J Netw Syst Manage (2008) 16:351–374
DOI 10.1007/s10922-008-9114-0

Probe Station Placement for Robust Monitoring of Networks

Maitreya Natu · Adarshpal S. Sethi

Published online: 13 November 2008
© Springer Science+Business Media, LLC 2008

Abstract We address the problem of selecting probe station locations from where probes can be sent to monitor all the nodes in the network. Probe station placement involves instrumentation overhead. Hence, the number of probe stations should be minimal to reduce the deployment cost. Also, probe station placement should be such that the network can be monitored even in the presence of failures. We present algorithms to select locations of probe stations so that the entire network can be monitored for computing various performance metrics. We aim to find a minimal set of probe station nodes so as to minimize the instrumentation overhead. The algorithm presented provides robust monitoring in presence of node failures. We then present algorithms to make the solution resilient to probe station failures, and to deal with weakly connected nodes. We provide an experimental evaluation of the proposed algorithms through simulation results.

Keywords Active probing · Probe station placement · Network monitoring · Fault diagnosis

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. 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.

M. Natu (✉)
Tata Research Development and Design Centre, 54-B Hadapsar Industrial Estate, Pune, MH 411013, India
e-mail: maitreya.natu@tcs.com

A. S. Sethi
Department of Computer and Information Sciences, University of Delaware, 103 Smith Hall, Newark, DE 19716, USA
e-mail: sethi@cis.udel.edu

1 Introduction

With the increase in complex networks and new services that require QoS guarantees, and the demand for sophisticated tools for monitoring performance of a network has increased rapidly. Tools for monitoring the network and detecting faults are critical for numerous important network management tasks, e.g., QoS guarantees to end applications, traffic engineering, ensuring SLA (Service Level Agreement) compliance, fault and congestion detection, and identifying performance bottlenecks.

Various approaches have been proposed in the past for monitoring an IP network in order to measure different performance metrics. Broadly the approaches can be classified into two groups, namely, the passive monitoring tools and the probing tools.

- *Passive monitoring tools* monitor the network by deploying monitoring agents on the networking devices using SNMP/RMON [1] or the Cisco NetFlow tool [2]. Stemm et al. [3] argue for use of passive monitoring because it does not send additional traffic to perturb actual Internet traffic. e.g., Cisco's NetFlow measurement tool allows NetFlow enables routers to collect detailed traffic data on packet flows between source-destination pairs. Key mechanism to enhancing NetFlow volume manageability is careful planning of NetFlow deployment. Although non-intrusive nature of passive monitoring is very appealing, it has several limitations. These tools require deployment of monitoring agents on a large number of devices. This might be an expensive operation and might not be possible in all cases due to privacy and security considerations. The tools can only measure regions of the Internet that application traffic has already traversed. Current measurement methods based on SNMP only provide device centric view of performance. They can not monitor end-to-end network parameters, like end-to-end path latency, that involve several components. Unlike end-to-end probes, neither of these methods can be used directly to measure end-to-end delay, loss, and impact of these performance metrics on service quality.
- *Probing tools* consist of connectivity, latency, and bandwidth measurement tools such as pathchar [4], pathrate [5], PathChirp [6], and NetTimer [7]. These tools measure end-to-end network performance metrics by sending probes in the network. e.g., PathChirp [6] is an active probing tool for estimating available bandwidth on a communication network path. Based on the concept of self-induced congestion, PathChirp features an exponential flight pattern of probes, called chirp. The end-to-end nature of probes allows measurement of end-to-end performance metrics involving several components. These tools however send network management traffic into the network. Thus care needs to be taken to optimize the probes that are being sent out in the network, so that the network functioning is not affected by the management traffic. One promising approach to optimize the probes is active probing [8, 9]. Active probing is an interactive probing approach, where probes to send are selected at run time based on previous probe results. In this scheme, probes get sent only to the points of

interest where there is a possibility of potential bottleneck or failure. Probing tools do not require special instrumentation of each node. Instead probes are sent from a small set of nodes, called probe stations, to the rest of the network. A probe station is able to monitor a limited set of nodes and links. Thus probe stations need to be deployed at few strategic points so as to maximize network coverage while minimizing infrastructure and maintenance costs.

In this paper we address the problem of selecting suitable node locations for probe station deployment. Instantiating probe stations at right locations is important for effective active probing solution. Improper probe station deployment can lead to higher non-optimized probe traffic. It can also reduce the diagnostic capability of the probing tool. Moreover a large number of probe stations would increase cost and incur instrumentation overhead. We present algorithms to find minimal set of probe stations for effective probing and propose various extensions to it.

1.1 Related Work

Network probing with low overhead has prompted development of new monitoring approaches. Various approaches have been presented in the past regarding the placement of probe stations. [10, 11, 12] propose intelligent distribution of probe stations at various traffic points in the network. Approaches presented in [4, 13, 14, 7] place probe stations at end-points of end-to-end path whose characteristics are of interest. Reference [15] proposes a single, predefined point in the network to be deployed as probe station. Below we describe the design rational of these approaches and then present our approach to probe station placement.

IDMaps project [10, 11] produces latency maps of Internet using special measurement servers called tracers that are distributed around the Internet so that any node is relatively close to one or more tracers. These tracers continuously probe each other to determine their latency which is used to approximate the latency of arbitrary network paths. The paper discusses different approaches of deploying tracers. IDMaps measurements may not be too accurate due to relatively small number of paths being monitored.

References [15, 16] use explicitly routed probes to reduce the number of monitoring stations required. Breitbart et al. [15] propose to use a single predefined point in the network as a monitoring station, called Network Operations Center, to perform all the required measurements. However in today's ISP and Enterprise environment, approaches based on explicitly routing of packets might not be feasible. Many routers frequently disable the source routing option. Authors in [17] and [18] propose placement of monitoring stations that is robust to route changes. Horton and Lopez-Ortiz [17] propose a strategy to use high arity nodes for beacon placement to test for connectivity on all relevant edges of the network. However the beacon set size could be quite large for real ISP topologies [18]. Kumar and Kaur [18] propose an approach for beacon placement by identifying edge sets that can be monitored by a beacon under all possible route configurations. Reddy et al. [19] propose to have each receiver monitor only a portion of path between itself and the source. On the other hand, in our approach only the probe stations nodes perform the task of sending probes and monitoring the network. Adler et al. [20] propose an approach to find minimum cost multicast trees that cover links of interest in the network.

Above approaches do not take network failures into account. The problem of probe station selection becomes harder when network components failure is taken into consideration. In this paper we propose a probe station placement strategy, to monitor the network while minimizing the probe traffic and probe station deployment overhead. We propose a monitoring scheme that is robust against failures in the network. A related work in probe station selection has been done by [21], where they 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. Bejerano and Rajeev Rastogi [21] focus on monitoring link delays and faults, while we consider node performance and node failures. A node specific perspective brings up various different node specific issues and optimization possibilities like dealing with weakly connected nodes, special treatment for neighboring nodes, node failures etc. As probe station itself is a node, we also address the case of failure of a probe station itself. We propose a probe station placement that is resilient to probe station failures.

1.2 Our Contributions

We present algorithms to select locations of probe stations so that the entire network can be monitored for computing various performance metrics. We aim to find a minimal set of probe station nodes so as to minimize the deployment cost. The algorithm presented provides robust monitoring in presence of node failures. We then present extensions to the algorithm to make the solution resilient to probe station failures, and to deal with weakly connected nodes. The failures assumed are permanent node failures. We assume static routing in the presented work. Considering route changes is part of our ongoing research.

2 Proposed Approach for Probe Station Selection

Given a set of nodes in the network, the probe station placement should be such that the probe stations are able to send enough probes to all the nodes to localize the fault. The number of probe station should be less to reduce the deployment overhead. It is possible to arrive at an optimal probe station placement by doing an exhaustive search, but this combinatorial approach is computationally too expensive to be deployed practically, unless the network is quite small. We propose a heuristic based approach that incrementally selects nodes which provide suitable locations to instantiate probe stations. The proposed algorithm involves much fewer computations than the combinatorial approach. These algorithms attempt to find the minimal probe station set but are not guaranteed to do so. We later show through simulation results that these algorithms give results that are close to optimal.

2.1 Design Issues in Deploying Probe Stations

The decision of probe station selection is based on nature of routes, nature of targeted failures, availability of dependency information etc. Below we discuss various such factors that contribute to the overall decision making of probe station selection:

- Nature of targeted failures: Probe station selection depends on the nature of faults that need to be diagnosed. Assuming a connected network, to detect a single node failure, a single probe station can be sufficient. However in the same network, to detect a link failure, we might need more than one probe station because, while the probe paths from a single probe station may be able to reach all other nodes, they might not cover all the edges. In this paper, we are targeting node failures.
- Maximum number of failures: The assumption of maximum number of faults that need to be localized in a network is an important factor in selecting the probe stations. In a connected network, a single node failure can be detected by just one probe station. However a single probe station might not be sufficient to detect two node failures, if both failures occur on the same probe path. Considering the extreme case where all nodes' failure needs to be diagnosed, the probe stations then need to be placed at the vertex cover of the graph formed from the network topology. We propose a k-fault resilient algorithm, assuming a limit of occurrence of at most k failures in the network at a time.
- Probe station failure: The problem becomes even more challenging when the probe station failure is taken into consideration. In case of probe station failure, probe stations must be chosen to provide the ability to detect such failures and make another probe station perform the job of the failed probe station. In the proposed algorithm, we consider probe station failures while deploying the probe stations.
- Topological constraints: Another important criterion involved in probe station selection is the topological constraint. The nodes with less connectivity need special treatment. For example, a probe station needs to be deployed at neighboring node of the leaf nodes, because otherwise if the leaf node's neighbor fails, the leaf node becomes unreachable by any probe station. Special topology structures like chains and rings also demand specific probe station placement requirements.
- Static versus dynamic probe station instantiations: The probe station selection criteria differ if a probe station location can be selected actively based on current diagnosis requirements. As opposed to static probe station selection, this approach provides more flexibility, but deploying probe stations dynamically might not be possible at all places in the network. In current work, we assume static instantiation of probe stations.
- Nature of routes: Probe station selection is also affected by the nature of routes taken by probes. Considering source routing, a node can probe another node through multiple routes which enhances its probing capacity. The symmetric or asymmetric nature of routes also provides additional information of the probing

capacity of the probes. Special care needs to be taken in the presence of loops. If the routes dynamically change, due to load-balancers, source routing, mobility, etc., the probe station might not be able to detect the same faults in the changed routing conditions. We make an assumption of static routes and a consistent routing model, as explained in next section. Considering the case of route changes is part of our ongoing research.

- Dependency information: The amount of routing information available for decision making poses some other practical problems in probe station selection. The accuracy and confidence in the information is expressed through the dependency model. Based on the information available, this model could be deterministic or probabilistic. It can be complete or incomplete. Moreover based on the nature of the network, the model might change with time. Changes can occur due to route changes or because of availability of more precise information.

2.2 Definitions

While presenting our research objectives and results, we use the general concepts that are defined in this section.

We represent a network by an undirected graph $G = (V,E)$, where $V = \{v_1, v_2, \ldots\}$ is the set of vertices that represent the network nodes, and $E = \{e_1, e_2, \ldots\}$ is the set of edges that represent the network links. An edge e_k is identified by an unordered pair (v_i, v_j) of vertices, where v_i and v_j are called the end vertices of e_k. A node v_i is said to be a neighbor of node v_j if (v_i, v_j) 2 E. We will denote the number of nodes in the graph, i.e. $|V|$, by N.

A path from node u to node v is a finite sequence of nodes through which a message passes to reach from node u to node v. An edge exists between each pair of consecutive nodes in the sequence and no vertex appears more than once. There can be many paths in a network between nodes u and v. However by the above definition of path, by $Path(u,v)$ we refer to the specific path that is used for packet transmission in the network from node u to node v. The number of nodes in a path is called the length of the path. We represent a path P from node u to node v as $P = Path(u,v) = (u, u_1, u_2, \ldots, \mathrm{v})$. Node u is called the source and node v is called the destination of path P. The node that precedes a node u_i in the sequence of nodes defined by path P is called the predecessor of u_i and is denoted by $Predecessor(P, u_i)$. The predecessor of the first node in the sequence is undefined. The node that succeeds a node u_i in the sequence of nodes defined by path P is called the successor of u_i and is denoted by $Successor(P, u_i)$. The successor of the last node in the sequence is undefined. Two paths $P = Path(u,v) = (u, u_1, u_2, \ldots, \mathrm{v})$ and $Q = Path(w,v) = (w, w_1, w_2, \ldots, \mathrm{v})$ to the same destination node v but from different sources u and w, are called independent paths if there does not exist any node in path P that is also present in path Q, except node v.

A probe is a test transaction sent from a probe station u to a node v such that the success or failure of the probe depends on the success or failure of the nodes on $Path(u,v)$. A successful probe on $Path(u,v)$ indicates good health of all nodes on

Path(*u*,*v*). A failed probe on *Path*(*u*,*v*) indicates poor health or failure of one or more nodes on *Path*(*u*,*v*).

Given a set of probe stations and the possibility of maximum k node failures, a non-probe station node is called a shadow node if there exists a scenario of k failures which makes the node unreachable from all probe stations. Shadow nodes hold the property that their health cannot be localized by any probe station in a certain scenario of k failures. Thus to determine health of all nodes in the network, the probe stations should be selected such that there do not exist any shadow nodes under the assumption of maximum k failures in the network.

2.3 Assumptions

We assume the availability of dependency information between the probes and network nodes. This information is stored in the form of a dependency model that represents which nodes are probed by a probe. We assume consistent IP routing such that packets to a particular destination are always forwarded to the same next hop by a forwarding node irrespective of the packet source. We discuss this routing model in Sect. 2.3.1. We place a limit on the maximum number of failures that can occur in the network and that need to be diagnosed. We also place a limit on the maximum number of probe stations that can be deployed. We consider only node failures for the following discussion. To simplify the problem we initially assume that probe station nodes do not fail. Later we relax this assumption by allowing probe station failures. We also deal with topology constraints that require special treatment for weakly connected nodes.

2.3.1 Consistent Routing Model

For the ongoing discussion, we assume static single-path routing. Thus packets between a particular source-destination pair always follow a single path that does not change with time. 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.

The following theorems hold for the consistent IP routing model. We have not presented proofs for all theorems, due to lack of space.

Theorem 2.1 *In the consistent IP routing model, if probe paths from two different probe stations to the same destination have one node in common, then all subsequent nodes following the common node on the two probe paths will be the same.*

Theorem 2.2 *Probe paths P1 and P2 to a destination node d are independent if and only if Predecessor(P1,d) is different from Predecessor(P2,d).*

2.4 Design Rationale

The consistent routing model provides an efficient test for determining whether or not two paths to a destination node *d* are independent. Instead of comparing the entire sequences of the two paths, it is sufficient to compare the predecessors of node *d* on the two paths. If the predecessors are the same, the paths are not independent. We now show with Theorem 2.3 that with the consistent routing assumption, localization of *k* failures requires the existence of *k* independent probe paths from the probe stations to every node.

Theorem 2.3 *Assuming a consistent IP routing model with at most k failures in the network, 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.*

Proof Assume a consistent IP routing model with at most *k* non-probe-station node failures in the network.

Case 1 If there exist *k* independent probe paths from a set of probe stations P to all non-probe-station nodes that are not neighbors of any probe station, then P can localize any *k* non-probe-station node failures in the network.

Proof by Way of Contradiction Assume that there exist *k* independent paths from a set of probe stations to all non-probe-station nodes that are not neighbors of any probe station. Also assume that at most *k* non-probe-station nodes can fail but *k* non-probe-station node failures cannot be localized. Without loss of generality, assume that there exists one non-probe station node u whose failure cannot be localized in the given scenario. Node u can belong to one of the following cases:

1. Node u is a neighbor of some probe station node. A direct probe can be sent from neighboring probe station to node u to localize the failure.
2. Node u is not a neighbor of any probe-station. As assumed, there can be at most *k* non-probe-station node failures in the network and failure of node u leaves the possibility of at most ($k-1$) other non-probe-station node failures. Since all of the *k* probe paths probing node u are independent, at most ($k-1$) probes can fail due to the ($k-1$) faults in the network. Thus there must exist one probe path through which a probe can be sent to probe node u and localize its failure.

Hence our assumption is wrong. This proves Case 1.

Case 2 If a set of probe stations can localize any *k* non-probe-station node failures, then there exist *k* independent probe paths to each non-probe-station node that is not a neighbor of any probe station.

Proof by Way of Contradiction Assume that there exists a set of probe stations that can localize any *k* non-probe-station node failures and there do not exist *k* independent probe paths to each non-probe-station node that is not a neighbor of any probe station. Consider a failed non-probe-station node u that has ($k-1$) independent probe paths. As node u is one of the failed nodes, with the assumption of at most *k* non-probe-station node failures, there can be at most ($k-1$) other failures. Consider a scenario, where these $(k-1)$ failures occur on the $(k-1)$ independent probe paths to node u, such that one node fails on each—*of the $(k-1)$ probe paths to node u. This makes node u unreachable to any probe station, such that no probe station is able to localize the failure of node u. Hence our assumption is wrong. This proves the statement in Case 2.

Both directions of "if and only if" have been proved separately, therefore their conjunction is also true. Hence the theorem is proved.

Consider the example shown in Fig. 1a. This figure shows how the availability of three independent paths to node 8 from probe station nodes 1, 4, and 6, help to diagnose a scenario of 3 node failures (nodes 2, 7, and 8). Failure of nodes 2 and 7 prevent 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 can lead to inadequate diagnostic power to detect failure of nodes 4 and 5. This figure shows paths from nodes 2 and 6 to reach node 5. It can be seen that the failure of node 4 makes node 5 unreachable from both the probe stations, 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.

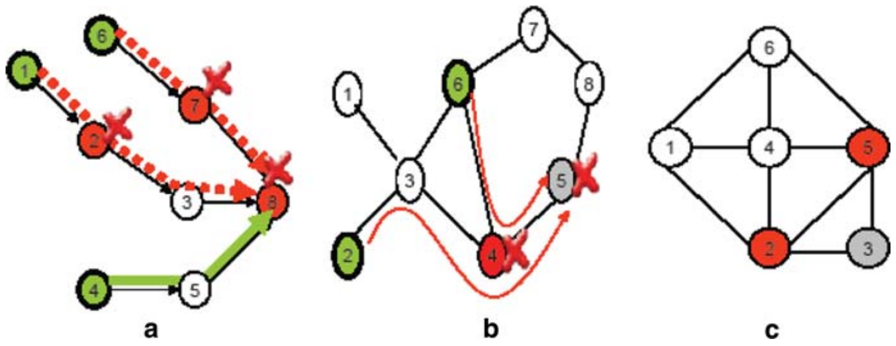

Fig. 1 **a** Three independent paths to node 8 from probe station nodes 1, 6, and 4, to detect failure of node 8 in a scenario of failure of three nodes (nodes 2, 7, and 8); **b** A scenario of failure of nodes 4 and 5, where inappropriate probe station placement (at nodes 2 and 6) makes node 5 a shadow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

2.5 Algorithms for Probe Station Selection

Given the assumptions made in Sect. 2.3, the objective of the algorithms presented in this section is to compute a smallest set of nodes to deploy probe stations such that each node that is not a neighbor of a probe station can be probed by *k* independent probe paths from these probe stations. As explained in Sect. 2.4, such a set of probe stations can monitor the health of all nodes to diagnose *k* node failures in the network. We first present an algorithm for optimal probe station placement based on exhaustive search by evaluating all possible node combinations. But this combinatorial approach is too expensive to be deployed practically for large networks. Hence, we propose a heuristic-based approach that incrementally selects nodes which provide suitable node positions to instantiate probe stations. The proposed algorithms involve much fewer computations than the combinatorial approach. These algorithms attempt to find the minimal probe station set but are not guaranteed to do so. We later show through simulation results that these algorithms give results close to optimal.

While presenting our algorithms, we use certain notations that are defined in the following sub-section.

(1) Algorithmic notations: We use the following notation for the algorithms presented in this section:

Graph G = (V,E): represents the managed network where V represents the network nodes and E represents the network links.
N: the total number of nodes in the network. Thus N = |V|.
Neighbor(u): the set of nodes that are neighbors of node u in the network.
ST(u): A spanning tree that represents the routes used to send probes from node u to all other nodes in the network. We assume the availability of spanning trees rooted at each node.
ST(u).parent(v): returns the parent node of node v in the spanning tree rooted at node u.

We use the following notation in the algorithms to represent specific data structures:

PS: the set of selected probe stations
SN: the set of shadow nodes
Parents(x): the set of parent nodes of node x in spanning trees rooted at all of the selected probe stations.
MAXFAULTS: the maximum number of faults that can occur in the network. We mostly refer to this value as k, but for clarity we use the term MAXFAULTS in the algorithms.
MAXPSFAULTS: the maximum number of probe station failures that can occur in the network.
MAXPSSIZE: the maximum size of the probe station set.

2.6 Combinatorial Algorithm

This algorithm takes a naive approach by evaluating all possible combinations of nodes to obtain a set of probe stations for localizing k faults. To localize k faults, we know from Theorem 2.3 that each non-probe station node should either be a neighbor of one of the probe station nodes or should have k independent probe paths to it. As the goal is to find a minimum set of probe station nodes which can localize k faults, the search starts by examining all singleton sets of one node and then continues exploring combinations of successive larger sizes until the desired set is found.

This algorithm explores all possible probe station combinations starting from combinations of smallest size, and then incrementally increasing the set size to a maximum limit of MAXPSSETSIZE. For each combination of probe station nodes, the shadow node set is computed. As defined in Sect. 2.2, shadow nodes for a network consist of the set of nodes whose health cannot be diagnosed by any probe station in a certain failure scenario. In the scenario of maximum k failures, the shadow nodes consist of all non-probe-station nodes which are not neighbors of probe station nodes and which do not have k independent probe paths from the probe stations.

The search can be improved by sorting the nodes in decreasing order of network degree and exploring nodes or adding nodes to an existing set in this order to try different combinations.

This algorithm performs $\sum_{i=1}^{MAXPSSETSIZE} \left(i^{MAXPSSETSIZE}\right).i.N = O\left(N^{MAXPSSETSIZE}\right)$ operations. If MAXPSSETSIZE is set to N, the number of nodes in the network, then the algorithm's computational complexity becomes exponential in N. In larger networks the number of probe station nodes is likely to be large. In such networks, the exhaustive algorithm can be computationally too expensive to be practically deployed.

We use this algorithm as a reference algorithm to compare with the heuristic algorithms of the next section, because it finds the optimal solution which provides minimum number of probe stations for localizing k faults.

2.7 Heuristic-Based 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 Sect. 2.4, 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 Sect. 2.4. 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. 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 all nodes in the selected probe station set to node x. Thus |Parents(x)| gives the total number of disjoint probe paths available through the selected probe station set. Lines 6 to 15 run a loop to find an independent probe path to each shadow node. In line 16, function ResetShadow-Nodes() is called to reset 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. Lines 6 to 15 when repeated, find 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 can not have k independent paths. Probe stations are selected so that such nodes are always neighbors of probe stations.

Figure 2 presents an example of how the probe station selection algorithm selects probe stations to detect any two node failures in the network. Figure 2a shows a network topology with nine nodes considering all nodes as shadow nodes. Figure 2b shows node 4, being the node with largest degree, as the selected probe station removing neighboring nodes 2, 6, 1, and 5 from the shadow node set. Figure 2c shows node 6 as the next selected probe station, which removes neighboring nodes 9 and 7 from the shadow node set. Nodes 3 and 8 are not neighbors of any probe station, but they have two independent probe paths from probe station 4 and 6 as shown in the Fig. 2c. Thus both nodes 3 and 8 are also removed from the shadow node set. Thus the probe station placement at nodes 4 and 6 can detect any two node failures in the network.

Algorithm SNR: shadow Node Reduction Algorithm (Probe station selection to localize any k faults in the network)

```
input   : MAXPSSETSIZE, MAXFAULTS
output  : Probe station set
1 Define N, ST(n), ST(n), parent(m),PS, SN, PPath(p,n), Neghbor(n), Parents(n);
2 CreateDataStructures();
3 u← SelectFirstProbeStation();
4 InitializeDataStructures(u);
5 repeat
6     repeat
7         foreach node c, where $c \notin PS$ do
8             S(c) ← ComputeShadowNodeset();
9         end
10        Select node c with smallest |S(c)| as next probe station;
11        UpdateDataStructures(c);
12        if|PS|←MAXPSSETSIZE then
13            return PS (Insufficient probe station set size);
14        end
15    until|SN|=0;
16    ResetShadowNodeSets();
17 until|SN|=0;
18 return PS as the probe station locations;
```

Procedure `CreatedataStructures`

```
begin
    PS←NULL;
    SN←V
    foreach node $w \in V$ do
        parents(w)←NULL;
  end
end
```

Procedure `SelectFirstProbeStation`

```
Select node u with highest node degree as first probe station :
return u :
```

Procedure `InitializeDataStructures` (*node u*)

```
Add u to PS:
Remove node u from SN :
Remove each node $v \in$ Neighbor(u) from SN :
foreach node $w \notin PS$ do
    Parents(w)←ST(u).Parent(w) :
end
```

Algorithm SNR: continued

```
Procedure ComputeShadowNodeSet (node c)
  S(c)←Null :
  foreach node $x \in$ V do
      if$((x \neq c)$ & $(x \notin Neighbor(c))$ & $(x \in SN)$ & $(ST(c)\ parent(x) \in parents(x)))$ 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 $\notin PS$ do
      Parents(y)←Parents(y) ∪ ST(c).Parents(y)
end
```

```
Procedure ResetShadowNodeSets
  SN←NULL:
  foreach node z do
      if$((z \notin PS)$ & $(z \notin \{\forall_{(p \in PS)} Neighbor(p)\})$ & $(|Parents(z)| < MAXFAULTS))$ then
          Add z to SN
      end
end
```

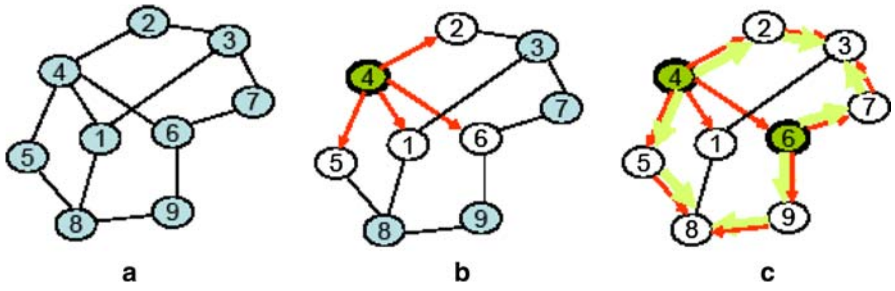

Fig. 2 **a** Original network topology with all nodes as shadow nodes; **b** Probe station placement at node 4 removing neighboring nodes from shadow node set; **c** Probe station placement at node 6 removing all nodes from shadow node set

2.8 Algorithm to Localize Probe Station Failures Along with Non-probe Station Node Failures

Algorithm SNR builds the probe station set assuming that probe stations do not fail. Thus no provision is made to localize a probe station failure.

However in a real network, these probe stations are also subject to failure, and thus probe stations should be selected such that a probe station node failure can 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. In Algorithm SNR, probe paths from a probe station to other probe station nodes were not considered. To localize a probe station failure, each probe station should also be probed by k independent probe paths.
2. In Algorithm SNR, all neighboring nodes to a probe station were assumed to be uniquely diagnosable and thus k independent paths were not found for such nodes. However, in a scenario where probe stations can fail, this assumption can not be made for neighboring nodes of the probe stations. Such nodes should also have k independent probe paths.
3. Till now, node failures could be localized for nodes with any degree. Nodes with degree less than k were made probe station neighbors as such nodes cannot have k independent probe paths. However when probe stations can also fail, this approach can not be used. Thus there may exist scenarios of k failures in which the failure of a node with degree less than k cannot be localized. One way to deal with nodes with smaller degree, can be to include a node with degree $d < k$ in the shadow nodes formed by a probe station only till d unique paths are found for such a node.

We present Algorithm SNR-PSF that deals with probe station failures and small degree nodes by modifying some of the functions of Algorithm SNR. The algorithm still incrementally finds independent paths for each node. However, unlike the previous algorithm, independent paths are searched for probe station nodes and their neighbors as well. This is done by including probe station and their neighbors in the shadow node set. Nodes with degree $d < k$, are included in the shadow node set till d unique probe paths are found for such nodes. To implement this functionality Algorithm SNR-PSF makes following modifications to Algorithm SNR.

Function InitializeDataStructurs() does not remove the first probe station and its neighbors from the shadow node set. Also the set Parents is computed for probe station nodes as well. Function ComputeShadowNodeSet() considers probe station c and its neighbors while building the shadow node set. Node with degree $d < k$ is included in shadow node set computation till number of parents for that node is less than d. While updating data-structures, set Parents is updated for probe station nodes as well. While resetting the shadow node set for next iteration, probe stations and probe station neighbors are also considered. Node with degree $d < k$ is included only if less than d parents exist in the node's parent set.

Algorithm SNR-PSF: Shadow Node Reduction with Probe Station Failures

Probe station selection to localize any k faults in the network limited probe station failures;

The algorithm is same as Algorithm SNR. However,some of the procedures used are implemented differently, as explained below in the function definitions;

Procedure `CreateDataStructures`

```
  This procedure is the same as one used by Algorithm SNR.
```

Procedure `SelectFirstProbeStation`

```
  This procedure is the same as one used by Algorithm SNR.
```

Procedure `InitializeDataStructures` (*node u*)

```
  Add u to PS
  foreach node w in V do
        Parents(w) ← ST(u)Parent(w);
  end
```

Procedure `ComputeShadowNodeSet` (*node C*)

```
   S(c) ← Null;
    foreach node x ∈ V do
    if ((n ∈ SN) & ((|Parents(x)| < degree(x)) & (ST(c) Parents(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 ∈ V do
         Parents(y) ← (parents(y) ∪ ST(c).Paernt(y);
   end
```

Procedure `ResetShadowNodeSets`

```
Update SN as follows;
  SN ← NULL foreach node z do
       if ((|Parents(z)| < MAXFAULTS) & (\Parents(z)\ < degree(z)) then
               Add z to SN;
       end
end
```

2.9 Algorithm to Localize Both Probe Station and Non-probe Station Node Failures with Different Node Failure-Limits

Algorithm SNR-PSF can be further modified by placing different limits on failure of probe station nodes and non probe station nodes. It may be the case that, once probe station nodes have been selected, extra resources can be committed to make these nodes more robust against failures, so that probe station nodes might have a lower probability of failure than non-probe-station nodes. We can then set different limits for maximum number of failures for probe stations and non-probe stations. We assume a limit l for probe station failures, where $l < k$, and $k - 1$ for non probe station failures. Thus each node requires k independent probe paths. However, note that nodes whose $l + 1$ neighbors are probe stations do not need k independent probe paths because, out of $l + 1$ probe station neighbors of the node, even if l probe station nodes fail, there will exist one probe station that can diagnose health of the node by sending a direct probe to it.

This in turn also allows nodes with degree less than k to be treated in a different manner. A node with degree less than k can be completely diagnosed if it has a degree $>l$, by having $l + 1$ neighbors as probe stations. Even for nodes with degree greater than k, if a node has $l + 1$ neighbors as probe stations then it does not need k independent probe paths and thus can be removed from the shadow node set.

Consider the example network shown in Fig. 1c. Assume that for this network $k = 3$ and $l = 1$, i.e., at most three nodes can fail such that maximum two non probe station nodes and one probe station node can fail. Node 3 has a degree two which is less than k. Thus node 3 can not have k independent probe paths. However if node 3 has $l + 1$, i.e., 2 neighbors as probe station nodes, then health of node 3 can always be determined.

The modified approach is presented in Algorithm SNR-CPSF by changing some of the functions of Algorithm SNR. The algorithm now maintains a count of neighboring probe stations for each node in the data structure NeighboringPSCount and excludes a node from the shadow node set when this count exceeds l. Functions CreateDataStructures(), InitializeDataStructures(), and UpdateDataStructures() would maintain the datastructure NeighboringPSCount for each node along with other data structures.

In functions ComputeShadowNodeSet() and ResetShadowNodeSets(), a node with degree degree(n) <= MAXPSFAULTS is included in the probe station selection criterion only till |Parents(n)| <degree(n), i.e. till degree(n) unique probe paths are found for them. Nodes with degree greater than MAXPSFAULTS are included till enough probe paths and probe stations are selected such that they are completely diagnosed.

Algorithm SNR-CPSF: Shadow Node Reduction with Constrained Probe Station Failures

Probe station selection to localize any k faults in the network including limited probe station failures:
The algorithm is same as Algorithm SNR. However, some of the procedures used are implemented differently, as explained below in the function definitions;

Procedure `CreateDataStructures`

```
begin
    PS ← NULL;
    SN ← V;
    foreach node w ∈ V do
        Parents(w) ← NULL;
    end
    foreach node u ∈ V do
        NeighboringPSCount(u) ← 0;
    end
end
```

Procedure `SelectFirstProbeStation`

This procedure is the same as one used by Algorithm SNR

Procedure `InitializeDataStructures` (*node u*)

```
  Add u to PS;
  foreach node w ∈ V do
        Parents(w) ← ST(u).Parent(w);
  end
foreach node u ∈ V do
        if n ∈ Neighbor(p) then
                NeighboringPSCount(n)++;
        end
end
```

Procedure `ComputeShadowNodeSet` (*node c*)

```
  S(c) ← Null;
  foreach node x ∈ V do
        if ((n ∈ SN) & ((|Parents(n)| < degree(n)) & (degree(n) <= MAXPSFAULTS))| (degree(n)
           > MAXPSFAULTS)& (ST(c).parent(n) ∈ Parents(n)))
        then
                Add x to S(c) ;
        end
  end
  return S(c)
```

Procedure `UpdateDataStructures` (*node c*)

```
  Add c to PS;
  SN ← S(c);
```

Algorithm SNR-CPSF: continued

```
foreach node $y \in V$ do
    Parents(y) ← (Parents(y) ∪ ST(c).Parent(y);
end
foreach node $u$ do
    if $u \in Neighbor(c)$ then
        NeighboringPSCount(u)++;
    end
end
```

Procedure `ResetShadowNodeSets`

```
SN ← NULL ;
foreach node $z$ do
    if $((|Parents(u)| < MAXFAULTS)$ & $(|\text{Parents(u)}| < degree(u))$ & $(NeighboringPSCount(n) <= MAXPSFAULTS))$ then
        Add z to SN;
    end
end
```

3 Simulation Results

In this section we present experimental evaluation of algorithms for probe station selection introduced in this paper. We apply the algorithms to build a probe station set that can localize at most *k* node failures, where the value of *k* is pre-defined for each execution. We present results for two sets of experiments. In the first set of experiments we assume that probe station nodes do not fail. We then perform another set of experiments where we consider probe station failures along with non probe station node failures. As the exhaustive search algorithm provides the smallest probe station set that can localize at most *k* failures, we use its result as a benchmark to compare size of the probe set built using the heuristic-based algorithm presented in this paper. Because of its high computational complexity, we were not able to run the exhaustive search algorithm for larger networks. For evaluation of the proposed algorithm on larger networks, we compared the Algorithm 1 with the algorithm where probes stations are selected at random locations in network till the desired diagnostic power is not obtained. Thus the random algorithm selects a node location as a probe station randomly and updates the shadow node set. The algorithm keeps selecting probe stations in this fashion till it achieves a null shadow node set.

3.1 Simulation Model

Let MD represent the maximum node degree in the network, AD represent the average node degree, and N represent the total number of nodes in the network. 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 size ranging from 20 to 100 with average network degrees ranging from 6 to 9 and maximum node degree set to min (20, network size). Each point plotted on the graph is an average of 20 runs. We compared the probe station set size and execution time of the SNR, SNR-PSF, and SNR-CPSF algorithms, with corresponding Exhaustive and Random placement algorithms.

In the experiments presented in Figs. 3, 4 we present results for SNR, Exhaustive and Random algorithm. We assume that probe station nodes do not fail, and set the maximum number of non-probe-station node failures to 4. The graphs shown in Figs. 5, 6 present the experiment results of the Algorithm SNR-PSF, Exhaustive and Random algorithm assuming at most 4 node failures and also considering probe station failures. The graphs shown in Figs. 7, 8 show the results of SNR-CPSF, Exhaustive and Random algorithms where we place explicit limits for probe station and non probe station node failures, making the placement resilient to at most 2 probe station failures and 2 non-probe-station node failures.

Figures 3, 5, 7 compare the performance of SNR, SNR-PSF, SNR-CPSF algorithms with the corresponding Exhaustive search algorithms. The graphs show that SNR algorithms results in probe station sets of sizes close to the optimal probe station set sizes computed by the respective Exhaustive search algorithms. Moreover the SNR algorithms run in significantly less time than the Exhaustive search algorithm.

Figures 4, 6, 8 compare the SNR, SNR-PSF, and SNR-CPSF algorithms with the corresponding Random placement algorithms. The results show that the probe station set size computed by SNR algorithms is significantly smaller than the Random placement algorithms, by taking a marginally longer time than the Random placement algorithms. Comparing the results of Fig. 3 with Figs. 5, 7 and the results of Fig. 4 with Figs. 6, 8, we can see that more probe stations are needed to make the localization resilient to probe station failures, as compared to the case where probe station failure is not considered.

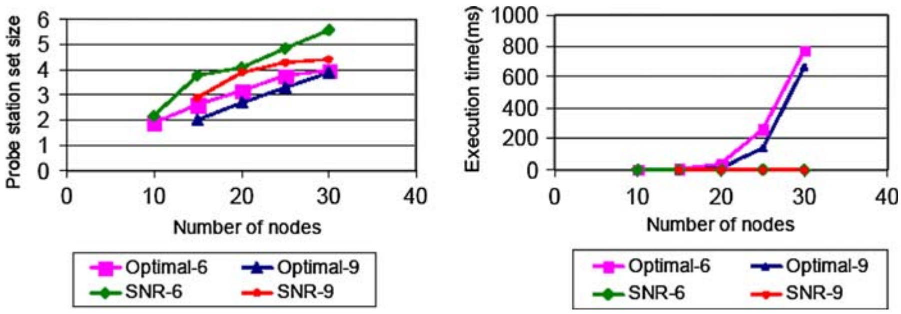

Fig. 3 Comparison of Exhaustive and SNR algorithm assuming no probe station failures on various networks with average node degree 6 and 9

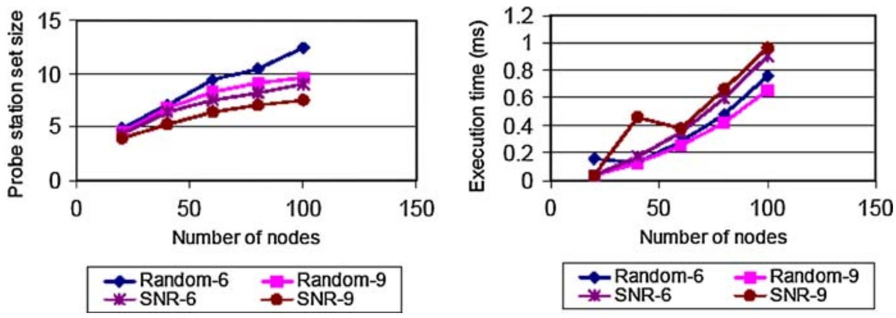

Fig. 4 Comparison of Random and SNR algorithm assuming no probe station failures on various networks with average node degree 6 and 9

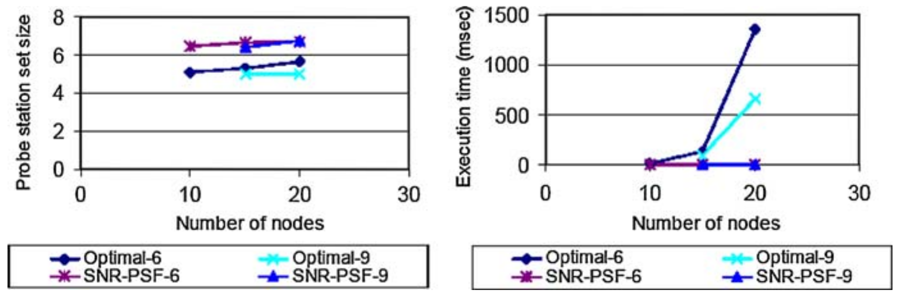

Fig. 5 Comparison of Exhaustive and SNR-PSF algorithm assuming probe station failures on various networks with average node degree 6 and 9

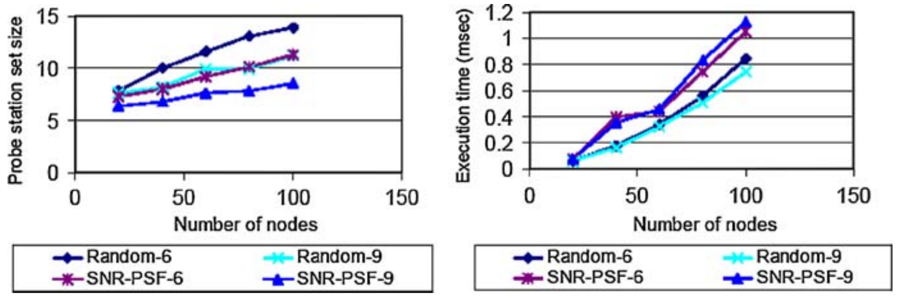

Fig. 6 Comparison of Random and SNR-PSF algorithm assuming probe station failures on various networks with average node degree 6 and 9

We compare SNR, SNR-PSF, and SNR-CPSF algorithms in Fig. 9. We can see that probe station set size computed by Algorithm SNR is smaller than that computed by Algorithm SNR-PSF, and Algorithm SNR-CPSF. This is because less probe stations need to be deployed when probe station failures are not considered. Algorithm SNR also takes advantage of probe station neighbor node property to reduce the probe station set size. Algorithm SNR-CPSF forms larger probe station set than SNR-PSF because Algorithm SNR-PSF does not ensure monitoring of

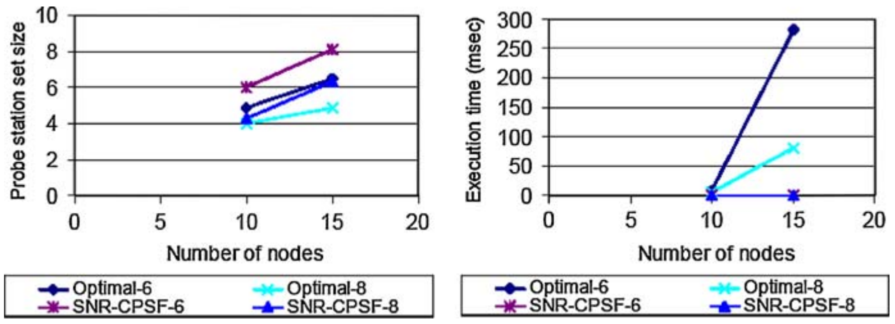

Fig. 7 Comparison of Exhaustive and SNR-CPSF algorithm assuming probe station failures on various networks with average node degree 6 and 8

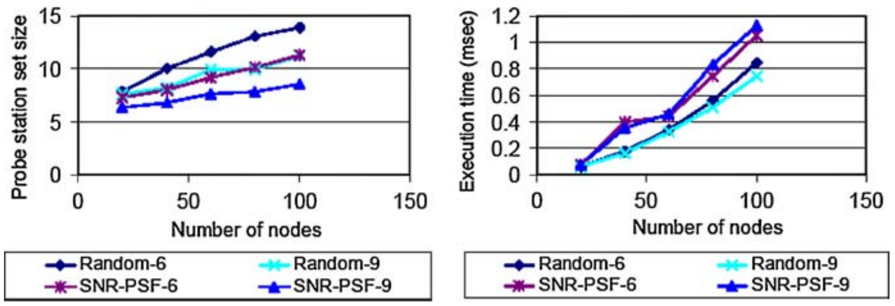

Fig. 8 Comparison of Random and SNR-CPSF algorithm assuming probe station failures on various networks with average node degree 6 and 9

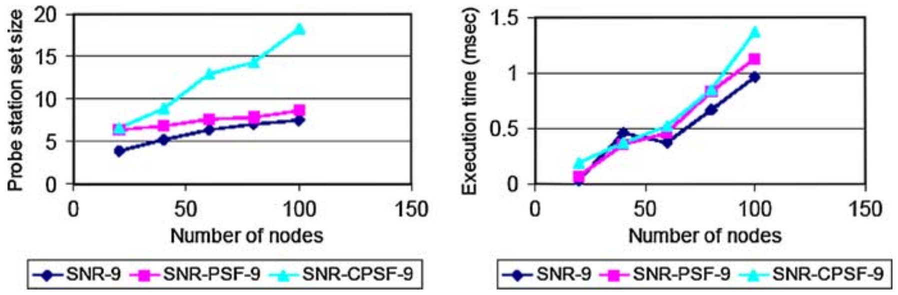

Fig. 9 Comparison of SNR, SNR-PSF and SNR-CPSF algorithms on various networks with average node degree 6 and 9

nodes with degree less than 4. On the other hand, since Algorithm SNR-CPSF assumes at most 2 probe station failures, it deals with nodes with degree less than 4 but greater than 1, by placing probe stations at the neighboring nodes of such weakly connected nodes. Algorithm SNR-CPSF provides more exhaustive monitoring capacity but in turn increases the size of probe station set.

4 Conclusions

We have addressed the problem of selecting suitable locations for probe station deployment to monitor the network even in presence of node failures. We discussed the design issues in probe station selection. We proposed a heuristic based approach of selecting node locations that minimize shadow nodes, and presented algorithm based on it for minimal probe station placement. We also addressed the case of probe station failure, and presented algorithms to make the probe station placement resilient to probe station failures. We proposed different optimizations in dealing with specific cases like weakly connected nodes, neighbor nodes, etc. We presented an experimental analysis of the algorithms through simulation results, and discussed the effectiveness and efficiency of different proposed algorithms.

References

1. Stallings, W.: SNMP, SNMPv2, SNMPv3, and RMON 1 and 2. 3rd edn. Addison-Wesley Longman Inc (1999)
2. Netflow Services and Applications. Cisco Systems (1999)
3. Stemm, M., Kaltz, R., Seshan, S.: A network measurement architecture for adaptive applications. In: IEEE INFOCOM, Mar 2000
4. Downey, B.: Using pathchar to estimate internet link characteristics. In: ACM SIGCOMM, Cambridge, MA (1999)
5. Dovrolis, C., Ramanathan, P., Moore, D.: What do packet dispersion techniques measure? In: IEEE INFOCOM'01, Alaska, Apr 2001
6. Ribeiro, V.J., Riedi, R.H., Baraniuk, R.G., Navratil, J., Cottrell, L.: Pathchirp: Efficient available bandwidth estimation for network paths. In: Passive and Active Measurement Workshop, 2003
7. Lai, K., Baker, M.: Measuring bandwidth. In: IEEE INFOCOM'99, New York City, NY, Mar 1999
8. Brodie, M., Rish, I., Ma, S., Grabarnik, G., Odintsova, N.: Active probing. Technical report, IBM, 2002
9. Natu, M., Sethi, A.S.: Active probing approach for fault localization in computer networks. In: E2EMON'06, Vancouver, Canada, 2006
10. Jamin, S., Jin, C., Jin, Y., Raz, Y., Shavitt, Y., Zhang, L.: On the placement of internet instrumentation. In: IEEE INFOCOM, Tel Aviv, Israel, Mar 2000
11. Francis, P., Jamin, S., Paxson, V., Zhang, L., Gryniewicz, D. F., Jin, Y.: An architecture for a global internet host distance estimation service. In: IEEE INFOCOM'99, Mar 1999
12. Theilman, W., Rothermel, K.: Dynamic distance maps of the internet. In: IEEE INFOCOM'2000, Mar 2000
13. Jacobsen, V.: Pathchar—a tool to infer characteristics of internet paths. April 1997
14. Bolot, J.C.: End-to-end packet delay and loss behavior in the internet. In: ACM SIGCOMM'93, Sep 1993
15. Breitbart, Y., Chong, C.Y., Garofalakis, M., Rastogi, R., Silberschatz, A.: Efficiently monitoring bandwidth and latency in ip networks. In: IEEE INFOCOM, Tel Aviv, Israel, Mar 2000
16. Li, F., Thottan, M.: End-to-end service quality measurement using source-routed probes. In: 25th Annual IEEE Conference on Computer Communications (INFOCOM), Barcelona, Spain, Apr 2006
17. Horton, J.D., Lopez-Ortiz, A.: On the number of distributed measurement points for network tomography. In: Internet Measurement Conference, IMC, 2003
18. Kumar, R., Kaur, J.: Efficient beacon placement for network tomography. In: Internet Measurement Conference, IMC, 2004
19. Reddy, A., Govindan, R., Estrin, D.: Fault isolation in multicast trees. In: ACM SIGCOMM, 2000
20. Adler, M., Bu, T., Sitaraman, R., Towsley, D.: Tree layout for internet network characterizations in multicast networks. In: NGC'01, London, UK, Nov 2001
21. Bejerano Y, Rajeev Rastogi.: Robust monitoring of link delays and faults in ip networks. In: IEEE INFOCOM, San Francisco, CA, Mar 2003

Author Biographies

Maitreya Natu received the M.S. and the Ph.D. degree in Computer and Information Sciences, both from the University of Delaware, Newark. He is current working as a scientist in Tata Research Design and Development Centre (TRDDC), Pune, India. His research interests include network management, fault management, management of enterprise systems, and network security.

Adarshpal S. Sethi is a Professor in the Department of Computer and Information Sciences at the University of Delaware, Newark, Delaware, USA. He has an M.Tech. degree in Electrical Engineering and a PhD in Computer Science, both from the Indian Institute of Technology, Kanpur, India. He has served on the faculty at IIT Kanpur, was a visiting faculty at Washington State University, Pullman, WA, and Visiting Scientist at IBM Research Laboratories, Zurich, Switzerland, and at the US Army Research Laboratory, Aberdeen, MD. He is on the editorial boards of the IEEE Transactions on Network and Service Management, International Journal of Network Management, and Electronic Commerce Research Journal. He is also active on the program committees of numerous conferences. His research interests include architectures and protocols for network management, fault management, quality-of-service and resource management, and management of wireless networks.