RC22928 (W0310-011) October 3, 2003
Computer Science

IBM Research Report

Strategies for Problem Determination Using Probing

Mark Brodie, Irina Rish, Sheng Ma, Alina Beygelzimer, Natalia Odintsova
IBM Research Division
Thomas J. Watson Research Center
P.O. Box 704
Yorktown Heights, NY 10598

Strategies for Problem Determination using Probing

Mark Brodie, Irina Rish, Sheng Ma, Alina Beygelzimer, Natalia Odintsova
IBM T.J. Watson Research Center, 19 Skyline Drive, Hawthorne, NY 10532
mbrodie,rish,shengma@us.ibm.com,beygel@cs.rochester.edu,nodintsova@hotmail.com

Abstract— **As distributed systems continue to grow in size and complexity, scalable and cost-efficient techniques are needed for performing tasks such as problem determination and fault diagnosis. In this paper, we address these tasks using *probes*, or test transactions, which replace traditional "passive" event-correlation techniques with a more active, real-time information-gathering approach. We provide a theoretical foundation and a set of practical techniques for implementing efficient probing strategies - the main issue is minimizing the cost of probing while maximizing the diagnostic accuracy of the probe set. We show that finding an optimal probe set is NP-hard and devise polynomial-time approximation algorithms that demonstrate excellent empirical performance, even on large networks. We also implement an active, on-line probing strategy that yields a significant reduction in the probe set size.**

I. Introduction

Accurate diagnosis and prediction of unobserved states of a large, complex, multi-component system by making inferences based on the results of various tests and measurements is a common problem occurring in practice. Numerous examples include medical diagnosis, airplane failure isolation, systems management, error-correcting coding, and speech recognition. Achieving high diagnostic accuracy may require performing a large number of tests, which can be quite expensive. Thus, our goal is developing cost-efficient techniques for real-time diagnosis in complex distributed systems, so that high accuracy can be achieved with a small number of tests.

The key component of our approach is an "active" measurement approach, called *probing*, instead of more "passive" data-analysis techniques. A probe is a test transaction whose outcome depends on some of the system's components; accurate diagnosis can be achieved by appropriately selecting the probes and analyzing the probe outcomes. Our main contribution is in providing a theoretical foundation and a set of practical techniques for implementing efficient probing strategies.

Although our methods are quite generic and are applicable to a wide variety of problem areas, we will focus specifically on the area of distributed systems management. The rapid growth in size and complexity of distributed systems makes performance management tasks such as problem determination – detecting system problems and isolating their root causes – an increasingly important but also extremely difficult task. For example, in IP network management, we would like to quickly identify which router or link has a problem when a failure or performance degradation occurs in the network. In the e-Commerce context, our objective could be to trace the root-cause of unsuccessful or slow user transactions (e.g. purchase requests sent through a web server) in order to identify whether it is a network problem, a web or back-end database server problem, etc. Another example is real-time monitoring, diagnosis and prediction of the "health" of a large cluster system containing hundreds or thousands of workstations performing distributed computations (e.g., Linux clusters or GRID-computing systems).

Two general approaches are commonly used for problem determination. The first is *event correlation* ([1], [2], [3]), in which every managed device is instrumented to emit an alarm when its status changes. By correlating the received alarms a centralized manager is able to identify the problem. However, this approach usually requires heavy instrumentation, since each device needs to have the ability to send out the appropriate alarms. Also it may be difficult to ensure that alarms are sent out, e.g. by a device that is down. To avoid these problems, which arise from using a fixed, "passive" data-gathering approach, a more active *probing technology* has been developed, which allows one to "ask the right questions at the right time" in order to provide more accurate and cost-efficient problem determination.

A. Probing Technology

In the context of distributed systems management, a probe is a program that executes on a particular machine (called a probe station) by sending a command or transaction to a server or network element and measuring the response. The *ping* program is probably the most popular probing tool that can be used to detect network availability. Other probing tools, such as IBM's EPP technology ([4]), provide more sophisticated, application-level probes. For example, probes can be sent in the form of test e-mail messages, web-access requests, and so on. Generally a distributed system (as well as many other applications) can be represented by a logical "dependency graph", where nodes

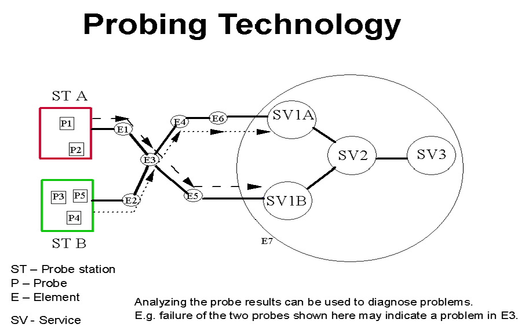

Probing Technology
ST A
P1
P2
E1
E4
E6
SV1A
SV2
SV3
E3
P3
P5
P4
E2
E5
SV1B
ST B
E7
ST – Probe station
P – Probe
E – Element
SV - Service
Analyzing the probe results can be used to diagnose problems.
E.g. failure of the two probes shown here may indicate a problem in E3.

Fig. 1. Probing Technology.

are either hardware elements (e.g., workstations, servers, routers, links) or software components and services, and links can represent both physical and logical connections between the elements.

Figure 1 shows a simple example of a distributed system with two probe stations, A and B, that send probes in some scheduled way. The probes test particular services, e.g., IP-connectivity, web-access to server SV1A, database access to server SV1B, which in turn depend on back-end database server SV2 and the availability of a particular database (“service” SV3). The probes go through particular network elements and problem diagnosis is performed by analyzing the results of different probes. For example, if two probes fail (e.g. the two shown in Figure 1), that may indicate a failure in some element that is common to both of them.

Probing technology has many advantages; it does not require extra instrumentation and works with any server that takes user transactions. It is very flexible; a probe station can be placed in any location with network access and can target multiple components. However using probes imposes costs, because of the additional network and server load and the need to collect, store and analyze probe results. It is important to control these costs in order to use probes effectively.

B. *Pre-planned and Active Probing*

In this work, we discuss using probing technology for the purpose of problem determination. We consider two basic approaches. In the first approach, a set of probes is selected and scheduled to run periodically. Problem determination is performed by analyzing the probe results. We call this approach **pre-planned** probing.

In the second approach, a set of probes is selected to run periodically as before, but only for the purpose of **detecting** when a problem occurs. Whenever occurrence a problem is detected by one or more of the probes, additional probes are sent out to obtain further information about the problem, and this process may repeat - as more data is obtained, decisions are made as to which probes to send next, until finally the problem is completely determined. We refer to this approach as **active probing**.

In both approaches, a critical problem is how to select probes. In the first approach, the set of probes must be preselected so that, no matter what problem occurs, the problem can be completely determined from the probe results without obtaining any additional information. In the second approach, we need to pre-select probes that can detect when problems occur, and in the active probing phase we need to determine which probe to send next, given what has been observed so far. To achieve cost-effective diagnosis, the size of the probe set should be minimized while providing wide coverage in order to locate or detect problems anywhere in the network.

C. *Contributions*

Probing technology is currently used for the purposes of quality of service measurement, and in practice probe selection is done using rather ad-hoc methodologies. This paper makes the following contributions:

1) We formulate the probe selection problem for problem determination in a general framework and show that both fault detection and fault localization are NP-hard.
2) We develop linear and polynomial-time approximation algorithms which utilize information-theoretic estimates of probe set quality, and show empirically that they find near-optimal probe sets.
3) We develop an algorithm for active probing and demonstrate the advantages of using it over pre-planned probing.

The outline of the paper is as follows. In Section II we introduce the basic concepts and in Section III we formulate the fault detection and localization problems precisely. Section IV shows that these problems are NP-hard and develops an information-theoretic measure of the quality of a probe set. Section V utilizes this measure to develop approximation algorithms which are practical for large networks and shows experimentally that they find near-optimal probe sets. Section VI investigates how to handle noise. Section VII considers the active probing scenario, where the selection of later probes depends on the results of earlier probes. Section VIII discusses related work and then we conclude.

II. Notation

Our formulation is a general one that applies to any situation in which tests, or probes, can be used to gain information about the state of objects that are connected together or are dependent on one other. Networks serve as a convenient illustrative example.

A. Faults

We assume there is a finite set O of objects each of which can be in one of two states, **"up"**, i.e. functioning correctly, or **"down"**, not functioning correctly. A **fault** can be any subset $f \subseteq O$. A fault f occurs if **all** the objects in f are down. If f is the empty set, then "occurrence" of the fault f corresponds to no problem - all objects are functioning correctly.

As an example, suppose we wish to detect problems in a network. A fault may be that a particular node or link (or combination of nodes and links) is down. For example, one fault may be $\{$node 1$\}$; this fault occurs if node 1 is down (whether or not any other node is up or down). Another fault might be $\{$node 1, node 2$\}$; this fault occurs if node 1 and node 2 are both down. Thus specification of the faults depends on many details of the situation and what problems we are interested in detecting.

B. Probes

A probe is a method of obtaining information about objects in O. We think of a probe as testing objects to determine whether or not they are up or down. Thus we regard a **probe** also as a subset $p \subseteq O$.

The occurrence of a fault affects some probes, while other probes remain unaffected. A probe p is affected by a fault f if p tests any of the elements of f; i.e. there is some element in f that is also in p:

A fault f **affects** a probe p if $f \bigcap p \neq \phi$.

In a network, the "objects" may be physical entities such as routers, servers, and links, or logical entities such as software components, database tables, etc. Probes are issued from machines, called probe-stations, where probing software is installed, and traverse the network, testing the availability and performance of the various objects. Probes can be low-level ping probes, or higher level test transactions such as web access, e-mail, etc. Each probe may depend on, and thus tests the functioning of, a wide variety of different objects in the network.

Thus there is a *logical* network associated with, but distinct from, the physical network. Nodes in the logical network represent objects in the physical network; thus links in the physical network can appear as nodes in the logical network. Links in the logical network represent dependencies between objects in the physical network. When we refer to "nodes" in our network examples, we will be referring to the logical network.

In a noise-free environment, if a probe is successful, then every node it passes through must be up; conversely, if a node is down then any probe which passes through that node fails to return. The probe-station detects whether or not the probe returns successfully. A probe fails to return if it passes through any failed node.

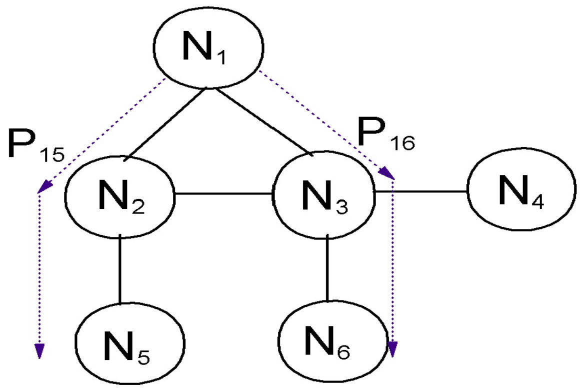


Fig. 2. An example network with two probes.

C. Dependency Matrix

At this point it is convenient to introduce the notion of a **dependency matrix** to capture the relationships between faults and probes. Given any set of faults $F = \{f_1, f_2, ..., f_n\}$ and probes $P = \{p_1, p_2, ..., p_r\}$, the dependency matrix $D_{P,F}$ is given by:

$$\begin{aligned} D_{P,F}(i,j) &= \text{1 if fault } f_j \text{ affects probe } p_i \\ &= \text{0 otherwise.} \end{aligned}$$

$D_{P,F}$ is an r-by-n matrix, where $r = |P|$ and $n = |F|$. Each row of $D_{P,F}$ represents a probe and each column represents a fault. Let c_j denote the j^{th} column of $D_{P,F}$ - c_j is the **probe "signal"** of fault f_j; it gives the results of the probes when only fault f_j occurs.

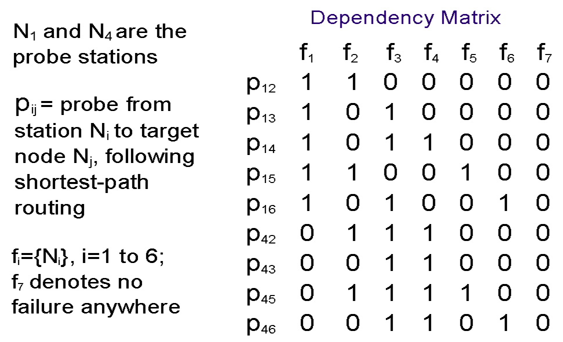
Example: To illustrate these ideas, consider the example network shown in Figure 2, with simple ping probes. Probe P_{ij} denotes a probe sent from node N_i to node N_j along the shortest path between the 2 nodes. For example, probe P_{15} follows the path $N_1 \rightarrow N_2 \rightarrow N_5$ while probe P_{16} follows the path $N_1 \rightarrow N_3 \rightarrow N_6$.

Suppose we wish to be able to detect and localize a single failure occurring in any of the nodes, or no failure anywhere in the network. In this case $f_i = \{N_i\}, i =$1 to 6, and $f_7 = \phi$ = the empty set. Then the dependency matrix will contain one column for each node, as well as an additional column for f_7, no failure anywhere in the network.

For this example let the probe-stations be nodes N_1 and N_4, and suppose that a probe can be sent from each probe-station to any node along the shortest path between them. The resulting dependency matrix is shown in Figure 3. For each probe (row), there is a 1 in each node (column) that the probe passes through, since each probe can detect a failure in any node that it passes through.

III. Problem Formulation

We assume that the faults F we wish to be able to detect and the collection of probes P that are available for use are given. We are interested in finding small sets of probes for



N_1 and N_4 are the probe stations

p_{ij} = probe from station N_i to target node N_j, following shortest-path routing

f_i={N_i}, i=1 to 6; f_7 denotes no failure anywhere

Dependency Matrix

	f_1	f_2	f_3	f_4	f_5	f_6	f_7
p_{12}	1	1	0	0	0	0	0
p_{13}	1	0	1	0	0	0	0
p_{14}	1	0	1	1	0	0	0
p_{15}	1	1	0	0	1	0	0
p_{16}	1	0	1	0	0	1	0
p_{42}	0	1	1	1	0	0	0
p_{43}	0	0	1	1	0	0	0
p_{45}	0	1	1	1	1	0	0
p_{46}	0	0	1	1	0	1	0

Fig. 3. The dependency matrix for the network in Figure 2.

the following two tasks: **detecting** whether or not a fault occurs, and **localizing** any fault that occurs, i.e. determining which $f \in F$ has in fact occurred. The final probe set must be a subset of the collection of probes P.

The tasks of fault detection and fault localization are important for the pre-planned and active probing strategies outlined in Section I-B. In one scenario fault localization is performed using only a pre-planned probe set; in the second scenario fault localization is performed by first using a pre-planned probe set for fault detection, and then sending additional probes using active probing to localize the fault.

A. Fault Detection

The task of fault detection is to find the smallest subset P' of the probe set P such that, if any (non-empty) $f \in F$ occurs, there is some probe $p \in P'$ that is affected by f. This can be formulated in terms of the dependency matrix:

Detection: Given $D_{P,F}$, find P^* that minimizes $|P'|$, where $P' \subseteq P$ such that there is at least one 1 in every column of $D_{P',F}$.

By monitoring the probes we will know, as soon as a probe fails to return, that there is a problem somewhere in the network, but we may not know exactly what the problem is.

B. Fault Localization

The task of fault localization is to find the smallest set of probes such that, if any $f \in F$ occurs, we can determine, from the return values of all the probes, exactly which fault has occurred. Note that we are assuming that only one fault occurs at a time; however, a fault may represent the simultaneous failure of one or more objects. We also assume that objects do not change state (from up to down or vice-versa) while probing is taking place.

Fault localization requires finding the smallest probe set such that every fault has a unique probe signal, since in that case exactly which fault has occurred can be determined from the probe results. Since the probe signal of fault f_j is the column c_j of $D_{P,F}$, each fault has a unique probe signal if and only if each column in $D_{P,F}$ is **unique**; i.e. differs from every other column. Since two columns c_i, c_j differ if and only if there is some entry where one of them has the value 1 while the other has the value 0 (i.e. there is some probe which is affected by one of the faults but not the other), fault localization can be expressed using the number of non-zero elements, denoted by n_{ij}, in $c_i \bigoplus c_j$, where $\bigoplus$ denotes exclusive-OR:

Localization: Given $D_{P,F}$, find P^* which minimizes $|P'|$, where $P' \subseteq P$ satisfies $\forall f_i, f_j \in F, n_{ij} \geq 1$.

Since fault detection only requires finding the smallest probe set such that every fault has a nonzero, but not necessarily unique, signal, clearly fault localization requires a larger probe set than fault detection.

Example: Returning to the example of Figure 3, fault detection requires finding the smallest number of rows such that every column (excluding, of course, f_7) has at least one 1. In this example, this means the smallest set of probes which pass through every node, so that, no matter which node fails, there is a probe that will detect it. The following set of 2 probes suffices:

$$\begin{array}{ccccccc} & f_1 & f_2 & f_3 & f_4 & f_5 & f_6 \\ p_{16} & 1 & 0 & 1 & 0 & 0 & 1 \\ p_{45} & 0 & 1 & 1 & 1 & 1 & 0 \end{array}$$

Since no single probe passes through all the nodes, this is clearly a smallest subset for fault detection. However this set fails for the task of fault localization because, for example, failures in nodes N_1 and N_6 cannot be distinguished from each other - they generate the same signal, since their columns are identical. However the following set of 3 probes is a minimal set for fault localization:

$$\begin{array}{cccccccc} & f_1 & f_2 & f_3 & f_4 & f_5 & f_6 & f_7 \\ p_{15} & 1 & 1 & 0 & 0 & 1 & 0 & 0 \\ p_{16} & 1 & 0 & 1 & 0 & 0 & 1 & 0 \\ p_{42} & 0 & 1 & 1 & 1 & 0 & 0 & 0 \end{array}$$

Note that the "fault" f_7, denoting no failure anywhere in the network, is included here, because we want its column to be unique, as well as the columns of each individual node failure. Since all 7 columns are unique, the results of these 3 probes allow us to determine exactly which node has failed. For example, if p_{15} and p_{16} both fail but p_{42} succeeds, then we infer that node N_1 has failed.

IV. Analysis

We now show that the tasks of fault detection and fault localization are NP-hard. We then define a measure of the quality of a probe set that will be useful in developing approximation algorithms.

A. NP-hardness

We are given a set of faults $F = \{f_1, \ldots, f_n\}$, a collection $P = \{p_1, \ldots, p_r\}$ of probes (and thus the $r \times n$ dependency matrix $D_{P,F} = \{d_{ij}\}$ with $d_{ij} = 1$ if probe p_i intersects fault f_j and 0 otherwise). A set of probes *covers* a fault if at least one of the probes in the set is affected by the fault. We want to find the smallest subset of P that covers all of F. The corresponding decision problem is to determine, given $D_{P,F}$ and some positive integer $k \leq r$, whether P contains such a covering subset of size at most k. We call this decision problem FAULT DETECTION.

Proposition 1: FAULT DETECTION is NP-hard.

Proof: FAULT DETECTION is precisely the MINIMUM SET COVER problem, which is known to be NP-hard [5]. ■

Given a set of faults F, a collection of probes P, and a positive integer $k \leq r$, we want to determine whether P contains a subcollection P' of size at most k such that for every pair of distinct faults $f_1, f_2 \in F$, there is a probe $p \in P'$ that intersects exactly one of f_1 and f_2 (thus p distinguishes between f_1 and f_2); or equivalently P' is such that the columns of the dependency matrix $D_{P',F}$ are all unique. We call this decision problem FAULT LOCALIZATION.

Proposition 2: FAULT LOCALIZATION is NP-hard.

Proof: FAULT LOCALIZATION can be shown to be NP-hard via a reduction from 3-DIMENSIONAL MATCHING. This problem (see [6]) is however not very well-known, so it is instructive to reduce FAULT LOCALIZATION from FAULT DETECTION. Intuitively, FAULT LOCALIZATION is a harder problem than FAULT DETECTION. However, because for any given instance the optimal solutions of these two problems can be very different, the proof is not straight-forward; the details can be found in the Appendix. ■

B. Localization Decomposition of a Probe Set

To develop polynomial-time algorithms that can approximate the probe subset of minimal size, we will define a measure which estimates the quality of a probe set. We will focus on the more difficult fault localization problem; similar ideas can be applied to the fault detection problem.

The quality of a probe set can be measured using the amount of information provided by the probe set about which fault has occurred. Given a dependency matrix $D_{P,F}$, faults in F that have the same column generate the same probe signal when they occur, and hence cannot be distinguished by the probe set P. For any faults $f_i, f_j \in F$, define **fault indistinguishability** as:

$$f_i \sim f_j \text{ if } c_i = c_j, \text{ where } c_i \text{ is the } i^{th} \text{ column of } D_{P,F}.$$

Fault indistinguishability is an equivalence relation and hence induces a decomposition of F into an exhaustive collection of disjoint subsets, which we denote by $S_{P,F}$ and call the **localization decomposition** of P. Thus, if there are k distinct columns in $D_{P,F}$,

$$S_{P,F} = \{G_i | i = 1 \text{ to } k\}, \text{ where } f_j \in G_i \text{ if and only if } c_j \text{ is identical to the } i^{th} \textbf{ distinct} \text{ column of } D_{P,F}.$$

Example: Consider the dependency matrix corresponding to the probe set P of Figure 2:

$$\begin{array}{c|ccccccc} & f_1 & f_2 & f_3 & f_4 & f_5 & f_6 & f_7 \\ p_{15} & 1 & 1 & 0 & 0 & 1 & 0 & 0 \\ p_{16} & 1 & 0 & 1 & 0 & 0 & 1 & 0 \end{array}$$

In this example the localization decomposition is:

$$S_{P,F} = \{\{f_1\}, \{f_2, f_5\}, \{f_3, f_6\}, \{f_4, f_7\}\}$$

Recall that $f_i = \{N_i\}, i = 1$ to 6, f_7 = no failure. Thus this decomposition reflects the facts that: If p_{15} and p_{16} both fail, this indicates a failure in N_1 (because both probes pass through it); if p_{15} fails while p_{16} succeeds, this indicates a failure in either N_2 or N_5; if p_{15} succeeds while p_{16} fails, this indicates a failure in either N_3 or N_6; finally if p_{15} and p_{16} both succeed, this indicates a failure in either N_4 or no failure.

The localization decomposition can be used to find the best probe to add to the current probe set. In this example we seek a third probe that will further decompose each of the 2-element subsets of $S_{P,F}$ into singleton sets. To do this we need a probe that passes through one of the 2 elements in each subset of $S_{P,F}$ but not the other. Examining the dependency matrix of Figure 3 shows that probe p_{42} does this, giving the following probe set P':

$$\begin{array}{c|ccccccc} & f_1 & f_2 & f_3 & f_4 & f_5 & f_6 & f_7 \\ p_{15} & 1 & 1 & 0 & 0 & 1 & 0 & 0 \\ p_{16} & 1 & 0 & 1 & 0 & 0 & 1 & 0 \\ p_{42} & 0 & 1 & 1 & 1 & 0 & 0 & 0 \end{array}$$

Since all the columns are unique, all faults can be localized:

$$S_{P',F} = \{\{f_1\}, \{f_2\}, \{f_3\}, \{f_4\}, \{f_5\}, \{f_6\}, \{f_7\}\}$$

Since each probe either succeeds or fails (we do not consider utilizing the actual value of the probe round-trip time), a set of r probes can generate no more than 2^r different probe signals. Hence at least $\lceil \log n \rceil$ probes are needed to localize a set of n faults. Thus the above set is minimal, because 7 faults require at least $\lceil \log 7 \rceil = 3$ probes. However in general the minimal probe set may be considerably larger, depending on which probes are available. Details of network structure, network routing, probe-station location, and so on may affect the set of available probes.

C. Localization Quality of a Probe Set

We define the **localization quality** $Q(P, F)$ of a set of probes P for the task of fault-localization of a set F of faults in terms of the amount of information P provides concerning which fault in F has occurred. If $\mathcal{X}$ and $\mathcal{Y}$ are random variables taking values $x_1, ..., x_n$, $y_1, ..., y_m$, then from information theory (see [7]) we have:

Entropy: $H(\mathcal{X}) = -\sum_{i=1}^{n} p(\mathcal{X} = x_i) \log p(\mathcal{X} = x_i)$

Conditional entropy:
$H(\mathcal{Y}|\mathcal{X}) = \sum_{i=1}^{n} p(\mathcal{X} = x_i) H(\mathcal{Y}|\mathcal{X} = x_i)$

Conditional entropy is the expected value of the entropies of the conditional distributions, averaged over the conditioning variable.

Now let $\mathcal{F}$ be the random variable denoting the fault; $\mathcal{F}$ has some prior probability distribution specifying $p(\mathcal{F} = f_i)$ for each $f_i \in F$. The localization decomposition $S_{P,F}$ is a collection of groups $\{G_1, ..., G_k\}$, where each group G_i contains the faults that cannot be distinguished from one another by P. Let $\mathcal{G}$ be the random variable denoting which group of $S_{P,F}$ contains the fault. Then we define the **localization quality** of P as the **conditional entropy** $H(\mathcal{F}|\mathcal{G})$:

$$Q(P, F) = H(\mathcal{F}|\mathcal{G})$$

Since the mutual information $I(\mathcal{F}; \mathcal{G})$ satisfies $I(\mathcal{F}; \mathcal{G}) = H(\mathcal{F}) - H(\mathcal{F}|\mathcal{G})$ (see [7]), we can think of localization quality as measuring the amount of additional information needed to localize the fault, beyond the information provided by the probe set P. Thus lower values of $Q(P, F)$ correspond to better probe sets.

The following proposition illustrates the use of this measure:

Proposition 3: If faults are independent and equally likely, then $Q(P, F) = \sum_{i=1}^{k} \frac{n_i}{n} \log n_i$, where n_i is the number of faults in group G_i of $S_{P,F}$, and $n = |F|$.

Proof:

If faults are independent and equally likely, then:

$$\begin{aligned} p(\mathcal{G} = G_i) &= n_i/n & (1)\\ p(\mathcal{F} = f_j|\mathcal{G} = G_i) &= 1/n_i \text{ if } f_j \in G_i & (2)\\ &= 0 \text{ otherwise} & (3) \end{aligned}$$

Hence:

$$\begin{aligned} Q(P, F) &= H(\mathcal{F}|\mathcal{G}) \\ &= \sum_{i=1}^{k} p(\mathcal{G} = G_i) H(\mathcal{F}|\mathcal{G} = G_i) \\ &\quad \text{(by definition of conditional entropy)} \\ &= \sum_{i=1}^{k} p(\mathcal{G} = G_i)[-\sum_{j=1}^{n} p(\mathcal{F} = f_j|\mathcal{G} = G_i) \\ &\quad \log p(\mathcal{F} = f_j|\mathcal{G} = G_i)] \\ &\quad \text{(by definition of entropy)} \\ &= \sum_{i=1}^{k} \frac{n_i}{n}[-\sum_{j=1}^{n_i} \frac{1}{n_i} \log \frac{1}{n_i}] \\ &\quad \text{(using (1), (2) and (3))} \\ &= \sum_{i=1}^{k} \frac{n_i}{n} \log n_i \end{aligned}$$

■

This expression has a natural interpretation. Since there are n_i faults in group G_i, at least $\log n_i$ additional probes are needed to localize all the faults in G_i. Since a random fault lies in G_i with probability n_i/n, the localization quality $Q(P, F)$ is simply the expected minimal number of additional probes needed to completely localize the fault. Note that lower values for $Q(P, F)$ correspond to better probe sets, and that $Q(P, F) = 0 \Leftrightarrow n_i = 1 \forall i \Leftrightarrow$ all faults are localizable.

Example:: Suppose probe set P_1 induces the decomposition $S_1 = \{\{f_1, f_2\}, \{f_3, f_4\}\}$ while probe set P_2 induces the decomposition $S_2 = \{\{f_1\}, \{f_2, f_3, f_4\}\}$. Although P_2 can uniquely localize one fault and P_1 cannot, it is possible to add just a single probe to P_1 and diagnose all the faults, whereas at least two additional probes must be added to P_2 before all faults can be diagnosed. Therefore, P_1 is a better probe set than P_2.

The localization quality reflects this:

$$\begin{aligned} Q(P_1, F) &= \frac{1}{2}\log 2 + \frac{1}{2}\log 2 = \log 2 = 1, \\ Q(P_2, F) &= \frac{1}{4}\log 1 + \frac{3}{4}\log 3 = \frac{3}{4}\log 3 = 1.19. \end{aligned}$$

V. Algorithms for Finding the Minimal Probe Set

We now examine algorithms for finding the minimal probe set for fault localization. Exhaustive search is impractical for large networks because the problem is NP-hard, so approximation algorithms must be considered. We present a quick (linear-time) "subtractive" search algorithm and a family of "greedy search" algorithms requiring polynomial time. An experimental comparison of the algorithms is presented in Section D.

The algorithms require computing the localization quality of probe sets, so we first describe how to do this efficiently.

A. Computing the Localization Decomposition

The localization decomposition $S_{P,F}$ is an exhaustive collection of disjoint sets, each containing the faults in F whose columns in $D_{P,F}$ are identical. Comparing each column with every other column requires $O(n^2 r)$ time, where r = number of probes (rows) and n = number of faults (columns). The decomposition can be computed in only $O(nr)$ time by proceeding row-by-row - the idea is that adding a row (i.e. a probe) results in a more extensive decomposition, because faults in distinct groups remain distinguishable; an additional probe only has the effect of distinguishing previously indistinguishable faults.

```
Input: Dependency matrix $D_{P,F}$, with rows $p_1, p_2, ..., p_r$
Output: Localization decomposition $S_{P,F}$
Algorithm:
  $S_0 = \{\{f_1, f_2, ..., f_n\}\}$
  For each probe $p_i$
    $S_i = \phi$
    For each $G$ in $S_{i-1}$
      $G_0 = \{f \in G | D(i,j) = 0\}$
      If $G_0 \neq \phi$, $S_i \leftarrow S_i \bigcup \{G_0\}$
      $G_1 = \{f \in G | D(i,j) = 1\}$
      If $G_1 \neq \phi$, $S_i \leftarrow S_i \bigcup \{G_1\}$
  Output $S_r$
```

Computing the localization decomposition $S_{P,F}$

The algorithm is shown above. At any step i, S_i represents the localization decomposition resulting from the first i probes. The $(i+1)^{th}$ probe p_{i+1} decomposes each set G in S_i into 2 subsets G_0 and G_1, depending on which faults in G affect p_{i+1}. G_0 and G_1 are added as elements to S_{i+1}, if they are non-empty. Note that there can never be more than $n = |F|$ sets in S_i, because the decomposition cannot proceed beyond $\{\{f_1\}, \{f_2\}, ..., \{f_n\}\}$.

Example: In the example of section IV-B, the first probe is:

$$\begin{array}{cccccccc} & f_1 & f_2 & f_3 & f_4 & f_5 & f_6 & f_7 \\ p_{15} & 1 & 1 & 0 & 0 & 1 & 0 & 0 \end{array}$$

This yields:

$$S_1 = \{\{f_1, f_2, f_5\}, \{f_3, f_4, f_6, f_7\}\}$$

Then consider the second probe:

$$\begin{array}{cccccccc} & f_1 & f_2 & f_3 & f_4 & f_5 & f_6 & f_7 \\ p_{16} & 1 & 0 & 1 & 0 & 0 & 1 & 0 \end{array}$$

This further decomposes each subset of S_1 according to whether the corresponding entry is 1 or 0. This yields:

$$S_2 = \{\{f_1\}, \{f_2, f_5\}, \{f_3, f_6\}, \{f_4, f_7\}\},$$

and so on.

B. Quick Search

Quick search starts with the initial set of r probes, considers each probe in turn, and *discards it if it is not needed*; i.e. if the localization quality remains the same even if it is dropped from the probe set. This process terminates in a subset with the same localization quality as the original set but which may not necessarily be of minimal size. The running time is linear in the size of the original probe set, because each probe is considered only once.

```
Input: Dependency matrix $D_{P,F}$, with rows $p_1, p_2, ..., p_r$
Output: Probe Set $P'$ (possibly non-minimal size)
Algorithm:
  $P' = \{p_1, p_2, ..., p_r\}$
  For $i = 1$ to $r$
    If $S_{P' \backslash p_i, F} = S_{P,F}$, $P' \leftarrow P' \backslash p_i$
  Output $P'$
```

Quick Search discards any unnecessary probe ($O(r)$ time)

The order of the initial probe set is quite important for the performance of this algorithm. Ordering the probes by probe station may reduce the opportunity of exploiting interactions among probes from different probe stations. The size of the final probe set can be reduced by randomly ordering the initial probe set, or ordering it by target node.

C. Greedy Search

Another approach is a greedy search algorithm. The simplest version of this algorithm is to add probes to the probe set one-by-one; at each step add the probe that results in the “most informative” probe set, using the quality measure $Q(P,F)$ defined above. Continue until all faults can be localized or until the localization quality of the original probe set is achieved. The running time of this algorithm is $O(r^2)$, where r is the size of the original probe set, because at each step the localization quality achieved by adding each of the remaining probes must be computed. The resulting probe set is not necessarily optimal because of the greedy nature of the search.

```
Input: Dependency matrix $D_{P,F}$, with rows $p_1, p_2, ..., p_r$
Output: Probe Set $P'$ (possibly non-minimal size)
Algorithm:
  $P' = \phi$ = empty set
  While $S_{P',F} \neq S_{P,F}$
    $p^* = argmin_{p \in P \setminus P'} Q(P' \bigcup \{p\}, F)$
    $P' \leftarrow P' \bigcup \{p^*\}$,
  Output $P'$
```

Greedy Search adds the best probe at each step ($O(r^2)$ time)

This approach can be extended by adding at each step the best remaining subset of t probes, for any fixed t. The overall running time is easily seen to be $O(r^{t+1})$. (Actually near the end of this process one must consider all subsets of t or fewer probes, to prevent adding probes that are unnecessary, but this doesn't change the essential results.) Of course the larger the value of t, the closer the probe set will be to the optimal one, but the longer the computation time required. (The above algorithm is the case $t = 1$.)

D. Experiments

This section describes the empirical behavior of the minimum probe set size and the performance of the approximation algorithms. The main result is that the algorithms find a probe set which is very close to the true minimum set size, and can be effectively used on large networks where exhaustive search is impractical.

For each network size n, we generated twenty random networks with n nodes by randomly connecting each node to four other nodes. Each link is given a randomly generated weight, to reflect network load. The probe stations are selected randomly. The available probes follow the least-cost path from each probe station to each node.

The faults we are interested in diagnosing are any single node being down or no failure anywhere in the network. We assume that each node has the same prior probability of failure, and that there is no noise in the probe results. Note that in this case n probes are sufficient, because one can always use just one probe-station and probe every single node. Thus we expect that the minimal number of probes should lie between $\log n$ and n.

Exhaustive search is performed to find the true minimum size probe set. Then the linear-time quick search algorithm and the quadratic-time greedy search algorithm are used to find probe sets.

Figure 4 shows the average probe set size, for the case of three probe stations. The minimal probe set size lies between $\log n$ and n, as expected. The minimal size is always larger than the theoretical lower bound of $\log n$, for three reasons:

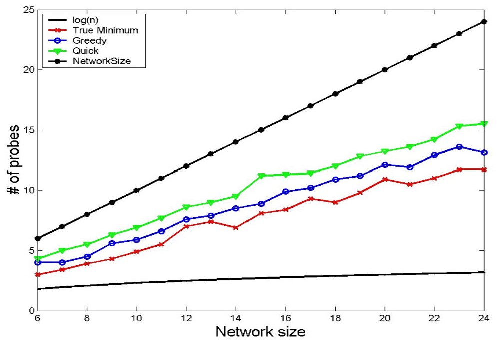


Fig. 4. Algorithms for Computing Probe Sets: True Minimum and Two Approximation Algorithms on Small Networks

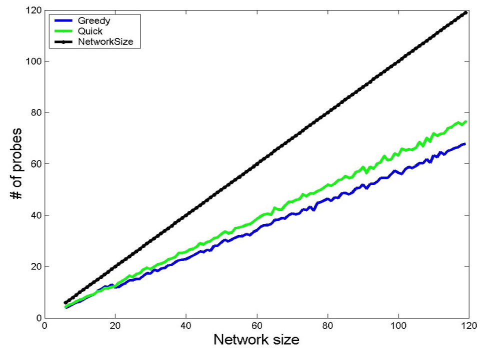


Fig. 5. The Approximation Algorithms on Large Networks.

- The networks are not very dense; the number of edges increases only linearly with network size. Thus many probe paths are simply not possible.
- Since the probes follow the least-cost path from probe station to node, the probe paths tend to be short, passing through few nodes. This reduces the opportunities for exploiting interactions between probe paths.
- The probe stations are selected randomly - a better placing of probe stations would produce fewer probes.

The results also show that the approximation algorithms perform well; the size of the probe set is much closer to the true minimum than to the upper bound. Figure 5 illustrates the performance of these algorithms on larger networks for which exhaustive search is not feasible. The greedy algorithm slightly outperforms the linear-time algorithm, but its quadratic computational cost is higher. An alternative approach is to run the linear-time algorithm many times with different initial orderings and take the best result.

VI. Noise

So far we have assumed a noise-free environment in which all the probe results are received correctly. In practice we cannot expect this to be the case; a probe may fail even if there is no fault anywhere along its intended path, or

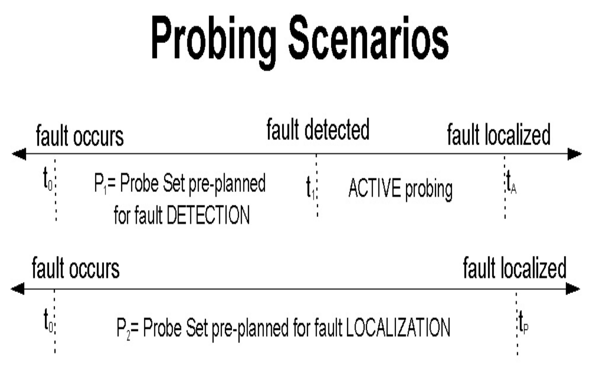


Fig. 6. Probing Scenarios.

a probe may succeed even if there is a fault on its path, e.g. because dynamic routing sends it along a different path.

Noise can be handled by expanding the probe set, in two fundamentally different ways - in space or in time. For example, suppose up to m probe results may be incorrectly received. In this case faults are defined to be equivalent if their probe signals differ in no more than $2m+1$ places; if n_{ij} is the number of non-zero elements in $c_i \bigoplus c_j$, where c_i is the i^{th} column of $D_{P,F}$, and $\bigoplus$ denotes exclusive-OR, then:

$$f_i \sim f_j \text{ if } n_{ij} \geq 2m+1.$$

The choice of $2m+1$ ensures that when the probe signal is received, even if the results of m probes are incorrect, there is only one column in the dependency matrix which is within Hamming distance m of the probe signal. Hence the fault can be uniquely localized, despite the noise. Of course the probe set size will increase as m increases.

Expanding the probe set in time refers to sending the same probes repeatedly and assuming that a probe result is valid if the probe yields the same result some number of times; transient probe results are assumed to be the result of noise. More complex models to handle noise can be developed; e.g. Bayesian networks are used in some of our other work, which provides some theoretical bounds on the diagnosis error and the number of probes required for asymptotically error-free diagnosis. ([8]).

VII. Active Probing

Here we extend the probing paradigm to allow for **active probing**. In this scenario the selection of later probes depends on the results of earlier probes. One decides in real-time which probes to send based on the current assessment of the network; after the probe results are received, one decides which further probes are needed, and so on. The advantage of this approach is that fewer probes can be used than if the entire probe set has to be pre-planned.

Two probing strategies are illustrated in Figure 6. In the active probing scenario, a pre-planned probe set is used that can **detect** that a fault has occurred; localizing the fault - **determining** which fault has occurred - is then achieved by sending additional probes in an active mode. In the pre-planned, or “passive”, scenario the entire probe set must be pre-planned so that it can determine exactly which fault has occurred.

A. Analysis

A pre-planned probe set for complete localization will always be larger than a probe set that is used for detection only, and so the time from fault occurrence (t_0 in Figure 6) to fault localization (t_P in Figure 6) will be larger than the time from fault occurrence to detecting that a fault has occurred somewhere (t_1 in Figure 6). However the important question is whether the combination of pre-planning for detection and then using active probing for localization is better or worse than pre-planning for localization; i.e. is t_P larger or smaller than t_A?

The following analysis indicates the issues involved. Let n_P be the number of probes needed for fault localization using pre-planned probing, and n_D the number needed for fault detection; we know that $n_D \leq n_P$. Probing is performed by sending probes at intervals, which may be scheduled either periodically or randomly. Suppose M probes are sent simultaneously at any time, and τ_1 is the average time between probing intervals. Assume for simplicity that in active mode only one probe is sent at a time; as soon as that probe returns, the next probe to send is determined and sent, and so on. Let n_A be the number of probes needed in the active phase to localize the problem and τ_2 the average time between these probes.

Then (assume $t_0 = 0$ for convenience):

$$\begin{aligned} t_P &\sim (n_P/M)\tau_1 \\ t_A &\sim (n_D/M)\tau_1 + n_A\tau_2 \end{aligned}$$

In practical applications $\tau_1 \gg \tau_2$. This is because τ_1 is the average time between intervals when probes are sent - to avoid overloading the network, τ_1 is usually on the order of magnitude of minutes. However in active mode, the next probe can be sent as soon as the result of the previous probe is received; thus τ_2 consists of probe-round-trip time together with the time for computing the next probe, so τ_2 may be on the order of magnitude of milliseconds. Since $n_D \leq n_P$, active probing will achieve faster fault localization unless n_A, the number of probes needed in active mode, is very large. We now describe how to decide which probe to send next, and present experimental results that show that n_A is small when compared with n_P.

In practice there will always be some costs of switching into active probing mode. Thus the gains yielded by active probing will depend on the frequency with which failures occur; if failures are very frequent an entirely pre-planned approach may be more cost-effective. The benefits of active probing increase as failure frequency decreases.

B. Selecting the Next Probe

Let $\mathcal{F}$ be a random variable denoting the fault. At each step k, $Pr(\mathcal{F}|T_k)$ denotes the probability distribution describing the joint probability of occurrence of faults, given the outcomes T_k of tests (probes) received so far. $Pr(\mathcal{F}|T_k)$ can be computed using, for example, a Bayesian approach for updating the prior probability of a hypothesis given new evidence.

Assume only one probe at a time is sent in active mode. Selection of the probe p_{k+1} to send next is done by computing the expected information gain each probe will provide, taking into account the likelihood that it fails or succeeds and what would be learned about the fault distribution; the most-informative probe is the one which maximizes the information gain.

For each candidate probe p_i, one computes $Pr(p_i$ fails) and $Pr(p_i$ succeeds). Then one computes the hypothetical fault distributions $Pr(\mathcal{F}|T_k \bigcup (p_i$ fails)) and $Pr(\mathcal{F}|T_k \bigcup (p_i$ succeeds)) The **information gain** of p_i failing is:

$$I_k(p_i \text{ fails}) = H(\mathcal{F}|T_k)) - H(\mathcal{F}|T_k \bigcup (p_i \text{ fails})), \text{ where } H \text{ denotes entropy,}$$

and similarly for p_i succeeding.

The probe to send next is selected to maximize the expected information gain:

$$p^*_{k+1} = argmax_i[Pr(p_i \text{ fails})I_k(p_i \text{ fails}) + Pr(p_i \text{ succeeds})I_k(p_i \text{ succeeds})]$$

The analysis generalizes directly to the case of sending s probes at a time; in practice s is usually quite small.

C. Results

The method described above was implemented in the experimental framework described in Section V-D. In this case, the algorithm for selecting the most-informative probe given the previous probe outcomes can be simplified as follows. We maintain the *target set* - the minimal node set which is guaranteed to contain the faulty node. (The “no failure” situation is viewed as an additional node). Initially, the target set includes all nodes. It is easy to see that the maximum information gain is provided by the probe that includes the largest number of nodes from the target set.

If the probe is successful, we remove the nodes on its path from the target set, and send the next most-informative probe. We continue doing so until we either get an unsuccessful probe, or the set of target nodes becomes empty (corresponding to the no-failure situation). As soon as a probe is unsuccessful, the faulty node is pinpointed by doing a binary search among target nodes on its path.

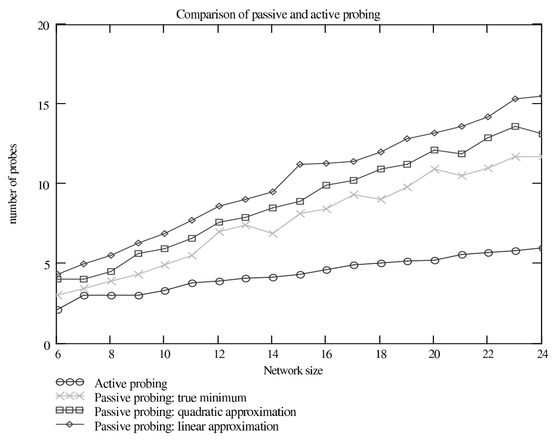


Fig. 7. Active Probing - small networks.

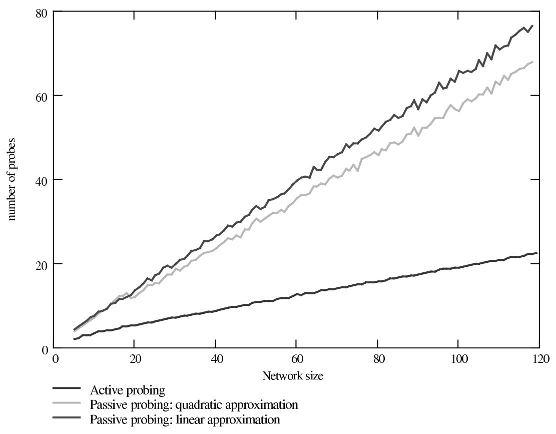


Fig. 8. Active Probing - large networks.

We ran simulations over the same set of networks described in Section V-D. For each network, each node was sequentially selected to be the faulty one, including the no-failure “pseudo-node”. The number of probes required to diagnose the fault was averaged over all networks of a given size.

The results presented in Figure 7 (small networks) and Figure 8 (large networks) clearly indicate the considerable improvement resulting from active probing when compared with pre-planned, or “passive”, probing. Thus we see that an active probing approach can greatly reduce the number of probes needed to successfully diagnose faults.

VIII. Related Work

Our previous work [9] and independently Ozmutlu et al. [10] studied the probe selection problem for the purpose of network management. Extending the previous work, this paper develops a more general framework for problem determination using probes, proves the NP-hardness of the problems, and studies the active probing approach and demonstrates its advantages in reducing probe size and time-to-decision. The active probing strategy, although very intuitive, has not been formally discussed before.

Our work relates to four broad categories of previous work: event correlation, system-level diagnosis, network fault diagnosis, and performance measurement. Event correlation ([1], [2]) for identifying root-causes has long been recognized as a critical issue in the system management domain. Problem determination is performed by analyzing alarms emitted by devices when a significant situation occurs. Unlike the probing scheme, alarms are "reactive" to a situation and this requires intensive instrumentation, only possible in a tightly managed environment. The probing approach uses test transactions that can be built easily without touching the existing devices.

Nonetheless, event correlation has many similarities to our work. The formulation of problem diagnosis as a "decoding" problem, where "problem events" are decoded from "symptom events", was first proposed by [3]. In our framework, the result of a probe constitutes a "symptom event", while a failure is a "problem event". However beyond this conceptual similarity the two approaches are quite different. The major difference is that we use an active probing approach versus a "passive" analysis of symptom events; namely [3] selects codebooks (a combination of symptoms encoding particular problems) from a specified set of symptoms, while we actively construct those symptoms (probes), a much more flexible approach. Another important difference is that [3] lacks a detailed discussion of efficient algorithms for constructing optimal codebooks.

The problem of fault diagnosis in a system of interconnected components dates back to [11] and [12]. Since that time a large body of literature has developed [13]. In contrast with that work, in our case it is not possible for every node in the network to be used to test other nodes - only a small number of nodes can be used as probe stations to generate the tests. As a result of this the probing problem becomes a "constrained-coding" problem, as explained above.

Other approaches to fault diagnosis in communication networks and distributed computer systems include Bayesian networks [14] and other probabilistic dependency models [15]; another approach is statistical learning to detect deviations from the normal behavior of the network [16]. The probabilistic diagnosis issues are addressed in our related work in [8], where a probabilistic approximation algorithm using Bayesian networks is provided for finding the most-likely problem diagnosis, and some theoretical bounds on the diagnosis error are derived.

Finally, probing has been used for the purpose of performance measurements. In particular, Duffield et al. [17] and Ji et al. [18] recently developed a framework for estimating the performance of a multi-cast network based on probes. Duffield et al. [19] and Paxson [20] studied performance measurements of end-to-end probing. Our work focuses on the problem determination aspect in a typical IP environment. We formulate and develop algorithms for the probe selection problem that has not been studied by the aforementioned authors.

IX. Conclusion

In this paper, we address the problem of diagnosis in distributed systems using test transactions, or probes. Probes offer an approach to diagnosis that is more active than traditional "passive" techniques like event correlation. Our main objective is developing a cost-efficient probing strategy; we want a small probe set which at the same time provides wide coverage for locating or detecting problems anywhere in the network.

We formulate the probe selection problem using a general optimization framework. By reasoning about the interactions among the probe paths, an information-theoretic estimate of which probes are valuable can be constructed. This yields a quadratic-time algorithm which finds near-optimal probe sets. We also implement a linear-time algorithm which can be used to find small probe sets very quickly.

We consider two probing approaches: entirely pre-planned and active. The active probing approach has two phases: fault detection by pre-planned probes, and fault localization by using additional probes as needed, based on previous observations. Since fault detection requires fewer probes than fault localization, the active probing approach is usually a more efficient strategy.

Directions for future work include real-time diagnosis with changing network state and intermittent faults, handling dynamic routing and lack of precise knowledge of the probe path, and adapting to non-stationary behavior of the system using on-line learning that yields a dynamic, adaptive, probing strategy. Our main focus remains the cost-efficiency and scalability requirements of problem determination, which will become increasingly important in the context of new technological challenges related to *autonomic computing*, a paradigm for new-generation IT systems capable of self-management and self-repair.

References

[1] A. Leinwand and K. Fang-Conroy. *Network Management: A Practical Perspective, 2nd Edition*. Addison-Wesley, 1995.

[2] B. Gruschke. Integrated Event Management: Event Correlation Using Dependency Graphs. In *Distributed Systems Operations and Management*, 1998.

[3] S. Kliger, S. Yemini, Y. Yemini, D. Ohsie, and S. Stolfo. A coding approach to event correlation. In *Intelligent Network Management (IM)*, 1997.

[4] A. Frenkiel and H. Lee. EPP: A Framework for Measuring the End-to-End Performance of Distributed Applications. In *Proceedings of Performance Engineering 'Best Practices' Conference, IBM Academy of Technology*, 1999.

[5] R. M. Karp. *Complexity of Computer Computations*, chapter Reducibility among combinatorial problems, pages 85–103. Plenum Press, 1972.

[6] M.R. Garey and D.S. Johnson. *Computers and Intractability; A Guide to the Theory of NP-completeness*. W.H. Freeman and Co., San Francisco, 1979.

[7] T.M. Cover and J.A. Thomas. *Elements of information theory*. New York:John Wiley & Sons, 1991.

[8] I. Rish, M. Brodie, and S. Ma. Accuracy vs. Efficiency Trade-offs in Probabilistic Diagnosis. In *Proceedings of the The Eighteenth National Conference on Artificial Intelligence (AAAI-2002), Edmonton, Alberta, Canada*, 2002.

[9] M. Brodie, I. Rish, and S. Ma. Optimizing probe selection for fault localization. In *Distributed Systems Operation and Management*, 2001.

[10] H.C. Ozmutlu, N. Gautam, and R. Barton. Zone recovery methodology for probe-subset selection in end-to-end network monitoring. In *Network Operations and Management Symposium*, pages 451–464, 2002.

[11] F.P. Preparata, G. Metze, and R.T. Chien. On the connection assignment problem of diagnosable systems. *IEEE Transactions on Electronic Computing*, pages 848–854, 1967.

[12] J. D. Russell and C. R. Kime. System fault diagnosis: Closure and diagnosability with repair. *IEEE Transactions on Computers*, pages 1078–1089, 1975.

[13] A.T. Dabhura. System level diagnosis. *Concurrent Computation: Algorithms, Architectures, Technologies*, pages 411–434, 1988.

[14] JF. Huard and A.A. Lazar. Fault isolation based on decision-theoretic troubleshooting. Technical Report 442-96-08, Center for Telecommunications Research, Columbia University, New York, NY, 1996.

[15] I.Katzela and M.Schwartz. Fault identification schemes in communication networks. In *IEEE/ACM Transactions on Networking*, 1995.

[16] C.S. Hood and C. Ji. Proactive network fault detection. In *Proceedings of INFOCOM*, 1997.

[17] N.G. Duffield, J. Horowitz, D. Towsley, W. Wei, and T. Friedman. Multicast-based loss inference with missing data. *Journal on Selected Areas in Communications*, 2002.

[18] C. Ji and A. Elwalid. Measurement-based network monitoring and inference: scalability and missing information. *Journal on Selected Areas in Communications*, 2002.

[19] N.G. Duffield, F. Lo Presti, V. Paxson, and D. Towsley. Inferring link loss using striped unicast probes. In *Proceedings of INFOCOM*, pages 915–923, 2001.

[20] V. Paxson. End-to-end internet packet dynamics. In *Proceedings of SIGCOMM*, pages 139–152, 1997.

APPENDIX

Here we prove that FAULT LOCALIZATION is NP-hard.

Proof: We will show that FAULT LOCALIZATION is NP-hard via a reduction from FAULT DETECTION.

Let the dependency matrix $D_{P,F}$ and the positive integer k be an arbitrary instance of FAULT DETECTION. We need to construct, in polynomial time, sets $\hat{F}$, $\hat{P}$, and an integer $\hat{k} \leq |\hat{P}|$, such that there exists a subset of P of size at most k covering F **if and only if** there exists a subset of $\hat{P}$ of size at most $\hat{k}$ such that $D_{\hat{P},\hat{F}}$ has unique columns.

The set $\hat{F}$ contains all the elements of $F = \{f_1, \ldots, f_n\}$, plus n new elements $g_1, \ldots, g_n$. The set $\hat{P}$ will contain all the probes in P, plus n additional probes $\{f_i, g_i\}$ for all $i \in \{1, \ldots, n\}$. The resulting dependency matrix will be of size $(r+n) \times (2n)$. It can be visualized as the four-block matrix

$$\left[\begin{array}{cc} D_{P,F} & 0 \\ I_n & I_n \end{array} \right],$$

where I_n denotes the $n \times n$ identity matrix. We also set $\hat{k} = k+n-1$ (for reasons that will become clear later). It is easy to see that the reduction can be done in polynomial time. We now show both the sufficient and necessary conditions.

For the "if" direction, we assume that $\langle D_{P,F}, k \rangle \in$ FAULT DETECTION, thus there exists a subset $P' \subseteq P$ of size at most k such that $D_{P',F}$ has at least one 1 in every column. We need to show that $\langle D_{\hat{P},\hat{F}}, \hat{k} \rangle$ is a positive instance of FAULT LOCALIZATION. Indeed, there exists a subset Q of at most rows $\hat{k}$ rows inducing a sub-matrix of $D_{\hat{P},\hat{F}}$ with unique columns – namely, Q contains P' and any $(n-1)$ of the n additional rows $\{f_i, g_i\}$. Since $D_{P',F}$ has a 1 in every column, every column of $D_{Q,\hat{F}}$ corresponding to a fault in F will differ from any column corresponding to a fault in $\{g_1, \ldots, g_n\}$. Since removing a single row from I_n leaves all the n columns distinct, the columns corresponding to the elements of F will all be distinct from each other, as are the columns corresponding to the elements of $\{g_1, \ldots, g_n\}$. Notice that $|Q| \leq k+n-1 = \hat{k}$, as desired.

To show the other direction, we assume that $\langle D_{P,F}, k \rangle$ is a negative instance of FAULT DETECTION. We need to show that $\langle D_{\hat{P},\hat{F}}, \hat{k} \rangle$ will be a negative instance of FAULT LOCALIZATION. Indeed, if any subset of at most k rows of $D_{P,F}$ induces an all-0 column, then any submatrix of $D_{\hat{P},\hat{F}}$ induced by a subset of at most $\hat{k}$ rows will contain duplicate columns. To see this, let Q be a subset of at most $\hat{k}$ rows. There are two cases to be considered:

1) Q contains at most k elements of P. Then, since there is an all-0 column in $D_{P,F}$ induced by $Q \cap P$, we know that $D_{Q,\hat{F}}$ will have identical columns corresponding to faults f_i and g_i, where i is the index of the all-0 column in $D_{P,F}$.
2) Q contains more than k elements of P. In this case Q cannot contain more than $n-2$ of the new rows (since $|Q| \leq \hat{k}$). But then Q will induce duplicate columns in the right half of $D_{\hat{P},\hat{F}}$, since removing at least two rows from I_n induces duplicate columns.

Thus any subset Q of at most $\hat{k}$ rows induces duplicate columns in $D_{\hat{P},\hat{F}}$, making it a negative instance of FAULT LOCALIZATION, as desired.

■