RC22817 (W0306-053) June 11, 2003
Computer Science

IBM Research Report

Active Probing

Mark A. Brodie, Irina Rish, Sheng Ma, Genady Grabarnik, Natalia V. Odintsova
IBM Research Division
Thomas J. Watson Research Center
P.O. Box 218
Yorktown Heights, NY 10598

Research Division
Almaden - Austin - Beijing - Delhi - Haifa - India - T. J. Watson - Tokyo - Zurich

Active Probing

Mark Brodie, Irina Rish, Sheng Ma, Genady Grabarnik, Natalia Odintsova
I.B.M. T.J. Watson Research
(mbrodie, rish, shengma, genady)@us.ibm.com, nodintsova@hotmail.com

Abstract

Problem determination is one of the most important tasks in managing distributed systems. Probing (both at the transaction and network levels) has been widely used for assessing compliance with Service Level Agreements and locating problems in distributed systems. However a probing scheme which uses a fixed set of regularly scheduled probes can be expensive in terms of the number of synthetic transactions needed, especially for the task of problem determination. This paper introduces an active probing scheme to reduce the number of probes needed. Our key idea is to divide the problem determination task into two steps. We first use a relatively small number of fixed, regularly scheduled, probes for detecting that a problem has occurred. In the second phase, once occurrence of a problem is detected, additional probes are issued on-the-fly to acquire additional information until the problem is localized. We develop algorithms for selecting an optimal set of probes for problem detection and choosing which probes to send next based on what is currently known. We demonstrate through both analysis and simulation that the active probing scheme can greatly reduce the number of probes and the time needed for localizing the problem when compared with a non-active probing scheme.

1 Introduction

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* ([13, 7, 12]), 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 test network and system components in order to provide more accurate and cost-efficient problem determination.

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. Previous work has focussed on selecting probes in an off-line, pre-planned fashion. Prior information about network and system dependencies, which problems need to be detected most urgently (perhaps because they are more important or more likely to occur), and other forms of prior knowledge are used to construct a set of probes that is small (thereby reducing probing costs such as data storage requirements and network load) yet provides extensive coverage so that problems can be diagnosed. These probes are scheduled to run periodically to provide information about what problems may be occurring. A typical example is IBM's EPP technology ([5]).

Using pre-planned probe sets suffers from considerable limitations. Because the probe set is computed off-line, it needs to be able to diagnose all possible problems which might occur. However in practice constructing such a probe set can be quite difficult, because one must envisage all problems one would like to be able to diagnose and construct probes for them. A pre-planned probe set may also be very wasteful, because many problems that might occur do not in fact ever happen. Probes to detect such problems enlarge the probe set unnecessarily, but knowing which probes can be safely omitted can usually only be determined on-line by monitoring which problems in fact occur.

Another disadvantage of pre-planned probe sets is that because the probes run periodically at regularly scheduled intervals, there may be a considerable delay in obtaining information when a problem occurs. It is clearly desirable to detect the occurrence of a problem as quickly as possible. Furthermore, once the occurrence of a problem has been detected, additional information may be needed to diagnose the problem precisely. This information may not be obtainable from the results of the pre-planned probes - additional probes may need to be sent to obtain it. These probes should be appropriately selected "on-demand", based on the results of the previous probes.

Our works develops a methodology called **active probing** that addresses these limitations. This involves probing in an interactive mode, where probe results are analyzed to determine the most likely diagnosis, and then additional probes are selected and sent in order to gain further information. This process may repeat - once additional probe results are obtained, the diagnosis is refined, and, if necessary, more probes are selected, and so on, until the problem is completely determined. The idea of this approach is to "ask the right questions at the right time".

Active probing selects and sends probes as needed in response to problems that actually occur. It therefore avoids both the difficulty of constructing probes for all possible problems as well as the waste inherent in using probes for problems that in fact never occur. Furthermore, because probes are selected on-line to obtain further information about particular problems that have occurred, they need not circulate regularly throughout the entire network; instead they can be targeted quickly and directly to the points of interest. Thus fewer probes are needed than in a pre-planned approach, allowing for a considerable reduction in probing costs.

Implementing active probing requires developing solutions for the following issues:

1. A **small** probe set must be pre-selected, so that when a problem occurs we can detect that something has gone wrong.

2. The probe results must be integrated and analyzed, to determine the most-likely network state.

3. The "most-informative" probes to send next must be selected, based on the analysis of previous probe results.

4. This process must be repeated until the problem diagnosis task is complete.

In this paper we describe efficient solutions for each of these issues and integrate them into a practical system for active probing. We also analyze the costs and benefits of active probing and show experimentally that active probing substantially reduces the number of probes needed to diagnose problems when compared with pre-planned probing. Our preliminary results suggest that active probing may be a powerful and effective technique for problem determination.

2 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 ([5]), 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 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.

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.

Figure 1 illustrates the core ideas of probing technology. The bottom left of the picture represents an external cloud (e.g. the Internet), while the greyed box in the bottom middle and right represents an example intranet - e.g. a web site hosting system containing a firewall, routers, web server, application server running on a couple of load

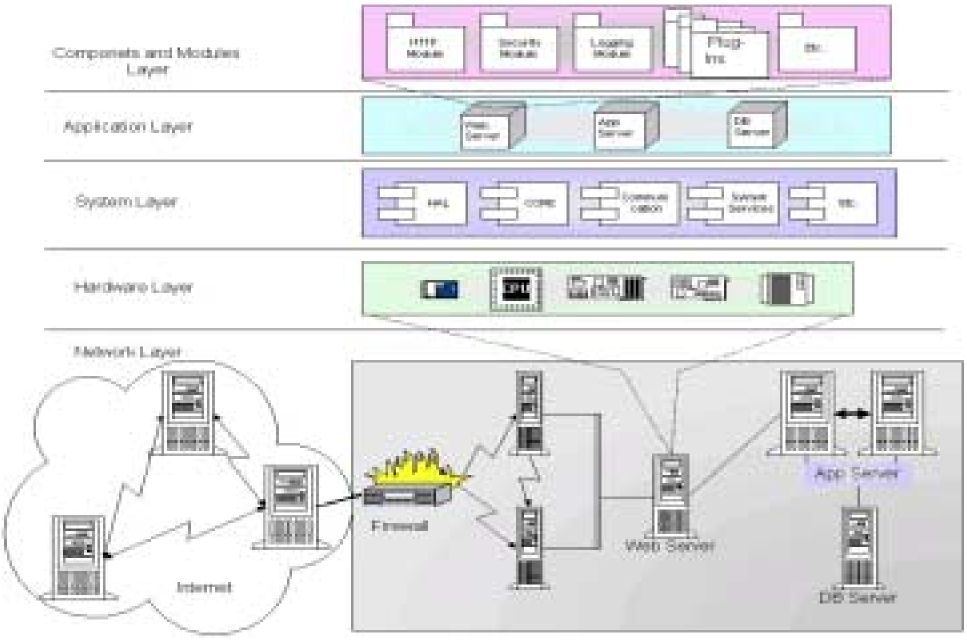


Figure 1: Illustrative Example where Probing can be used at Multiple Levels.

balanced boxes, and database server. Each of these contains further substructure - the figure illustrates the various layers underlying the components.

Probing can take place at multiple levels of granularity; the appropriate choice of granularity depends on the task probing is used for. For example, to test a Service Level Agreement stating response time one need only probe one point of Figure 1, the point of contact of the external cloud and the intranet. In order to find more detailed information about the system one could probe all network segments as well the web server, application server, database server - all the elements of the intranet in the network layer. If we need to do problem determination or tune up the system for better performance, we may also need to consider more detailed information; e.g. from the system layer (some systems allow instrumentation to get precise information about system components) or component and modules layer, and so on. For each task appropriate probes must be selected and sent and the results analyzed.

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.

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.

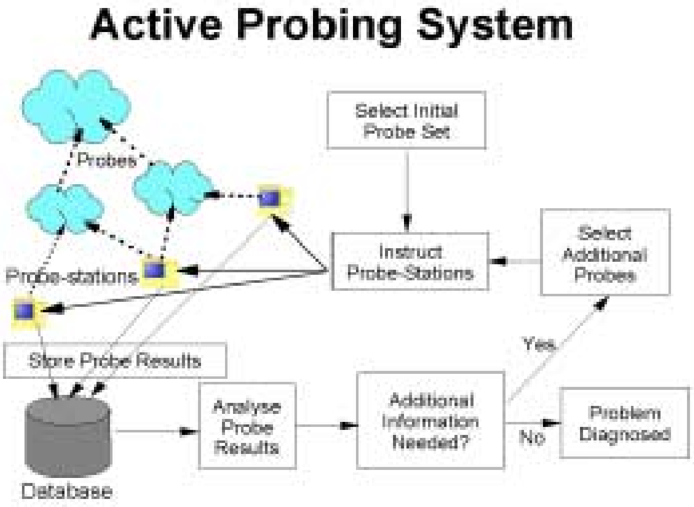


Figure 2: Active Probing System.

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 probing effectively.

3 Active Probing

In this work, we discuss using probing technology for the purpose of problem determination. An initial set of probes is selected off-line to run periodically through the network, for the purpose of **detecting** when a problem occurs. Whenever occurrence of a problem is detected, additional probes are selected on-line and 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**.

An active probing system is outlined in Figure 2. Probe-stations issue probes which traverse different parts of the network. The results of the probes are stored in a database. These results are analyzed by an inference engine that infers what problems might be occurring in the network. If additional information is needed in order to locate the problem more precisely, the analysis engine determines what would be the most useful probes to send. Instructions are then issued to the probe-stations. Once additional results are received, further inferences can be drawn, and the process repeats until enough information has been gained to completely determine the problem.

We now describe each of the following steps: (1) Selecting the initial set of probes to detect problems; (2) Analyzing the probe results to determine the most-likely state of the network; (3) Determining on-line the most useful probes to send next to gain additional information. We then quantify the advantages of an active probing methodology when compared with an entirely pre-planned approach, and present experimental results that show that active probing can greatly the number of probes and the time needed to perform problem determination.

3.1 Selecting the Initial Probes

Here we briefly summarize our previous work on off-line selection of a pre-planned probe set. For more details see [1].

The relationships between the probes that are available to be used and the problems that need to be detected can be described using a *dependency matrix*, where the probes are the rows, the problems are the columns, and a matrix entry is nonzero precisely when the corresponding probe tests the occurrence of the corresponding problem. Thus each row (probe) may have multiple nonzero entries, corresponding to the components it tests, and each problem (column) may have multiple nonzero entries, corresponding to the probes that test it.

The task of **problem detection** is to find the smallest set of probes such that, no matter which problem occurs, there is some probe that will fail; i.e. will detect that a problem has occurred somewhere. Using the dependency matrix formulation, this corresponds to finding the smallest set of rows such that each column has a nonzero entry.

The task of problem detection should be distinguished from the task of **problem determination**, which requires not simply detecting that a problem has occurred, but also identifying, from the results of the probes, precisely which problem has occurred. In the dependency matrix formulation this requires finding the smallest set of rows such that every column is unique.

The tasks of finding the smallest probe sets for both problem detection and problem determination can be shown to be NP-hard ([6]), but approximation algorithms perform well in practice, finding near-optimal probe sets in polynomial time. Thus a small set of probes can be initially selected which provides wide coverage throughout the network to detect when problems occur (see [1] for further details).

3.2 Analyzing Probe Results

The initial set of probes is computed and scheduled off-line and runs periodically throughout the network. When a problem occurs, the probe results must be analyzed in order to detect which problem(s) has most likely occurred. If further information is needed, additional probes must be selected and sent.

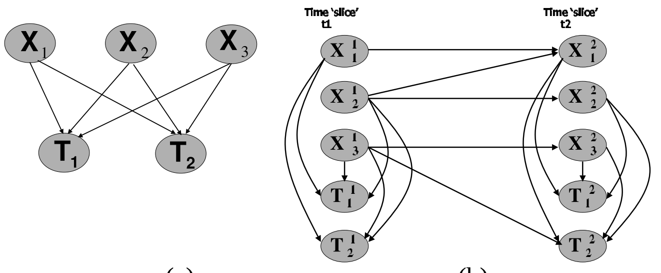
In a deterministic environment analyzing the probe results is straight-forward: If a probe fails, then at least one of the elements it tests has a problem, while if a probe succeeds then all the elements it tests are OK. By examining the results of the different probes and reasoning about the interactions among their paths inferences can be drawn about which problems have most likely occurred.

More precisely, diagnosis can be formulated as a constrained optimization problem, as follows. Let X_i denote the current state of a network component (1 means “OK”, 0 means 'failed'). The states of n network elements are denoted by a vector $\mathbf{X} = (X_1, ..., X_n)$ of *unobserved* Boolean variables. Let T_j denote the j-th probe. A vector $\mathbf{T} = (T_1, ..., T_m)$ of *observed* Boolean variables denotes the outcomes (0 - failure, 1 - OK) of m probes. Lower-case letters, such as x_i and t_j, denote the values of the corresponding variables, i.e. $\mathbf{x} = (x_1, ..., x_n)$ denotes a particular assignment of n node states, and $\mathbf{t} = (t_1, .., t_m)$ denotes a particular outcome of m probes.

Each probe outcome $T_i = t_i$ imposes a logical-AND constraint $t_i = x_{i_1} \wedge ... \wedge x_{i_k}$ on the values of its parent nodes $X_{i_1}, ..., X_{i_k}$. Diagnosis can be viewed as the problem of finding the *most-likely* assignment to all network components given the probe outcomes. Assume a prior probability of fault $p_i = P(X_i = 0)$ for every node. Then we wish to find a vector $\mathbf{x}^* = \arg\max_{x_1,...,x_n} \prod_{j=1}^{n} p_j$ subject to those constraints imposed by observed probes. The problem can also be cast as constraint satisfaction rather than optimization if there exists a unique solution satisfying the constraints.

Although constrained optimization and constraint satisfaction problems are generally NP-hard, it is interesting to note that for uniform priors the probing domain yields a tractable set of constraints which can be solved in linear time (see [19] for more details). However, if arbitrary fault priors are allowed, the problem becomes NP-complete and thus intractable for large networks. Still, if the fault probability is small enough (usually it is much lower than 0.5), we can make simplifying assumptions such as no more than i simultaneous faults. Then it can be shown that diagnosis will be only exponential in i (e.g., linear for single-fault assumption).

Note that until now we assumed no noise in the probe outcomes. However in a realistic scenario analysis of probe results must take into account an environment of noise and uncertainty. For example, a probe can fail even though all the nodes it goes through are OK (e.g., due to packet loss). Conversely, there is a chance that a probe succeeds even if a node on its path has failed (e.g., dynamic routing may result in the probe following a different path). Thus a more sophisticated inference mechanism is needed to determine the *most likely* configuration of the states of the network elements.



(a) (b)

Figure 3: (a) A two-layer Bayesian network structure for a set $\mathbf{X} = (X_1, X_2, X_3)$ of network elements and a set of probes $\mathbf{T} = (T_1, T_2)$, and (b) its extension to a Dynamic Bayesian Network.

We use the graphical framework of Bayesian networks [16] that provides both a compact factorized representation for multivariate probabilistic distributions as well as a convenient tool for probabilistic inference. An example of a simple Bayesian network (BN) for problem diagnosis is shown in Figure 3a: a is a bipartite (two-layer) graph where the top-layer nodes represent marginally independent faults or other problems (if the problems are not marginally independent, appropriate edges must be added between them) and the bottom-layer nodes represent probe results. The network represents a joint probability $P(\mathbf{x}, \mathbf{t}) = \prod_{i=1}^{n} P(x_i) \prod_{j=1}^{m} P(t_j|\mathbf{pa}(t_j))$, where $P(t_j|\mathbf{pa}(t_j))$ is the *conditional probability distribution (CPD)* of node T_i given the set of its *parents* $\mathbf{pa_i}$, i.e. the nodes pointing to T_i in the directed graph, and $P(x_i)$ is the prior probability that $X_i = x_i$. Formally, a Bayesian network BN over a set of variables $X_1, ..., X_k$ is a tuple (G, P) where G is the directed acyclic graph encoding the independence assumptions of the joint distribution $P(\mathbf{X})$, and $P = \{P(x_i|\mathbf{pa}(\mathbf{x_i}))\}$ is the set of all CPDs.

In general, a CPD defined on binary variables is represented as a k-dimensional table where $k = |\mathbf{pa}(t_j)|$. Thus, just the specification complexity is $O(2^k)$ which is very inefficient, if not intractable, in large networks with long probe paths (i.e. large parent sets). It seems reasonable to assume that each element on the probe's path affects the probe's outcome independently, so that there is no need to specify the probability of T_i for all possible value combinations of $X_{i_1}, ..., X_{i_k}$ (the assumption known as *causal independence*). For example, in the absence of uncertainty, a probe fails if and only if at least one node on its path fails, i.e. $T_i = X_{i_1} \wedge ... \wedge X_{i_k}$, where $\wedge$ denotes logical AND, and $X_{i_1}, ..., X_{i_k}$ are all the nodes probe T_i goes through; therefore, once it is known that some $X_{i_j} = 0$, the probe fails independently of the values of other components.

In practice, however, this relationship may be disturbed by "noise". For example, a probe can fail even though all nodes it goes through are OK (e.g., if network performance degradation leads to high response times interpreted as a failure). Vice versa, there is a chance that the probe succeeds even if a node on its path has failed, e.g. due to routing changes. Such uncertainties yield a *noisy-AND* model which implies that several causes (e.g., node failures) contribute independently to a common effect (probe failure), and is formally defined as follows:

$$P(t = 1|x_1, \ldots, x_k) = (1 - l) \prod_{x_i = 0}^{n} q_i, \text{ and} \tag{1}$$

$$P(t = 1|x_1 = 1, ..., x_k = 1) = 1 - l, \tag{2}$$

where l is the *leak probability* which accounts for the cases of a probe failing even when all the nodes on its path are OK, and the *link probabilities* q_i account for the second kind of "noise", namely, cases when a probe succeeds with a small probability q_i even if node X_i on its path has failed.

Once a Bayesian network is specified, the diagnosis task can be formulated as finding the *maximum probable explanation (MPE)*, i.e. a most-likely assignment to all X_i nodes given the probe outcomes, i.e. $\mathbf{x}^* = \arg\max_{\mathbf{x}} P(\mathbf{x}|\mathbf{t})$. Since $P(\mathbf{x}|\mathbf{t}) = \frac{P(\mathbf{x},\mathbf{t})}{P(\mathbf{t})}$, where $P(\mathbf{t})$ does not depend on $\mathbf{x}$, we get $\mathbf{x}^* = \arg\max_{\mathbf{x}} P(\mathbf{x}, \mathbf{t})$ (see [19] for our approach on efficient algorithms for solving this problem).

In order to represent temporal dependencies, Bayesian network can be extended to a k-slice *Dynamic Bayesian Network* where each time-slice contains a copy of the above

BN, and inter-slice dependencies are encoded by transition probabilities, as shown in Figure 3b (see [18] for details).

3.3 Selecting Additional Probes

At each stage additional probes must be selected based on the results of the previous probes. For each probe, one can compute:

1. The likelihood that the probe will succeed or fail, which depends on inferences drawn about the probability of different network states;

2. The additional information about the network that will result from sending that probe and receiving a successful or failed result.

Using this one can compute the **expected information gain** of each probe and select the probe to send next to maximize the expected information gain. For example, if a particular probe, assuming it were successful, would provide considerable information about the location of the problem, but the likelihood of that probe being successful was very low, then that probe might have a relatively low expected information gain.

This approach can be generalized directly to consider sending many probes simultaneously. In practice the number of probes sent simultaneously in active probing mode will usually be quite small.

Once the probes to send next are selected, they are sent, their results obtained, further inferences about the network state are made, and additional probes, if necessary, are selected and sent. This process continues until the problem determination is complete.

4 Probing Scenarios

The active probing approach can be compared with a different approach in which the entire set of probes is pre-selected in such a way that, no matter what problem occurs in the network, the nature of the problem can be determined by analyzing the probe results without the necessity of sending additional probes. It is clear that this may require an inordinately large number of probes. Active probing, which only issues probes "on-demand", should allow fewer probes to be used. However the system requirements for active probing are somewhat more complex because of its interactive nature.

These two probing strategies are illustrated in Figure 4. 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 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.

4.1 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 4) to

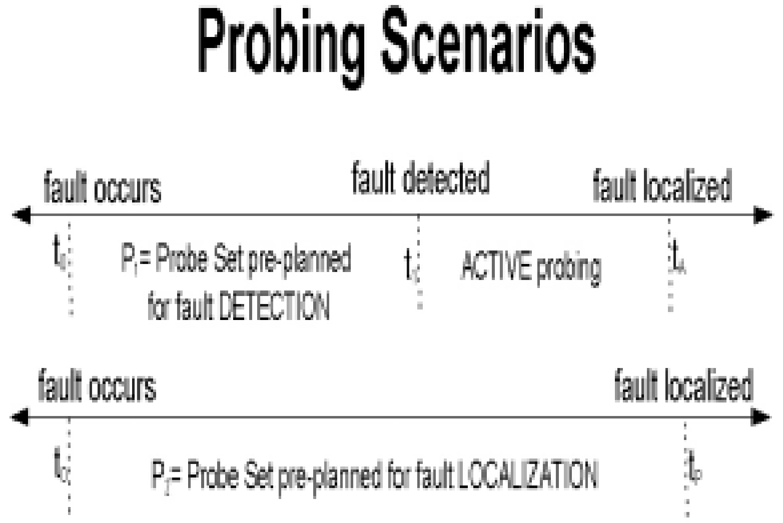


Figure 4: Probing Scenarios.

fault localization (t_P in Figure 4) will be larger than the time from fault occurrence to detecting that a fault has occurred somewhere (t_1 in Figure 4). 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 simple 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 present experimental results that show that n_A is small when compared with n_P.

5 Experiments

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. Three 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.

Exhaustive search is performed to find the true minimum size probe set for "passive probing", i.e. the smallest set of probes whose results can be used to determine unambiguously exactly which node has failed. Since this exhaustive search requires exponential time and is therefore prohibitively expensive for large networks, a linear-time quick search algorithm and a quadratic-time greedy search algorithm are also used to find near-optimal probe sets.

In active probing mode, 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.

5.1 Results

For each network, each node was sequentially selected to be the faulty one. The number of probes required to diagnose the fault was averaged over all networks of a given size.

The results presented in Figure 5 (small networks) and Figure 6 (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 and the time needed to successfully diagnose faults.

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.

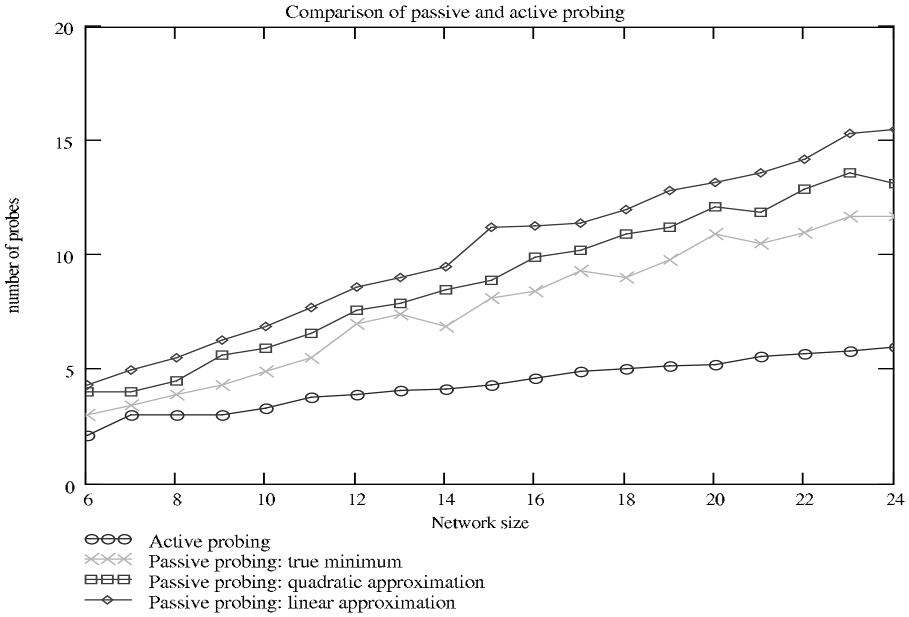


Figure 5: Active Probing - small networks.

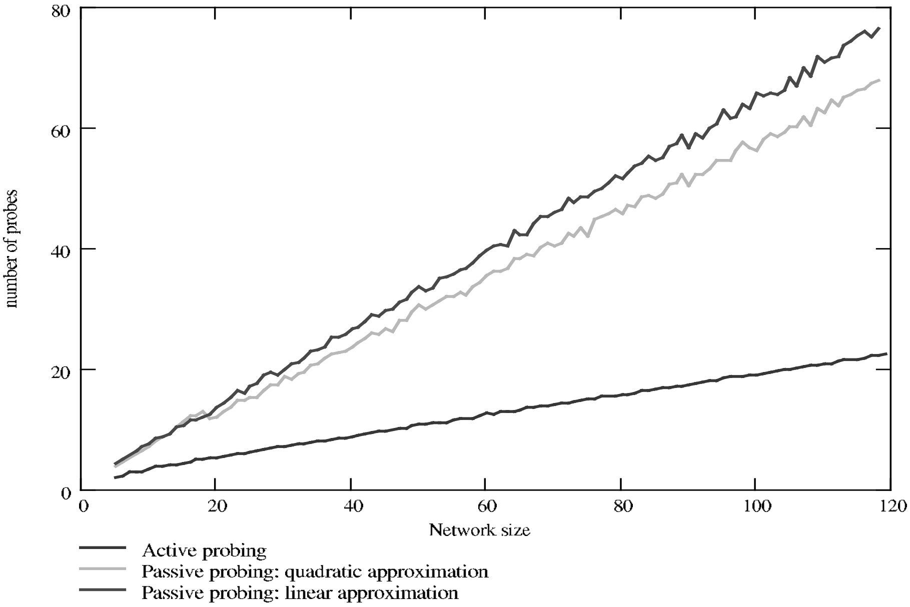


Figure 6: Active Probing - large networks.

6 Related Work

Our previous work [1] and independently Ozmutlu et al. [14] studied the probe selection problem for the purpose of network management. Extending the previous work, this paper 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 ([13, 7]) 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 [12]. 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 [12] 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 [12] 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 [17] and [20]. Since that time a large body of literature has developed [2]. 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 [9] and other probabilistic dependency models [10]; another approach is statistical learning to detect deviations from the normal behavior of the network [8]. The probabilistic diagnosis issues are addressed in our related work in [19], 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. [3] and Ji et al. [11] recently developed a framework for estimating the performance of a multi-cast network based on probes. Duffield et al. [4] and Paxson [15] 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.

7 Conclusion

In this paper we propose an active probing scheme for the task of problem determination. The key idea is to divide problem determination into two steps: detecting occurrence of a problem and then actively probing in order to localize the problem. The first step is to detect whether there is a problem; once occurrence of a problem is detected, the second step repeatedly issues additional probes based on the results of previous probes until sufficient information to localize the problem has been obtained.

We have developed algorithms for selecting a small set of probes for detecting occurrence of a problem and actively choosing the optimal probes to send next based on the information available. Experimental simulation shows that this scheme can dramatically reduce the total number of probes needed, and the time required, to localize problems when compared with a "passive" scheme that uses a fixed probe set.

The next steps of this work include developing a real-time diagnosis algorithm which takes into account dynamically changing network state and intermittent occurrence of faults, as well as testing our prototype on a real production system. Although much remains to be done, the idea of active probing offers exciting possibilities and undoubted potential for problem determination in particular and distributed system management in general.

References

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

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

[3] 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.

[4] 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.

[5] 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.

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

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

[9] 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.

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

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

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

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

[14] 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.

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

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

[17] 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.

[18] I. Rish, M. Brodie, and S. Ma. Accuracy versus efficiency in probabilistic diagnosis. Technical report, IBM T.J. Watson Research Center, 2002.

[19] 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.

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