Probe Station Placement for Fault Diagnosis [1]

Maitreya Natu and Adarshpal S. Sethi
Dept. of Computer and Information Sciences
University of Delaware, Newark, DE, USA, 19716
Email: {natu, sethi}@cis.udel.edu

***Abstract*— Probe stations are specially instrumented nodes from where probes can be sent to monitor the network. Probe station locations affect probing efficiency, monitoring capability, and deployment cost. We present probe station selection algorithms and aim to minimize the number of probe stations and make the monitoring robust against failures in a deterministic as well as a non-deterministic environment. We provide experimental evaluation of the proposed algorithms through simulation results.**

***Index Terms*— Adaptive probing, Probe station selection, Fault diagnosis, Network monitoring, Probabilistic dependency model.**

I. Introduction

Probing provides an effective tool for monitoring the network for fault diagnosis. Probing tools monitor the network by sending probes over the network and analyzing various network parameters from information obtained from the probes. An important problem to address while deploying probing solutions is to identify node locations from where probes can be sent and evaluated. We refer to such nodes as the probe stations. The location of these probe stations should be selected such that probes can be sent to all the network components of interest. Presence of failures in the network can make certain network components unreachable from some part of the network. The probe station placement needs to be made robust to such failures. The number of probe stations should also be optimized for cost-effectiveness. In this paper, we present algorithms to select suitable node locations to deploy probe stations. We discuss various failures scenarios and present probe station selection algorithms to provide robustness against them.

Probe station selection algorithms use the information about the routes used by the probes to reach various network components. This information is captured in a dependency matrix and then used to identify appropriate node locations to deploy probe stations. We first use a deterministic dependency model assuming complete knowledge of the network routes. However, in many scenarios there might be uncertainties present in the obtained information about the network. The probe station selection problem becomes even more challenging when this information is not complete or accurate. We present algorithms for probe station placement where this information about the dependencies is not deterministic.

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

II. Related work

Network probing with low overhead has prompted development of many monitoring approaches. Due to space reasons, we survey only those approaches that directly relate to probe station placement. [8], [5], [12] propose intelligent distribution of probe stations at various traffic points in the network. Approaches presented in [4], [7], [2], [10] place probe stations at end-points of end-to-end paths whose characteristics are of interest. [11] and [3] use explicitly routed probes to reduce the number of monitoring stations required. [6] proposes a strategy to use high arity nodes for beacon placement. [9] proposes an approach for beacon placement by identifying edge sets that can be monitored by a beacon under all possible route configurations.

The above approaches do not take network failures into account. The problem of probe station selection becomes harder when a network component's failure is taken into consideration. In this paper we propose a probe station placement strategy that, along with minimizing the probe traffic and probe station deployment overhead, makes the monitoring robust against failures in the network. A related work in probe station selection has been done by [1], where the authors propose algorithms to compute locations of a minimal set of monitoring stations such that all links are covered, even in the presence of link failures. However there are some significant differences. [1] focuses on monitoring link delays and faults, while we consider node performance and node failures, which brings up different node specific issues and optimization possibilities like dealing with weakly connected nodes, special treatment for neighboring nodes, node failures etc. As a probe station itself is a node, we also address the case of failure of a probe station itself. The above approaches also do not consider the uncertainty in the obtained dependency information about the routes. Most of the approaches assume a deterministic environment with availability of complete and accurate dependency information. We propose a probe station selection approach to deal with the uncertainties involved in the collected information about the underlying network.

III. Proposed approach for probe station selection

In this section, we present a heuristic based approach that incrementally selects nodes which provide a suitable location to instantiate a probe station. The algorithm is based on the

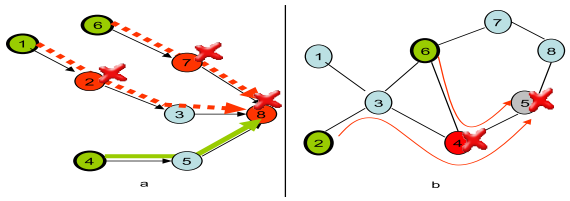

Fig. 1. (a.) 3 independent paths to node 8 from probe station nodes 1, 6, and 4, to detect failure of node 8 in a scenario of failure of 3 nodes (nodes 2, 7, and 8); (b.) A scenario of failure of nodes 4 and 5, where inappropriate probe station placement (at nodes 2 and 6) makes node 5 a shadow node.

concept that to diagnose k failures in a network, the probe stations should be placed such that each node can be probed through k independent (node disjoint) paths.

A. Assumptions

We build probe station selection algorithms to diagnose node failures in a network. The algorithms can be extended to monitor other network components of interest, e.g., link failures. We initially assume the availability of deterministic dependency information between the probes and the network nodes. In this section, we assume a consistent routing model which we discuss later in this section. We place a limit on the maximum number of node failures that can be diagnosed and the maximum number of probe stations that can be deployed. To simplify the problem, we initially assume that the probe station nodes do not fail. Later, we relax this assumption by considering probe station failures. We relax the assumptions about the dependency model and the routing model in Section V where we consider the presence of incomplete or inaccurate dependency information.

1) Routing model: In this section, we assume static single-path routing. We also assume a traditional IP routing model in which a forwarding node, on receiving packets for a destination node, sends the packets to the next hop listed in its forwarding table for that destination. Under this assumption, a forwarding node will always forward packets for a particular destination to the same next hop, irrespective of the source of the packets. We also assume that all routes in the network are free of loops. In what follows, we refer to this set of assumptions as the *consistent IP routing model*.

Algorithm SNR: Shadow Node Reduction Algorithm (Probe station selection to localize any k faults)

```
input : MAXPSSETSIZE, MAXFAULTS
output: Probe station set
Define N, ST(n), ST(n).parent(m), PS=NULL, SN=V, PPath(p,n),
Neighbor(n), Parents(n)=NULL;
Select node u with highest degree as first probe station;
InitializeDataStructures(u);
repeat
    repeat
        foreach node c, where c ∉ PS do
            S(c) ← ComputeShadowNodeSet();
        end
        Select node c with smallest |S(c)| as probe station;
        UpdateDataStructures(c);
        if |PS| ← MAXPSSETSIZE then
            return PS (Insufficient probe station set size);
        end
    until |SN| = 0 ;
    ResetShadowNodeSets();
until |SN| = 0 ;
return PS as the probe station locations;
;
Procedure InitializeDataStructures(node u)
Add u to PS; Remove u and Neighbor(u) from SN;
foreach node w ∉ PS do
    Parents(w) ← ST(u).Parent(w);
end
;
Procedure ComputeShadowNodeSet (node c)
S(c) ← Null;
foreach node x ∈ V do
    if ((x ≠ c) & (x ∉ Neighbor(c)) & (x ∈ SN) & (ST(c).parent(x) ∈
    Parents(x))) then
        Add x to S(c);
    end
end
return S(c);
;
Procedure UpdateDataStructures (node c)
Add c to PS;
SN ← S(c);
foreach node y ∉ PS do
    Parents(y) ← (Parents(y) ∪ ST(c).Parent(y)
end
;
Procedure ResetShadowNodeSets()
SN ← NULL;
foreach node z do
    if ((z ∉ PS) & (z ∉ {∀(p∈PS) Neighbor(p)}) & (|Parents(z)| <
    MAXFAULTS)) then
        Add z to SN
    end
end
```

B. Proposed approach

It can be proved that the paths following the consistent routing model will have the following two properties:

- If probe paths from two different probe stations to the same destination have one node in common, then all subsequent nodes on the two probe paths will be the same.
- Probe paths P1 and P2 to a destination node d are independent (i.e., node-disjoint) if and only if P1 and P2 have different predecessors to the node d.

These properties provide an efficient test to identify the independence of two paths to the same destination d. Instead of comparing the entire sequences of the two paths, it is sufficient to verify the predecessors of node d on the two paths to be different.

With the assumptions made in Section III-A, it can be proved that *a set of probe stations can localize any k non-probe-station node failures in the network if and only if there exist k independent probe paths to each non-probe-station node that is not a neighbor of any probe station.*

Consider the example shown in Figure 1a. Failure of nodes 2 and 7 prevents probe stations 1 and 6 respectively from diagnosing the health of node 8. However, with the assumption of detecting at most 3 faults, and the availability of 3 independent paths, there is one probe path to node 8 (path

from 4 to 8) with no intermediate failed nodes. Thus probe station node 4 can detect the failure of node 8. An incorrect probe station placement can make some nodes unreachable in a certain failure scenario. We use the term *shadow nodes* to represent such nodes. Figure 1b shows how an incorrect probe station placement at nodes 2 and 6, makes node 5 unreachable on failure of node 4, making node 5 a shadow node.

The neighbor nodes of a probe station do not need k independent paths as they can be uniquely diagnosed through direct probes from the probe stations irrespective of the faults present in the network. Thus all non-probe-station nodes that are not neighbors of probe stations and that do not have k independent paths from the probe stations belong to the shadow node set. And the objective of probe station placement algorithms is to minimize the shadow node set. The heuristic used is to select a node as a probe station that minimizes the shadow node set.

IV. Algorithms for probe station selection

A. Algorithm to localize node failures assuming no probe station failures

For clarity, we first do not take probe station failures into consideration. We assume a limit k on the maximum number of faults that need to be diagnosed in the network. Initially the selected probe station set is empty and all nodes belong to the shadow node set. As explained in Section III-B, the heuristic used is to choose a node that minimizes the shadow node set. The first probe station is selected based on the node degree. The node with the largest number of neighboring nodes can remove maximum number of nodes from the shadow node set during the first probe station selection and hence is a good candidate to be selected as the first probe station. However in scenarios where some pre-placement of probe stations already exists, this step of first probe station selection can be avoided.

When only one probe station has been selected, all nodes that are not neighbors of the selected probe station belong to the set of shadow nodes. All the nodes that do not belong to the selected probe station set are candidates for the next probe station selection. For each candidate probe station, the algorithm determines how the shadow node set would change if the candidate was selected as a probe station. This shadow node set will consist of i) nodes that are not neighbors of selected probe stations, and ii) nodes that do not have k unique paths from the selected probe stations. Of all the candidate probe station nodes, the node that produces the smallest set of shadow nodes is selected as the next probe station node. Note that the independence of paths can be verified simply by comparing the predecessors of the destination node on the two probe paths as explained in Section III. The algorithm iteratively adds a new node to the probe station set till the desired capacity of diagnosing k faults is achieved. The algorithm terminates when no shadow nodes are present or the probe station set size reaches the maximum limit.

This approach is presented in Algorithm SNR. In this algorithm, probe stations are first selected to find any two faults in the network. Then new nodes are added to localize any three faults in the network and so on. In other words, in the first iteration, probe stations are added such that each non probe station node is either a neighbor of a probe station or has two independent probe paths. In the next iteration, more probe stations are selected such that each non probe station node that is not a probe station neighbor has three independent probe paths. The nodes are added in this fashion by incrementally increasing the overall diagnostic capability, till k faults in the network can be localized. In Algorithm SNR, we use a data structure Parents(x) representing the set of nodes that are parents of node x on probe paths from the selected probe stations to node x. Thus |Parents(x)| gives the total number of disjoint probe paths available through the selected probe station set. The inner loop finds an independent probe path to each shadow node. Function ResetShadowNodes() resets the shadow node set to all nodes that are not probe stations, are not probe station neighbors, and have less than k independent probe paths to them. The inner loop when repeated, finds another set of probe stations such that each shadow node has one additional independent path. This increases the diagnostic capability of the probe station set to localize one more fault in the network. Also note that nodes with degree less than k cannot have k independent paths. Probe stations are selected so that such nodes are always neighbors of probe stations.

B. Algorithm to localize probe station failures along with non-probe station node failures

In a real network, the probe stations are also subject to failure, and thus probe stations should be selected such that a probe station node failure can also be localized by the other probe stations. To find a probe station set that has the capability to localize probe station failures, there are three main points to note:

1) Each probe station should also be probed by k independent probe paths.
2) All neighboring nodes to a probe station should also have k independent probe paths.
3) There may exist scenarios of k failures where the failure of a node with degree less than k cannot be localized.

Details of this algorithm are omitted for reasons of space.

V. Probe station placement in a non-deterministic environment

In this section, we deal with this issue of uncertainty involved in the dependencies while computing the probe station set to monitor a network. We use a probabilistic dependency model to represent the uncertainty involved in the dependencies between the end-to-end paths and the nodes used on these paths. The weight assigned to the dependency between an end-to-end path p and a node n represents the probability that node n is used by the probe on the end-to-end path p.

A. Routing model:

Unlike the previous assumption, in a non-deterministic environment a node might forward a packet for a particular

destination to different next hops with different probabilities. For instance, in scenarios with load balancers or dynamic routing, traffic for a particular destination might be routed to different next hops with different probabilities.

Algorithm PSNR: Probabilistic Shadow Node Reduction (Probe station selection to localize any k faults in the network in a non-deterministic environment)

```
The algorithm is same as Algorithm SNR, with some procedures
implemented differently. Path(m,n):nodes on path from m to n;
PSPath(w):nodes on path from the selected probe stations to node w;
PathCount(n):number of independent probe paths to node n;
;
Procedure InitializeDataStructures(node u)
Add u to PS;
Remove node u and Neighbor(u) from SN ;
Initialize PathCount(w) for each w ∉ PS, to 1;
foreach node w ∉ PS do
    foreach node t ∉ PS do
        P(PSPath(w), t) ← P(Path(u,w),t);
    end
end
;
Procedure ComputeShadowNodeSet (node c)
S(c) ← Null ;
PathCount(c,n) ← PathCount(n) for each node n ∉ PS;
foreach node x ∈ V do
    if ((x ≠ c) & (x ∉ Neighbor(c)) & (x ∈ SN) then
        Overlap ← 0;
        foreach node n ∈ (Path(c,x) ∩ PSPath(x)) do
            Overlap ← ⋃(Overlap, P(PSPath(x),n).P(Path(c,x),n))
        end
        if (1-Overlap) > Threshold then
            Add x to S(c)
        end
        else
            Increment PathCount(c,n) by 1
        end
    end
end
return S(c);
;
Procedure UpdateDataStructures (node c)
Add c to PS;
SN ← S(c);
PathCount(n) ← PathCount(c,n) for each node n ∉ PS;
foreach node y ∉ PS do
    foreach node t ∉ PS do
        P(PSPath(y),t) ← ⋃(P(PSPath(y),t), P(Path(c,y),t));
    end
end
;
Procedure ResetShadowNodeSets()
SN ← NULL;
foreach node z do
    if ((z ∉ PS) & (z ∉ {∀(p∈PS) Neighbor(p)}) & (|PathCount(z)| <
    MAXFAULTS)) then
        Add z to SN
    end
end
```

B. Test of independent paths:

In a non-deterministic environment, to verify the independence of two paths, all nodes on one path need to be verified to be different from all nodes on another path. Also, the independence of the two paths p and q needs to be represented with a confidence in our belief that the two paths are independent. The belief value can be computed as follows:

$B(I_{(p,q)}) = 1 - \bigcup_{n \in Nodes(p) \cap Nodes(q)} P(p,n).P(q,n)$

where the term $\bigcup_{n \in Nodes(p) \cap Nodes(q)} P(p,n).P(q,n)$ represents the probability that the two paths p and q overlap, by using the nodes that are common on both paths. The higher the value of $B(I_{(p,q)})$, the stronger is the belief that probes p and q are independent.

C. Algorithm for probe station placement in a non-deterministic environment

For clarity, we do not consider probe station failures and assume that probe stations can always probe their neighbors irrespective of other faults in the network. As discussed in Section IV, this algorithm can be extended to relax these assumptions. As before, the first probe station is selected as the node with highest degree. We represent the already selected probe stations with a set S. P(Path(S,n),m) represents the probability that paths from probe stations in set S to node n use node m. For a candidate probe station c, the probability that c provides an independent path to a shadow node s, can be obtained by computing

$B(I_{(Path(S,n),Path(c,n))}) =$
$1 - \bigcup P(Path(S,n),m).P(Path(c,n),m)$

where $m \in Nodes(Path(S,n)) \cap Nodes(Path(c,n))$.

If this value is greater than a threshold, then the path is considered to be independent. A candidate node that provides maximum number of independent paths is selected as the next probe station. Once a probe station c is selected, it is added to the set S and the probability P(Path(S,n), m) for each node n to which node c provides an independent path and for each node m used by these paths, is updated as follows:

$P(Path(S \cup c, n), m) = \bigcup(P(Path(S,n),m), P(Path(c,n),m))$

The algorithm stops when each non-probe station node that is not a probe station neighbor has k independent paths. We present this approach in Algorithm PSNR.

VI. Simulation results

In this section we present experimental evaluation of algorithms for probe station selection introduced in this paper. We do not consider probe station failures in these experiments. We use exhaustive-search-based Optimal algorithm as a benchmark. Because of its high computational complexity, we were not able to run the Optimal algorithm for larger networks. For evaluation of the proposed algorithm on larger networks, we compared the proposed algorithms with alternative algorithms where probes stations are selected at random locations in the network till the desired diagnostic power is obtained.

A. Simulation model

Let MD, AD, and N represent the maximum node degree, average node degree, and the total number of nodes in the network respectively. Given these three parameters, we create a network of N nodes, randomly introducing N*AD links such that no node has a network degree greater than MD, and also ensuring that the network is connected. We conducted experiments on network sizes ranging from 10 to 200 nodes with average network degrees ranging from 3 to 9 and maximum

Fig. 2. Comparison of SNR and Optimal algorithm for networks with average node degrees 6 and 9.

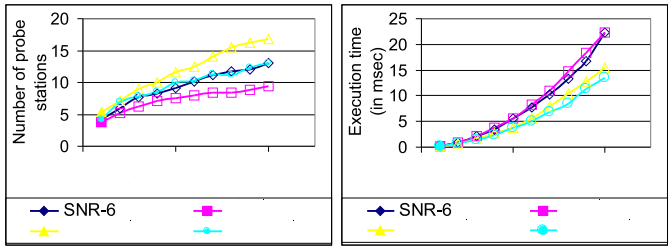

Fig. 3. Comparison of SNR and Random algorithm for networks with average node degrees 6 and 9.

node degree set to min(20, network size). Each point plotted on the graph is an average of 20 runs. We compared the computed probe station set size and the execution time of the algorithms to detect 4 failures in the network in a deterministic and a non-deterministic environment.

Figures 2 and 3 present the results for the SNR, Exhaustive and Random algorithms for a deterministic environment. Figure 2 shows that the SNR algorithm computes probe station sets of sizes close to the optimal probe station set sizes computed by the Optimal algorithm (greater by only 0.5 for 30 node network with average node degree 9). Moreover the SNR algorithm runs in significantly less time (750ms less for 30 node network) than the Optimal algorithm. Figure 3 shows that the probe station set sizes computed by the SNR algorithm are smaller than the Random algorithm taking marginally longer time than the Random algorithm. Moreover, the difference in the probe station set sizes computed by SNR and Random algorithms grows bigger with increasing network size.

We built a probabilistic dependency model by computing multiple paths between a source-destination pair and assigning probabilities to the nodes on these paths on the basis of the lengths of the paths. The nodes on the shorter path are assigned higher probability. From Figure 4 we can see that Algorithm PSNR performs close to optimal (computing a probe station set size bigger by only 1.15 nodes for a 25 node network with degree 6). Figure 4 also compares the probe station set sizes computed by PSNR and Random algorithm showing a significant improvement by PSNR algorithm over the Random algorithm (computing a probe station set size smaller by 20 nodes for a 50 node network).

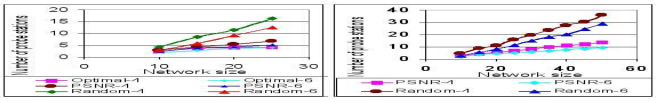

Fig. 4. Comparison of PSNR, Optimal, and Random algorithms for networks with average node degrees 4 and 6.

VII. Conclusion

We presented algorithms to select suitable locations to deploy the probe stations. We first presented the algorithms assuming the availability of complete and accurate information about the underlying network. Later we relaxed this assumption and considered the presence of uncertainties in the dependency information. We made the placement robust against various failures in the network. We evaluated the proposed algorithms through simulation results and compared the proposed heuristics with the Optimal and Random placement algorithms. As part of on-going research, we are working on developing algorithms to select appropriate probes to be sent from these probe stations to monitor the network for fault diagnosis.

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

References

[1] Y. Bejerano and Rajeev Rastogi. Robust monitoring of link delays and faults in IP networks. In *IEEE INFOCOM, San Francisco, CA*, Mar 2003.

[2] J. C. Bolot. End-to-end packet delay and loss behavior in the Internet. In *ACM SIGCOMM'93*, Sep 1993.

[3] Y. Breitbart, C. Y. Chong, M. Garofalakis, R. Rastogi, and A. Silberschatz. Efficiently monitoring bandwidth and latency in IP networks. In *IEEE INFOCOM, Tel Aviv, Israel*, Mar 2000.

[4] A. B. Downey. Using pathchar to estimate Internet link characteristics. In *ACM SIGCOMM, Cambridge, MA*, 1999.

[5] P. Francis, S. Jamin, V. Paxson, L. Zhang, D. F. Gryniewicz, and Y. Jin. An architecture for a global Internet host distance estimation service. In *IEEE INFOCOM'99*, Mar 1999.

[6] J. D. Horton and A. Lopez-Ortiz. On the number of distributed measurement points for network tomography. In *Internet Measurement Conference, IMC*, 2003.

[7] V. Jacobsen. pathchar- a tool to infer characteristics of Internet paths. April 1997.

[8] S. Jamin, C. Jin, Y. Jin, Y. Raz, Y. Shavitt, and L. Zhang. On the placement of Internet instrumentation. In *IEEE INFOCOM, Tel Aviv, Israel*, Mar 2000.

[9] R. Kumar and J. Kaur. Efficient beacon placement for network tomography. In *Internet Measurement Conference, IMC*, 2004.

[10] K. Lai and M. Baker. Measuring bandwidth. In *IEEE INFOCOM'99, New York City, NY*, Mar 1999.

[11] F. Li and M. Thottan. End-to-end service quality measurement using source-routed probes. In *25th Annual IEEE Conference on Computer Communications (INFOCOM), Barcelona, Spain*, Apr 2006.

[12] W. Theilman and K. Rothermel. Dynamic distance maps of the Internet. In *IEEE INFOCOM'2000*, Mar 2000.