Robust Monitoring of Link Delays and Faults in IP Networks

Yigal Bejerano and **Rajeev Rastogi**
Bell Laboratories, Lucent Technologies
600 Mountain Avenue, Murray Hill, NJ 07974
Email: {bej,rastogi}@research.bell-labs.com

Abstract – **In this paper, we develop failure-resilient techniques for monitoring link delays and faults in a Service Provider or Enterprise IP network. Our two-phased approach attempts to minimize both the monitoring infrastructure costs as well as the additional traffic due to probe messages. In the first phase of our approach, we compute the locations of a minimal set of monitoring stations such that all network links are covered, even in the presence of several link failures. Subsequently, in the second phase, we compute a minimal set of probe messages that are transmitted by the stations to measure link delays and isolate network faults. We show that both the station selection problem as well as the probe assignment problem are NP-hard. We then propose greedy approximation algorithms that achieve a logarithmic approximation factor for the station selection problem and a constant factor for the probe assignment problem. These approximation ratios are provably very close to the best possible bounds for any algorithm.**

Keywords – Latency and Fault monitoring, Network Failures, Set Cover Problem, Approximation algorithms.

I. Introduction

The demand for sophisticated tools for monitoring network utilization and performance has been growing rapidly as Internet Service Providers (ISPs) offer their customers more services that require quality of service (QoS) guarantees, and as ISP networks become increasingly complex. Tools for monitoring link delays and faults in an IP network are critical for numerous important network management tasks, including providing QoS guarantees to end applications (e.g., voice over IP), traffic engineering, ensuring service level agreement (SLA) compliance, fault and congestion detection, performance debugging, network operations, and dynamic replica selection on the Web. Consequently, there has been a recent flurry of both research and industrial activity in the area of developing novel tools and infrastructures for measuring network parameters.

Existing network monitoring tools can be divided into two categories. *Node-oriented* tools collect monitoring information from network devices (routers, switches and hosts) using SNMP/RMON probes [5] or the Cisco NetFlow tool [6]. These are useful for collecting statistical and billing information, and for measuring the performance of individual network devices (e.g., link bandwidth usage). However, in addition to requiring monitoring agents to be installed at every device, these tools cannot monitor network parameters that involve several components, like link or end-to-end path latency. The second category contains *path-oriented* tools for connectivity and latency measurement like `ping`, `traceroute` [11], `skitter` [12] and tools for bandwidth measurement such as `pathchar` [13], `Cprobe` [14], `Nettimer` [15] and `pathrate` [16]. As an example, `skitter` sends a sequence of probe messages to a set of destinations and measures the latency of a link as the difference in the round-trip times of the two probes to the endpoints of the link. A benefit of path-oriented tools is that they do not require special monitoring agents to be run at each node. However, a node with such a path-oriented monitoring tool, termed a *monitoring station*, is able to measure latencies and monitor faults for only a limited set of links in the node's routing tree (e.g., *shortest path tree*). Thus, monitoring stations need to be deployed at a few strategic points in the ISP or Enterprise IP network so as to maximize network coverage while minimizing hardware and software infrastructure costs, as well as maintenance costs for the stations.

A. Related Work

The need for low-overhead network monitoring has prompted the development of new monitoring platforms. The IDmaps [8] project produces "*latency maps*" of the internet using special measurement servers called *tracers* that continuously probe each other to determine their distance. These times are subsequently used to approximate the latency of arbitrary network paths. Different methods for distributing tracers in the internet are described in [9], one of which is to place them such that the distance of each network node to the closest tracer is minimized. A drawback of the IDMaps approach is that latency measurements may not be too accurate. Essentially, due to the small number of paths actually monitored, it is possible for errors to be introduced when round-trip times between tracers are used to approximate arbitrary path latencies. Recently, in [7], the authors propose a monitoring scheme where a single *network operations center* (NOC) performs all the required measurements. In order to monitor links not in its routing tree, the NOC uses the IP source routing option to explicitly route probe packets along the link. Unfortunately, due to security problems, many routers frequently disable the IP source routing option. Consequently, approaches that rely on explicitly routed probe packets for

delay and fault monitoring may not be feasible in today's ISP and Enterprise environments.

In other recent work on monitoring, [10] proposes to solve a linear system of equations to compute delays for smaller path segments from a given a set of end-to-end delay measurements for paths in the network. Similarly, [2] considers the problem of inferring link-level loss rates and delays from end-to-end multicast measurements for a given collection of trees. Finally, [4] studies ways to minimize the monitoring communication overhead for detecting alarm conditions due to threshold violations.

The problems studied in [18], [1] are most closely related to our work. [18] considers the problem of fault isolation in the context of large multicast distribution trees. The schemes in [18] achieve efficiency by having each receiver monitor only a portion of the path (in the tree) between it and the source, but require receivers to have some monitoring capability (e.g., the ability to do multicast traceroute). In contrast, in our approach, only the monitoring stations (or sources) transmit probe messages, and network nodes are assumed to have very limited support for monitoring. [1] focuses on the problem of determining the minimum cost set of multicast trees that cover links of interest in a network, which is similar to the station selection problem tackled in this paper. However, there are some significant differences. For instance, [1] does not consider network failures, or issues like minimizing the monitoring overhead due to probe messages. Also, our problem of selecting the minimum number of monitoring stations whose routing trees cover links of interest is more restrictive than the link covering problem tackled in [1] since routing trees usually are more constrained (e.g., shortest path trees) than multicast trees.

B. Our Contributions

Most of the infrastructures for monitoring described above suffer from three major drawbacks: (1) The systems do not guarantee that all links of interest in the network are monitored, especially in the presence of network failures, (2) The systems have limited support for accurately pinpointing the location of a fault when a network link fails, and (3) The systems pay little attention to minimizing the overhead (due to additional probe messages) imposed by monitoring on the underlying production network. In this paper, we propose a novel *two-phased* approach for fully and efficiently monitoring link latencies and faults in an ISP or Enterprise IP network, using path-oriented tools. Our schemes are failure-resilient, and ensure complete coverage of measurements by selecting monitoring stations such that each network link is always in the routing tree of some station. Our methods also reduce the monitoring overhead which consists of two costs: the infrastructure and maintenance costs associated with monitoring stations, and the additional network traffic due to probe packets. Minimizing the latter is especially important when information is collected frequently (e.g., every 15 minutes) in order to continuously monitor the state and evolution of the network.

In the first phase of our approach, we compute a minimal set of monitoring stations (and their locations) that always cover all links in the network, even if some links were to fail. Subsequently, in the second phase, we compute the minimal set of probe messages transmitted by each station such that the latency of every network link can be measured, and every link failure can be detected. The main contributions of our work can be summarized as follows.

- **Novel algorithms for Station Selection.** We show that the problem of computing the minimum set of stations whose routing trees cover all network links is NP-hard. The station selection problem maps naturally to the set cover problem [3], and thus a polynomial-time greedy algorithm yields a solution that is within a logarithmic factor of the optimal. Further, using sophisticated reductions from set cover, we are able to prove that the logarithmic factor is indeed a lower bound on the degree of approximation achievable by any algorithm. In the presence of network failures, we show that the station selection problem maps to a variant of the set cover problem, which we again solve using a greedy algorithm, while guaranteeing a logarithmic approximation ratio.
- **Novel algorithms for Probe Assignment.** We show that the problem of computing the optimal set of probe messages for measuring the latency of network links is NP-hard. We devise a polynomial-time greedy algorithm that computes a set of probes whose cost is within a factor of 2 of the optimal solution. Again, this approximation factor is very close to the best possible approximation result. Finally, we show how our framework for monitoring link latencies can also be extended to monitor link failures with a near-optimal number of probe messages.

The remainder of this paper is organized as follows. Section II presents the network model and a description of the network monitoring framework is given in Section III. Section IV describe our design for link monitoring systems and we extend this design with fault detection mechanism in Section V. Then, Section VI presents a robust link monitoring system that is resilience to several failures and we conclude this work in Section VII.

II. Network model

We model the Service Provider or Enterprise IP network by an undirected graph $G(V, E)$, where the graph nodes, V, denote the network routers and the edges, E, represent the communication links connecting them. The number of nodes and edges is denoted by $|V|$ and $|E|$, respectively. Further, we use $P_{s,t}$ to denote the path traversed by an IP packet from a source node s to a destination node t. In our model, we assume that packets are forwarded using standard IP forwarding, that is, each node relies exclusively on the destination address in the packet to determine the next hop. Thus, for every node $x \in P_{s,t}$, $P_{x,t}$ is included in $P_{s,t}$. In addition, we also assume that $P_{s,t}$ is the routing path in the opposite direction from node t to node s. This, in turn, implies that, for every node $x \in P_{s,t}$, $P_{s,x}$ is a prefix of $P_{s,t}$. As a consequence, it follows

that for every node $s \in V$, the subgraph obtained by merging all the paths $P_{s,t}$, for every $t \in V$, must have a tree topology. We refer to this tree for node s as the *routing tree* (RT) for s, and denote it by T_s. Note that tree T_s defines the routing paths from node s to all the other nodes in V and vice versa.

Observe that for a Service Provider network consisting of a single OSPF area, the RT T_s of node s is its shortest path tree. However, for networks consisting of multiple OSPF areas or autonomous systems (that exchange routing information using BGP), packets between nodes may not necessarily follow shortest paths. In practice, the topology of RTs can be calculated by querying the routing tables of nodes.

In case a link f in the network fails, the IP routing protocols define a new delivery tree, $T_{s,f}$, for every node $s \in V$[1]. The new tree $T_{s,f}$ has the property that every path $P_{s,t}$ in T_s, $t \in V$, that does not contain link f is also included in the tree $T_{s,f}$. The reason for this is that the failure of link f only affects those routing paths in T_s that contain f. Thus, it may be possible to infer the topology of a significant portion of $T_{s,f}$ directly from T_s without any knowledge of the route computation algorithms followed by the routers. Note that if $f \notin T_s$, then $T_{s,f} = T_s$.

We associate a positive cost $c_{s,t}$ with sending a message along the path $P_{s,t}$ between any pair of nodes $s, t \in V$. For every intermediate node $x \in P_{s,t}$ both $c_{s,x}$ and $c_{x,t}$ are at most $c_{s,t}$ and $c_{s,x} + c_{x,t} \geq c_{s,t}$. Typical examples of this cost model are the fixed cost model, where all messages have the same cost, and the hop count model, where the message cost is the number of hops in its route. Moreover, we denote by $h_{s,t}$ the number of hops in path $P_{s,t}$.

III. Delay Monitoring Framework

In this section, we describe our methodology for complete measurement of round-trip latency of network links in an IP network. Our proposed framework can also be extended to detect link failures as well as be resilient to them; we discuss these extensions in more detail in Sections V and VI, respectively. For monitoring the round-trip delay of a link $e \in E$, a node $s \in V$ such that e belongs to s's RT (*i.e.,* $e \in T_s$), must be selected as a monitoring station. Node s sends two probe messages[2] to the end-points of e, which travel almost identical routes except for the link e. Upon receiving a probe message, the receiver replies immediately by sending a probe reply message to the monitoring station. Thus, the monitoring station s can calculate the round-trip delay of the link by measuring the difference in the round-trip times of the two probe messages (see also the `skitter` tool [12]).

From the above description, it follows that a monitoring station can only measure the delays of links in its RT. Consequently, a monitoring system designated for measuring the delays of all network links has to find a set of *monitoring stations* $S \subseteq V$ and a *probe assignment* $\mathcal{A} \subset \{m(s,u) | s \in S, u \in V\}$, where each message $m(s,u)$ represents a probe message that is sent from the monitoring station s to node u. The set S and the probe assignment $\mathcal{A}$ are required to satisfy two constraints: (1) A *covering set* constraint that guarantees that all links are covered by the RTs of the nodes in S, *i.e.,* $\bigcup_{s \in S} T_s = E$, and (2) A *covering assignment* constraint which ensures that for every edge $e = (u,v) \in E$, there is a node $s \in S$ such that $e \in T_s$ and $\mathcal{A}$ contains the messages $m(s,u)$ and $m(s,v)$[3]. The covering assignment constraint essentially ensures that every link is monitored by some station. A pair $(S, \mathcal{A})$ that satisfies the above constraints is referred to as a *feasible solution*. Note that although we only consider the problem of monitoring all network links in this paper, our results also apply to the problem of monitoring only a subset of links of interest.

The overhead of a monitoring system consists of two components, the overhead of installing and maintaining the monitoring stations, and the communication cost of sending probe messages. In practice, it is preferable to have as few stations as possible since this reduces operational costs, and so we adopt a two-phased approach to optimizing monitoring overheads. In the first phase, we select an optimal set of monitoring stations, while in the second, we compute the optimal probes for the selected stations. An *optimal station selection* S is one that satisfies the covering set requirement while simultaneously minimizing the number of stations. After selecting the monitoring stations S, an *optimal probe assignment* $\mathcal{A}$ is one that satisfies the covering assignment constraint and minimizes the sum $\sum_{m(s,v) \in \mathcal{A}} c_{s,v}$. Note that choosing $c_{s,v} = 1$ essentially results in an assignment $\mathcal{A}$ with the minimum number of probes while choosing $c_{s,v}$ to be the hop distance, $h_{s,v}$, yields a set of probes that traverse the fewest possible network links.

A final component of our monitoring infrastructure is the *network operations center* (NOC) which is responsible for coordinating the actions of the set of monitoring stations S. The NOC queries network nodes to determine their RTs, and subsequently uses these to compute a near-optimal set of stations and a probe assignment for them, as described in the following section. In Section V, we extend our proposed mechanisms to also detect link failures, and finally in Section VI, we present a link monitoring system that is resilient to multiple link failures.

IV. Delay Monitoring Algorithms

In this section, we present polynomial-time approximation algorithms for solving the station selection and probe assignment problems, for a scenario that does not consider network link failures. After showing both problems to be NP-hard, we develop a $\ln(|V|)$-approximation algorithm for station selection, and a 2-approximation algorithm for probe assignment.

[1] This typically takes a few seconds to a few tens of seconds depending on the IP routing protocol parameter settings.

[2] The probe messages are implemented by using "ICMP ECHO REQUEST/REPLY" messages similar to `ping`.

[3] If one of the endpoints of $e = (u,v)$ is in S, say $u \in S$, then $\mathcal{A}$ is only required to contain the probe $m(u,v)$.

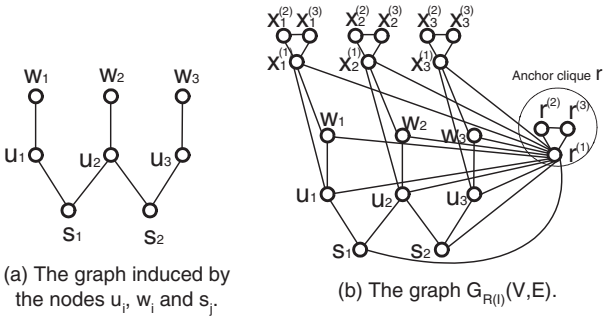

Fig. 1. The graph $G_{\mathcal{R}(I)}(V, E)$ for the given instance of the SC problem.

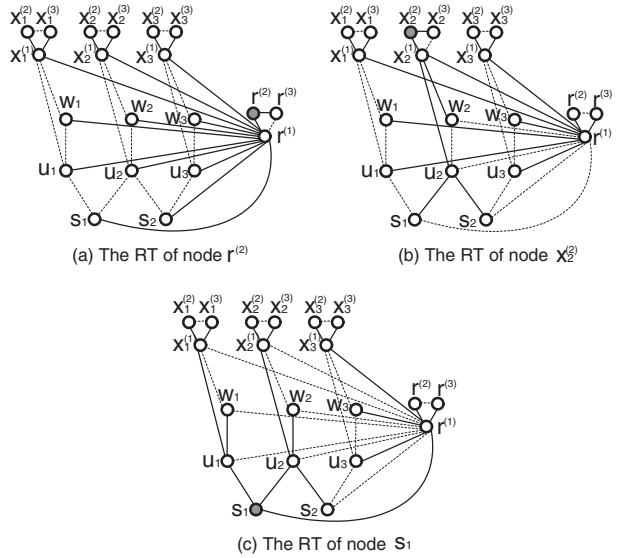

Fig. 2. The RTs of nodes $r^{(2)}$, $x_2^{(2)}$ and s_1.

A. An Efficient Station Selection Algorithm

We are interested in solving the problem of covering all graph edges with a small number of RTs.

Definition 1 (The Link Monitoring Problem - LM): Given a graph $G(V, E)$ and a RT, T_v, for every node $v \in V$, find the smallest subset $S \subseteq V$ such that $\bigcup_{v \in S} T_v = E$. □

Due to space constraints, we only consider the unweighted version of the LM problem. However, our results can be easily extended to the weighted version of the problem, where each node has an associated cost, and we seek a set $S \subseteq V$ that minimizes the total cost of the monitoring stations in S. The latter can be used, for instance, to find a station selection when monitoring stations can be installed only at a restricted set of nodes. For restricting the station selection, nodes that cannot support monitoring stations are assigned infinite cost. The LM problem is similar to the *set cover* (SC) problem, which is a well-known NP-hard problem. In an instance $I(Z, \mathcal{Q})$ of the SC problem, $Z = \{z_1, z_2, \cdots, z_m\}$ is a universe of m elements and $\mathcal{Q} = \{Q_1, Q_2, \cdots, Q_n\}$ is a collection of n subsets of Z (assume that $\bigcup_{Q \in \mathcal{Q}} Q = Z$). The SC problem seeks to find the smallest collection of subsets $\mathcal{S} \subseteq \mathcal{Q}$ such that their union contains all the elements in Z, *i.e.,* $\bigcup_{Q \in \mathcal{S}} Q = Z$.

A.1. The Hardness of the LM Problem

Theorem 1: The LM problem is NP-hard, even when the RT of each node is restricted to be its shortest path tree.

Proof: We show that the LM problem is NP-hard by presenting a polynomial reduction from the *set cover* problem to the LM problem. Consider an instance $I(Z, \mathcal{Q})$ of the SC problem. Our reduction $\mathcal{R}(I)$ constructs the graph $G_{\mathcal{R}(I)}(V, E)$ where the RT of each node $v \in V$ is also its shortest path tree. For determining these RTs, each edge is associated with a weight[4], and the graph contains the following nodes and edges. For each element $z_i \in Z$, it contains two connected nodes u_i and w_i. For each set $Q_j \in \mathcal{Q}$, we add a node, labeled by s_j, and the edges (s_j, u_i) for each element $z_i \in Q_j$. In addition, we use an auxiliary structure, termed an *anchor clique* x, which is a clique with three nodes, labeled by $x^{(1)}$, $x^{(2)}$ and $x^{(3)}$, and only node $x^{(1)}$ has additional incident edges. For each element $z_i \in Z$, the graph $G_{\mathcal{R}(I)}$ contains one anchor clique x_i whose attachment point, $x_i^{(1)}$, is connected to the nodes u_i and w_i. The weights of all the edges described above is 1. Finally, the graph $G_{\mathcal{R}(I)}$ contains an additional anchor clique r that is connected to the remaining nodes and anchor cliques of the graph, and the weights of these edges is $1 + \epsilon$. An example of such a graph is depicted in Figure 1 for an instance of the SC problem with 3 elements $\{z_1, z_2, z_3\}$ and two sets $Q_1 = \{z_1, z_2\}$ and $Q_2 = \{z_2, z_3\}$. Moreover, we assume that the RT, T_v, of every node $v \in V$ is its shortest path tree.

We claim that there is a solution of size k to the given SC problem if and only if there is a solution of size $k+m+1$ to the LM instance defined by the graph $G_{\mathcal{R}(I)}(V, E)$. We begin by showing that if there is a solution to the SC problem of size k then there exists a set S of at most $k+m+1$ stations that covers all the edges in $G_{\mathcal{R}(I)}$. Let the solution of the SC problem consist of the sets $Q_{i_1}, \ldots, Q_{i_k}$. The set S of monitoring stations contains the nodes $r^{(2)}$, $x_i^{(2)}$ (for each element $z_i \in Z$) and $s_{i_1}, \ldots, s_{i_k}$. We show that the set S contains $k + m + 1$ nodes that cover all the graph edges. The tree $T_{r^{(2)}}$ covers edges $(r^{(1)}, r^{(2)})$, $(r^{(2)}, r^{(3)})$, all edges $(u_i, r^{(1)})$, $(w_i, r^{(1)})$, $(x_i^{(1)}, r^{(1)})$, $(x_i^{(1)}, x_i^{(2)})$, $(x_i^{(1)}, x_i^{(3)})$, for each element z_i, and the edges $(s_j, r^{(1)})$ for every set $Q_j \in \mathcal{Q}$. An example of such a $T_{r^{(2)}}$ is depicted in Figure 2-(a). Similarly, for every $z_i \in Z$, the RT $T_{x_i^{(2)}}$ covers edges $(x_i^{(2)}, x_i^{(3)})$, $(x_i^{(2)}, x_i^{(1)})$, $(x_i^{(1)}, u_i)$ and $(x_i^{(1)}, w_i)$. $T_{x_i^{(2)}}$ also covers all edges (s_j, u_i) for every set Q_j that contains element z_i, and edges $(r^{(1)}, r^{(2)})$ and $(r^{(1)}, r^{(3)})$. An example of the RT $T_{x_2^{(2)}}$ is depicted in Figure 2-(b). Thus, the only remaining uncovered edges are (u_i, w_i), for each element z_i. Since Q_{i_j}, $j = 1, \ldots, k$, is a solution to the SC problem, these edges are covered by the RTs $T_{s_{i_j}}$, as depicted in Figure 2-(c). Thus, S is a set of at most $k + m + 1$ stations that covers all the edges in the graph $G_{\mathcal{R}(I)}$.

[4]These weights do not represent communication costs.

$\mathcal{S} \leftarrow \emptyset$, $C \leftarrow Z$
While $C \neq \emptyset$ do
    $Q \leftarrow arg\max_{Q_v \in \mathcal{Q}-\mathcal{S}} |Q_v \bigcap C|$
    $\mathcal{S} \leftarrow \mathcal{S} \bigcup \{Q\}$
    $C \leftarrow C - Q$
End While
Return $\mathcal{S}$

Fig. 3. A formal description of the Greedy Algorithm.

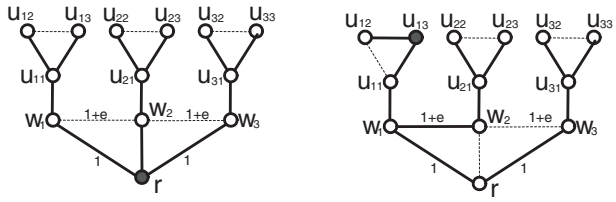

Fig. 4. The RTs of nodes r and u_{13}.

Next, we show that if there is a set of at most $k+m+1$ stations that covers all the graph edges then there is a solution for the SC problem of size at most k. Note that there needs to be a monitoring station in each anchor clique and suppose w.l.o.g that the selected stations are $r^{(2)}$ and $x_i^{(2)}$ for each element z_i. None of these $m+1$ stations covers edges (u_i, w_i) for elements $z_i \in Z$. The other k monitoring stations are placed in the nodes u_i, w_i and s_j. In order to cover edge (u_i, w_i), there needs to be a station at one of the nodes u_i, w_i or s_j for some set Q_j containing element z_i. Also, observe that the RTs of u_i and w_i cover only edge (u_i, w_i) for element z_i and no other element edges. Similarly, the RT of s_j covers only edges (u_i, w_i) for elements z_i contained in set Q_j. Let $\mathcal{S}$ be a collection of sets defined as follows. For every monitoring station at any node s_j add the set $Q_j \in \mathcal{Q}$ to $\mathcal{S}$, and for every monitoring station at any node u_i or w_i we add to $\mathcal{S}$ an arbitrary set $Q_j \in \mathcal{Q}$ such that $z_i \in Q_j$. Since the set of monitoring stations cover all the element edges, the collection $\mathcal{S}$ covers all the elements of Z, and is a solution to the SC problem of size at most k. □

The above reduction $\mathcal{R}(I)$ from the SC problem can be extended to derive a lower bound for the best approximation ratio achievable by any algorithm (see [17] for proof).

Theorem 2: The lower bound of any approximation algorithm for the LM problem is $\frac{1}{2} \cdot \ln(|V|)$.

A.2. A Greedy Algorithm for the LM Problem

We now present an efficient algorithm for solving the LM problem. The algorithm maps the given instance of the LM problem, involving graph $G(V, E)$, to an instance of the SC problem, and then uses a greedy heuristic for solving the SC instance. In the mapping, the set of edges E defines the universe of elements Z, and the collection of sets $\mathcal{Q}$ includes the subsets $Q_v = \{e | e \in T_v\}$ for every node $v \in V$. The greedy heuristic, depicted in Figure 3, is an iterative algorithm that selects, in each iteration, the set Q with the maximum number of uncovered elements. According to [3], the greedy algorithm is a $(\ln(\Delta)+1)$-approximation algorithm for the SC problem, where Δ is the size of the biggest subset. Thus, since for the LM problem, every subset includes all the edges of the corresponding RT and its size is exactly $|V|-1$, we have the following result.

Theorem 3: The greedy algorithm computes a $(\ln(|V|)+1)$-approximation for the LM problem.

Note that the worst-case time complexity of the greedy algorithm can be shown to be $O(|V|^3)$.

B. An Efficient Probe Assignment Algorithm

Once we have selected a set S of monitoring stations, we need to compute a probe assignment $\mathcal{A}$ for measuring the latency of network links. Recall from Section III that a *feasible probe assignment* is a set of probe messages $\{m(s,u) | s \in S, u \in V\}$, where each $m(s,u)$ represents a probe message that is sent from station s to node u. Further, for every edge $e = (u,v) \in E$, there is a station $s \in S$ such that $e \in T_s$ and $\mathcal{A}$ contains the probes[5] $m(s,u)$ and $m(s,v)$. The cost of a probe assignment $\mathcal{A}$ is $COST_{\mathcal{A}} = \sum_{m(s,u) \in \mathcal{A}} c_{s,u}$ and the *optimal probe assignment* is the one with the minimum cost.

B.1. The Hardness of the Probe Assignment Problem

In the following, we show that computing the optimal assignment is NP-hard.

Theorem 4: Given a set of stations S, the problem of computing the optimal probe assignment is NP-hard.

Proof: We show a reduction from the *vertex cover* problem, which is defined as follows: Given $k \in \mathbb{Z}^+$ and a graph $\hat{G} = (\hat{V}, \hat{E})$, find a subset $V' \subseteq \hat{V}$ containing at most k vertices such that each edge in $\hat{E}$ is incident on a node in V'. For a graph $\hat{G}$, we define an instance of the probe assignment problem, and we show that there is a vertex cover of size at most k for $\hat{G}$ if and only if there exists a feasible probe assignment $\mathcal{A}$ with cost at most $COST_{\mathcal{A}} = 5 \cdot |\hat{V}| + |\hat{E}| + k$. We assume that the cost of any probe $c_{s,u} = 1$, thus, $COST_{\mathcal{A}}$ is the number of probes in $\mathcal{A}$ (the proof for $c_{s,u} = h_{s,u}$ is similar).

For a graph $\hat{G}$, we construct the network graph $G(V, E)$ and set of stations S for the probe assignment problem as follows. In addition to a root node r, graph G contains for each node $\hat{v}_i$ in $\hat{G}$ four nodes, w_i, u_{i1}, u_{i2} and u_{i3}, connected by the edges (w_i, r), (w_i, u_{i1}), (u_{i1}, u_{i2}), (u_{i1}, u_{i3}) and (u_{i2}, u_{i3}). Also, for every edge $(\hat{v}_i, \hat{v}_j)$ in $\hat{G}$, we add the edge (w_i, w_j) to G. Figure 4 shows an example of the constructed G for the graph $\hat{G}$ containing nodes $\hat{v}_1$, $\hat{v}_2$ and $\hat{v}_3$, and edges $(\hat{v}_1, \hat{v}_2)$ and $(\hat{v}_2, \hat{v}_3)$. Finally, we assign a weight $1+\epsilon$ to each edge (w_i, w_j) in G, while the remaining edges are assigned a weight of 1. These weights are used only for determining the RTs that in our reduction are the shortest path trees, and we assume that there are monitoring stations at node r and nodes u_{i3} for each vertex $\hat{v}_i \in \hat{G}$. Figure 4 illustrates the RTs of nodes r and

[5] If s is one of the edge endpoints, say node v, then the probe $m(s,v)$ is omitted from $\mathcal{A}$.

u_{14}. Note that edge (w_i, w_j) is only contained in the RTs of u_{i3} and u_{j3}, and (u_{i1}, u_{i2}) is not contained in the RT of u_{i3}.

We first show that if there is a vertex cover V' of size at most k for $\hat{G}$, then there exists a feasible assignment $\mathcal{A}$ containing no more than $5 \cdot |\hat{V}| + |\hat{E}| + k$ probes. For measuring the latency of the five edges corresponding to $\hat{v}_i \in \hat{V}$, $\mathcal{A}$ contains five probe messages: $m(r, w_i)$, $m(r, u_{i1})$, $m(r, u_{i2})$, $m(u_{i3}, u_{i1})$ and $m(u_{i3}, u_{i2})$. So edges (w_i, w_j) (corresponding to $(\hat{v}_i, \hat{v}_j) \in \hat{E}$) are the only edges in G whose latency still remains to be measured. Since V' is a vertex cover of $\hat{G}$, it must contain one of $\hat{v}_i$ or $\hat{v}_j$. Suppose $\hat{v}_i \in V'$. Then, $\mathcal{A}$ contains the following two probes $m(u_{i3}, w_i)$ and $m(u_{i3}, w_j)$ for each edge (w_i, w_j). Because the probe message $m(u_{i3}, w_i)$ is common to the measurement of all edges (w_i, w_j) corresponding to edges covered by $\hat{v}_i \in V'$ in $\hat{G}$, and size of V' is at most k, $\mathcal{A}$ contains at most $5 \cdot |\hat{V}| + |\hat{E}| + k$ probes messages.

We next show that if there is a feasible probe assignment $\mathcal{A}$ containing at most $5 \cdot |\hat{V}| + |\hat{E}| + k$ probes, then there exists a vertex cover of size at most k for $\hat{G}$. Let V' consist of all nodes $\hat{v}_i$ such that $\mathcal{A}$ contains the probe $m(u_{i3}, w_i)$. Since each edge (w_i, w_j) is in the RTs of only u_{i3} and u_{j3}, $\mathcal{A}$ must contain one of $m(u_{i3}, w_i)$ or $m(u_{j3}, w_j)$, and thus V' must be a vertex cover of $\hat{G}$. Further, we can show that V' contains at most k nodes. Suppose that this is not the case and V' contains more than k nodes. Then, $\mathcal{A}$ must contain greater than k probes $m(u_{i3}, w_i)$ for $\hat{v}_i \in \hat{V}$. Further, in order to measure the latencies of all edges in E, $\mathcal{A}$ must contain $5 \cdot |\hat{V}| + |\hat{E}|$ additional probes. Of these, $|\hat{E}|$ are needed for edges (w_i, w_j), $3 \cdot |\hat{V}|$ for edges (u_{i3}, u_{i1}), (u_{i3}, u_{i2}) and (r, w_i), and $2 \cdot |\hat{V}|$ for edges (u_{i1}, u_{i2}). $\mathcal{A}$ contains 2 probe messages for each edge (u_{i1}, u_{i2}) because the edge does not belong to the RT of u_{i3} and thus 2 probe messages (v, u_{i2}) and (v, u_{i1}), $v \neq u_{i3}$ are needed to measure the latency of edge (u_{i1}, u_{i2}). This, however, leads to a contradiction since $\mathcal{A}$ would contain more than $5 \cdot |\hat{V}| + |\hat{E}| + k$ probes. Thus V' must be a vertex cover of size no greater than k. □

B.2. The Simple Probe Assignment Algorithm

We now describe a *simple probe assignment* algorithm that computes an assignment whose cost is within a factor of 2 of the optimal solution. Consider a set of monitoring stations S and for every edge $e \in E$, let $S_e = \{s | s \in S \wedge e \in T_s\}$ be the set of stations that can monitor e. For each edge $e = (u, v) \in E$, the algorithm selects as the monitoring station of e the node $s_e \in S_e$ for which the cost $c_{s_e,u} + c_{s_e,v}$ is minimum. In case of ties (that is, multiple stations have the same cost), it selects s_e to be the station with the minimum identifier among the tied stations. Then, it adds the probe messages $m(s_e, u)$ and $m(s_e, v)$ to $\mathcal{A}$.

Theorem 5: The approximation ratio of the simple probe assignment algorithm is 2.

Proof: For monitoring the delay of any edge $e \in E$, at least one station $s \in S$ must send two probe messages, one to each endpoint of e. As a result, in any feasible probe assignment, at least one probe message can be associated with each edge e. Let it be the message that is sent to the farthest endpoint of e from the monitoring station. Let $\mathcal{A}^*$ be the optimal probe assignment and let s_e^* be the station that monitors edge e in $\mathcal{A}^*$. So, in $\mathcal{A}^*$, the cost of monitoring edge $e = (u, v)$ is at least $\max\{c_{s_e^*,u}, c_{s_e^*,v}\}$. Let s_e be the selected station for monitoring edge e in the assignment $\mathcal{A}$ returned by the simple probe assignment algorithm. s_e minimizes the cost $c_{s,u} + c_{s,v}$, for every $s \in S_e$. Thus, $c_{s_e,u} + c_{s_e,v} \leq c_{s_e^*,u} + c_{s_e^*,v} \leq 2 \cdot \max\{c_{s_e^*,u}, c_{s_e^*,v}\}$. Thus, $COST_{\mathcal{A}} \leq 2 \cdot COST_{\mathcal{A}^*}$. □

Note that the time complexity of the simple probe assignment algorithm can be shown to be $O(|S| \cdot |V|^2)$.

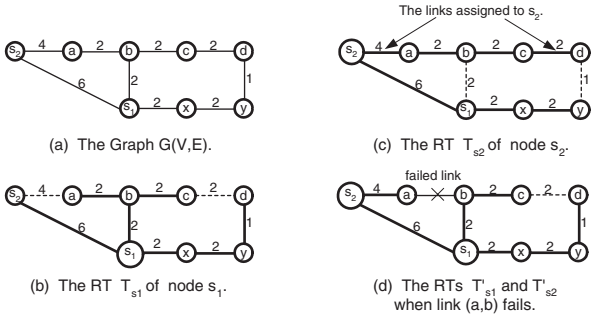

Fig. 5. The affect of a link failure on the monitoring system.

V. Delay Monitoring with Fault Detection

The probe-based delay monitoring system described in the previous section can be extended to also detect network link failures. Using probes to pinpoint network faults has several advantages over monitoring routing protocol messages (e.g., OSPF LSAs) or using SNMP traps to indentify failed links. First, probe-based techniques are routing protocol agnostic; as a result, they can be used with a range of protocols like OSPF, IS-IS, RIP etc. Second, SNMP trap messages may be somewhat unreliable since they are transported using UDP datagrams. The probe-based fault detection algorithms proposed in this section can be used either stand-alone, or in conjunction with SNMP traps, to build a robust infrastructure for accurately pinpointing network faults.

Our probe-based methodology from the previous section, while suitable for estimating link delays, may not be able to identify failed network links (unless it is modified).

Example 1: Consider the graph $G(V, E)$ depicted in Figure 5-(a), where each link is labeled with its weight. Let $S = \{s_1, s_2\}$ be the set of selected monitoring stations. The RTs T_{s_1} and T_{s_2} are the shortest path trees rooted at nodes s_1 and s_2 as illustrated in Figures 5-(b) and 5-(c), respectively. The simple probe assignment algorithm assigns all graph links to be monitored by s_1 except the links (s_2, a) and (c, d) which are monitored by s_2. Note that s_2 transmits two probes $m(s_2, c)$ and $m(s_2, d)$ that traverse nodes a, b and c to measure the latency of link (c, d). Now, consider the failure of link (a, b) that causes the RTs of s_1 and s_2 to be modified as shown

in Figure 5-(d)[6]. Specifically, the new RT for s_1 contains the link (s_2, a) instead of (a, b), while the tree for s_2 contains links (s_1, b) and (y, d) instead of (a, b) and (c, d). Clearly, neither s_1 nor s_2 detects the failure of link (a, b), and further since the probes $m(s_2, c)$ and $m(s_2, d)$ traverse a diverse set of nodes ($\{s_1, b\}$ and $\{s_1, x, y\}$, respectively), they are no longer measure the latency of link (c, d). □

In this section, we extend our delay monitoring framework to also detect link failures. Our proposed fault monitoring infrastructure utilizes the same set of stations S used for delay monitoring – thus, the stations in S cover all links in the network. However, as shown above in Example 1, we require a new set of probe messages for identifying failed links (in addition to measuring link delays). In the following subsections, we begin by first computing a near-optimal set of probes for detecting the failure of a network link, and then we describe an algorithm for accurately pinpointing the faulty link based on probe results from the various stations. Note that our methods can be easily extended to also handle node failures.

A. Detecting a Network Link Failure

In our fault monitoring solution, probe messages use the *time-to-live* (TTL) field of the IP header. The TTL field provides an efficient way to bound the number of hops that an IP packet traverses in its path [11]. Essentially, the TTL field of an IP packet is initialized by the source node to the maximum number of hops that the packet is allowed to traverse. Each router along the path followed by the packet then decrements the TTL value by one until it reaches zero. When a router decrements the TTL field to zero, it discards the packet and sends an ICMP "time expired" reply to the packet's source node[7].

Now suppose that we would like link $e = (u, v) \in T_s$ to be monitored by station $s \in S$, and let node v be the node that is further from s. Our strategy is to appropriately set the TTL values in the two probe messages $m(s, u)$ and $m(s, v)$ that measure link e's delay, such that the probes can also detect changes in T_s due to e's failure. One straightforward option is to simply set the TTL field of probe message $m(s, v)$ to $h_{s,v}$, the hop distance in T_s between nodes s and v. This guarantees that the probe message does not traverse more than $h_{s,v}$ hops. Thus, a reply message from a node other than v indicates a link failure along the path $P_{s,v}$ in T_s. While this observation enables us to detect some failures, it may miss others, as illustrated in Example 2 below.

Example 2: Consider the graph in Example 1, and assume a failure of the link (a, b) monitored by s_1. The hop distances from s_1 to nodes a, b before the failure are $h_{s_1,b} = 1$ and $h_{s_1,a} = 2$, and they remain the same after (a, b) fails. Thus, s_1 cannot detect the failure of link (a, b). □

[6]RTs typically adapt to failures in a few seconds or a few tens of seconds depending on the IP routing protocol parameter settings.

[7]This method is used by the `traceroute` application for discovering routers in the path between a given pair of nodes.

The above problem can be fixed by associating the same destination address with the two probe messages for link $e = (u, v)$. Let $m(s, t, h)$ denote the probe message sent by source s to a destination node t with TTL value set to h. Further, let $R_{m(s,t,h)}$ be the node that replies to the probe message. Assuming that node u is closer to station s than node v, *i.e.,* $h_{s,u} < h_{s,v}$, station s can monitor both the delay as well as failure of link e by sending the following two probes: $m(s, v, h_{s,u})$ and $m(s, v, h_{s,v})$. These messages have the same destination, v, but they are sent by s with different TTL values. Clearly, in the absence of failures, the reply for the first message is sent by node u while the reply for the second message is sent by node v, *i.e.,* , $R_{m(s,v,h_{s,u})} = u$ and $R_{m(s,v,h_{s,v})} = v$. Thus, the difference in the round-trip times of the two probes gives link e's delay. Further, if link e fails, then since u and v are no longer adjacent in the new RT $T_{s,e}$ for s, then at least one of these replies will be originated by a different node. This means that either $R_{m(s,v,h_{s,u})} \neq u$ or $R_{m(s,v,h_{s,v})} \neq v$ or both. Thus, the probe assignment $\mathcal{A}$ essentially contains, for each edge $e = (u, v)$, the probes $m(s_e, v, h_{s_e,u})$ and $m(s_e, v, h_{s_e,v})$, where $s_e \in S_e$ is the station for which $c_{s_e,u} + c_{s_e,v}$ is minimum (ties are broken in favor of stations with smaller identifiers). Further, if for $e = (u, v)$, $R_{m(s_e,v,h_{s_e,u})} \neq u$ or $R_{m(s_e,v,h_{s_e,v})} \neq v$, then station s_e informs the NOC that a network link has failed.

Revisiting Example 2, s_1 sends probe messages $m(s_1, a, 1)$ and $m(s_1, a, 2)$ to monitor link (a, b). In case link (a, b) fails, then with the new RT for s_1, $R_{m(s_1,a,1)}$ would be s_2 instead of b, and so s_1 would detect the failure of link (a, b). The probe assignment $\mathcal{A}$ described above thus monitors all network links for both delay as well as faults. Further, using the same arguments of Theorem 5, we can prove that the cost of assignment $\mathcal{A}$ is within a factor of 2 of the optimal probe assignment.

Let $P_{s,e}$ denote the path between station s and link e (and including e) in T_s. While assignment $\mathcal{A}$ ensures that the failure of a link e will be detected by the station s that monitors e, s may not always be able to infer (by itself) whether the faulty link is e or some other link in $P_{s,e}$. The reason for this is that replies for the probe messages for link e may be affected by the failure of a link f in $P_{s,e}$, and f may not be monitored by s. Thus, s may be unable to conclude whether the erroneous replies to e's probes were caused by the failure of e or f.

Example 3: Consider the graph in Example 1 where station s_2 monitors links (s_2, a) and (c, d), and s_1 monitors the remaining links. Suppose that for probe $m(s_2, d, 3)$ that monitors link (c, d), $R_{m(s_2,d,3)}$ is y instead of c. This could be the result of the failure of either link (c, d) or (a, b). In both cases, the new routing path from s_2 to d traverses nodes s_1, x and y. Since s_2 does not monitor link (a, b), it cannot conclude by itself, which of links (a, b) or (c, d) have failed. □

In the following subsection, we present an algorithm for accurately pinpointing the faulty link, in which each monitoring station sends its failure information to the NOC.

B. Identifying the Failed Link

As shown in Example 3, with probe assignment $\mathcal{A}$, when a link fails, the station monitoring the link detects the failure, but may not always be able to accurately identify the location of the failed link. However, station s can narrow down the possible candidates to links in the path connecting it to the failed link.

Lemma 1: If for link (u, v) monitored by station s, $R_{m(s,v,h_{s,u})} \neq u$ or $R_{m(s,v,h_{s,v})} \neq v$, then there is a failure along the path $P_{s,v}$ in T_s.

Proof: Suppose in contrast that there is no failure in the path $P_{s,v}$. Thus, the messages $m(s, u, h_{s,u})$ and $m(s, v, h_{s,v})$ traverse the paths $P_{s,u}$ and $P_{s,v}$, and reply messages are sent by nodes u and v, respectively. This contradicts the statement of the lemma that $R_{m(s,v,h_{s,u})} \neq u$ or $R_{m(s,v,h_{s,v})} \neq v$. □

In the remainder of this subsection, we show how Lemma 1 can be used to identify the faulty link after a monitoring station s detects a failure. We assume a single link failure, since the likelihood of multiple concurrent link failures within a short time interval is very small. Let $E_s \subseteq E$ be the set of links monitored by a station $s \in S$ as a result of assignment $\mathcal{A}$. Since a link failure may affect the paths of multiple probe messages sent by s, we denote by $F_s \subseteq E_s$ the set of links $(u, v) \in E_s$ for which $R_{m(s,v,h_{s,u})} \neq u$ or $R_{m(s,v,h_{s,v})} \neq v$. When a station s concludes a failure, it computes the set F_s, and then finds the link $f_s \in F_s$ closest to s. Thus, there is no other link $e \in F_s$ in the path P_{s,f_s}, and due to Lemma 1, the faulty link must be included in this path.

Clearly, if all the links in P_{s,f_s} are in E_s, then f_s must be the failed link since none of the other links in P_{s,f_s} are in F_s. However, it is possible that some links in P_{s,f_s} are not in E_s (in Example 3, link (a, b) is not in $P_{s_2,(c,d)}$). Thus, s cannot conclude that f_s is the failed link since a link in $P_{s,f_s} - E_s$ may have failed. One option is for s to simply send additional probe messages, $m(s, v, h_{s,u})$ and $m(s, v, h_{s,v})$ for monitoring every link $(u, v) \in P_{s,f_s} - E_s$. Station s can then declare the faulty link to be the link (u', v') closest to it in $P_{s,f_s} - E_s$ for which $R_{m(s,v',h_{s,u'})} \neq u'$ or $R_{m(s,v',h_{s,v'})} \neq v'$. With this technique, s sends $O(|V|)$ additional messages in the worst case. Further, since the faulty link may be detected by multiple monitoring stations, the total number of extra messages is $O(|S| \cdot |V|)$.

We now present a centralized approach for identifying the faulty link at the system NOC, without sending additional probe messages. In the NOC-based approach, each station s that detects a failure transmits to the NOC a "FAULT DETECTED" message containing the identity of link f_s. When the NOC receives the message it calculates the set C_s of the potentially failed links detected by s as:

$$C_s = (P_{s,f_s} - E_s) \bigcup \{f_s\}$$

Note that the NOC may receive a FAULT DETECTED message from more than one station, and for these stations C_s will be non-empty. For the remaining stations, $C_s = \emptyset$. Once the NOC has received the FAULT DETECTED message from all stations that detected a failure, it computes the identity of the faulty link by evaluating the following expression:

$$C = \bigcap_{C_s \neq \emptyset} C_s - \bigcup_{C_s = \emptyset} E_s \quad (1)$$

In the above equation, the second term prunes from the candidate set, links that are monitored by stations which did not detect failures.

Example 4: Consider the graph in Example 1 where station s_2 monitors links (s_2, a) and (c, d), and s_1 monitors the remaining links. Suppose that link (a, b) fails. Both s_1 and s_2 detect a failure, and send to the NOC FAULT DETECTED messages containing links (a, b) and (c, d), respectively. The NOC calculates the sets $C_{s_1} = \{(a, b)\}$ and $C_{s_2} = \{(a, b), (b, c), (c, d)\}$, and computes the set $C = C_{s_1} \cap C_{s_2} = \{(a, b)\}$, that contains only the failed link. □

We next prove that C indeed contains a single element which is the failed link.

Lemma 2: In assignment $\mathcal{A}$, let $e_1, e_2 \in E$ be a pair of links monitored by two different stations $s_1, s_2 \in S$, respectively. If $e_1 \in P_{s_2,e_2}$, then $e_2 \notin P_{s_1,e_1}$.

Proof: Suppose that $e_2 \in P_{s_1,e_1}$. Without loss of generality, let s_1 have a smaller identifier than s_2. Also, for station s and edge $e = (u, v)$, let $c_{s,e} = c_{s,u} + c_{s,v}$. Now since e_2 is closer to s_1 than e_1 in T_{s_1}, we conclude that $c_{s_1,e_2} \leq c_{s_1,e_1}$. Link e_1 is monitored by s_1 and not s_2; therefore, $c_{s_1,e_1} \leq c_{s_2,e_1}$. Because $e_1 \in P_{s_2,e_2}$, we get that $c_{s_2,e_1} \leq c_{s_2,e_2}$. Thus, it follows that $c_{s_1,e_2} \leq c_{s_2,e_2}$, and since s_1 has a smaller identifier, e_2 must be monitored by s_1, which leads to a contradiction. □

Theorem 6: Set C in Equation 1 contains only the faulty link f.

Proof: First, we show that f is in C. The set C_s for every station $s \in S$ is either empty or contains the link f (since $f \in P_{s,f_s}$). Thus, $f \in \bigcap_{C_s \neq \emptyset} C_s$. Note also that for stations s such that $C_s = \emptyset$, $f \notin T_s$, and so $f \notin \bigcup_{C_s = \emptyset} E_s$. Thus, $f \in C$. We next show that $|C| = 1$. Clearly, C does not contains any link $e \in E_s$ such that $C_s = \emptyset$. Further, for links $e \in E_s$ such that $C_s \neq \emptyset$ and $e \neq f_s$, it is the case that $e \notin C_s$, and thus $e \notin C$. As a result, it follows that $C \subseteq \{f_s : C_s \neq \emptyset\}$. Now let s' be the monitoring station for the failed link f. Consider a station $s \neq s'$ such that $C_s \neq \emptyset$ and $f_s \neq f$. Since $f \in P_{s,f_s}$, due to Lemma 2, it follows that $f_s \notin P_{s',f}$ and thus, $f_s \notin C_{s'}$. As a result, $f_s \notin C$ and C contains only f. □

VI. Robust Link Monitoring

A system for monitoring links is *robust* if it continues to monitor network links even if one or more links in the network fail. A key challenge in designing such a robust monitoring system is selecting a set of stations whose RTs *always* cover all the active network links. The problem is that when a link f fails, the new RT $T_{s,f}$ for a station s may be different from T_s, the RT prior to the failure. As a result, a station s that was responsible for monitoring a link $e \in T_s$ may be unable to monitor the link once link f fails. The problem is further complicated by the fact that the RTs $T_{s,f}$ for the various stations may not be known in advance (when stations are selected).

As an example, consider the graph in Example 1 with RTs T_{s_1} and T_{s_2} as shown in Figures 5-(b) and 5-(c), respectively. The failure of link $f = (a, b)$ causes the RTs of s_1 and s_2 to be modified as shown in Figure 5-(d). Clearly, link $(c, d) \in T_{s_2}$ can no longer be monitored after f fails since $(c, d) \notin T_{s_2,f}$. Thus, the monitoring system (with stations s_1 and s_2) in Figure 5-(a) is not robust.

In the following subsection, we consider the problem of efficient placement of monitoring stations that guarantee delay and fault monitoring of all active links in the presence of at most $K - 1$ failures. We refer to this problem as the *K-Fault Resilient Monitoring* (K-FRM) problem, and develop a solution with an approximation ratio of $\ln(|E|)$ for it. Once the set S of stations is computed, the probe assignment is computed as described in previous sections. For simplicity, we only consider link failures; however, our general approach can be easily enhanced to support nodes failures as well.

A. The K-Fault Resilient Monitoring Problem

A set S of stations is resilient to one fault if and only if it satisfies the following *fault resilience* property: For every link $f \in E$, for every other link $e \neq f$, $e \in T_{s,f}$ for some station $s \in S$. The fault resilience property ensures that when an arbitrary link f fails, every other active link is contained in the new RT of some station in S. However, finding a set of stations that satisfies the property may be difficult since the trees $T_{s,f}$ may not be known in advance. Further, the property becomes extremely complex when we consider K-fault resilience, since any combination of $K - 1$ links can potentially fail.

Due to the above-mentioned reasons, we instead require S to satisfy a stronger but simpler condition that implies the above fault resilience property. The condition does not rely on the knowledge of $T_{s,f}$, but exploits the fact that T_s and $T_{s,f}$ are identical with respect to paths that do not contain the failed link f. Let $F_e(T_s)$ be the parent link of link e in T_s. Then the stronger condition is based on the key observation that S is resilient to a single link failure if one of the following two conditions holds for every link $e \in E$:

1) One of e's endpoints is in S.
2) Link e is in the RTs of at least two monitoring stations $s_1, s_2 \in S$, and $F_e(T_{s_1}) \neq F_e(T_{s_2})$.

The following lemma presents a more general sufficient condition for any K-fault resilient monitoring system.

Lemma 3: A set S of monitoring stations is K-fault resilient if for every link $e = (u, v) \in E$, at least one of the following conditions is satisfied.
(1) One of nodes u or v is in S, or
(2) There are K nodes in S, denoted by $S_e = \{s_1, s_2, \cdots, s_K\}$, whose RTs T_{s_i} contain link e, and for every pair of distinct nodes $s_i, s_j \in S_e$, it is the case that $F_e(T_{s_i}) \neq F_e(T_{s_j})$.

Proof: Clearly, if one of the endpoints of link $e = (u, v)$ is in S, then this endpoint can monitor e as long as it is active. On the other hand, if neither u nor v are in S, then there must exist a subset $S_e \subseteq S$ as described above. We show that the paths $P_{s_i,e}$ for every $s_i \in S_e$ define K disjoint paths (excluding link e). Thus, even if $K - 1$ links fail, for some $s_i \in S_e$, $P_{s_i,e}$ contains none of the failed links, and therefore s_i will be able to monitor link e. Suppose that the K paths are not disjoint, that is, for a pair of stations $s_i, s_j \in S_e$, paths $P_{s_i,e}$ and $P_{s_j,e}$ have a common node w which is not an endpoint of e. Since e belongs to both trees T_{s_i} and T_{s_j}, and messages are forwarded according to their destination address (the IP forwarding technique), we can conclude the following: (1) Link e is included in the tree T_w, and (2) Every message from s_i or s_j to nodes u or v passes through node w and is subsequently forwarded along the same path $P_{w,e}$. However, these facts contradict the assumption that $F_e(T_{s_i}) \neq F_e(T_{s_j})$. □

Thus, we can define the *K-Fault Resilient Monitoring* (K-FRM) problem as follows.

Definition 2 (The K-FRM Problem): Given are a constant K, a graph $G(V, E)$ and a RT T_v for every node $v \in V$. Find the smallest subset $S \subseteq V$ such that for every link $e \in E$, at least one of the following two conditions is satisfied.
(1) One of nodes u or v is in S, or
(2) There are K nodes in S, denoted by $S_e = \{s_1, s_2, \cdots, s_K\}$, whose RTs T_{s_i} contain link e, and for every pair of distinct nodes $s_i, s_j \in S_e$, it is the case that $F_e(T_{s_i}) \neq F_e(T_{s_j})$. □

The K-FRM problem is a generalization of the LM problem defined in Section IV, and any instance of the LM problem can be represented as an instance of the K-FRM problem with $K = 1$. Thus, Theorems 1 and 2 imply the following result.

Theorem 7: The K-FRM problem is NP-hard. Further, the lower bound of any approximation algorithm for the problem is $O(\ln(|V|))$.

B. The Partial Multi-Set Cover Problem

In order to solve the K-FRM problem, we map it to an extended version of the set cover (SC) problem, which we refer to as the *Partial Multi-Set Cover* (PMSC) problem.

Definition 3 (The PMSC Problem): Given are a constant K, a universe of elements Z, and the following two collections of subsets of Z: $\mathcal{Y} = \{Y_1, Y_2, \cdots, Y_m\}$ and $\mathcal{Q} = \{Q_1, Q_2, \cdots, Q_n\}$. Each $Y_j \in \mathcal{Y}$ contains at least K elements, and is disjoint from other members of $\mathcal{Y}$. Find the smallest collection $\mathcal{S} \subseteq \mathcal{Q}$ such that for every $Y_j \in \mathcal{Y}$, $|\bigcup(\mathcal{S}) \cap Y_j| \geq K$. □

Above, $\bigcup(\mathcal{S}) = \bigcup_{Q \in \mathcal{S}} Q$ is the union of the collection $\mathcal{S}$. It is easy to see that the PMSC problem is more general than the SC problem. Every instance $I(Z, \mathcal{Q})$ of the SC problem can be reduced to an instance of the PMSC problem by selecting $K = 1$ and defining the collection $\mathcal{Y} = \{\{z\} | \forall z \in Z\}$ in which every subset Y_i contains a single element of Z. Thus, the optimal solution of the calculated PMSC instance is also the optimal solution of the given SC instance, $I(Z, \mathcal{Q})$. Therefore, PMSC is NP-hard, and it has a lower bound of at least $\ln(|Z|)$.

We now describe a greedy algorithm for solving the PMSC problem (see Figure 6 for pseudocode). Our proposed algorithm uses ideas similar to those employed by the greedy SC algorithm; specifically, in each iteration, our greedy PMSC

$\mathcal{S} \leftarrow \emptyset$
$u_{total} \leftarrow mK$
While $u_{total} > 0$ do
  For every $Y_j \in \mathcal{Y}$ do
    $u_j \leftarrow K - \min\{K, |\bigcup(S) \cap Y_j|\}$
  For every $Q_i \in \mathcal{Q} - \mathcal{S}$ do
    $n_i = \sum_{j=1}^{m} \min\{u_j, |Q_i \cap (Y_i - \bigcup(\mathcal{S}))|\}$
  $Q_i^* \leftarrow arg\max_{Q_i \in \mathcal{Q}-\mathcal{S}} n_i$
  $\mathcal{S} \leftarrow \mathcal{S} \bigcup \{Q_i^*\}$
  $u_{total} \leftarrow u_{total} - n_i^*$
End While
Return $\mathcal{S}$

Fig. 6. A Formal Description of the Greedy PMSC Algorithm.

algorithm selects the most cost-effective set $Q_i^* \in \mathcal{Q}$ until all the sets in $\mathcal{Y}$ are covered, as explained below. In the algorithm, $\mathcal{S} \subseteq \mathcal{Q}$ is the collection of subsets that have been selected so far. We say that set Y_j is *covered* by $\mathcal{S}$ if $\bigcup(\mathcal{S})$ contains at least K elements from Y_j, *i.e.,* $|\bigcup(S) \cap Y_j| \geq K$. With each set $Y_j \in \mathcal{Y}$, we associate a variable u_j that specifies the number of *uncovered* elements in $Y_j - \bigcup(S)$ that still need to be selected to cover Y_j. Thus, $u_j = 0$ if Y_j is already covered, otherwise $u_j = K - |\bigcup(S) \cap Y_j|$. We use $u_{total} = \sum_{j=1}^{m} u_j$ to represent the total number of uncovered elements that should be selected for covering all the sets $Y_j \in \mathcal{Y}$. Note that when $u_{total} = 0$, the calculated selection $\mathcal{S}$ is a feasible solution. With every set $Q_i \in \mathcal{Q}$, $Q_i \notin \mathcal{S}$, we associate a value n_i that is the total over all uncovered sets Y_j, of the number of uncovered elements in Y_j that are also contained in Q_i, and can be selected to cover Y_j. Thus, $n_i = \sum_{j=1}^{m} \min\{u_j, |Q_i \cap (Y_i - \bigcup(\mathcal{S}))|\}$. Note that adding Q_i to $\mathcal{S}$ causes u_{total} to decrease by amount n_i. Therefore, in each iteration, the greedy algorithm (see Figure 6) adds to $\mathcal{S}$, the *most cost-effective* set $Q_i \in \mathcal{Q} - \mathcal{S}$ that maximizes the ratio n_i.

We now calculate the approximation ratio of the greedy PMSC algorithm using a technique similar to the one used for proving the approximation ratio of the greedy SC algorithm in [3]. Consider the solution $\mathcal{S}$ returned by the greedy PMSC algorithm. Its cost is $Cost(\mathcal{S}) = |\mathcal{S}|$. Let Q_r^* be the set added to $\mathcal{S}$ in the r^{th} iteration, and n_r^* be the amount u_{total} is reduced due to the addition of Q_r^* to $\mathcal{S}$. Since initially $u_{total} = mK$, it follows that $\sum_{r=1}^{|\mathcal{S}|} n_r^* = mK$. Let OPT be the optimal solution and let $Cost(OPT) = |OPT|$ denote its cost.

Lemma 4: In the r^{th} iteration of the greedy algorithm, $n_r^* \geq \frac{\sum_{l=r}^{|\mathcal{S}|} n_l^*}{Cost(OPT)}$.

Proof: At the beginning of each iteration, u_{total} is the number of uncovered elements that need to be selected for covering all the sets $Y_j \in \mathcal{Y}$. In any iteration, the unselected sets of OPT can cover the remaining uncovered sets from $\mathcal{Y}$ with cost at most $Cost(OPT)$. Therefore, among these sets of OPT there must be a set Q_i such that the number n_i of uncovered elements in $\mathcal{Y}$ that are covered by Q_i is at least $\frac{u_{total}}{Cost(OPT)}$. The algorithm selects the set Q_i that maximizes n_i. Thus, in the r^{th} iteration, it must be the case that for the selected set Q_r^*, $n_r^* \geq \frac{u_{total}}{Cost(OPT)}$. Further, since at the beginning of the r^{th} iteration, $u_{total} = \sum_{l=r}^{|\mathcal{S}|} n_l^*$, the lemma follows. □

From Lemma 4, it follows that $Cost(\mathcal{S}) = |\mathcal{S}| \leq Cost(OPT) \cdot \sum_{r=1}^{|\mathcal{S}|} \frac{n_r^*}{\sum_{l=r}^{|\mathcal{S}|} n_l^*}$. Since $\sum_{r=1}^{|\mathcal{S}|} n_r^* = mK$, a series of algebraic manipulations lead to the following result.

Theorem 8: The approximation ratio of the greedy PMSC algorithm is $\ln(K) + \ln(m) + 1$, where $m = |\mathcal{Y}|$.

C. K-FRM Problem Solution

We solve the K-FRM problem by first reducing it to the PMSC problem, and then applying the greedy PMSC algorithm shown in Figure 6. In order to map a K-FRM instance involving the graph $G(V, E)$ to a PMSC instance, we need to define the collections $\mathcal{Y}$ and $\mathcal{Q}$. The collection $\mathcal{Y}$ contains $m = |E|$ disjoint sets, where each set $Y_e \in \mathcal{Y}$ results from a link $e \in E$ and contains at least K elements. The collection $\mathcal{Q}$ contains $n = |V|$ sets, where each set $Q_v \in \mathcal{Q}$ is derived from the RT T_v of node v.

Let $S \subseteq V$ be any subset of nodes and let $\mathcal{S} \subseteq \mathcal{Q}$ be the corresponding collection of sets such that $\mathcal{S} = \{Q_v | v \in S\}$. Our reduction guarantees that S is a feasible solution for the K-FRM problem if and only if $\mathcal{S}$ is a feasible solution for the corresponding PMSC instance. Here, a feasible solution S for the K-FRM problem is one for which every link $e \in E$ satisfies one of the two conditions of Definition 2, while $\mathcal{S}$ is a feasible solution for a PMSC instance if for every $Y_j \in \mathcal{Y}$, $|\bigcup(S) \cap Y_j| \geq K$.

In order to achieve the above-mentioned goal, we include in Z two types of elements, where each type is used to ensure that one of the conditions in Definition 2 is captured. Let A_v and A_e denote the set of links in E that are incident on node v and endpoints of edge e, respectively. For capturing the first condition, we define for every node $v \in V$ and every one of its outgoing links $e \in A_v$, K different elements that are included in both sets Y_e and Q_v. Each element is represented by a triple $< e, v, k >$, for every $1 \leq k \leq K$. Thus, selecting a set Q_v ensures that all the sets Y_e, $e \in A_v$ are covered. The second condition is reflected in a more straightforward manner. For every link $e = (u, v) \in E$ and each one of its adjacent links $e' \in A_e$ we define an element $< e, e' >$ that is included in the set Y_e. The element $< e, e' >$ is also included in the set Q_v of every node v such that link e' is the parent of link e in the tree T_v, *i.e.,* $e' = F_e(T_v)$. Thus, selecting one of the sets Q_v ensures that a single uncovered element in Y_e is covered.

In summary, for every $e = (u, v) \in E$, the set $Y_e \in \mathcal{Y}$ is defined as,

$$Y_e = \{< e, v, k >, < e, u, k > | 1 \leq k \leq K\} \bigcup \{< e, e' > | e' \in A_e\}$$

and for every node $v \in V$, the set Q_v is defined as,

$$Q_v = \{< e, v, k > | e \in A_v, 1 \leq k \leq K\} \bigcup \{< e, F_e(T_v) > | e \in T_v \wedge e \notin A_v\}$$

Suppose the greedy PMSC algorithm returns collection $\mathcal{S}$ as the solution to the above PMSC instance. Then, the solution to the original K-FRM problem is computed as $S = \{v | Q_v \in$

$\mathcal{S}\}$. Clearly, since $\mathcal{S}$ covers every set $Y_e \in \mathcal{Y}$, every link $e \in E$ either has an endpoint in S, or it appears in at least K RTs of nodes in S, with distinct parent links. Thus, S is a feasible solution to the K-FRM problem. Further, since the mapping between $\mathcal{S}$ and S does not alter the cost of the solutions, due to Theorem 8, it follows that the cost of solution S is within a $\ln(K) + \ln(|E|) + 1$ factor of the optimal solution to the K-FRM problem.

Note that the greedy algorithm takes $O(|V|^3)$ steps to solve the PMSC instance corresponding to the K-FRM problem since $|Q_v| = O(|V|)$ and $|\mathcal{Q}| = |V|$.

D. Probe Assignment in the Presence of Failures

Once we have selected a set S of K-fault resilient monitoring stations, initially each link $e = (u, v)$ is monitored by the station $s \in S$ such that $e \in T_s$ and $c_{s,u} + c_{s,v}$ is minimum. The NOC keeps track of failed network links in the variable X. When the NOC detects the failure of a network link f, it adds the link to X. Further, for each link e currently being monitored by a station s such that $P_{s,e}$ contains the failed link f, the NOC computes a new station s' for monitoring e. The new station s' for e, in addition to satisfying the conditions $e \in T_{s'}$ and $c_{s',u} + c_{s',v}$ is minimum, also needs to satisfy the condition that $P_{s',e} \cap X = \emptyset$. This ensures that the routing path from s' to e does not contain any of the failed links. Note that since S is K-fault resilient, there are at least K disjoint routing paths from stations in S to each link e not adjacent to a station in S. Thus, each active network link can be continuously monitored by some station in S as long as the number of failures $|X| < K$. To monitor both the fault and delay of a link $e = (u, v)$, the station s monitoring e sends the probes $m(s, v, h_{s,u})$ and $m(s, v, h_{s,v})$, as described in Section V.

VII. Summary

In this paper, we proposed a two-phased approach to monitoring that ensures complete coverage of the network in the presence of link failures, and minimizes the monitoring overhead on the underlying production network. In the first phase of our approach, we computed the locations of a minimal set of monitoring stations such that all network links are covered, even if some links in the network were to fail. Subsequently, in the second phase, we computed the minimal set of probe messages to be sent by each station such that the latency of every network link can be measured, and faulty network links can be isolated. Unfortunately, both the station selection and the probe assignment problems are NP-hard. However, our proposed polynomial-time greedy approximation algorithms achieve close to the best possible approximations to both the station selection and the probe assignment problems.

References

[1] M. Adler, T. Bu, R. Sitaraman and D. Towsley, "Tree Layout for Internal Network Characterizations in Multicast Networks", In *Proceedings of NGC'01*, London, UK, November 2001.

[2] T. Bu, N. Duffield, F. Lo Presti and D. Towsley, "Network Tomography on General Topologies", In *Proceedings of the ACM SIGMETRICS*, 2002.

[3] V. Chavatal, "A Greedy Heuristic for the Set-Covering Problem", *Math. of Operation Research*, Vol. 4, No. 3, pp 233-235, 1979.

[4] M. Dilman and D. Raz, "Efficient Reactive Monitoring", In *Proceedings of the IEEE INFOCOM'2001*, Alaska, April 2001.

[5] William Stallings, "SNMP, SNMPv2, SNMPv3, and RMON 1 and 2", *Addison-Wesley Longman* Inc., 1999, (Third Edition).

[6] "NetFlow Services and Applications", Cisco Systems, 1999.

[7] Y. Breitbart, C-Y. Chan, M. Garofalakis, R. Rastogi and A. Silberschatz, "Efficiently Monitoring Bandwidth and Latency in IP Networks", In *Proceedings of the IEEE INFOCOM'2000*, Tel-Aviv, Israel, March 2000,

[8] P. Francis, S. Jamin, V. Paxson, L. Zhang, D. F. Gryniewicz, and Y. Jin, "An Architecture for a Global Internet Host Distance Estimation Service", In *Proceedings of IEEE INFOCOM'99*, New York City, New York, March 1999.

[9] S. Jamin, C. Jin, Y. Jin, Y. Raz, Y. Shavitt, and L. Zhang, "On the Placement of Internet Instrumentation", In *Proceedings of IEEE INFOCOM'2000*, Tel Aviv, Israel, March 2000.

[10] Y. Shavitt, X. Sun, A. Wool and B. Yener, "Computing the Unmeasured: An Algebraic Approach to Internet Mapping", In *Proceedings of IEEE INFOCOM'2001*, Alaska, April 2001.

[11] S. W. Richard, "TCP/IP illustrated", *Addison-Wesley Publishing Company*, 1994.

[12] Cooperative Association for Internet Data Analysis (CAIDA), http://www.caida.org/.

[13] V. Jacobsen, "Pathchar – A Tool to Infer Characteristics of Internet Paths", April 1997. ftp://ftp.ee.lbl.gov/pathchar.

[14] R. L. Carter and M. E. Crovella, "Server Selection Using Dynamic Path Characterization in Wide-Area Networks", In *Proceedings of IEEE INFOCOM'99*, Kobe, Japan, April 1997.

[15] K. Lai and M. Baker, "Measuring Bandwidth", In *Proceedings of IEEE INFOCOM'99*, New York City, New York, March 1999.

[16] C. Dovrolis, P. Ramanathan and D. Moore, "What Do Packet Dispersion Techniques Measure?", In *Proceedings of IEEE INFOCOM'2001*, Alaska, April 2001.

[17] Y. Bejerano and R. Rastogi, "Efficient Monitoring Schemes for IP Networks", Research Report, Bell Labs, 2001.

[18] A. Reddy, R. Govindan and D. Estrin, "Fault Isolation in Multicast Trees", In *Proceedings of the ACM SIGCOMM*, 2000.