, in the following, we only assume all received symptoms are negative ones.

Based on received overlay symptoms (o_i) and *ONP*, we can dynamically construct an Overlay Symptom-Fault-Action (*O-SFA*) model (e.g., Fig.5) as the following:

- For every observed overlay path (p_i) related symptom, associate a vertex o_i ($o_i \in O$);
- For every overlay link l_i ($l_i \in p_i$) related symptom, associate a vertex s_i ($s_i \in S$);
- For each component in the *3-tuple model* of each involved overlay link, associate a vertex f_i ($f_i \in F$);
- For every overlay action, associate an action vertex a_i of weight 1, $a_i \in A$;
- For every component in F, associate an edge to its corresponding overlay link symptom vertex with a weight equal to $P(s_i|f_j)$, which is obtained from *ONP*;
- For every overlay link symptom vertex s_i, associate an edge to all relevant overlay path symptom vertex o_j that contains s_i
- For every overlay action in A, associate an edge to its corresponding component in F with a weight equal to the administrative action cost, which is specified by *OvNOC* administrators.

Please note, in an overlay network, all the potential faults being considered are directly observable via some actions (i.e., the up/down state of every overlay node and network path between them could be independently and directly measured). O-AIR uses passively obtained symptoms to determine which potential faults to test first, as a cost optimization. Therefore, the overlay fault diagnosis task can be further defined as the following two sub-tasks: (1) Given a set of received overlay end-to-end symptoms $O = \{o_i\}$, find a fault hypothesis h that is comprised of a set of faults ($h = \{f_i\}$) that can best explain overlay symptoms O; (2) If h can not be verified after taking verification actions, find a set of actions with the least cost that can lead to find the root faults.

The overlay fault diagnosis (*O-AIR*) process includes three functional modules: Symptom Mining (SM), Fault Reasoning (FR), and Action Selection (AS). The Symptom Mining module uses observed overlay symptoms (O) to dynamically create *O-SFA* based on *ONP*. The Fault Reasoning module takes *O-SFA* as input and returns a fault hypothesis h as output. The fault hypothesis h contains a set of faulty components that explains all observed symptoms so far. The corresponding overlay actions are selected to verify the hypothesis. If all faults in h are verifiable, the overlay fault diagnosis process can terminate. Otherwise, the action results will be used to update previously constructed *O-SFA* by removing explained overlay symptoms, as well as irrelevant components, and adding new symptoms and related components. This process is repeated until a verifiable hypothesis is found.

C. Overlay Fault Localization

In the following, because of the similarity with the corresponding nodules in *AIR* discussed in Section IV, we will introduce a new overlay contribution function used in the Fault Reasoning (*FR*) module, and briefly discuss Symptom the Mining (*SM*) and the Action Selection (*AS*) Modules.

1) Overlay Symptom Mining: The overlay Symptom-Fault-Action model (*O-SFA*) needs to be dynamically created and updated. There are two basic functions conducted in the overlay Symptom Mining (SM) module: (1) to collect overlay symptoms (O) and query *ONP*; (2) to construct/update *O-SFA*. The input of SM can be from different overlay application monitoring agents (*oAgent*), or from the return results of active investigation actions.

2) Overlay Fault Reasoning: Given *O-SFA*, the next task is to find the most likely root causes from F, which can best explain all observed overlay path symptoms O. In the Overlay Fault Reasoning module, we use a *Overlay Contribution Function,* $C^o(f_i)$, as a criterion to find faults that have the maximum indications of the observed symptoms. In *O-SFA*, overlay link symptom s_i is introduced because of the observation of O. s_i can be caused by at least one component f_i, ($f_i \in F_{s_i}$) in the *3-tuple model* with different possibilities $p(s_i|f_i) \in (0,1]$. We assume that the Symptom-Fault correlation model in *O-SFA* is sufficient enough to neglect other undocumented faults (i.e., prior fault probability is very low). Thus, we can also assume that symptom s_i will not occur if no component in *3-tuple model* F_{s_i} is faulty.

However, conditional probability $p(s_i|f_i)$ itself may not truly reflect the chance of fault f_i occurrence by observing symptom s_i. For example, in Fig. 5, by observing s_{ef}, in order to measure the indication for each fault (f_e, f_{ef}, f_f) to the creation of s_{ef}, we normalize the conditional probability $p(s_i|f_i)$ to the normalized conditional probability $\tau(s_i|f_i)$, the same as Eq. 1, in order to reflect the relative indication of each fault f_i to s_i.

Similarly, we use Eq. 2 to calculate $\mu(f_i|s_i)$, the relative probability of f_i happening by observing s_i. For example, in Fig. 5, assuming overlay components (i.e., overlay node E and F) have much higher prior probability (e.g., 20%) than underlay components (e.g., N_{ef} with prior fault probability 10%), then $\mu(f_e|s_{ef}) = 0.8*0.2/(0.8*0.2+0.5*0.1+0.7*0.2) = 0.45$, $\mu(f_{ef}|s_{ef}) = 0.14$, and $\mu(f_f|s_{ef}) = 0.41$.

Thus, we can measure the prior fault probability of each related overlay link (represented as s_i) given an overlay path symptom o_i ($s_i \in o_i$) as the following:

$$\sigma(s_i|o_i) = \frac{\sum_{f_i \in F_{s_i}} \tau(s_i|f_i)p(f_i)}{\sum_{s_i \in S_{o_i}} \sum_{f_i \in F_{s_i}} \tau(s_i|f_i)p(f_i)} \qquad (6)$$

In *O-AIR*, we developed the following *Overlay Contribution Function* $C^o(f_i)$ to evaluate all contribution factors $\mu(f_i|s_i)$ ($s_i \in S_{f_i}$) and decide which f_i is the best candidate with maximum indication value $C^o(f_i)$ to the currently not yet explained overlay path symptoms. Here, O_{f_i} is the set of overlay symptoms that could be caused by f_i.

$$C^o(f_i) = \sum_{s_i \in S_{f_i}, o_i \in O_{f_i}} \mu(f_i|s_i)\sigma(s_i|o_i) \qquad (7)$$

Therefore, overlay fault reasoning becomes a process of searching for the fault (f_i) with maximum $C^o(f_i)$ such that it can be included in the corresponding fault hypothesis h. This process continues until all observed symptoms are explained. We omitted the details of the overlay fault reasoning algorithm, which is similar to Algorithm 1. However, in the overlay fault localization process, it is impossible to evaluate the fidelity of the hypothesis because of incomplete symptom observation. Instead, the corresponding verification actions need to be taken in order to verify the correctness of the hypothesis.

3) Overlay Action Cost Estimation: There are different ways to choose the action cost. We here provide an intuitive idea on how to estimate the cost in an overlay environment. The action cost is a function of "benefit" minus the "overhead". The benefit of an action is determined by the criticality and impact of the associated fault. The criticality reflects the severity of this fault on business value and SLA which could be estimated from previous history or business policies. The fault impact is used to estimate the magnitude of the damage caused by this fault (e.g., like number of users/customers has been effected by this fault). The overhead of an action is estimated in term of bandwidth, delay (number of overlay component and underlay links), and labor cost. These factors can be converted to real cost (e.g., dollars) based on business practices and policies and then aggregated to calculate the total cost of an action. In our approach, the total cost is assumed as a weighted sum of number of normalized cost factors as explained. We also assume that the cost factors and weights can be provided by the network administrators, as the action-symptom association.

4) Overlay Action Selection: Depending on the verified overlay faults (e.g. end-to-end packet loss), the corresponding overlay actions can be selected using simple utilities such as *ping* and *traceroute*, user scripts of individual tools such as *sting, sprobe* and *IPerf*, or customized existing measurement framework such as *ScriptRoute*. Once an overlay fault hypothesis is generated, the corresponding overlay actions can be remotely invoked from a management workstation to verify the corresponding monitored overlay faults. As multiple tools (actions) can be used to verify the same type of overlay fault but with different cost (e.g. running time, intrusiveness, etc.), we developed an Action Selection module for O-AIR similar to the one described in Section IV-C to search for a set of overlay actions with minimal administrative action cost.

VI. System Evaluation

In this section, we describe system evaluation in Action Integrated fault Reasoning (*AIR*) and Overlay Action Integrated fault Reasoning (*O-AIR*) frameworks via extensive simulation studies and Internet experiments on the Planet-lab [2]. The evaluation study considers fault detection time (T) as the performance parameter and the detection rate (α) and false positive rate (β) as the accuracy parameters.

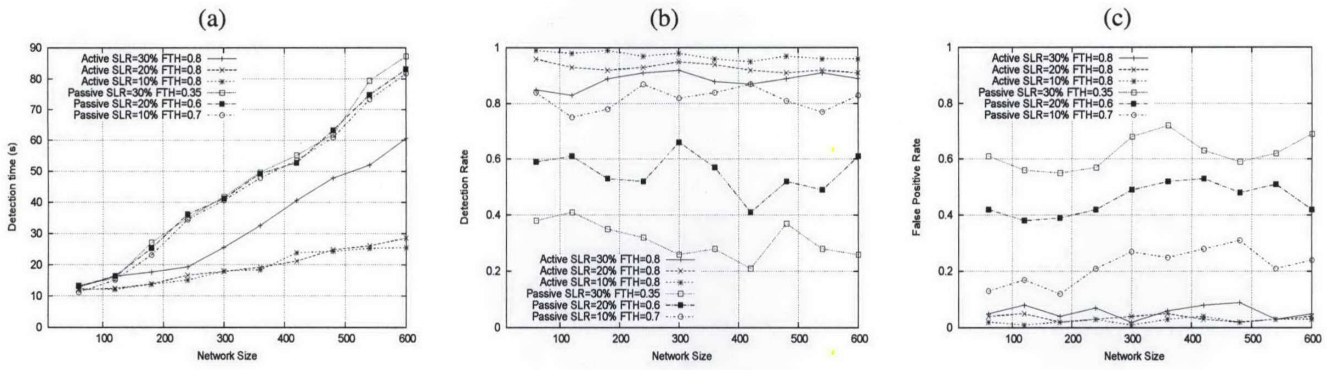

Fig. 6. The Impact of Symptom Loss Ratio in AIR (a) Detection time T (b) Detection rate α (c) False positive rate β.

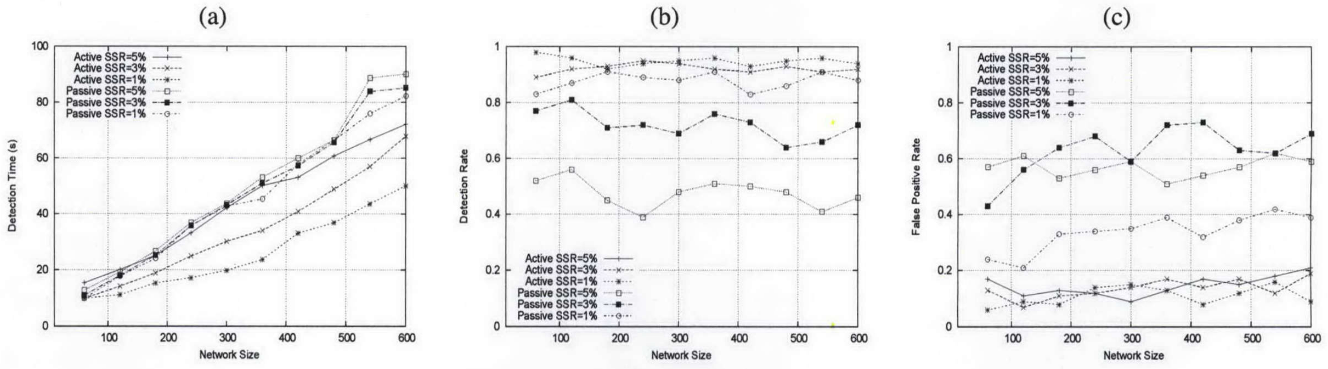

Fig. 7. The Impact of Spurious Symptoms in AIR (a) Detection time T (b) Detection rate α (c) False positive rate β.

A. Evaluation on AIR via Simulations

In our simulation, the number of network objects varies between $[60, 600]$. Each network object generates different faults, and each fault is associated with $[2, 5]$ symptoms. Thus, the total symptoms vary from 120 to 3,000. We use fault cardinality (FC), symptom cardinality (SC) and action cardinality (AC) to describe the Symptom-Fault-Action matrix, such that FC defines the maximal number of symptoms that can be associated with one fault; SC defines the maximal number of faults to which one symptom might correlate; and AC defines the maximal number of symptoms that one action can verify. We set $p(f_i)$ and $p(s_i|f_j)$ in ranges [0.001, 0.01] and (0, 1], respectively. Our simulation model also considers the following parameters: Initial Passive Period (IPP); Symptom Active Collecting Rate ($SACR$); Symptom Passive Collecting Rate ($SPCR$); Symptom Loss Ratio (SLR); Spurious Symptom Ratio (SSR); and Fidelity Threshold FD_{TH}.

The major contribution of this work is to offer an efficient fault reasoning technique that provides accurate results even in the worst cases (e.g., SLR and SSR are high). We show how these factors affect the performance (T) and accuracy (α and β) of our approach and a passive fault reasoning approach.

1) The Impact of the Symptom Loss Ratio: Symptom loss hides fault indications, which negatively affects both the accuracy and performance of the fault localization process. In this simulation, we set $SSR = 0$; $IPP = 10sec$; $SACR = SPCR = 100$ symptoms/sec; and vary SLR from 10% to 30%. With the increase of SLR, the passive fault reasoning system may become infeasible. In this case, we have to reduce the fidelity threshold so that the passive reasoning process can converge in a reasonable time. In Fig. 6(a), we can see that, in contrast to the passive approach, the *AIR* system always reaches a relatively high fidelity threshold with average performance improvement of 20% to 40%. In addition to the performance improvement, the *AIR* system also shows high accuracy. With the same settings, Fig. 6(b) and (c) show that the active approach gains 20-50% improvement of the detection rate and 20-60% improvement of the false positive rate, even with much higher fidelity criteria over the passive reasoning approach.

2) The Impact of Spurious Symptoms: The spurious symptoms are also regarded as the observation noise, which could seriously affect fault reasoning because they provide misleading information rather than losing information. In this simulation, we set $SLR = 0$; $IPP = 10s$; $SACR = 100$ symptoms/sec. The relative signal-noise ratio can be calculated as $SNR = \frac{1-SSR}{SSR}$ if $SLR = 0$. Fig. 7(a) shows that on average *AIR* shows 10-20% better performance than the passive approach, even with high fidelity value. With the same experiment settings, in Fig. 7(b) and (c), *AIR* shows accuracy improvement of 10-50% for the detection rate and 10-40% for the false positive rate over the passive approach.

3) The Impact of Network Size: In this section, we examine the scalability of *AIR* when the network size and the number of symptoms significantly increase. To show this, we measure *AIR* detection time under different scenarios: (1) without symptom loss and spurious symptom (Fig. 8(a)); (2) with symptom loss only (Fig. 8(b)), and (3) with spurious symptoms only (Fig. 8(c)). In all three cases, when the network size increases 10 times (from 100% to 1000%), the detection

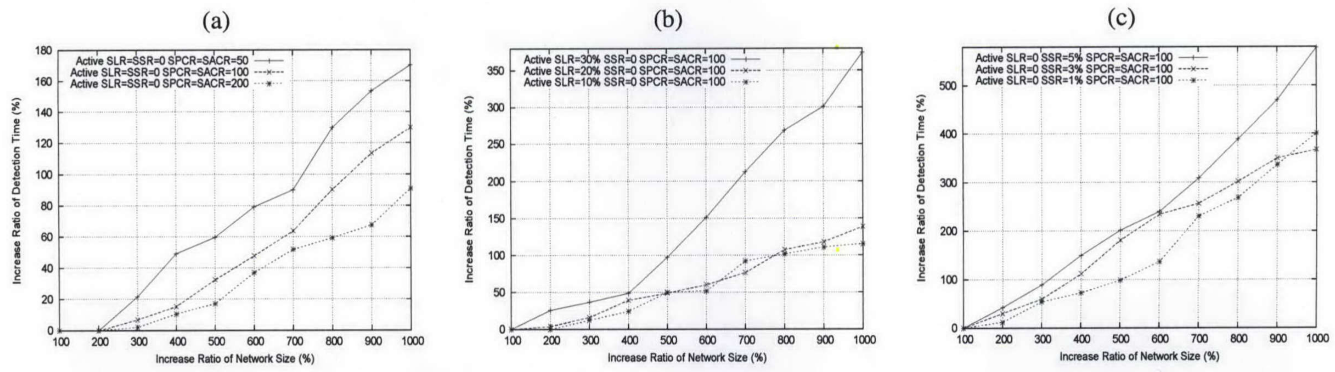

Fig. 8. The Impact of Network Size in AIR (a) Without symptom loss and spurious symptoms (b) With symptom loss (c) With spurious symptoms

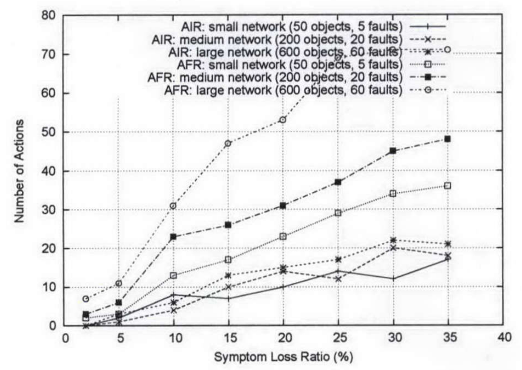

Fig. 9. Intrusiveness Evaluation.

time slowly increases by 1.7 times (170%), 3.7 times (370%), and 5.8 times (580%) in Fig. 8(a), (b) and (c), respectively. This shows that even in the worst case scenario (Fig. 8(c)), the growth in network size causes a slow linear increase of *AIR* performance.

4) The Impact of Symptom Loss on AIR *Intrusiveness:* *AIR* intrusiveness is measured by the number of total actions performed to localize faults. As shown in Section IV, the intrusiveness of *AIR* is algorithmically minimized by (1) considering the fault hypothesis of high credibility, and (2) selecting the minimum cost actions based on the greedy algorithm described in Section IV-C. We conducted experiments to assess the intrusiveness (i.e., action cost) when the symptom loss ratio increases. In this simulation, we set Action Cardinality (AC) to 3. Apparently, the higher the AC, the more symptoms can be verified by taking a single action and thus the less the total required actions. For comparison, we also implemented an Active Fault Reasoning algorithm called AFR. AFR takes the observed symptoms (S_O) as the input to find all related faults (i.e., F_{S_O}). Then, AFR selects the actions to verify all possible symptoms that can be caused by F_{S_O}. Fig. 9 shows that, with a different scale of network sizes and the prior fault probability as high as 10%, the number of actions required for AIR increases slowly linearly (from 1 - 22) even when the symptom loss ratio significantly increases (from 2%-35%). However, compared to AIR, AFR may take up to 200% more actions to find the satisfactory fault reasoning result. For example, in a large-scale network of 600 objects, the number of actions performed by AIR did not exceed the 0.37 action/fault ratio, but exceeded the 1.19 action/facult ratio in AFR.

B. Evaluation of O-AIR *via Simulations*

In the following, we describe our simulations of *O-AIR* framework. *O-AIR* can be applied to locate various network faults, such as delay, loss, and jitter. In our simulations, we assume the same type of overlay symptoms are collected. We consider the following dimensions and parameters for simulations and experiments.

- *Underlying Network Topology and Size:* We use a synthetic topology generator BRITE [19] with three types of topology models. In addition, we import to BRITE using real network AS topology data from Skitter [20] with each set more than 20,000 ASes for the evaluation.
- *Overlay Network Topology:* For given N overlay nodes, we create $\lceil 10\% N \rceil$ number of overlay applications. Each overlay application constructs a tree with 10 overlay paths. The number of overlay links in each overlay path is uniformly distributed between $[2, 6]$.
- *User Observation Ratio:* For each overlay application, it can decide how many critical overlay paths are monitored.
- *Fault Ratio:* The prior fault probability of the overlay nodes and underlay networks are uniformly distributed in $[0.01, 0.2]$ and $[0.001, 0.1]$. Here we assume the underlay nodes (e.g., network routers) are more reliable than the overlay nodes (e.g., user workstations).

1) The Impact of Network Topology and Size: We use BRITE [19] to create four types of topologies with different network sizes from 100 to 30,000 network objects: (1) AS-level Waxman model; (2) AS-level Barabasi-Albert model; (3) Hierarchical model; (4) Skitter [20]. As shown in Fig.10, when the network size increased 300 times, the corresponding detection time increased only 11 times (approximately from 5 to 55 seconds). For the detection rate and the false positive rate, the change rate is within 10% of the increase in network size.

2) The Impact of Overlay Symptom Loss: The identification and explanation of power laws has become an increasingly dominant theme in the recent body of network topology research [19]. In the following simulation, we use only BRITE

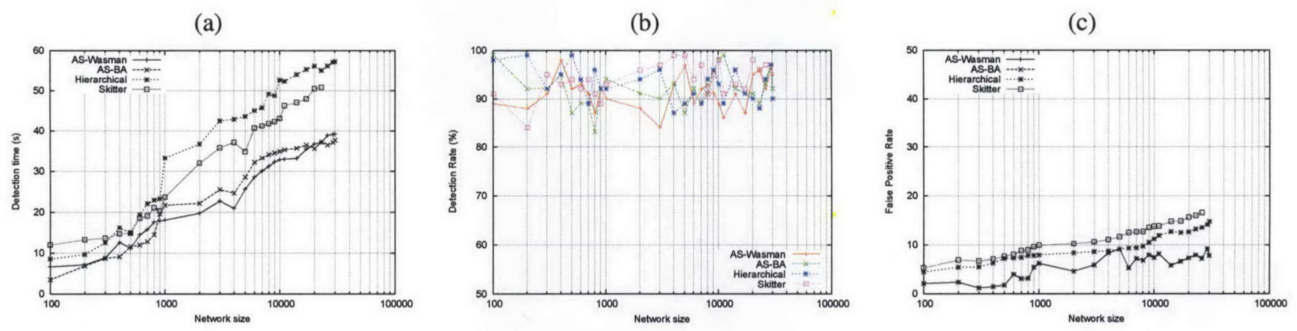

Fig. 10. The Impact of Overlay Network Topology and Size (a) Detection time T (b) Detection rate α (c) False positive rate β

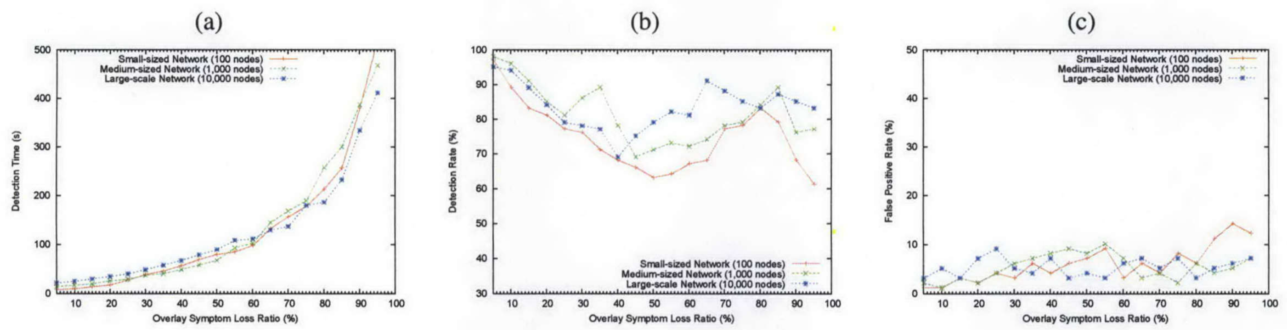

Fig. 11. The Impact of Overlay Symptom Loss Ratio (a) Detection time T (b) Detection rate α (c) False positive rate β

Hierarchical Topology with the ASBarabasiAlbert and Router-Waxman model, which can properly represent power law node distribution. In this simulation, we simulate three different scenarios: (1) small-size network (100 nodes); (2) medium-size network (1,000 nodes); (3) large-scale network (10,000 nodes). For all generated network faults, we choose an Overlay Symptom Loss Ratio increased from 10% to 100% with a increase of 10%. Obviously, with the increase of the Overlay Symptom Loss Ratio, the detection time is significantly increased (can be 50 times higher) as shown in Fig. 11. One cause is insufficient symptom observation, thus we need take more actions to collect the relevant information. Also, the incomplete observation can result in an uncredible hypothesis, which requires further actions to enhance the reasoning result. However, the detection rate is relatively stable, even decreased with the increase in the symptom loss ratio. The false positive ratio is also effectively controlled because of the integration of the active actions.

TABLE III
OVERLAY NETWORK DISTRIBUTION

	Overlay Service Providers	
	Planet-Lab	Akamai
Nodes	~800	~25,000
Networks	~400	~1,000
Countries	~30	~70
Applications	CoDeeN, CoMon	Yahoo, Facebook

TABLE IV
EXPERIMENTAL OVERLAY TOPOLOGY

Planet-Lab Node Dist	OSP Infrastructure
N. America (edu) (346)	40
N. America (non-edu) (28)	10
S. America (18)	8
Europe (230)	20
Asia (138)	20
Oceania (9)	2

C. The Experiments on Planet-Lab

We experimented with the *O-AIR* system on the Planet-Lab testbed. In the following, we elaborate the major steps.

1) Creating an OSP: The Planet-Lab nodes are mainly distributed in five geographic zones: North America, South America, Europe, Asia and Oceania zones. North America nodes can be further classified as EDU and non-EDU nodes. Thus, the Planet-Lab nodes can be classified into six categories as shown in Table IV. We create an OSP using the following parameters:

- the total number of overlay nodes and their distribution: Based on the industrial OSP infrastructure and the available resource from the Planet-Lab as shown in Table III, we select 100 well-distributed Planet-Lab nodes, each from a different organization/network to construct an experimental OSP as shown in Table IV.
- the total number of overlay links: Technically, for 100 overlay nodes, we can construct $100 \times 99 = 9,900$ overlay links. (Here, we regard l_{ab}^{o} as different from l_{ba}^{o}.) For simplicity without losing generality, in our experiments we randomly select $3,000$ overlay links.
- the total number of overlay paths: Each overlay path consists of a sequence of overlay links. From the above selected $3,000$ overlay links, we construct 500 overlay paths, and each overlay path consists of 3-6 acyclic overlay links.

2) Building the ONP: In this experiment, we use *Packet Loss* as the monitored object. We built the ONP in different periods of one, two and four weeks, respectively, and denoted the corresponding ONP as ONP_1, ONP_2 and ONP_4. When constructing the ONP, each overlay node sends a 40-byte UDP packet every 30s to the corresponding overlay nodes that are on the same selected overlay links. In order to monitor the status of the selected overlay nodes (e.g., when an overlay node is down), from the same network of each node, we choose another 100 overlay nodes as the mirrored overlay nodes to *ping* their peers and log the test results. In practice, there is potentially a scalability issue in constructing ONP if the number of overlay nodes and the corresponding overlay links is significantly increased. Clustering overlay nodes based on their network properties (e.g., the nodes on the same network or using the same ISP can be clustered together) could be an effective approach to overcoming this problem. We leave this as our future work. For all $3,000$ overlay links during the period of constructing ONP_1, ONP_2 and ONP_4, the overall link loss ratio is 1%, 4% and 7% respectively. Moreover, for all lossy links in each ONP, 37%, 33% and 29% lossy links were caused by overlay node failures. The ONP information is aggregated to a management workstation (MW) and updated periodically. In our experiment, constructing the ONP is a continuous process and the corresponding network measurement is conducted at a fixed rate (i.e., once every 30s). The monitoring results can be used to later verify the fault reasoning results. In practice, the network sampling period in constructing ONP can be significantly increased over time to reduce the network intrusiveness.

3) Monitoring Overlay Paths: An end-to-end overlay path is represented as a sequence of overlay links. In practice, the overlay paths are formed and can be tracked by the corresponding overlay applications. In our experiment, a source overlay routing table, which contains the corresponding sequence of overlay links for each overlay path, is assigned to every source overlay node of each constructed overlay path. A source node sends UDP testing packets to the next overlay node according to the source overlay routing table, until the last one along this overlay path. Once received the testing packets, the last overlay node sends an acknowledgement packet with the corresponding sequence number of each test packet directly to the source overlay node without following the reverse source overlay routing table. At the same time, a log is also saved and sent to the management workstation (MW) in order to later verify if a symptom is a false alarm due to the loss of the acknowledgment packets. If a source node cannot receive the expected acknowledgements before timeout, a symptom will be generated and sent to the management workstation (MW) with an NTP timestamp and the corresponding overlay path information (e.g., the sequence of overlay links).

4) Overlay Fault Reasoning via O-AIR: For all received symptoms within the same observation window (we use 15 minutes, which is a reasonable estimated average fault period [13]), we dynamically construct an O-SFA model based on an existing overlay network profile (i.e., ONP_1, ONP_2 or ONP_4). In this experiment, the overlay actions are designed, based on 3-tuple overlay model of each overlay link, to check the status of the source and destination overlay nodes and the link quality (i.e., the packet loss ratio recorded when constructing the corresponding ONP).

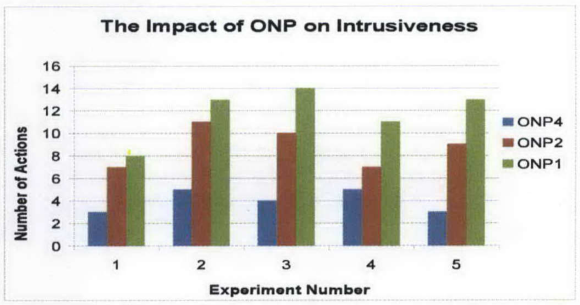

Fig. 12. *O-AIR* Intrusiveness Evaluation on Planet-Lab

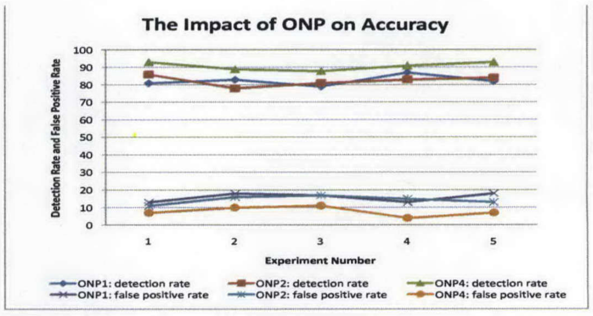

Fig. 13. *O-AIR* Accuracy Evaluation on Planet-Lab

5) Experiment Results: We consider an experiment result (also called an experiment snapshot) valid only if there are more than 100 overlay components (i.e., $|F_{S_O}| > 100$) related to the observed overlay symptoms within the same observation window (i.e., 15 minutes). Since the fault rate is relatively low in the real network, in our experiments we collected five snapshots in total from the same overlay experimental environment as described above. Firstly, we evaluate the intrusiveness (i.e. the total number of verification actions) of O-AIR with the different ONP. Fig.12 shows that *O-AIR* presented very promising results regarding its intrusiveness factor, with an average of four, eight and 11 verification actions when using ONP_4, ONP_2 and ONP_1, respectively. The detection time of five experiments based on different ONPs is distributed between five to 25 seconds, which is proportional to the number of required actions. Please note that by using a ONP built up within a shorter period (e.g., ONP_1), O-AIR may take much more actions because of lacking enough statistic information of the overlay network. For example, if an overlay link is first time reported with some observed symptoms (i.e., no related record can be found in the ONP), exhaustively taking all verification actions on every related 3-tuple component is inevitable. This could explain why using ONP_4 can reduce the required verification actions.

It is also interesting to note how an ONP impacts the accuracy of *O-AIR*. Accordingly, for each of the experiments discussed above, let O-AIR adopt ONP_1, ONP_2 and ONP_4, respectively, to analyze the root causes. As shown in Fig. 13, there are no obvious differences in the fault reasoning accuracy by using ONP_1 and ONP_2. However, the reasoning accuracy based on ONP_4 outperforms both ONP_1 and ONP_2. This result seems to show that the statistics based on the long-term network monitoring results can provide a more reliable reference to get a more accurate reasoning result.

VII. Conclusion and Future Work

In this paper, we present a novel technique called Active Integrated fault Reasoning or *AIR*. *AIR* is designed to minimize the intrusiveness of investigation actions (e.g., probings) via incrementally enhancing the fault hypothesis and optimizing the investigation action selection process. With the advent of the overlay service model, how to effectively managing overlay service networks becomes a significant challenge. In this paper, we also propose a novel end-to-end user-level overlay fault diagnosis framework (called *O-AIR*). *O-AIR* creates the correlation matrix dynamically as *O-SFA* and seamlessly integrates the passive and active fault reasonings. The results of our simulation and Internet experiments show that both *AIR* and *O-AIR* are efficient and accurate.

In our future work, we will investigate the automatic creation of the Symptom-Fault-Action correlation matrix from the network topology and high-level service specifications, especially for dynamic overlay networks.

References

[1] Akamai *Technology Overview*, http://www.akamai.com/html/technology.

[2] PlanetLab: http://www.planet-lab.org.

[3] J. L Hellerstein, K. Katircioglu, and M. Surendra, "A Framework for Applying Inventory Control to Capacity Management for Utility Computing," *IEEE/IFIP Integrated Management (IM'2005)*, May 2005.

[4] G. J. Lee and L. Poole, "Diagnosis of TCP Overlay Connection Failures using Bayesian Networks," *SIGCOMM'06 Workshops*, Sept. 2006, Pisa, Italy.

[5] S. Kliger, S. Yemini, Y. Yemini, D. Ohsie, and S. Stolfo, "A coding approach to event correlation," in *Proc. Fourth International Symposium on Intelligent Network Management*, 1995.

[6] E. Al-Shaer and Y. Tang, "QoS Path Monitoring for Multicast Networks," *Journal of Network and System Management (JNSM)*, 2002.

[7] Y. Tang, E. S. Al-Shaer and R. Boutaba, "Active Integrated Fault Localization in Communication Networks," *IEEE/IFIP Integrated Management (IM'2005)*, May 2005.

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

[9] M. Steinder and A. S. Sethi, "Probabilistic Fault Diagnosis in Communication Systems Through Incremental Hypothesis Updating," *Computer Networks,* vol. 45, no. 4, pp. 537-562, July 2004.

[10] I. Rish, M. Brodie, N. Odintsova, S. Ma, and G. Grabarnik, "Real-time Problem Determination in Distributed Systems using Active Probing," in *Proc. IEEE/IFIP (NOMS)*, Seoul, Korea, 2004.

[11] M. Brodie, I. Rish, and S. Ma, "Optimizing Probe Selection for Fault Localization," in *Proc. IEEE/IFIP (DSOM)*, 2001.

[12] K. Applety *et al.*, "Yemanja–a layered event correlation system for multi-domain computing utilities," *J. Network and Systems Management*, 2002.

[13] V. Paxson, "End-to-End Routing Behavior in the Internet," *IEEE/ACM Trans. Networking*, 1996.

[14] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein, *Introduction to Algorithms, Second Edition.* The MIT Press.

[15] J. Guo, G. Kar, and P. Kermani, "Approaches to Building Self Healing System using Dependency Analysis," in *Proc. IEEE/IFIP (NOMS)*, Seoul, Korea, 2004.

[16] J. Pearl, *Probabilistic Reasoning in Intelligent Systems.* Morgan Kaufmann Publishers, San Francisco, CA, 1988.

[17] G. Liu, A. K. Mok, and E. J. Yang, "Composite events for network event correlation," *Integrated Network Management VI*, pp. 247260, Boston, MA, May 1999.

[18] G. Jakobson and M. D. Weissman, "Alarm correlation," *IEEE Network*, pp. 5259, Nov. 1993.

[19] A. Medina, I. Matta, and J. Byers, "On the origin of power laws in Internet topologies," *ACM Computer Commun. Rev.*, Apr. 2000.

[20] Skitter, CAIDA's topology measurement tool, http://www.caida.org/tools/measurement/skitter/.

[21] Y. Zhao, Y. Chen, and D. Bindel, "Towards Unbiased End-to-End Network Diagnosis," in *Proc. ACM SIGCOMM*, 2006.

[22] M. Coates, A. Hero, R. Nowak, and B. Yu, "Internet Tomography," *IEEE Signal Processing Mag.*, vol. 19, no. 3, 2002.

[23] M. Zhang, C. Zhang, V. Pai, L. Peterson, and R. Wang, "PlanetSeer: Internet Path Failure Monitoring and Characterization in Wide-Area Services," in *Proc. Sixth Symposium on Operating Systems Design and Implementation*, Dec. 2004.

Yongning Tang has been an assistant professor in the School of Information Technology at Illinois State University since 2007. His primary research areas are network monitoring, fault management, policy-driven network management, and network security. Prof. Tang has many journal and conference publications in the area of distributed monitoring, fault localization, and multicast and overlay management.

Ehab Al-Shaer has been an Associate Professor in the School of CTI at DePaul University since 1998. He is Co-Editor of a number of books in the area of multimedia management and end-to-end monitoring. Prof. Al-Shaer has served as a Program Co-chair for a number of conferences in area including IM 2007, POLICY 2008, MMNS 2001, and E2EMON 2003-2005. He was a Guest Editor for a number of journals in his area. He has also served as a TPC member, session chair, and tutorial presenter for many IEEE, ACM, USENIX and IFIP conferences in his area including INFOCOM, ICNP and IM/NOMS and others. His research is funded by NSF, Cisco, Intel, and Sun Microsystems. He received his M.S. and Ph.D. in Computer Science from Northeastern University, Boston, MA and Old Dominion University, Norfolk, VA in 1994 and 1998 respectively.

Raouf Boutaba received the MSc. and PhD. Degrees in Computer Science from the University Pierre & Marie Curie, Paris, in 1990 and 1994 respectively. He is currently a Professor of Computer Science at the University of Waterloo. His research interests include network, resource and service management in wired and wireless networks. He is currently a distinguished lecturer of the IEEE Communications Society, the chairman of the IEEE Technical Committee on Information Infrastructure and the IEEE Technical Committee on Autonomic Communications. He has received several best paper awards and other recognitions such as the Premier's research excellence award.