# *Optimizing Probe Selection for Fault Localization*

Mark Brodie, Irina Rish, Sheng Ma
*IBM T.J. Watson Research Center, Hawthorne, NY 10532, USA*
*(mbrodie, rish, shengma)@us.ibm.com*

We investigate the use of probing technology for the purpose of problem determination and fault localization in networks. We present a framework for addressing this issue and implement algorithms that exploit interactions between probe paths to find a small collection of probes that can be used to locate faults. Small probe sets are desirable in order to minimize the costs imposed by probing, such as additional network load and data management requirements. Our results show that although finding the optimal collection of probes is expensive for large networks, efficient approximation algorithms can be used to find a nearly-optimal set.

**Keywords:** probes, fault localization, problem determination, event correlation

## 1. Introduction

As networks continue to grow in size and complexity, system administrators are faced with an ever-increasing volume of event data, and tasks such as fault localization and problem determination become more difficult. As a result, tools are needed that can assist in performing these management tasks by responding quickly and accurately to the large number of events and alarms that are usually generated by even a single fault.

Probing offers the opportunity to develop an approach to problem determination that is more active than traditional event correlation and other methods. A probe is a program that executes on a particular machine (called a probing station) by sending a command or transaction to a server or network element and measuring the response. Probing technology is widely used to measure the quality of network performance, often motivated by the requirements of service-level-agreements. Examples of probing technology include the T. J. Watson EPP technology [2] and the Keynote product [7].

To use probes, probing stations must first be selected at one or more locations in the network. Then the probes must be configured – it must be decided which network elements to target and which station each probe should originate from. Both probe stations and probes impose a cost – probe stations because the probing code must be installed, operated, and maintained, probes because of the additional network load that their use entails and because the probe results must be collected, stored and analysed. There is a trade-off between these costs, since having more probe stations allows fewer probes to be used. Identifying these costs is of considerable interest for probing practitioners.

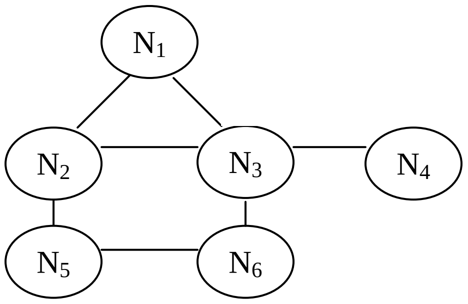
As a first step towards this goal we investigate the question of configuring the probe set in order to perform fault localization. The objective is to obtain a probe set which is both small, thereby minimizing probing costs, yet also provides wide coverage, in order to locate problems anywhere in the network. We describe a system architecture that provides a general framework for addressing this issue. Within this framework we present and implement various algorithms in order to determine the relationship between computational cost and the quality of the probe set that is obtained. Our results show that achieving the minimal probe set can be accurately approximated by fast, simple algorithms that scale well to large networks.

## 2. Approach

### 2.1 Problem Formulation

Finding the minimal set of probes requires answering the following questions: (1) Which probes are available as "candidates" for use in a network? (2) Which faults can be successfully identified by a given set of probes? (3) What is the smallest number of probes that can identify the same collection of faults as a given set? Suppose the network has *n* nodes. Each probe is represented as a binary string of length *n*, where a 1 in position j denotes that the probe passes through node $N_j$. This defines a **dependency matrix** D(i,j), where D(i,j)=1 if probe $P_i$ passes through node $N_j$, D(i,j)=0 otherwise. D is an *r*-by-*n* matrix, where *r* is the number of probes. (This formulation is motivated by the "coding" approach to event correlation suggested by [8].)

For example, consider the network in Figure 1. Suppose one probe is sent along the path $N_1$->$N_2$->$N_5$ while another is sent along the path $N_1$->$N_3$->$N_6$. The resulting dependency matrix is shown to the right of the network (probes are indexed by their start and end nodes).



$$\begin{array}{c|cccccc} & N_1 & N_2 & N_3 & N_4 & N_5 & N_6 \\ \hline P_{15} & 1 & 1 & 0 & 0 & 1 & 0 \\ P_{16} & 1 & 0 & 1 & 0 & 0 & 1 \end{array}$$

Figure 1: An Example Network and Dependency Matrix

Each probe that is sent out either returns successfully or fails to do so. If a probe is successful, then every node and link along its path must be up. Conversely, if a node or link is down then any probe passing through that node or link fails to return. Thus *r* probes result in a "signal" – a binary string of length *r*, each digit denoting whether or not that probe returned successfully. For example, if only $N_2$ is down then $P_{15}$ fails but $P_{16}$ succeeds. Similarly if only $N_5$ is down then $P_{15}$ fails but $P_{16}$ succeeds. Thus these two failures result in the same signal, because their columns in the dependency matrix are identical. Any problem whose column in the dependency matrix is unique generates a unique signal and as a result can be unambiguously diagnosed.

We begin by considering the situation where only one node in the network can fail at any given time. (In Section 6.1 we discuss how to extend the dependency matrix to deal with multiple simultaneous failures.) In Figure 1, examining the columns of the dependency matrix shows that that a failure in node $N_1$ can be uniquely diagnosed, because both probes fail and **no other** single node failure results in the same signal;

$N_1$'s column in the dependency matrix is unique. However, as explained above, a failure in $N_2$ cannot be distinguished from a failure in $N_5$, and similarly a failure in $N_3$ cannot be distinguished from a failure in $N_6$. Although $N_4$'s column is unique, a failure in $N_4$ cannot be distinguished from no failure anywhere in the network, because there is no probe passing through $N_4$. Adding an extra "node" whose column is all zeroes, representing no failure, avoids this technicality.

Thus a dependency matrix decomposes the network into a disjoint collection of nodes, where each group consists of the nodes whose columns are identical; i.e. each group contains those nodes whose failure cannot be distinguished from one another by the given set of probes. This defines the **diagnostic power** of a set of probes. For example, in Figure 1 the diagnostic power is the decomposition {{1}, {2,5}, {3,6}, {4,7}}, where index j represents node $N_j$ and $N_7$ is the extra "node" representing no failure anywhere in the network. A failure is diagnosable if and only if its group is a singleton set.

It is important to note that the network model is quite general. For example, layering can be accommodated: if a web-server depends on TCP/IP running which depends on the box being up, this can be modeled as a node for the box with a link to TCP/IP from the box and a further link from TCP/IP to the web-server. Thus nodes may represent applications and links dependencies between those applications. Similarly, a node may represent a sub-network of many nodes whose inter-connections are unknown. In this case probing will determine that the problem lies somewhere in that sub-network, at which point some form of local system management (perhaps including local probing) may be used to pinpoint the problem.

## 2.2 Architecture

The system architecture is shown in Figure 2. First the candidate probes are identified and the dependency matrix and its diagnostic power are determined. Then a subset of the candidate probes is found with the same diagnostic power as the entire set. Various algorithms are available to compute this, depending on whether the minimal set of probes is required or if a non-minimal set is adequate.

The candidate probes may be provided as an input from an external source, such as a human expert, or they may be computed as a function of factors like the structure of the network, the location of the probe stations, the routing strategy, and so on. This is described in Section 3.1. The procedure for determining the diagnostic power of a set of probes is explained in Section 3.2. Given the dependency matrix, the algorithm for finding the minimal subset with the same diagnostic power is described in Section 3.3(i). Approximation algorithms which are much faster but are not guaranteed to find the exact minimal set are explored in Sections 3.3(ii) and 3.3(iii). The performance of all the algorithms is evaluated in section 4. The choice of algorithm may depend on a decision criterion which specifies the cost/optimality trade-off.

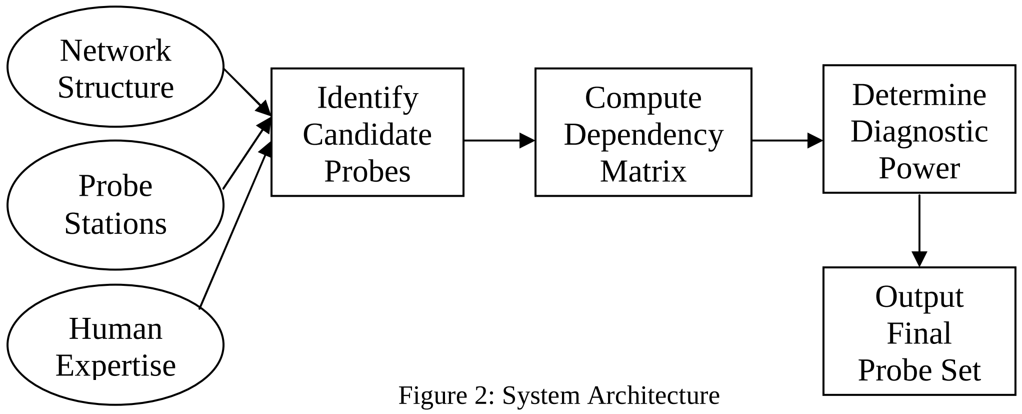

Network Structure
Probe Stations
Human Expertise
Identify Candidate Probes
Compute Dependency Matrix
Determine Diagnostic Power
Output Final Probe Set

Figure 2: System Architecture

## 3. Implementation

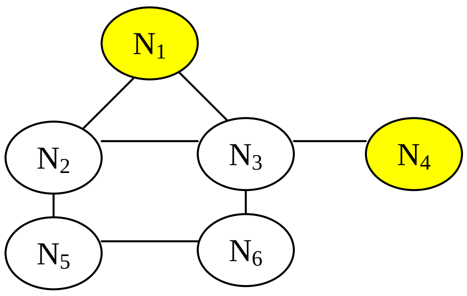
### *3.1 Determining the Initial Probe Set*

It is important to point out that the architecture described above allows the set of candidate probes to be provided from whatever sources are available; for example a human expert may specify which probes are possible. However it may also be useful to compute the available probes from the network structure and the location of the probe stations.

We begin by selecting from the *n* nodes a subset of *k* nodes as the probe stations. In this work we do not address the question of how to select the probe stations, since they usually cannot be chosen to optimize the probing strategy; other considerations, such as gaining access to the machines, may be more important for choosing probe stations.

A probe can be sent to any node from any probe station. Thus the candidate set of probes could theoretically contain a probe for every possible route between every probe station and every node. In practice it cannot be guaranteed that a probe follows a particular path through the network, and thus routing strategies restrict the set of available probes; for example a probe may follow the shortest (i.e. least-cost) path through the network. This creates a candidate set of probes of size $r=O(n)$[1]; note that this set is sufficient to diagnose any single node being down because one can simply use one probe station and send a probe to every node.

As an example, in Figure 3, with $N_1$ and $N_4$ as probe stations, and weights of 1 on each link, the candidate probes in the case of shortest path routing are:



| | $N_1$ | $N_2$ | $N_3$ | $N_4$ | $N_5$ | $N_6$ | $N_7$ |
|---|---|---|---|---|---|---|---|
| $P_{12}$ | 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 |

Figure 3: The Initial Probe Set

### *3.2 Determining the Diagnostic Power of a Set of Probes*

Given a dependency matrix, the decomposition places all problems with the same column into the same group. Thus a naïve approach would compare each column with every other column. We can do better by proceeding row-by-row and computing the decomposition incrementally. The key is that adding a row (i.e. a probe) always results in a more extensive decomposition, because nodes in distinct groups remain distinguishable; an additional probe can only have the effect of distinguishing previously indistinguishable nodes. For example, recall the probe set and decomposition from Figure 1:

$N_1$ $N_2$ $N_3$ $N_4$ $N_5$ $N_6$ $N_7$

[1] To avoid repetitions, probes need only be considered from probe station *i* to probe station *j* if *j>i*. Thus the size of the initial probe set is actually exactly $kn-k(k+1)/2$.

$$\begin{array}{cccccccc} P_{15} & 1 & 1 & 0 & 0 & 1 & 0 & 0 \\ P_{16} & 1 & 0 & 1 & 0 & 0 & 1 & 0 \end{array} \qquad \text{Decomposition}=\{\{1\}, \{2,5\}, \{3,6\}, \{4,7\}\}$$

Suppose we add the probe $N_4$->$N_3$->$N_2$, giving the following dependency matrix:

$$\begin{array}{cccccccc} & N_1 & N_2 & N_3 & N_4 & N_5 & N_6 & N_7 \\ P_{15} & 1 & 1 & 0 & 0 & 1 & 0 & 0 \\ P_{16} & 1 & 0 & 1 & 0 & 0 & 1 & 0 \\ P_{42} & 0 & 1 & 1 & 1 & 0 & 0 & 0 \end{array} \qquad \text{Decomposition}=\{\{1\}, \{2\}, \{3\}, \{4\}, \{5\}, \{6\}, \{7\}\}$$

Since each column is unique, any single node failure among the 6 nodes can be uniquely diagnosed; for example a failure in $N_3$ is the only possible cause of probe $P_{15}$ succeeding and probes $P_{16}$ and $P_{42}$ failing. Note that $P_{42}$ achieved this decomposition by going through exactly one of the nodes in each group of the previous decomposition – it passed through $N_2$ but not $N_5$, through $N_3$ but not $N_6$, through $N_4$ (but not $N_7$ – no probe can pass through $N_7$ because it represents no failure anywhere in the network and doesn't actually exist as a node).

We should point out that if $N_1$ were the only probe station, so that the initial probe set was restricted to only the first five probes given in Figure 3, no set of three probes could diagnose every single node failure. Having $N_4$ available as a second probe station helps to reduce the number of probes needed.

Each additional probe decomposes every group of the current decomposition into two subgroups depending on which nodes the probe passes through. This process is repeated for each probe – each of the nodes remains grouped with precisely those nodes it has not yet been distinguished from. The algorithm, shown in Figure 4, terminates with the complete decomposition after considering each probe only once.

```
Input: Dependency matrix D. Output: Diagnostic Power of the Probe Set

$S_0$ ={1, 2, …, n}
For i=1 to r
    For each set in $S_{i-1}$, create two subsets – one for those nodes j which the i^th probe passes
        through (D(i,j)=1) and the other for those it does not pass through (D(i,j)=0)
    $S_i$ = collection of all nonempty subsets
Output $S_r$
```

Figure 4: Computing the Diagnostic Power of a Probe Set

As an illustrative example, consider the three probes shown above. Let $S_0=\{\{1,2,\ldots,7\}\}$ be the initial decomposition. The first probe, $P_{15}$, passes through nodes $N_1$, $N_2$, and $N_5$, inducing the decomposition $S_1=\{\{1, 2, 5\}, \{3, 4, 6, 7\}\}$. Now consider the second probe, $P_{16}$, which passes through $N_1$, $N_3$ and $N_6$. The next decomposition is computed by traversing each group in $S_1$, creating sub-groups for those nodes which $P_{16}$ does and does not pass through; this yields $S_2=\{\{1\}, \{2,5\}, \{3,6\}, \{4,7\}\}$. Now traverse $S_2$ with the third probe, $P_{42}$, (passing through $N_2$, $N_3$ and $N_4$) yielding $S_3=\{\{1\},\{2\},\{3\},\{4\},\{5\},\{6\},\{7\}\}$. The successive decompositions can be efficiently computed using linked lists.

## *3.3 Finding the Minimal Set of Probes*

We now investigate the question of finding the minimal set of probes that has the same diagnostic power as a given set. For example, we have seen that the initial set of nine probes for the six-node network in Figure 3 has a subset of only three probes that suffices to diagnose any single node being down. Clearly the minimal set of probes may not be unique, although the minimal number of probes is.

In general, one probe station and *n* probes are needed to locate any single down node, because a probe can be sent to every node. However in many situations far fewer probes may suffice. Because *r* probes generate $2^r$ possible signals (one of which corresponds to the case that there is no failure), in the ideal situation only *log(n)+1* probes are needed to locate a single failure in any of *n* nodes. However this is only achievable if all the necessary links exist in the network and it is possible to guarantee that a probe follows a pre-specified path. In the case of shortest-path routing with an arbitrary network structure, the minimal number of probes may lie anywhere between *log(n)+1* and *n*; the exact value depends on the network structure and the location of the probe stations. We expect that the size of the minimal set should decrease as more probe stations are added – this is confirmed in Section 4.

We examine three algorithms for finding the minimal probe set: an exponential time exhaustive search and two approximation algorithms – one requiring linear time and the other quadratic time. Estimation of their computational complexity ignores sub-processes that are common to them all, for example, finding the shortest paths in the network, and determining the diagnostic power of a probe set. An experimental comparison of the algorithms is presented in Section 4.

### *(i) Exhaustive Search*

The minimal set can of course be found by exhaustive search. Not every probe subset of size between *log(n)+1* and *n* need be considered. Since a node can only be diagnosed if there is at least one probe passing through it, probes can be added incrementally in all feasible combinations until the minimal set is reached.

Let Probes(j) denote the set of probes passing through $N_j$, and S x T denote all subsets with one element from S and a distinct element from T. $S_i$=Probes(1) x Probes(2) … Probes(i) is a set of subsets, each of size i, such that the minimal set must be a superset of at least one of them. Hence, if *min* is the true size of the minimal set, then $S_{min}$ contains this set (and all other minimal sets). Thus the minimal set can be found by checking each subset in $S_i$ for successive i until one is found with the same diagnostic power as the original probe set; this occurs when i=*min*.

The exhaustive search algorithm is given in Figure 5. To estimate its computational complexity, note that each node has at least *k* distinct probes through it, so $S_{min}$ contains at least $k^{min}$ subsets which may need to be considered before the algorithm terminates; this gives a computational complexity of $k^n$ in the worst case. This is clearly prohibitive unless the network is quite small.

```
Input: Dependency matrix D. Output: Probe Set of Minimal Size

Compute the diagnostic power DP induced by D
$S_1$=Probes(1)
While no subset in $S_k$ has diagnostic power DP
        $S_{k+1}$=$S_k$ x Probes(k)
Output the subset with diagnostic power DP
```

Figure 5: Exhaustive Search (Exponential Time)

We now consider two approximation algorithms that heuristically attempt to find the minimum set but are not guaranteed to do so.

### *(ii) Subtractive Search*

Subtractive 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 diagnostic power remains the same even if it is dropped from the probe set. This process terminates in a subset with the same diagnostic power 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; this gives a computational complexity of *O(r)*, which is *O(n)* if *r=O(n)*, as in Section 3.1.

```
Input: Dependency matrix D. Output: Probe Set (possibly non-minimal size)

Compute the diagnostic power DP induced by D
S={$P_1$, $P_2$, …, $P_r$} ($P_i$ is the $i^{th}$ row of D)
For i=1 to r
    If S\{$P_i$} has diagnostic power DP, then S= S\{$P_i$}
Output S
```

Figure 6: Subtractive Search (Linear Time)

The order of the initial probe set is quite important for the performance of this algorithm. If the probes are ordered by probe station, the algorithm will remove all the probes until the last *n* (all of which are from the last probe station), since these suffice to diagnose any node. This reduces the opportunity of exploiting probes from different probe stations. The size of the probe set can be reduced by randomly ordering the initial probe set, or ordering it by target node.

### *(iii) Additive (Greedy) Search*

Another approach is a greedy search algorithm where at each step we add the probe that results in the "most informative" decomposition. For example, suppose probe set $P_1$ induces the decomposition $S_1=\{\{1,2\},\{3,4\}\}$ while probe set $P_2$ induces the decomposition $S_2=\{\{1\},\{2,3,4\}\}$. Although $P_2$ can uniquely diagnose one of the nodes and $P_1$ cannot, it is possible to add just a single probe to $P_1$ and thereby diagnose all the nodes, whereas at least two additional probes must be added to $P_2$ before all the nodes can be diagnosed. Therefore $S_1$ is a more "informative" decomposition than $S_2$.

A decomposition S is a collection of groups of indistinguishable nodes with $n_i$ nodes in the $i^{th}$ group. If a node is in the $i^{th}$ group, at least $log(n_i)$ additional probes are needed to uniquely diagnose the node. Since a random node lies in the $i^{th}$ group with probability $n_i/n$, the average number of additional probes needed to uniquely diagnose all the nodes is $H(S)=\Sigma_i(n_i/n)log(n_i)$ and measures the "uncertainty" induced by the decomposition S. (In information theory terms, if X={1, ..., n} is the variable denoting the node, and D={1, ..., k} is the variable denoting which group contains the node, then H(S) is the conditional entropy H(X|D) [1]. This assumes that failures are equally likely in any node. If this is not the case prior knowledge about the likelihood of different types of failures can be incorporated into the measure.)

The additive algorithm is shown in Figure 7. It starts with the empty set and repeatedly adds the probe which gives the most informative decomposition. This algorithm also finds a probe set with the same diagnostic power as the original set but which is not necessarily minimal. The running time of this algorithm is quadratic in *r*, the size of the original probe set, because at each step the information content of the decomposition induced by each of the remaining probes must be computed. This gives a computational complexity of $O(n^2)$ if *r=O(n)*, as in Section 3.1.

```
Input: Dependency matrix D. Output: Probe Set (possibly non-minimal size)

Compute the diagnostic power DP induced by D; S=∅;
While S does not have diagnostic power DP
    For each probe P not in S compute H(decomposition induced by S∪P)
    S=S∪$P_m$, where $P_m$ is the probe with minimum H value
Output S
```

Figure 7: Additive Search (Quadratic Time)

## 4. Experiments

This section investigates experimentally both the general behavior of the minimum set size and how the two approximation algorithms compare with exhaustive search in computing the probe set. The main result is that the approximation 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.

### *4.1 Setup*

For each network size *n*, we generate a network with *n* nodes by randomly connecting each node to four other nodes. Each link is then given a randomly generated weight, to reflect network load. The probe stations are selected randomly. One probe is generated from each probe station to every node using shortest-path routing. The three algorithms described in Section 3.3 are then implemented. This process is repeated ten times for each network size and the results averaged.

### *4.2 Results*

#### *(i) Probe Set Size*

Figure 8 shows the case of three probe stations. The size of the probe set found by all the algorithms lies between *log(n)+1* and *n*, as expected. The minimal size is always larger than the theoretical lower bound of *log(n)+1*, for two reasons: (1) The networks are not very dense; since each node is linked to four other nodes, the number of edges increases only linearly with network size. Thus many probe paths are simply not possible. (2) 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.

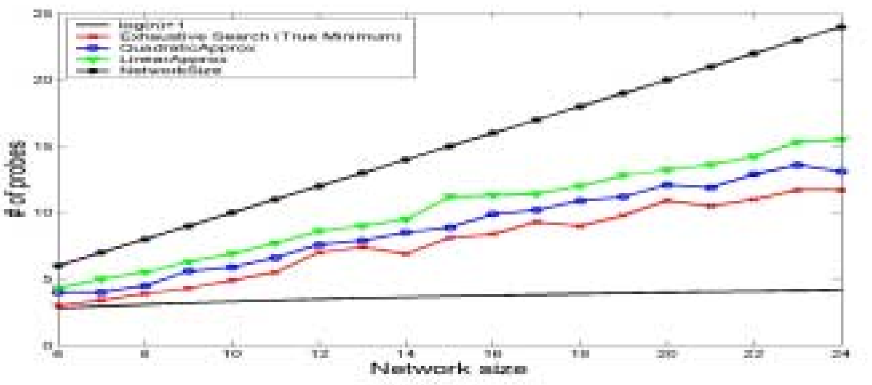


Figure 8: Three Algorithms for Computing Probe Sets

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 9 illustrates the performance of these algorithms on larger networks for which exhaustive search is not feasible. The quadratic-time algorithm slightly outperforms the linear-time algorithm, but its 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.

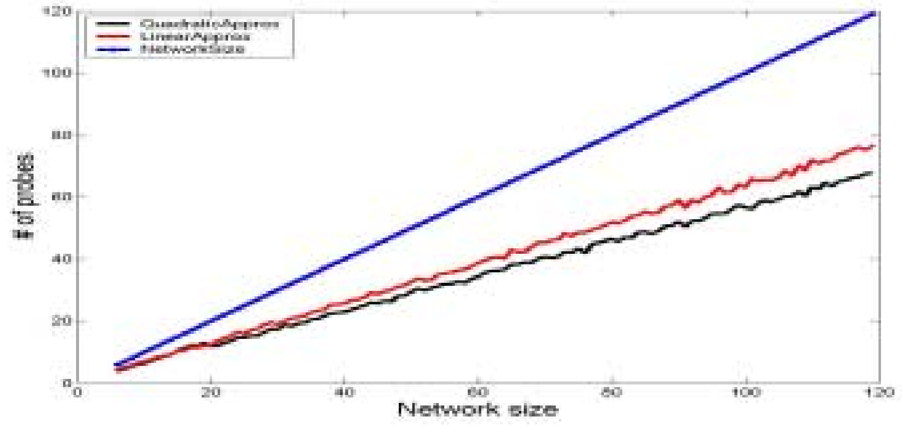


Figure 9: Approximation Algorithms - Large Networks

*(ii) Number of Probe Stations*

Although it is sufficient to have just one probe station, the interactions between probe paths increase if probe stations are added, and so the minimal probe set size decreases. Figure 10 shows the average true minimum set size for one, two, and three randomly placed probe stations. This confirms that adding probe stations reduces the network load imposed by probing. However additional probe stations can be quite expensive, and the process may soon reach a point of diminishing returns where the cost of an additional probe station exceeds the benefit gained by reducing the size of the probe set.

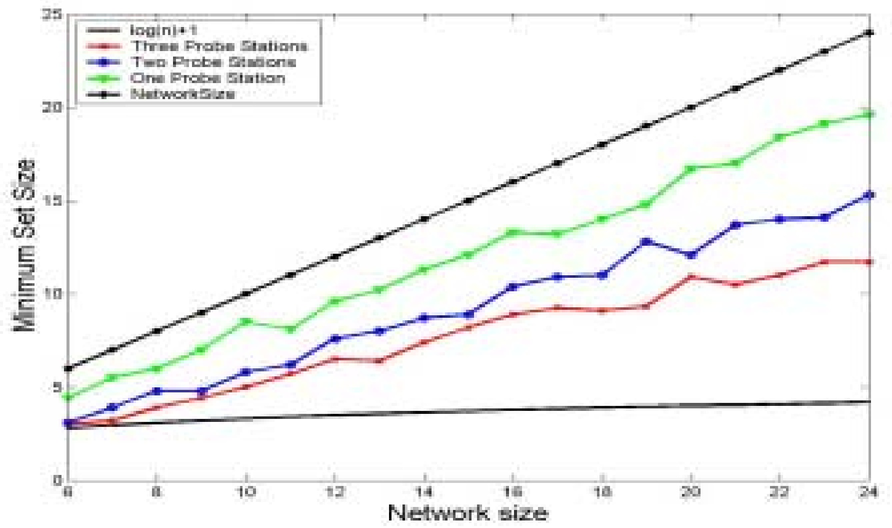


Figure 10: Increasing the number of Probe Stations decreases the number of Probes needed

## 5. Related Work

The formulation of problem diagnosis as a "decoding" problem, where "problem events" are decoded from "symptom events", was first proposed by [8]. In our framework, the result of a probe constitutes a "symptom event", while a node 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 [8] 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 [8] lacks a detailed discussion of efficient algorithms for constructing optimal codebooks; they mention only a greedy pruning algorithm. For more detail on event correlation see also [9] and [3].

Other approaches to fault diagnosis in communication networks and distributed computer systems have been presented during the past decade. For example, Bayesian networks [5] and other probabilistic dependency models [6] can be used to encode dependencies between network objects; another approach is statistical learning to detect deviations from the normal behavior of the network [4].

The approach in [6] uses a graph model in which the prior and conditional probabilities of node failures are given and the objective is to find the most likely explanation of a collection of alarms. It is shown that the problem is NP-hard and a polynomial-time approximation algorithm is given; the performance of this algorithm can be improved by assuming that the probabilities of node failure are independent of one another. Using the dependency matrix formulation enables us to take a more straightforward approach which does not require searching the space of possible explanations for a set of alarms. Using this approach probe sets can be found which are constructed to locate precisely those problems one is interested in detecting.

A decision-theoretic approach using Bayesian networks is presented in [5]. The goal is to find the minimum-cost diagnosis of problems occurring in a network. Dependencies between a problem and its possible causes and symptoms are represented using Bayesian networks, which are manually constructed for each problem, and probabilities are assigned using expert knowledge. The goal is to minimize the total cost of tests needed to diagnose a fault; a single fault at a time is assumed. This approach may become intractable in large networks due to the NP-hardness of inference in Bayesian networks; also, considering more than one fault at a time leads to an exponential increase in complexity. Therefore, approximation methods as proposed in this paper will be needed in practical applications that involve a large number of dependent components.

## 6. Extensions

### *6.1 Multiple Simultaneous Failures*

In this section the assumption that only one node in the network may fail at any given time is relaxed. The general approach remains the same; for each potential problem there is a column in the dependency matrix that represents the probe signal resulting from the occurrence of that problem. These columns can be computed directly from the columns for single nodes. Given a candidate set of probes, the optimal subset or approximations of it can be found as before.

For example consider again the three probes from section 3.2:

$$\begin{array}{cccccccc} & N_1 & N_2 & N_3 & N_4 & N_5 & N_6 & N_7 \\ P_{16} & 1 & 1 & 0 & 0 & 1 & 0 & 0 \\ P_{15} & 1 & 0 & 1 & 0 & 0 & 1 & 0 \\ P_{42} & 0 & 1 & 1 & 1 & 0 & 0 & 0 \end{array}$$

Suppose we would also like to diagnose the simultaneous failure of nodes $N_1$ and $N_4$. We simply add a new column to the dependency matrix, denoted by $N_{14}$, which is easily computed by "Or"-ing together the columns for the individual nodes, since a probe might pass through either member of a pair. This gives the dependency matrix shown below:

|  | $N_1$ | $N_2$ | $N_3$ | $N_4$ | $N_5$ | $N_6$ | $N_{14}$ | $N_7$ |
|---|---|---|---|---|---|---|---|---|
| $P_{16}$ | 1 | 1 | 0 | 0 | 1 | 0 | 1 | 0 |
| $P_{15}$ | 1 | 0 | 1 | 0 | 0 | 1 | 1 | 0 |
| $P_{42}$ | 0 | 1 | 1 | 1 | 0 | 0 | 1 | 0 |

Since each column is unique, all the problems can be diagnosed. This procedure can be generalized to detect the simultaneous failure of any subset of nodes.

Clearly the number of probe stations and candidate probes may need to be increased to allow any possible problem to be diagnosed. For example, to diagnose the simultaneous failure of every possible pair of nodes will generally require a number of probes quadratic in the number of nodes, every triple of nodes will require a cubic number of probes, and so on. If any possible subset of nodes may fail simultaneously the number of candidate probes will be exponential in the number of nodes; this is a consequence of the inherent complexity of the task of fault localization. Whatever the candidate probe set is, the algorithms described above can be used to find smaller probe sets with the same diagnostic power as the given set.

### *6.2 Dealing with Uncertainty*

In this paper we have assumed that the probe signal is received correctly; no data in the network is lost or spuriously altered. If network errors are possible then we require that the distance between probe signals (the codebook "radius" in the terminology of [8]) is larger than a single bit, thereby providing robustness to noise and lost packets. Dynamic network routing is another source of uncertainty, since the path probes through the network may not be known accurately. Other changes to the network may occur; for example nodes and links are continually being added, removed, and reconfigured. For these reasons the dependency matrix may need to be regularly updated. Although our initial results are promising, much work remains to be done to extend the technique to deal with the uncertainties and complex changes that occur in real networks.

### *6.3 Adaptive Probing*

Finally, we should note that in this work we have not considered **adaptive** probing, where the decision of which probes to send depends on the result of earlier probes; instead we have treated probe scheduling as a planning step where the entire probe set is selected before the probes are sent. Adaptive probing creates the possibility of using an intelligent probing strategy that adjusts the probe set dynamically in response to the state of the network. This adaptive probing scenario is an additional challenge for future work.

## 7. Conclusion

Using probing technology for the purposes of fault localization requires that the number of probes be kept small, in order to avoid excessive increases in network load. In this paper we have proposed a framework in which this can be done by exploiting interactions among the paths traversed by the probes. However, finding the smallest number of probes that can diagnose a particular set of problems is computationally expensive for large networks. In this paper we have shown that approximation algorithms can be used to find small probe sets that are very close to the optimal size and still suffice for problem diagnosis. These approximation algorithms enable system managers to select their own trade-off between the computational cost and increased network load required by effective fault localization. Probing provides a flexible approach to fault localization because of the control that can be exercised in the process of probe selection.

## 8. References

[1] T. M. Cover and J. A. Thomas. Elements of Information Theory, New York, John Wiley & Sons, 1991.

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

[3] B. Gruschke. Integrated Event Management: Event Correlation Using Dependency Graphs, DSOM 1998.

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

[5] J-F. 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.

[6] I.Katzela and M.Schwartz. Fault Identification Schemes in Communication Networks, IEEE/ACM Transactions on Networking, 1995.

[7] "Using Keynote Measurements to Evaluate Content Delivery Networks", available at http://www.keynote.com/services/html/product_lib.html

[8] S. Kliger, S. Yemini, Y. Yemini, D. Ohsie, S. Stolfo. A Coding Approach to Event Correlation, IM 1997.

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