Telecommun Syst (2008) 37: 109–125
DOI 10.1007/s11235-008-9069-1

Efficient probe selection algorithms for fault diagnosis

Maitreya Natu · Adarshpal S. Sethi · Errol L. Lloyd

Published online: 16 April 2008

Abstract Increase in the network usage for more and more performance critical applications has caused a demand for tools that can monitor network health with minimum management traffic. Adaptive probing has the potential to provide effective tools for end-to-end monitoring and fault diagnosis over a network. Adaptive probing based algorithms adapt the probe set to localize faults in the network by sending less probes in healthy areas and more probes in the suspected areas of failure. In this paper we present adaptive probing tools that meet the requirements to provide an effective and efficient solution for fault diagnosis for modern communication systems. We present a system architecture for adaptive probing based fault diagnosis tool and propose algorithms for probe selection to perform failure detection and fault localization. We compare the performance and efficiency of the proposed algorithms through simulation results.

Keywords Adaptive probing · Probe selection · Network monitoring · Failure detection · Fault localization

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

M. Natu (✉) · A.S. Sethi · E.L. Lloyd
Department of Computer and Information Sciences, University of Delaware, Newark, DE 19716, USA
e-mail: natu@cis.udel.edu

A.S. Sethi
e-mail: sethi@cis.udel.edu

E.L. Lloyd
e-mail: elloyd@cis.udel.edu

1 Introduction

Fault diagnosis is the process of detecting a failure and localizing the cause of the failure from observed failure indications. With the widespread usage of computer networks in e-commerce, telecommuting, web services, grid services, etc., fault diagnosis has become a vital task for network administrators. Furthermore, increasing advances in developing performance critical applications, increasing importance on quality of service, and growth of large and complex systems make quick detection and isolation of faults essential for robustness, reliability, and system accessibility.

A promising approach to fault diagnosis is *probing*, which is based on actively sending out probes in the network to infer the health of network components. Probes are test transactions whose success or failure depends on the health of the probed network components. Probing is typically used to obtain end-to-end statistics such as latency, throughput, loss, and route availability. For instance, a probe could be a *ping* to collect information about connectivity of nodes. More sophisticated probes could be a train of probe packets of various sizes to collect information about the total bandwidth, or available bandwidth [17, 20, 28]. Probes could also be HTTP requests to specific servers to collect information about the response time statistics for fault diagnosis [14]. Besides generating traffic between selected points of interest to obtain desired end-to-end statistics, probing provides other advantages. Probing provides flexibility in the design of probe streams with particular properties to match measurement requirements. For example, after localizing a fault to a particular network point by measuring quantities such as average delay and loss on a route, finer probes can be sent to identify the bottleneck and available bandwidth [18, 28], or to estimate cross traffic [27]. Another application of this

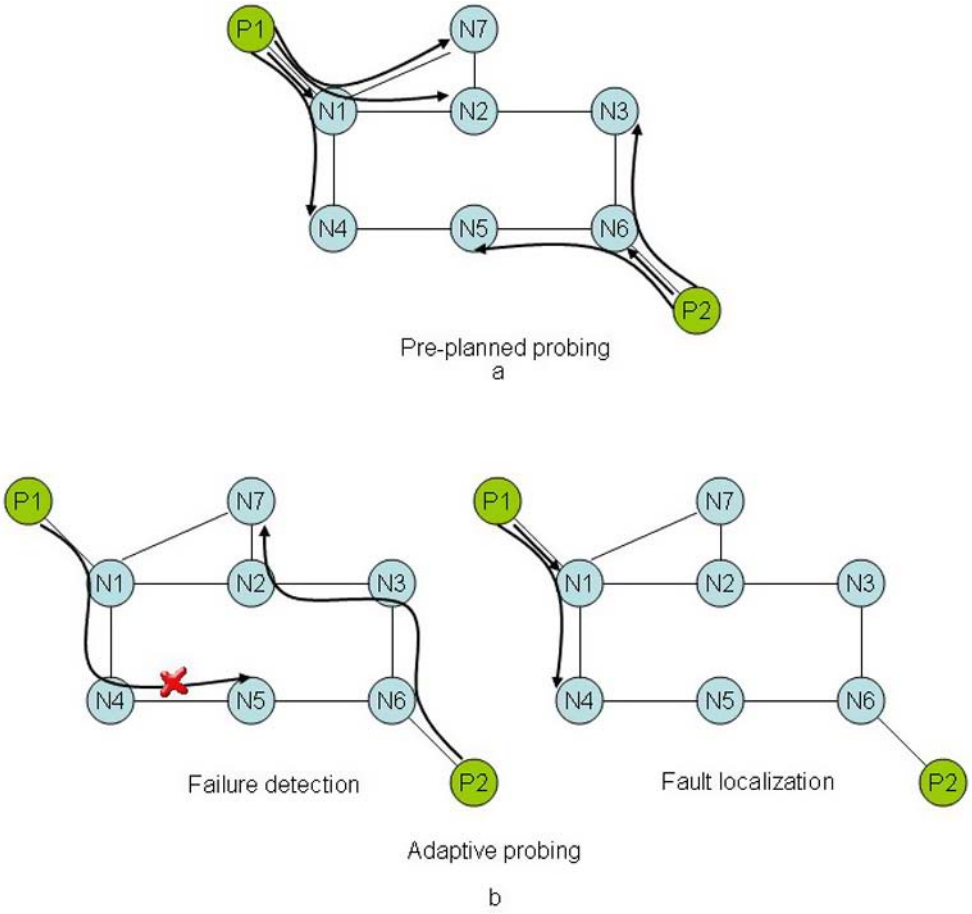

Fig. 1 Example of preplanned and adaptive probing to localize one node failure. Nodes P1 and P2 are probe station nodes. (**a**) Network nodes probed by a large preplanned set of probes such that any node failure can be diagnosed from the probe results. (**b**) Network nodes probed using adaptive probing, where first a small set of probes is sent to detect a failure in the network. On observing a failure on the path from P1 to N5, additional probes are sent for localizing the failed node on this path

feature of probing is to localize a bottleneck node through probes, and then send specialized probes to further diagnose the exact cause of failure at a finer level such as the system or middleware level within the node. A drawback of probing is its invasive character. Probes may modify route conditions and perturb the very traffic being monitored (a phenomenon known as *artifact*). To minimize artifact, it is essential to use probe streams of low average bandwidth, and to optimize the probe traffic.

Probing has been used broadly in two ways: *preplanned probing* and *adaptive probing*. Preplanned probing [5, 6] involves offline selection of a set of probes which are selected such that all possible failures can be diagnosed by observing the probe results. These probes are periodically sent out over the network. Then a passive data mining phase infers the network state by analyzing the probe results. A major drawback of this approach is the large amount of management traffic sent out over the network, a large part of which might be wasteful as not all failures occur all the time. Other drawbacks of this approach include the difficulty involved in envisaging such a probe set, and the delay involved in sending a large set of probes, and collecting the probe results. These drawbacks lead to inaccuracies and increased localization time.

One promising approach to effective and efficient fault diagnosis is *adaptive probing*, where a probe set is adapted to the observed network conditions by sending fewer probes in healthy areas and more probes in suspicious areas. Instead of sending probes to localize all potential failures, a small set of probes is first sent to monitor the health of all network components. These probes detect the presence of a failure, but they are not comprehensive enough to identify the failure's exact location. Based on the probe results, suspected areas of failure are identified, and additional probes are selected that further help to localize the exact cause of failure. Figure 1 shows an example of probes sent using preplanned probing and adaptive probing to localize one node failure in the network. Figure 1a shows an example of preplanned probing where a large number of probes are sent over the network to localize any node failure in the network. Figure 1b shows probes sent in an adaptive manner where first a small set of probes is sent to detect any node failure in the network. On observing a failed probe path (path P1 to N5), additional probes are sent over this failed probe path for further diagnosis.

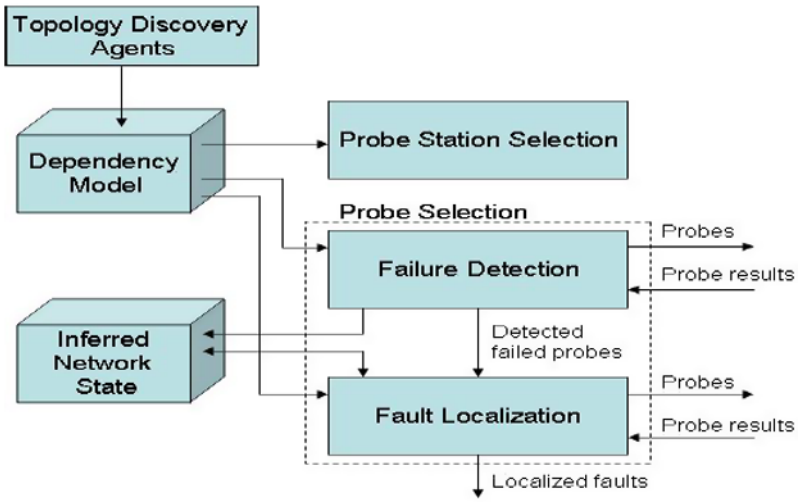

Fig. 2 System architecture for fault diagnosis using adaptive probing

2 System architecture

In this section, we present a system architecture for an adaptive-probing-based fault diagnosis methodology. Figure 2 presents the components of the proposed system architecture. Below we describe these components:

- Probe Station Selection: This component determines the locations in the network where probe stations should be deployed. A minimum requirement is that probe stations should be deployed such that probes could be sent from these locations to monitor the entire network for faults even in the presence of multiple failures.
- Probe Selection: Once probe stations are selected, out of all available probes that can be sent from these probe stations, the Probe Selection component selects suitable probes to be sent over the network. Using adaptive probing, we divide the probing task into two sub-tasks which we call *Failure Detection* and *Fault Localization*.
  - The *Failure Detection* module periodically sends a small number of probes such that the failure of any managed component can be detected. These probes are few in number, and are designed to only detect the presence of a failure in the network. They might be unable to exactly localize a failure.
  - Once a failure is detected in a network, the *Fault Localization* component analyzes probe results, and selects additional probes that provide maximum information about the suspected area of the network. This process of probe analysis and selection is performed to localize the exact cause of failure. Figure 1b presents an example of probes sent for failure detection and fault localization.
- Topology Discovery Agents: The Probe Station Selection and Probe Selection modules of the architecture use dependencies between probes and nodes on the probe path. The dependency information between probes and nodes is obtained through *Topology Discovery Agents*, and is stored in a dependency model. The collection of dependency information could be done in a variety of ways. For instance, the dependency between the end-to-end network probe path and the network nodes can be obtained using traceroutes or the topology tables of the underlying network layer.
- Dependency Model: This model stores the relationships between probes and nodes. In particular, it maintains information about the nodes that are visited by the probes, so that the health of nodes can be inferred from the success or failure of the probes. The dependency model could be deterministic when complete and accurate information is available about the probe paths. For situations in which complete and accurate information about the probe paths is unavailable, a probabilistic model is used where the dependency between probe paths and nodes is represented using the probability of causal implication of a node failure on a probe failure.
- Inferred Network State: The Failure Detection and Fault Localization components of the architecture store intermediate diagnosis results in the Inferred Network State. These results are refined by the Fault Localization component by sending and analyzing additional probes.

3 Related work

Various approaches have been proposed for probing a network to measure different performance metrics. Probing tools proposed in the past consist of connectivity, latency and bandwidth measurement tools such as pathchar [13], Cprobe [8], Skitter [19], pathrate [12], PathChirp [29], NetTimer [21] etc. These tools measure end-to-end network performance metrics by sending probes in the network. For instance, PathChirp [29] is a 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. Skitter [19] sends a sequence of probes to a set of destinations and measures the latency of a link as the difference in round-trip times of the two probes to link end-points.

Probes have also been used in overlay networks. Accurate monitoring systems in overlays can detect loss, link stress, path outages, and periods of degraded performance facilitating management of distributed systems. In order to create a good overlay topology in dynamic environments, probes have been used by overlay nodes to periodically monitor the quality of paths from overlay nodes to other nodes. Narada [16] and RON [2] employ a straight-forward approach of pair-wise probing. This approach causes a significant probing overhead and can incur high link stress. Several methods have been proposed in the past to reduce probing overhead in overlay networks. Most of these approaches reduce probing overhead at the cost of probing completeness. In structured peer-to-peer systems, a node maintains connections to $O(\log(n))$ [31, 35] or $O(n^{1/d})$ [26] neighbors. Scalable multicast systems such as NICE [3] and HMTP [34] organize nodes in a hierarchy based on their distances to each other. Each node periodically selects a particular set of nodes to probe, usually keeping the overall probing overhead constant. In [9, 10], a basis set of paths is selected to monitor loss rates and is used to infer loss rates of all end-to-end paths from this basis set. An approach proposed in [32] also trades probing overhead for estimation accuracy. It uses network-level path composition information to infer path quality without full-scale probing.

Various criteria have been used in the past for probe selection. Li and Thotten [22] propose to use source routed probes to measure end-to-end performance metrics. Bejarano and Rastogi [4] propose a probe selection algorithm for monitoring network links based on a greedy heuristic of selecting a probe that covers maximum number of uncovered network components. Ozmutlu et al. [25] propose to identify zones in topologies using end-to-end delay data and common occurrences of high delays on paths and then selecting a subset of paths to probe to identify each zone uniquely. The methodology proposed in [7] relies on active perturbation of the system to identify dependencies and the use of statistical modeling to compute dependency strengths. Active Integrated fault Reasoning (AIR) proposed in [33], integrates the advantages of both passive and active monitoring into one framework. The AIR framework uses active probing to discover critical symptoms when passive reasoning is insufficient to explain the problem. It proposes a greedy algorithm to select probes that provide maximum coverage. On similar lines, both offline and online tools are used in [15] to analyze the dependency relationship among various software components. If the offline transaction log analysis cannot pinpoint the root cause of error, [15] proposes to run more transactions to collect sufficient symptoms for localizing the root cause. The transactions are chosen such that maximum information can be gained from the transaction result. Al-Shaer and Tang [1] propose a monitoring framework called SMRM (SNMP-based Multicast Reachability Monitoring), for actively monitoring the health and quality of service of multicast networks. The SMRM framework can be used to generate directed multicast traffic and collect real-time reports about the reachability and quality of the multicast delivery at any point or segment in enterprise networks.

In the past, some work has been done in optimizing the probe set to be used for monitoring purposes. Optimization of probes by using an interactive approach of adaptive probing has been proposed by I. Rish, M. Brodie and their colleagues [6, 30]. Brodie et al. [5, 6] proposed some general approaches to be used for probe set selection for both failure detection and fault localization. These approaches attempt to find a small set of probes using certain heuristics. In [5], a subtractive search has been proposed that starts with all probes, considers each probe in turn, and discards a probe if it does not add to the diagnostic capability of the probe set. The effectiveness of this approach in finding the minimal set depends on the order in which probes are explored. Another approach proposed in [5] is additive search, where at each step the probe giving the most informative decomposition is added to the probe set. In [30], Rish et al. have proposed an adaptive probing approach for fault localization. Optimized monitoring strategies have been proposed by Ozmutlu et al. [24] by formulating a graph-theoretic Constrained Coverage Problem to optimally select a subset of traceroute like probes to monitor the network. They propose a heuristic algorithm to efficiently approximate the probe selection solution in polynomial time.

Most of the probe selection work proposed in the past suffers from the following main limitations:

- A preplanned approach is used to build a probe set to localize all possible failures in the network. Besides the high probe traffic, analysis of these probes might take considerable time before diagnosing the problem. An adaptive approach is more suitable in terms of generated probe traffic, management overhead, and localization time.
- The existing algorithms suffer from efficiency issues.

An important contribution of this paper is an *adaptive probing based* algorithm for localizing faults. We demonstrate the efficiency and effectiveness of adaptive probing over preplanned probing through experimental evaluation.

4 Failure detection

In this section, we introduce a probe selection algorithm to perform failure detection. We use this algorithm together with the fault localization algorithms presented in Sect. 5 to develop a complete fault diagnosis tool.

Probes for failure detection should be selected such that, in the presence of a fault in the network, some of the selected probes should fail, causing the detection of the failure by the network manager. As the probes for failure detection are sent at periodic intervals, they should be optimized to prevent overwhelming the network resources and affecting the performance of other applications using the network. Ideally the aim is to find a minimal set of probes that can detect the health of all the managed components such that if any component fails, at least one of the probes should report failure. Informally, the probe set selection for failure detection problem is:

> *Given a set of probes and given the dependency information between the probes and the network nodes, select a smallest set of probes such that every node in the network is covered by some probe.*

In this section, we first prove that probe set selection for failure detection is NP-Complete. We then present an approximation algorithm for the selection of such a probe set.

4.1 Probe set selection for failure detection is NP-Complete

Consider a set of nodes N, each of which can be either up (functioning correctly), or down (functioning incorrectly). Consider a set of probes P. The set of nodes visited by a probe $p \in P$ is denoted by $S(p) \subseteq N$. A probe either succeeds or fails. A successful probe indicates that every node visited by the probe is healthy. On the other hand, if a node fails, then every probe visiting this node fails.

In this context then the probe set selection for failure detection problem is to select a smallest set of probes $Q \subseteq P$ such that every node $n \in N$ is visited by at least one of these probes. We refer to this probe set as the *detection set*.

In what follows, we prove the NP-Completeness of the MFDPS problem by reducing the Minimum Set Cover problem [11] to MFDPS. As per standard practice, we use the decision versions of the MFDPS and the Minimum Set Cover problem in the following proof. We extend the definition of *detection set* to refer to a probe set $Q \subseteq P$ of size k such that every node $n \in N$ is visited.

4.1.1 Definition of problem instances

- MFDPS problem:

  Instance: A set of nodes N, a set of probes P, a set $S(p) \subseteq N$ for each probe p indicating the nodes visited by probe $p \in P$, and an integer k.
  Question: Does there exist a *detection set* of size k? That is, does there exist a set of k probes $Q \subseteq P$, such that every node in N is visited by at least one of the probes in Q? In other words, $\bigcup_{q \in Q} S(q) = N$.

- Set cover problem:

  Instance: A finite set of elements E, a collection C of subsets of E, and an integer k.
  Question: Does there exist a subset $C' \subseteq C$ of size k such that every element in E is covered by at least one set in C'?

4.1.2 MFDPS is in NP

In this section, we prove that the MFDPS problem is in NP, by showing that a certificate for the problem can be verified in polynomial time. Here a certificate consists of a set of probes $Q \subseteq P$.

Verification that $|Q| \leq k$, can be done in $O(k)$ operations. To check if Q provides the *detection set*, we compute a set R as follows:

$$R = \bigcup_{q \in Q} S(q)$$

If $R = N$, then Q is the *detection set* for the MFDPS problem. The computation and verification of the set R can be done in $O(|N|)$ operations.

Since a certificate can be verified in polynomial time, it follows that MFDPS is in NP.

4.1.3 Reduction

To complete the proof of the NP-Completeness of the MFDPS problem, we provide a reduction from Minimum Set Cover to MFDPS such that there exists a set cover of size k for the Minimum Set Cover instance if and only if there exists a *detection set* of size k for the MFDPS instance.

Construction Given an instance of Minimum Set Cover problem (set of elements E, collection C of subsets of E, integer k), create an instance of MFDPS as follows:

- Create a set of nodes N such that there is one node in N for each element in E.
- Create a set of probes P such that there is one probe in P for each set in C.
- For probe $p \in P$ corresponding to a set $C_j \in C$, construct a set $S(p) \subseteq N$ such that a node $N_i \in S(p)$ if the corresponding element $E_i \in C_j$. Thus a probe p visits a node n if the set corresponding to probe p covers the element corresponding to node n.

Algorithm GFD: Greedy algorithm for probe set selection for failure detection

input : N: The set of nodes. PS: The set of probe stations. AvailableProbes: Set of probes that can be sent from probe stations to other nodes in the network.
output : Probe set for failure detection
1 Initialize a set NonProbedNodes to all non-probe station nodes, i.e., $N - PS$;
2 **while** ((NonProbedNodes $\neq$ Null)&(AvailableProbes $\neq$ Null)) **do**
3 Select a node m from NonProbedNodes, that is probed by least number of probes;
4 Out of all the probes in AvailableProbes probing node m, select a probe q that probes largest number of NonProbedNodes;
5 Remove all nodes probed by probe q from NonProbedNodes;
6 Remove probe q from AvailableProbes;
7 Add probe q to SelectedProbes;
8 **end**
9 **if** NonProbedNodes = Null **then**
10 Return the SelectedProbes;
11 **end**
12 **else**
13 Report "Insufficient probes";
14 **end**

Given a finite collection C and a finite set of elements E, the complexity of the construction is $O(|C| * |E|)$.

Proof Forward (MSC implies MFDPS): Here we prove that if there is a solution to the Minimum Set Cover instance then there must be a solution to the MFDPS instance.

A solution to Minimum Set Cover consists of a collection C' of k sets such that every element in E is covered by some set of C'. As explained in the construction, if a set C_j covers an element E_i, then the probe corresponding to the set C_j will visit the node corresponding to the element E_i. Thus, as k sets in C' cover every element in E, the k probes representing the sets in C' will visit every node in N. The probes representing the sets in C' thus form a detection set of size k.

Reverse (MFDPS implies MSC): Here we prove that if there is a solution to the MFDPS instance then there must be a solution to the Minimum Set Cover instance.

A solution to MFDPS consists of a detection set of size k which is a set of k probes such that every node in the set N is visited by these probes. As explained in the construction, a node n is visited by a probe p if the element corresponding to node n is covered by the set corresponding to probe p. Thus if k probes in the *detection set* visit all nodes then the k sets corresponding to these k probes will cover all elements represented by the nodes.

Thus the MFDPS problem is NP-Complete.

4.2 Heuristic-based algorithm for probe set selection for failure detection

In this section, we present a heuristic-based algorithm to select probes for failure detection. Different nodes are probed by different numbers of probes, depending on the routes used by the probes. Nodes that are probed by fewer probes narrow down the search space for probe selection. Consider the case where a node n is probed by only one probe. In this case, the only probe probing node n must always be selected, irrespective of the number of nodes it covers. Consider another case, where only two probes pass through a node n. Then one of the two probes must be selected to cover node n. Amongst the two probes, the probe covering a larger number of uncovered nodes is the better choice.

Algorithm GFD describes a Greedy approximation algorithm that explores the information contained in the dependencies between probes and network components. The algorithm selects the network element n which is probed by the least number of probes, using the dependency information between probes and probed elements. Out of all the probes probing element n, the algorithm selects the probe which goes through a maximum number of nodes that are not yet probed.

As an example, consider a dependency graph represented by the matrix in Fig. 3, where rows represent probes and columns represent nodes. $Cell(i, j) = 1$ indicates that probe i probes node j. In this matrix, node 1 is probed by only one probe, i.e., probe C. Thus probe C must be selected. Nodes 1 and 5 are probed by probe C. Out of the remaining nodes (i.e., nodes 2, 3, and 4), node 2 is probed by the least number of probes (probes A and B). Thus next probe should be selected to probe node 2. Probe A covers 2 non-probed nodes while probe B covers 3 non-probed nodes. Thus probe B is a better choice. Since all nodes are now covered, the algorithm ends with probes B and C as the final solution.

Considering the total number of nodes equals N and the total number of probes equals P, the algorithm performs $O(N)$ operations to choose a node to probe and $O(P)$ operations to select a probe in each iteration. The algorithm performs $O(N)$ iterations to select probes to cover all the nodes, making the total complexity of the algorithm $O(N * (N + P))$. Assuming an upper limit s on the number of probe stations, the total number of probes P is $O(s * N)$ with each of the s probe stations sending probes to reach other non-probe-station nodes. This makes the complexity of the algorithm $O(N^2)$. In Sect. 6 we provide simulation results that include an analysis of the performance of our algorithm GFD in comparison with an algorithm from [5].

5 Fault localization

In this section we present algorithms for probe selection for fault localization. We first describe the preplanned approach

Node/ Probe	1	2	3	4	5
A	0	1	0	1	0
B	0	1	1	1	0
C	1	0	0	0	1
D	0	0	1	1	1
E	0	0	1	0	1

a

Node/ Probe	2	3	4
A	1	0	1
B	1	1	1
D	0	1	1
E	0	1	0

b

Node/ Probe
A
D
E

c

Fig. 3 Matrix representing dependencies between probes and nodes such that $cell(i, j) = 1$ implies that probe i probes node j; (**a**) Selection of Probe C to probe Node 1; (**b**) Selection of Probe B to probe Node 2 on the reduced matrix; (**c**) No more uncovered nodes left in the matrix

for probe selection for fault localization and show that the problem of selecting such a probe set is NP-Complete. We then present adaptive probing algorithms for fault localization based on different heuristics.

5.1 Preplanned-probing-based fault localization

Preplanned probing proposes to select a set of probes such that any node failure can be uniquely diagnosed. To build such a probe set, the dependency matrix is used that contains dependencies between the probes represented by the rows and the nodes represented by the columns. Based on the dependencies, a scenario of node failure produces a specific pattern of probe outcomes. Thus each node failure represented by a column can be represented by a vector of probe outcomes. Assuming that each node failure is uniquely diagnosable from the probe outcomes, the failed node can be deduced by observing the vector of probe outcomes. However, there might not be the need to send all probes to perform such a diagnosis, as many probe outcomes might give redundant information and thus might not be needed for the unique diagnosis of node failures. Informally the probe set selection for fault localization problem is:

> *Given a set of probes and the dependencies between probes and node failure states, select the smallest set of probes such that every node failure state can be uniquely diagnosed.*

In this section, we prove that probe set selection for fault localization is NP-Complete. In the proof provided below we assume that the goal is localization of a single node failure. It follows immediately that the more general problem of localizing an arbitrary number of simultaneous node failures is also NP-complete.

5.1.1 *Probe set selection for fault localization is NP-Complete*

Consider a set of nodes N, each of which can either be up (functioning correctly), or down (functioning incorrectly). Consider a set of probes P. Each probe is represented by a binary string of length $|N|$. If a probe visits a node $N_j \in N$, then the string contains a 1 at position j, and a 0 otherwise. This defines a 0–1 dependency matrix D of size $|P| * |N|$ with rows representing probes and columns representing nodes, such that $D[i, j] = 1$ if probe i visits node j and 0 otherwise.

A probe either succeeds or fails. A successful probe indicates that every node it visits is healthy. On the other hand, if a node fails, then every probe visiting this node fails. Thus, the outcome of a set of r probes results in a binary string of length r with each digit indicating the success or failure of a probe. We refer to this binary string of probe results as a *probe result vector*. A node failure that generates a unique probe result vector relative to the probe result vectors of other nodes can be diagnosed unambiguously from the observed probe result vector. Here, we assume a single node failure at a given time.

In this context then the probe set selection for fault localization problem is to select a smallest set of probes $Q \subseteq P$ such that every node failure produces a unique probe result vector for the selected probes. We refer to this set of probes as the *localization set*.

In what follows, we prove the NP-Completeness of the MFLPS problem by reducing the Minimum Test Collection problem to MFLPS. As per standard practice, we use the decision versions of the MFLPS and the Minimum Test Collection problems. We extend the definition of *localization set* to refer to a probe set $Q \subseteq P$ of size k such that there exists a unique probe result vector for every node $n \in N$.

Definition of problem instances

- MFLPS problem:

Instance: A set of nodes N, a set of probes P, a 0–1 dependency matrix D, where $D[i, j] = 1$ if probe i visits node j, and 0 otherwise, and an integer k.
Question: Does there exist a *localization set* of size k? That is, does there exist a set of k probes $Q \subseteq P$, such that for the k probes there exists a unique probe result vector for every node?

- Minimum Test Collection problem:

Instance: A finite set of elements E, a collection C of subsets of E, and an integer k.
Question: Does there exist a collection $C' \subseteq C$ of k sets such that for each pair of distinct elements $E_i, E_j \in E$, there is some set $c \in C'$ that covers exactly one of E_i or E_j?

MFLPS is in NP

We now prove that the MFLPS problem is in NP by showing that a certificate for the problem can be verified in polynomial time. Here a certificate consists of a set of probes $Q \subseteq P$.

Verification that $|Q| \leq k$, can be done in $O(k)$ operations. The set Q generates $|N|$ probe result vectors each of size k. To check if Q provides the *localization set*, we verify that the $|N|$ probe result vectors are unique. This can be done in $O(k|N|^2)$ operations.

Since a certificate can be verified in polynomial time, it follows that MFLPS is in NP.

Reduction

To complete the proof of the NP-Completeness of the MFLPS problem, we provide a reduction from Minimum Test Collection to MFLPS such that there exists a test collection of size k for the Minimum Test Collection instance if and only if there exists a *localization set* of size k for the MFLPS instance.

Construction Given an instance of the Minimum Test Collection problem (set of elements E, collection C of subsets of E, integer k), create an instance of MFLPS as follows:

- Create a set of nodes N such that there is one node in N for each element in E.
- Create a set of probes P such that there is one probe in P for each set in C.
- Build a dependency matrix such that if set C_i covers an element E_j, then the probe corresponding to set C_i visits the node corresponding to element E_j. Thus, create a 0–1 dependency matrix D, such that $D[i, j] = 1$ if $E_j \in C_i$, and $D[i, j] = 0$ otherwise.

Given a finite collection C and a finite set of elements E, the complexity of matrix creation and population is $O(|C| * |E|)$.

Proof Forward (MTC implies MFLPS): Here we prove that if there is a solution to the Minimum Test Collection (MTC) instance then there must be a solution to the MFLPS instance.

A solution to MTC consists of a collection C' of k sets such that for every two elements $E_i, E_j \in E$ there is some set in C' that covers exactly one of E_i and E_j.

As explained in the construction, if a set C_i covers an element E_j, then $D[i, j] = 1$, and $D[i, j] = 0$ otherwise. Thus, if for each pair of elements there exists a set that covers exactly one of the two, then for each pair of nodes in the dependency matrix D, there must exist some probe entry that is different for the two nodes (1 for one node and 0 for the other). This makes the probe set corresponding to the solution of the MTC instance provide a unique probe result vector for each node, making the probe set the *localization set* for the MFLPS instance.

Reverse (MFLPS implies MTC): Here we prove that if there is a solution to the MFLPS instance then there must be a solution to the Minimum Test Collection instance.

A solution to MFLPS consists of a *localization set* of k probes such that every node in the set N produces a unique probe result vector for the k probes. If k probes in the localization set generate unique probe result vectors for each node, then for each pair of nodes, there must exist at least one probe in the localization set that provides different values to the probe result vector of the two nodes. Thus, for each pair of nodes, there must exist at least one probe that visits exactly one of the two nodes, thus providing '0' for one node and '1' for the other node's probe result vector.

As explained in the construction, a node n visited by a probe p corresponds to inclusion of the element corresponding to the node n in the set corresponding to the probe p. Thus, a 1 entry in a $D[i, j]$ corresponds to inclusion of the node j in the set i. If for a pair of nodes, there exists a probe p that visits exactly one of the two nodes, then for the corresponding pair of elements in the MTC instance the set corresponding to the probe p will cover exactly one of the two elements.

If k probes have the property that for every pair of nodes there exists a probe that visits exactly one of the two nodes, then the corresponding k sets in the MTC problem instance will hold the property that for every pair of elements, there exists a set that covers exactly one of the two elements.

Thus the MFLPS problem is NP-Complete.

In the past, heuristic-based algorithms have been proposed [5] for probe set selection for preplanned probing. In Sect. 6 we compare the adaptive probing algorithms presented in this paper with the preplanned probing algorithm presented in [5] and show that adaptive probing performs fault localization using significantly smaller number of probes than preplanned probing.

5.2 Adaptive-probing-based fault localization

We now present an adaptive probing approach for fault localization. The fault localization process is triggered when a failure in the network is detected by the failure of one or more detection probes. This failure indicates the presence of one or more faults over the failed probe paths. However these probes might not be able to locate the exact cause of failure. Hence additional probes must be sent over the identified problem areas to localize the fault. In this section, we present various approaches to select additional probes for localization to a finer granularity.

The probes for fault localization need to be sent such that the health of all the nodes on the failed probe paths can be determined. It should be noted that the presence of failures may cause certain nodes to be unreachable from certain probe paths. This is demonstrated by the example shown in Fig. 4. Figure 4 shows probe paths from probe stations 1 and 9 to other nodes in the network. Consider the probe path $9 \rightarrow 7 \rightarrow 5$. Failure of node 7 makes node 5 unreachable for the probe station node 9. Also, as certain node failures are identified, probe paths need to be selected to probe the rest of the nodes such that the probes do not pass through the already identified failed nodes. In this section we first present an approach to analyze probe results for inferring node health and then present various ways to select appropriate probes for localization. We compare the number of probes and time required by each of these methods to localize the faults in the network in Sect. 6.

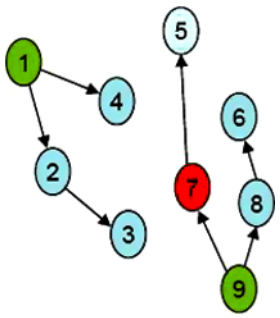

Fig. 4 Example shows probe station nodes 1 and 9 and probes sent from nodes 1 and 9 to other nodes. Failure of node 7 makes node 5 unreachable from probe station node 9

5.2.1 *Probe analysis*

The sets of failed, passed, and suspected nodes are maintained for analysis of the health of network nodes. The set of suspected nodes contains the nodes whose health needs to be determined. This suspected node set is initialized to all nodes that are present on the failed probe paths. The success and failure of the probes sent affect the sets of failed, passed, and suspected nodes. The nodes lying on the paths of successful probes are added to the set of passed nodes and removed from the set of suspected nodes. A node n is declared as failed and added to the failed nodes set when a failed probe goes through a set of nodes such that all nodes other than node n on that path have already been found to have good health. In other words, no other node on that path is present in the suspected node set. In each iteration, the algorithm builds a probe set to be sent over the network to determine the health of the remaining suspected nodes.

Figure 5 shows how the sets of suspected nodes, failed nodes, and passed nodes are maintained with the observed success and failure of probes. Figure 5 shows a network with nodes 1 and 7 as probe station nodes. Failure of probe $1 \rightarrow 8$ brings nodes 4, 5, 6, and 8 into the suspected node set. On observing success of probe $7 \rightarrow 4$, nodes 4 and 5 are removed from the set of suspected nodes and put into the set of passed nodes. Failure of probe $1 \rightarrow 6$, together with the information about the good health of nodes 4 and 5, narrows down the failure to node 6. Thus node 6 is removed from the set of suspected nodes and put into the set of failed nodes. Success of probe $7 \rightarrow 8$ brings node 8 in the set of passed nodes. Thus the probes classify the suspected nodes into the set of passed and failed nodes.

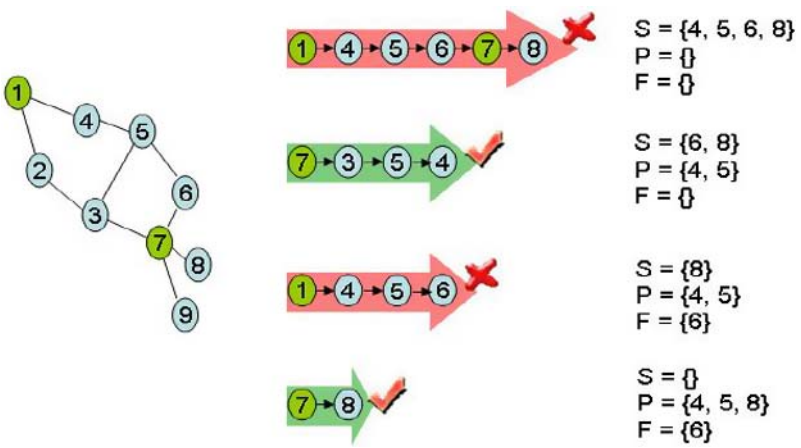

Fig. 5 Analysis of probe success and failure to classify suspected nodes into passed and failed nodes

Algorithm NM: Network Monitoring Algorithm (Algorithm to send probes for monitoring network health and localizing causes of failure)

```
   Input  : NetworkNodes, ProbeSet
   Output: FailedNodes
1  FDProbes ← Failure Detection Probes (from Algorithm 1);
2  while no failure do
3      Send FDProbes; Wait for t time units;
4  end
5  Identify the PassedProbes, FailedProbes;
6  FailedNodes ← NULL; PassedNodes ← NULL;
7  while |FailedProbes| ≠ NULL do
8      (FLProbes, FailedNodes, PassedNodes) ←
       GFL(FailedProbes, PassedProbes, FailedNodes,
       PassedNodes, ProbeSet);
9      Send FLProbes and identify PassedProbes and FailedProbes;
10     Remove probes ∈ FLProbes from ProbeSet;
11 end
12 Return FailedNodes and exit;
```

Algorithm GFL: Greedy Fault Localization Algorithm (Probe selection for fault localization in the network)

```
   Input  : FailedProbes, PassedProbes, FailedNodes,
            PassedNodes, ProbeSet
   Output: Probe set for further fault localization, FailedNodes,
            PassedNodes
1  Add nodes on the passed probe paths to PassedNodes;
2  Initialize a set SuspectedNodes to nodes that lie on failed probe
   paths and are not present in the PassedNodes set and are not the
   ProbeStationNodes;
3  foreach probe p ∈ FailedProbes do
4      PathSuspectedNodes ← ProbePathNodes(p) ∩
       SuspectedNodes;
5      if |PathSuspectedNodes| = 1 then
6          add PathSuspectedNodes to the FailedNodes;
7      end
8  end
9  Remove nodes ∈ FailedNodes from SuspectedNodes;
10 Build a set AvailableProbes to the probes from ProbeSet that
   pass through SuspectedNodes and do not pass through
   FailedNodes;
11 FLProbes ← SelectFLProbes (SuspectedNodes,
   AvailableProbes);
12 return FLProbes, FailedNodes, PassedNodes;
```

Procedure `SelectFLProbes` (SuspectedNodes, AvailableProbes) {Probe selection using Max search}

```
begin
    LocalizationProbes ← NULL;
    TargetNodes ← SuspectedNodes;
    ProbeSpace ← AvailableProbes;
    while |TargetNodes| ≠ NULL do
        NextProbe ← Probe ∈ ProbeSpace that covers
        maximum number of TargetNodes;
        Add NextProbe to LocalizationProbes;
        Remove NextProbe from ProbeSpace;
        Remove ProbedNodes(NextProbe) from TargetNodes;
    end
    return LocalizationProbes;
end
```

Procedure `SelectFLProbes` (SuspectedNodes, AvailableProbes) {Probe selection using Min search}

```
LocalizationProbes ← NULL;
TargetNodes ← SuspectedNodes;
ProbeSpace ← AvailableProbes;
foreach node n ∈ TargetNodes do
    NextProbe ← Probe that passes through node n and through
    minimum number of other TargetNodes;
    Add NextProbe to the LocalizationProbes;
    Remove NextProbe from ProbeSpace;
    Remove ProbedNodes(NextProbe) from TargetNodes;
end
Return LocalizationProbes;
```

5.2.2 *Greedy algorithm*

In this section we present algorithms based on the Greedy approach to build a probe set for fault localization. We first present Algorithm NM that describes the operations performed by the network manager. The network manager first sends the probe set for failure detection. If no failure is observed on these probes, the manager waits for some time and then resends the same probe set for failure detection. However, if a failure is observed, the manager performs deeper diagnosis of failed probe paths by executing Algorithm GFL (Greedy Fault Localization). Algorithm GFL receives the information about the passed and failed probes and any already collected information about the passed and failed nodes. The algorithm then updates the passed and failed nodes sets by analyzing the received information. The algorithm computes a set of probes to be sent for further analysis of nodes whose health is yet undetermined. The network manager sends these additional probes and repeats the same process of probe result analysis and new probe selection until the health of all nodes is determined.

We present two approaches to select the probes for probing the nodes in the suspected node set. One approach is to iteratively select a probe that covers a maximum number of suspected nodes as shown in Fig. 6b. The success of such a probe gives a large amount of information by removing all the nodes on that probe path from the suspected node set. However if the probe fails then the probe does not give much information to significantly narrow the search space of a failed node. For instance, in Fig. 7a success of probe $1 \rightarrow 8$ gives information about good health of nodes 4, 5, 6, and 8, while its failure does not narrow down the set of suspected nodes. The suspected nodes set would need more probes for further diagnosis. Hence another approach could be to select a probe for each suspected node such that it goes through the least number of other suspected nodes as shown in Fig. 6c. The success of such a probe gives the information about good health of a small number of nodes, reducing

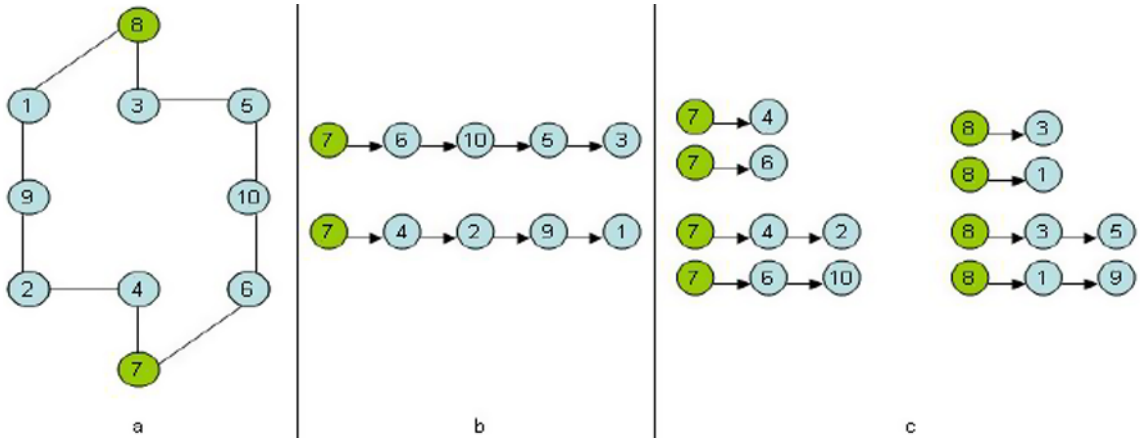

Fig. 6 (**a**) Network with nodes 7 and 8 as probe stations; (**b**) Nodes probed by a small set of long probes; (**c**) Nodes probed by a large set of short probes

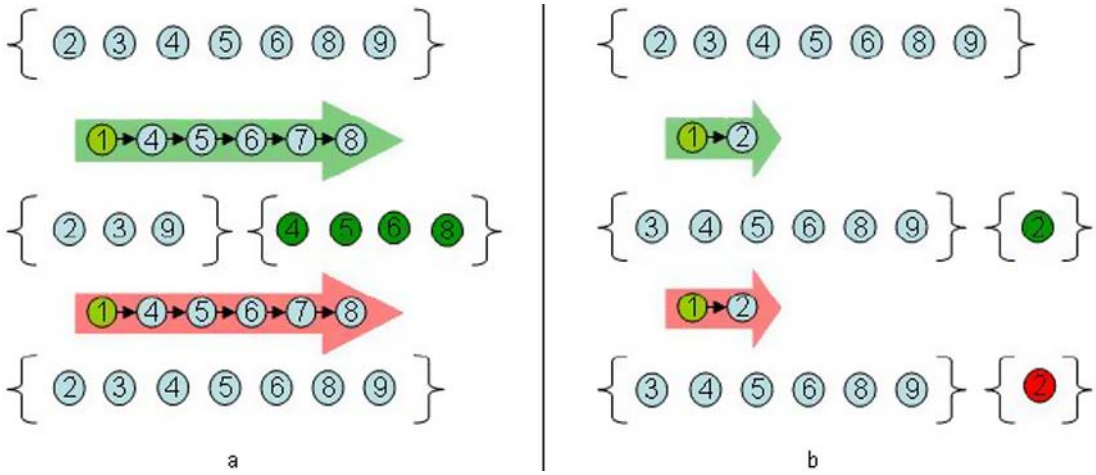

Fig. 7 (**1**) Successful probes give more information in Max search; (**2**) Failed probes give more information in Min search

the suspected node set only by a small amount. However, the failure of such a probe narrows down the search space significantly. For instance, in Fig. 7b success of a smaller probe $1 \rightarrow 2$ gives little information indicating good health of the single node 2, but failure of this probe narrows the fault localization to a single node, node 2. This set requires no additional probes for fault localization.

Based on these concepts we first present two algorithms to select probes for fault localization. The basic algorithm presented in Algorithm GFL stays the same for the two approaches. The two approaches differ in the probe selection procedure. The procedure SelectFLProbes() in Algorithm GFL selects probes from the set of available probes to diagnose the suspected nodes. We present two approaches to select this probe set, namely Max Search and Min Search. Thus we present two different implementations of the SelectFLProbes procedure based on the Max search and Min search to present a Greedy Fault Localization Algorithm with Max search and Min Search.

- *Max search*: As explained above, the Max search approach selects a probe that covers a maximum number of suspected nodes. In this implementation, the procedure SelectFLProbes() returns a set of probes from the available probes by iteratively selecting probes that cover a maximum number of uncovered nodes till all the suspected nodes are covered.
- *Min search*: The Min search approach works on the concept of selecting a probe for each suspected node set such that the selected probe goes through a minimum number of other nodes in the suspected node set. For implementing the Min search approach, the procedure SelectFLProbes() returns a set of probes for fault localization using this approach.

Considering the total number of nodes equals N and the total number of probes equals P, the for loop in Algorithm GFL performs $O(N)$ operations in each iteration in building the *PathSuspectedNodes*. The for loop runs for each failed

Algorithm BSFL: Binary Search Fault Localization Algorithm (Probe selection for fault localization in the network)

```
   input : Nodes, ProbeSet
   output: FailedNodes
1  FDProbes = FDetection (Nodes, ProbeSet) (from Algorithm 1);
2  Send FDProbes;
3  if no probe failed then
4      Declare FailedNodes and return;
5  end
6  SuspectedNodes ← NULL;
7  foreach failed probe p = (n1 → n2 → ··· → nm) do
8      n ← BinarySearch (n1, n1, nm);
9      Add n to FailedNodes;
10     Add all nodes following n on the path
       (n1 → n2 → ··· → nm) to SuspectedNodes;
11     Remove probes passing through node n from the ProbeSet;
12 end
13 BSFL (SuspectedNodes, ProbeSet);
```

Procedure `BinarySearch` (StationNode, StartNodePosition, EndNodePosition)

```
if StartNodePosition = EndNodePosition then
    return StartNodePosition;
end
TargetPosition ← ⌈(StartNodePosition + EndNodePosition)/2⌉;
Send probe from StationNode to node at position TargetPosition;
if probe fails then
    BinarySearch (StationNode, StartNodePosition,
    TargetPosition);
end
else
    BinarySearch (StationNode, TargetPosition,
    EndNodePosition);
end
```

probe. The total number of probes chosen by Algorithm GFL to be sent is at most N as at most N nodes can fail and the algorithm can select at most N probes one for each suspected node. Thus the complexity of the for loop in Algorithm GFL is $O(N^2)$. The operation of building the probe *AvailableProbes* set performs $O(P)$ operations.

In the SelectFLProbes function using Max search approach, the while loop runs at most N times. The probe selection operation in the loop is an $O(P)$ operation and updating the *TargetNodes* data structure is an $O(N)$ operation, making the total complexity of the function to be $O(N*(N+P))$. Thus the computational complexity of Algorithm GFL using Max search is $O(N*(N+P))$. Assuming an upper limit s on the maximum number of probes stations that can be deployed, the probe set size will be $O(s*N)$, assuming one probe from each of the s probe stations to $O(N)$ non-probe station nodes. Thus the probe set size of $O(P)$ can be considered to be $O(s*N)$. This makes the $O(N*(N+P))$ complexity of Algorithm GFL using Max search become $O(N^2)$.

The for loop in the SelectFLProbes function using the Min search approach, performs $O(P)$ operations to choose a probe and $O(N)$ operations to update the *TargetNodes* data structure. Thus one iteration of this for loop performs $O(N+P)$ operations. As explained above P equals $s*N$ making the complexity of one iteration to be $O(N)$. The for loop runs an iteration for each suspected node thus running at most $O(N)$ times. This makes the complexity of the SelectFLProbes function implementing Min search to be $O(N^2)$.

5.2.3 Binary search

In this section, we present another approach for selecting probes for fault localization. The Greedy approach does not diagnose each probe path independently. Instead, it builds a single set of suspected nodes consisting of nodes on all failed probe paths. In the Binary search approach, we propose to diagnose each failed probe path independently. On each failed probe path, additional probes are sent till one failure on that path is diagnosed. We propose to send these probes in a binary search fashion. On a failed probe path, a probe is first sent from the probe station to the node that is half way on the probe path. If this probe fails, further diagnosis is done on the first half of the probe path. On the other hand, if this probe succeeds, then the later half of the probe path is diagnosed in similar fashion. Figure 8 shows an example of how probes are sent in a binary search fashion to identify a failed node on the probe path. In this figure, probes are sent from probe station node 7. Consider that node 5 has failed and that the failure was detected on observing a failure of probe $7 \rightarrow 3$. Binary search probe selection then analyzes first half of this path by sending probe $7 \rightarrow 10$. On observing success on this path, nodes 6 and 10 are inferred to be in good health, and the second half of the probe path is analyzed focusing on nodes 5 and 3. Continuing probe selection in the binary search fashion, probe $7 \rightarrow 5$ is sent. Failure of this probe together with information about good health of nodes 6 and 10 indicates a failure of node 5.

This process identifies one failed node on each probe path. On each of these probe paths the health of the nodes behind the identified failed node might still be unknown. Hence a suspected node set is again created that consists of the unidentified nodes that lie behind the failed nodes on the probe paths. The nodes that are already known to be in good health or failed, are removed from this set. The similar fault localization process is repeated for this newly formed suspected node set. The process is repeated till the health of all the nodes is determined. Algorithm BSFL presents the Binary Search Fault Localization Algorithm.

Algorithm BSFL executes the Greedy algorithm to select probes for failure detection which performs $O(N^2)$ operations as shown in Sect. 4. An iteration of the for loop in Algorithm BSFL executes the BinarySearch procedure

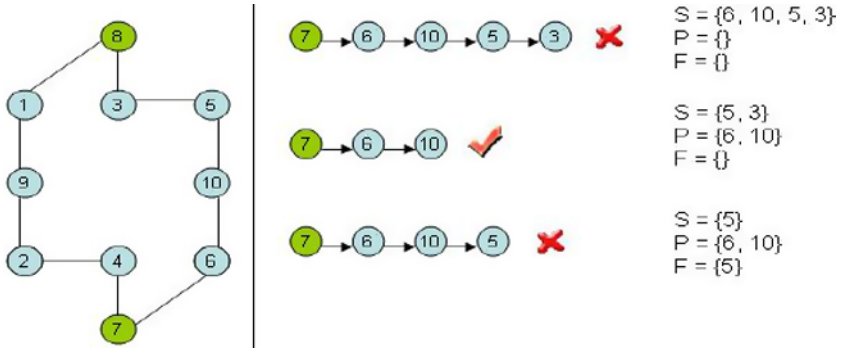

Fig. 8 Probes sent in a binary search fashion on a failed probe path 7 → 3 to identify failed node on the path

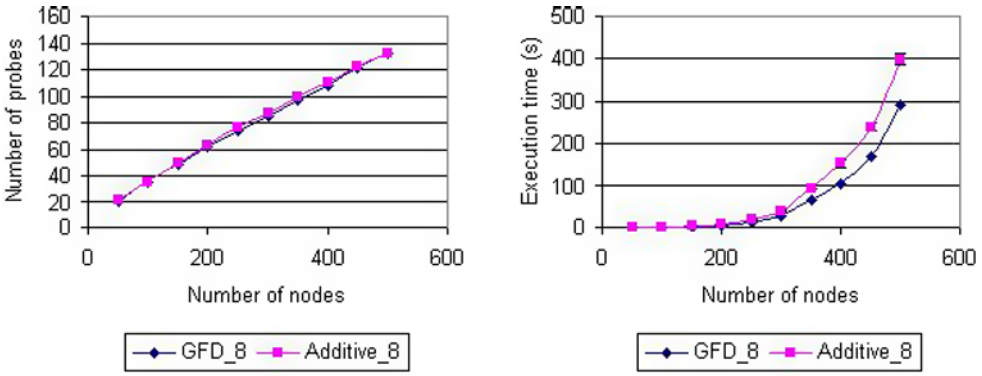

Fig. 9 Comparison of the probe set size and execution time of GFD and Additive algorithms for probe selection for failure detection in networks with different sizes and an average node degree of 8

which performs $O(\log(n))$ operations as discussed later in this section. The operation of updating the probe set is of $O(P)$ complexity. Thus one iteration of the for loop performs $O(\log(N) + P)$ operations. The for loop runs one iteration for each failed probe. Algorithm BSFL sends the probe set that covers all suspected nodes. This probe set can at most be of size N assuming sending one probe for each suspected node. In practice, the number of suspected nodes is much less than N and the number of probes to cover suspected nodes is much less than the total number of suspected nodes. Thus we can consider the number of failed probes and hence the for-loop iterations to be of $O(N)$. Hence, the computational complexity of Algorithm BSFL is $O(N * (P + \log(N)))$. As discussed earlier, assuming at most s probe stations, the probe set size P is $O(k * N)$, making the complexity of Algorithm BSFL to be $O(N^2)$.

6 Experimental evaluation

In this section, we present experimental evaluation via simulation of the algorithms proposed in this paper.

6.1 Simulation model

In this section we present the simulation model used for the experiment results presented in this paper. We simulated various network topologies with different network sizes and average node degrees. 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 degree greater than MD, and also ensuring that the network is connected. Each point plotted on the graphs is an average of 20 experiment runs where each experiment was run on a different network topology. We also plot the 95% confidence intervals. We present results to localize node failures in the network. We build a deterministic dependency model to represent the dependencies between the nodes and probes. For each probe selection experiment, we first ran a probe station selection algorithm (Algorithm SNR) [23]. This identified a set of nodes as probe stations from where probes can be sent to monitor the network. We used the probes that can be sent from these probe stations to build the set of available probes.

6.2 Simulation results

6.2.1 Failure detection

In this section we present results to show the effectiveness of Algorithm GFD in computing the probe set for failure detection. We compared Algorithm GFD with the Additive algorithm presented in [5] for network sizes varying from 50 to 500 nodes, with an average node degree of 8. The comparison of the two algorithms is shown in Fig. 9. Figure 9

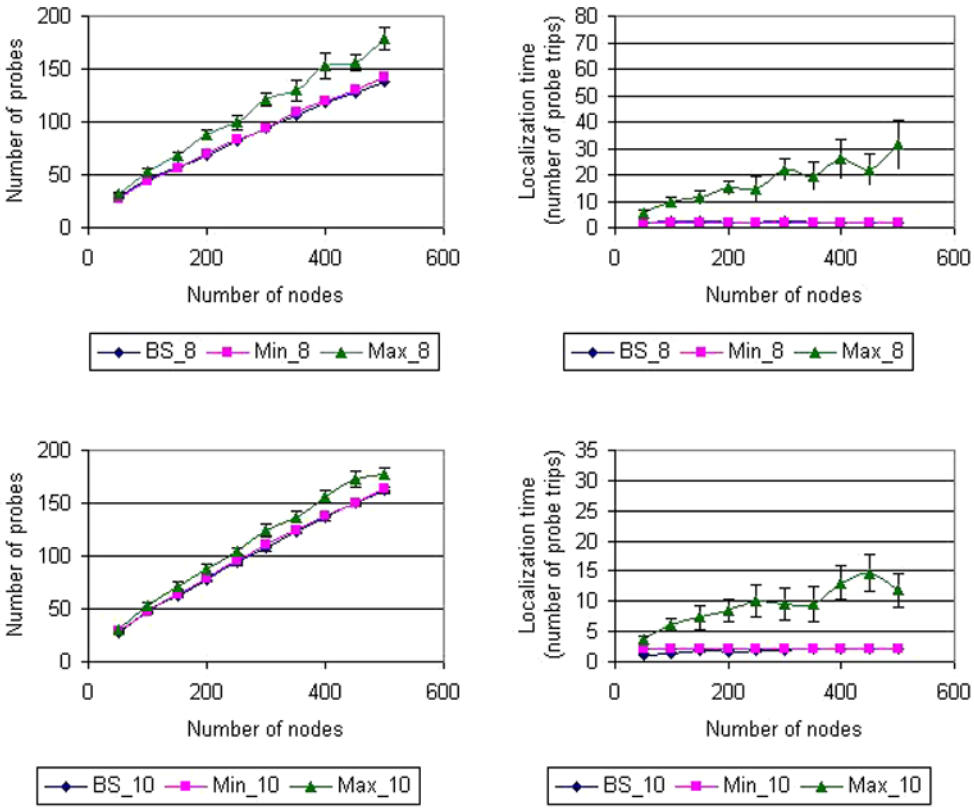

Fig. 10 Comparison of the probe set size computed and the localization time taken by Min, Max and Binary search for fault localization on networks with different sizes and with average node degrees 8 and 10. Time is measured in terms of probe trips where a probe trip time is the average time taken to send a set of probes and get back the probe results

shows that execution time of the Greedy algorithm is smaller than that of the Additive algorithm; while the probe set sizes computed by the Greedy algorithm are close to those computed by the Additive algorithm.

6.2.2 Fault localization

In this section, we present experimental evaluation of the proposed probe selection algorithms for fault localization. We randomly introduced three node failures and ran Algorithm BSFL and Algorithm NM using Min Search and Max Search approaches for probe selection. We calculated the number of probes used by the three algorithms and the time required by the three algorithms in terms of probe trip times. We consider the average time taken to send a set of probes and get back the probe results as one probe trip time. Using this metric, we compare the time taken by the three algorithms in localizing the introduced failures. We ran the three algorithms on network sizes ranging from 50 to 500 nodes with average node degrees of 8 and 10.

Figure 10 presents graphs for the probe set size computed and localization time taken by the Min search, Max search, and Binary search for fault localization. Figure 10 shows that Max search requires more probes than Min search and Binary search. Figure 10 also compares the time required by the three algorithms in localizing the failure where the time is measured in terms of probe trips. We also observe that Max search takes a longer time to localize the fault than Min search and Binary search.

Max search requires more probes for localization because of the use of long probes which are more likely to fail in the presence of failed nodes. A failed long probe gives less information to localize the fault. A failed long probe thus results in a need for more probes to perform diagnosis and increased number of iterations for fault diagnosis. Min search diagnoses all suspected nodes in parallel. Moreover, the small probe size leads to quicker diagnosis of node health. Binary search also operates in parallel on all probe paths leading to a faster diagnosis. Unlike Min search, Max search does not treat each suspected node in parallel but uses probes of larger lengths to cover all suspected nodes resulting in a larger number of iterations. However, the performance of Max search improves as the average node degree increases. With the increase in the average node degree, the average probe length decreases. A shorter probe is less likely to pass through the failed node, and hence more probes sent by Max search succeed, giving more information about the network health.

Next we compare adaptive probing with preplanned probing based probe selection algorithms proposed in the earlier work [5]. As explained in Sect. 1, preplanned probing computes a set of probes that is capable of diagnosing all possible failure scenarios of interest and sends this probe set periodically over the network. We compare the number of

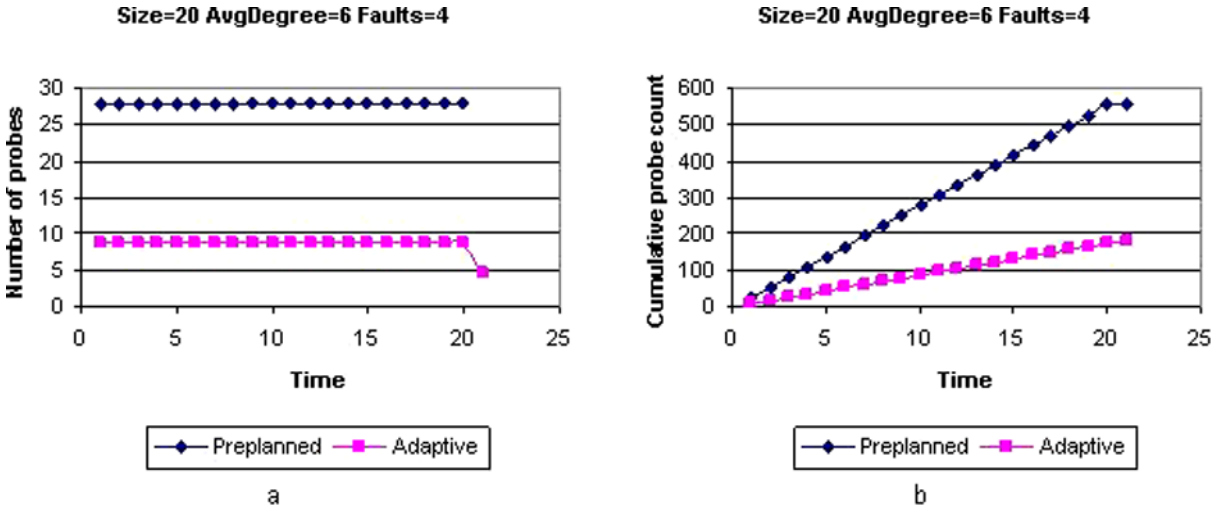

Fig. 11 (**a**) Number of probes sent per interval; (**b**) Total number of probes sent in 20 intervals by Adaptive and Preplanned approaches to localize 4 faults in a network of 20 nodes

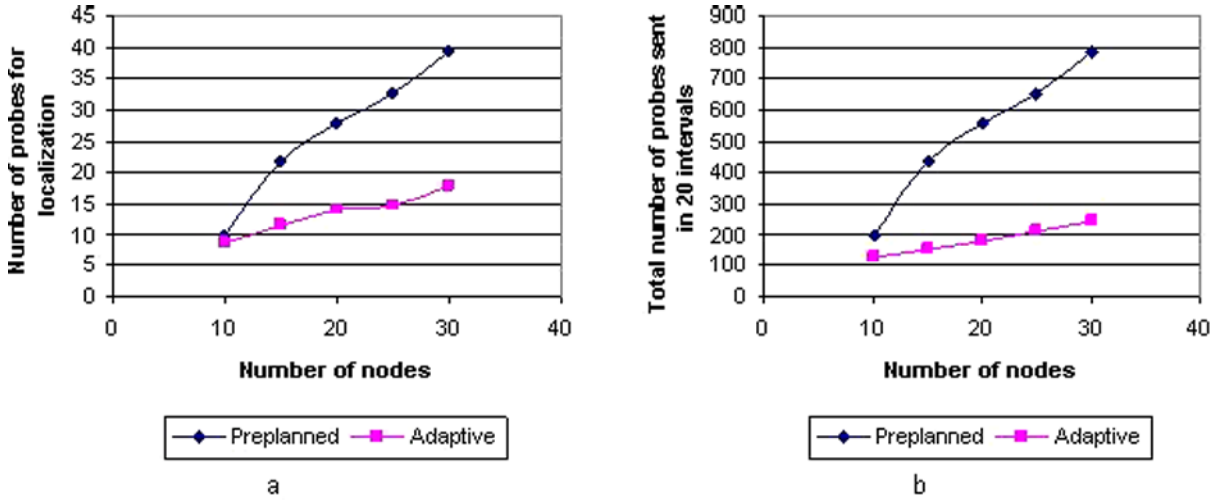

Fig. 12 (**a**) Number of probes sent per interval; (**b**) Total number of probes sent in 20 intervals by Adaptive and Preplanned approaches to localize 4 faults in networks with different network sizes

probes sent by adaptive and preplanned algorithms to perform fault localization in various scenarios. We first compare the number of probes sent by adaptive and preplanned probing on a network of 20 nodes to localize 4 faults in the network. We consider a scenario where both adaptive and preplanned approaches probe the network periodically and a failure is introduced at the 20th interval. We compare the number of probes sent by both approaches to localize the introduced failure. Figure 11a shows the number of probes sent per interval by adaptive and preplanned approaches. Preplanned probing sends out a fixed probe set for all 20 intervals. Adaptive probing on the other hand, first sends a probe set for failure detection for first 20 intervals and then sends the probes for fault localization on detecting failure at the 20th interval. Figure 11b compares the total number of probes sent by the two approaches in this scenario. From the figure we can see that adaptive probing sends a significantly smaller number of probes that preplanned probing.

Figure 12 shows the number of probes sent by the two approaches to localize 4 faults in networks with different sizes. It can be seen that the difference in the number of probes sent by adaptive and preplanned probing increases with the increase in network size. Figure 13 compares the number of probes sent by the two approaches in a network of 20 nodes to localize different number of faults in the network. We can see that as the number of faults to localize increases, the difference between the number of probes used

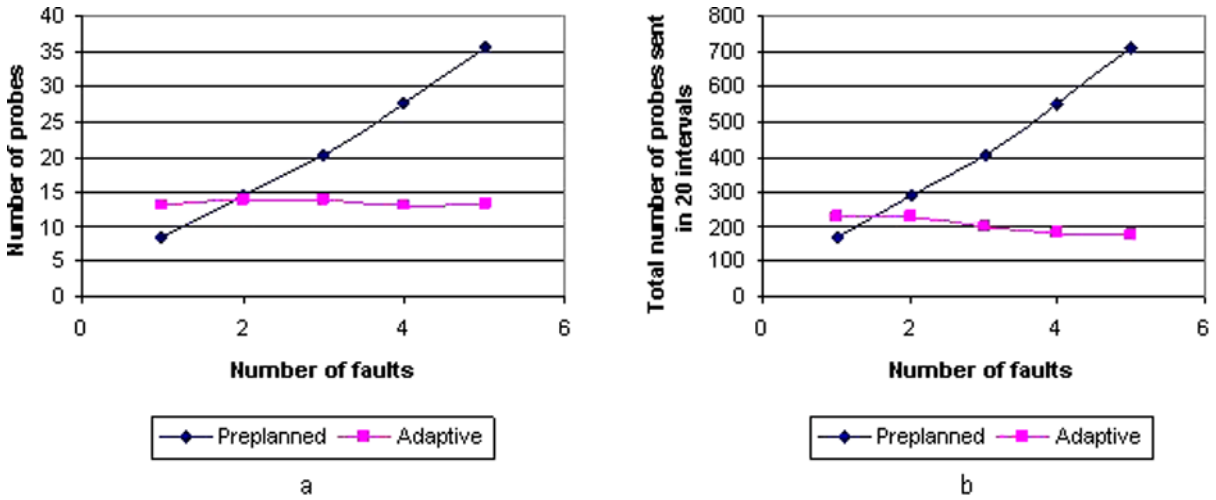

Fig. 13 (**a**) Number of probes sent per interval; (**b**) Total number of probes sent in 20 intervals by Adaptive and Preplanned approaches to localize different number of faults in a network of 20 nodes

by the two approaches increases, making adaptive probing significantly more efficient than preplanned probing.

7 Conclusion and future work

In this paper, we presented an adaptive probing based approach to perform fault localization in networks. We proposed fault localization algorithms to adapt the probe set to the observed network conditions and send probes in an interactive manner to perform the diagnosis. We presented three algorithms to select probes for fault localization in an adaptive manner and presented the performance evaluation of these algorithms. We also compared adaptive probing with preplanned probing and showed that adaptive probing performs localization using significantly smaller number of probes than preplanned probing. The end-to-end nature of probes and optimized traffic overhead makes adaptive probing a promising tool for various monitoring applications.

As part of future research, we aim to consider node mobility and scenarios with incomplete or inaccurate dependency information while selecting probes for fault localization. We also aim to develop a probe station communication protocol and build a decentralized fault localization system.

Acknowledgements 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 US Government.

References

1. Al-Shaer, E., & Tang, Y. (2002). QoS path monitoring for multicast networks. *Journal of Network and Systems Management*, *10*(3), 357–381.
2. Andersen, D. G., Balakrishnan, H., Kaashoek, M. F., & Morris, R. (2001). Resilient overlay networks. In *Symposium on operating systems principles* (pp. 131–145). Chateau Lake Louise, Banff, Canada.
3. Banerjee, S., Bhattacharjee, B., & Kommareddy, C. (2002). Scalable application layer multicast. In *ACM SIGCOMM 2002* (pp. 205–217). Pittsburgh, PA.
4. Bejerano, Y., & Rastogi, R. (2003). Robust monitoring of link delays and faults in IP networks. In *IEEE INFOCOM 2003* (pp. 1092–1103). San Francisco, CA.
5. Brodie, M., Rish, I., & Ma, S. (2001). Optimizing probe selection for fault localization. In *IFIP/IEEE international workshop on distributed systems: operations and management* (pp. 1147–1157). Nancy, France.
6. Brodie, M., Rish, I., Ma, S., Grabarnik, G., & Odintsova, N. (2002). *Active probing* (Technical report). IBM.
7. Brown, A., & Patterson, D. (2001). An active approach to characterizing dynamic dependencies for problem determination in a distributed environment. In *IFIP/IEEE international symposium on integrated network management* (pp. 377–390). Seattle, WA.
8. Carter, R. L., & Crovella, M. E. (1997). Server selection using dynamic path characterization in wide-area networks. In *IEEE INFOCOM, 1999* (pp. 1014–1021). Kobe, Japan.
9. Chen, Y., Bindel, D., & Katz, R. H. (2003). Tomography-based overlay network monitoring. In *ACM SIGCOMM conference on Internet measurement, 2003* (pp. 216–231). Miami, FL.
10. Chen, Y., Bindel, D., Song, H., & Katz, R. H. (2004). An algebraic approach to practical and scalable overlay network monitoring. In *ACM SIGCOMM 2004* (pp. 55–66).
11. Cormen, T. H., Stein, C., Rivest, R. L., & Leiserson, C. E. (2001). *Introduction to algorithms*. New York: McGraw-Hill.
12. Dovrolis, C., Ramanathan, P., & Moore, D. (2001). What do packet dispersion techniques measure? In *IEEE INFOCOM 2001* (pp. 905–914). Anchorage, Alaska.
13. Downey, A. B. (1999). Using Pathchar to estimate Internet link characteristics. In *ACM SIGCOMM 1999* (pp. 241–250). Cambridge, MA.
14. Frenkiel, A., & Lee, H. (1999). EPP: A framework for measuring the end-to-end performance of distributed applications. In *Performance engineering 'best practices' conference*. IBM Academy of Technology.
15. Gao, J., Kar, G., & Kermani, P. (2004). Approaches to building self-healing systems using dependency analysis. In *IEEE/IFIP*

network operations and management symposium (pp. 119–132). Seoul, Korea.
16. Chu, Y. H., Rao, S. G., Seshan, S., & Zhang, H. (2002). A case of end system multicast. *IEEE Journal on Selected Areas in Communications*, 1456–1471.
17. Hu, N., & Steenkiste, P. (2003). Evaluation and characterization of available bandwidth probing techniques. *IEEE Journal on Selected Areas in Communications*, *21*(6), 879–894. Special issue in Internet and WWW measurement, mapping and modeling.
18. Hu, N., & Steenkiste, P. (2003). Towards tunable measurement techniques for available bandwidth. In *Bandwidth estimation workshop (BEst03)*. San Diego, CA.
19. Huffaker, B., Plummer, D., Moore, D., & Claffy, K. (2002). Topology discovery by active probing. In *Symposium on applications and the Internet* (pp. 90–96). Nara, Japan.
20. Jain, M., & Dovrolis, C. (2002). End-to-end available bandwidth: Measurement methodology, dynamics, and relation with TCP throughput. In *ACM SIGCOMM 2002* (pp. 537–549). Pittsburgh, PA.
21. Lai, K., & Baker, M. (1999). Measuring bandwidth. In *IEEE INFOCOM 1999* (pp. 235–245). New York, NY.
22. Li, F., & Thottan, M. (2006). End-to-end service quality measurement using source-routed probes. In *IEEE INFOCOM 2006* (pp. 1–12). Barcelona, Spain.
23. Natu, M., & Sethi, A. S. (2008, to appear). Probe station placement for robust monitoring of networks. *Journal of Network and Systems Management*.
24. Ozmutlu, H. C., Gautam, N., & Barton, R. (2002). Managing end-to-end network performance via optimized monitoring strategies. *Journal of Network and Systems Management*, *10*(1), 107–126.
25. Ozmutlu, H. C., Gautam, N., & Barton, R. R. (2002). Zone recovery methodology for probe-subset selection in end-to-end network monitoring. In *IEEE/IFIP network operations and management symposium* (pp. 451–464). Florence, Italy.
26. Ratnasamy, S., Francis, P., Handley, M., Karp, R., & Schenker, S. (2001). A scalable content-addressable network. In *ACM SIGCOMM 2001* (pp. 161–172). San Diego, CA.
27. Ribeiro, V., Coates, M., Riedi, R., & Sarvotham, S. (2000). Multifractional cross-traffic estimation. In *ITC specialist seminar on IP traffic measurement, modeling and management 2000*. Monterey, CA.
28. Ribeiro, V. J., Riedi, R. H., & Baraniuk, R. G. (2004). Spatio-temporal available bandwidth estimation with STAB. In *ACM SIGMETRICS 2004* (pp. 394–395). New York, NY.
29. Ribeiro, V. J., Riedi, R. H., Baraniuk, R. G., Navratil, J., & Cottrell, L. (2003). pathChirp: Efficient available bandwidth estimation for network paths. In *Passive and active measurement workshop*. La Jolla, CA.
30. Rish, I., Brodie, M., Ma, S., Odintsova, N., Beygelzimer, A., Grabarnik, G., & Hernandez, K. (2005). Adaptive diagnosis in distributed systems. *IEEE Transactions on Neural Networks*, *6*(5), 1088–1109.
31. Rowstron, A., & Druschel, P. (2001). Pastry: Scalable, decentralized object location, and routing for large-scale peer-to-peer systems. In *Lecture notes in computer science* (Vol. 2218, pp. 329–350). Berlin: Springer.
32. Tang, C., & McKinley, P. (2003). On the cost-quality tradeoff in topology-aware overlay path probing. In *IEEE international conference on network protocols, 2003* (pp. 268–179). Atlanta, GA.
33. Tang, Y., Al-Shaer, E. S., & Boutaba, R. (2005). Active integrated fault localization in communication networks. In *IFIP/IEEE international symposium on integrated network management* (pp. 543–556). Nice, France.
34. Zhang, B., Jamin, S., & Zhang, L. (2002). Host multicast: A framework for delivering multicast to end users. In *IEEE INFOCOM 2002* (pp. 1366–1375). New York, NY.
35. Zhao, B. Y., Kubiatowicz, J. D., & Joseph, A. D. (2001). *Tapestry: An infrastructure for fault-tolerant wide-area location and routing* (Technical report). University of California at Berkeley, Berkeley, CA, USA.

Maitreya Natu received M.S. degree and Ph.D. degree in Computer and Information Sciences from the University of Delaware. His research interests include fault localization for wired and wireless networks, probing techniques for network monitoring and management, and network security problems like intrusions and denial of service attacks. He will be joining Tata Research Design and Development Center, Pune, India in February 2008.

Adarshpal S. Sethi is a Professor in the Department of Computer & Information Sciences at the University of Delaware, Newark, Delaware, USA. He has an M.Tech. degree in Electrical Engineering and a Ph.D. 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. Dr. Sethi 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. Dr. Sethi's research interests include architectures and protocols for network management, fault management, quality-of-service and resource management, and management of wireless networks.

Errol L. Lloyd is a Professor of Computer and Information Sciences at the University of Delaware. Previously he served as a faculty member at the University of Pittsburgh and as Program Director for Computer and Computation Theory at the National Science Foundation. From 1994 to 1999 he was Chair of the Department of Computer and Information Sciences at the University of Delaware. Concurrently, from 1997 to 1999 he was Interim Director of the University of Delaware Center for Applied Science and Engineering in Rehabilitation. Professor Lloyd received undergraduate degrees in both Computer Science and Mathematics from Penn State University, and a PhD in Computer Science from the Massachusetts Institute of Technology. His research expertise is in the design and analysis of algorithms, with a particular concentration on approximation algorithms. In 1989 Professor Lloyd received an NSF Outstanding Performance Award, and in 1994 he received the University of Delaware Faculty Excellence in Teaching Award.