This full text paper was peer reviewed at the direction of IEEE Communications Society subject matter experts for publication in the IEEE INFOCOM 2007 proceedings.

Diagnosing Link-level Anomalies Using Passive Probes

Shipra Agrawal
Department of Computer Science
Stanford University, California, USA

K.V.M. Naidu, Rajeev Rastogi
Bell Laboratories, Lucent Technologies
Bangalore, India

***Abstract*—In this paper, we develop passive network tomography techniques for inferring link-level anomalies like excessive loss rates and delay from path-level measurements. Our approach involves placing a few passive monitoring devices on strategic links within the network, and then passively monitoring the performance of network paths that pass through those links. In order to keep the monitoring infrastructure and communication costs low, we focus on minimizing (1) the number of passive probe devices deployed, and (2) the set of monitored paths. For mesh topologies, we show that the above two minimization problems are NP-hard, and consequently, devise polynomial-time greedy algorithms that achieve a logarithmic approximation factor, which is the best possible for any algorithm. We also consider tree topologies typical of Enterprise networks, and show that while similar NP-hardness results hold, constant factor approximation algorithms are possible for such topologies.**

I. Introduction

Real-time services like voice over IP and IPTV are fueling the need for sophisticated monitoring tools capable of continuously tracking network performance parameters like end-to-end delay and loss rates. To meet the stringent requirements of voice and video services, these next-generation monitoring systems will not only need to quickly detect degradation in network performance, but they will also be required to isolate the root cause for service quality problems. Consequently, these monitoring tools, by enabling timely resolution of network problems (e.g., routing traffic around a congested link), will play a crucial role in both ensuring service quality as well as reducing service downtimes.

Recent years have witnessed a considerable amount of activity within the research community as well as industry to develop tools for monitoring network parameters (e.g., delay, packet loss). Existing tools can be divided into two categories. *Active monitoring* tools inject additional traffic into the network from key vantage points. For example, the *skitter* [1] tool measures the latency of a link as the difference in the round-trip times of two transmitted probe messages. In contrast, *passive monitoring* tools make inferences about network performance by snooping on the existing traffic traversing network links. In comparison to active monitoring, the passive approach has a number of benefits. First, it does not burden the underlying network infrastructure with additional synthetic traffic for making measurements. This can be especially critical when a network interface or link gets congested – during such times, injecting additional traffic for active measurements will only further exacerbate the situation. Second, since all measurements are based on actual traffic flowing within the network, passive mechanisms can accurately assess the network performance as experienced by end-users.

In this work, we study the problem of designing a *low-cost* passive monitoring system for identifying anomalous links with excessive delays or loss rates from path-level measurements. Our overall system consists of a few passive monitoring devices distributed at strategic links within the network. These devices passively monitor the performance of network paths that pass through them, and then use these to accurately pinpoint network anomalies. Our focus in this paper is on optimizing monitoring costs – this includes (1) the cost of installing and operating the passive monitoring infrastructure, and (2) the communication overhead of collecting performance statistics for monitored paths from the monitoring devices for analysis. To deal with (1), we look to deploy as few probes as possible at carefully chosen locations, while we address (2) by selecting a minimal set of paths whose monitoring enables anomalies to be effectively diagnosed.

A. Related Work

Network tomography involves estimating link-level parameters like delay and loss from end-to-end path-level measurements, and is an active area of research (see [2] for a comprehensive survey of the field). Tomographic techniques can be broadly classified into two categories: *continuous* and *boolean* [3]. Continuous tomographic methods use sophisticated statistical algorithms to solve an (under-constrained) linear system of equations that relate path measurements to individual link characteristics. For instance, [4] employs the least-squares method to compute delays for smaller path segments, while [5], [6] use the expectation-maximization (EM) algorithm to infer link-level loss rates from end-to-end multicast probe packet measurements. In [7], the authors use the Markov Chain Monte Carlo (MCMC) algorithm to identify lossy links in the network based on passive observations of traffic at a server. For overlay networks, [8] proposes an approach that selectively monitors a minimal set of linearly independent paths so that the behavior (loss rates and latency) of all paths can be inferred.

Our work is more closely related to boolean tomographic methods [9], [10], [11]. These assign *bad* labels to paths whose

0743-166X/07/$25.00 ©2007 IEEE

loss rates or delays exceed a certain threshold (the remaining paths are labeled *good*), and then use the labels to identify the *bad* links. [9] proposes a simple inference algorithm that attributes the bad paths to the smallest set of consistent bad links. In [10], the authors consider the problem of identifying links with high loss rates using passive measurements of end-to-end application traffic in a sensor network setting. They assume that some paths may be wrongly labeled as being good or bad, which complicates their link diagnosis problem. They propose a heuristic for computing the smallest set of lossy links that minimizes the number of incorrectly labeled paths. However, the authors do not address the problems of passive probe placement or the paths to monitor to accurately identify the problem links.

For a given set of paths in an active monitoring system, [11] proposes a polynomial-time algorithm for determining the smallest (independent) subset of paths that can distinguish identical anomalous-link sets as distinguished by the (bigger) input path set. A drawback of the approach of [11] is that it assumes that an arbitrarily large subset of links can simultaneously experience an anomaly – this can lead to many paths being monitored. In practice, however, due to traffic and path diversity [5], [8], it is very unlikely that a large number (e.g., > 3) of network links will get congested at the same time. In contrast, in our work, we put a bound on the number of concurrent anomalous links – thus, our approach will result in fewer paths being monitored compared to [11].

Recently, for an active monitoring setup, [12], [13], [14] proposed algorithms for computing a minimal set of nodes at which to locate monitoring stations such that their routing trees cover all the network links. Each station monitors the links in its routing tree by sending probe messages. In contrast, in our passive monitoring framework, a pair of monitoring devices located at the two end-points of a path are required to monitor the path. [15], [16] further enrich the power of active monitoring systems by allowing probe messages to be explicitly routed (e.g., using source routing), and propose schemes for computing a minimum cost set of probes to cover a given set of edges.

Finding optimal locations for monitoring devices has also been studied in the passive monitoring context [17], [18], [19]. Although, the goal in these works is to cover as many IP flows as possible, and maximize the flow sampling rate. In contrast, our focus is on diagnosing anomalies. To the best of our knowledge, ours is the first work to answer the questions of where to locate measurement points and which paths to monitor to facilitate the identification of problematic links in a passive monitoring setting.

B. Network Model

We model the network as a directed graph $G = (V, E)$, where the vertex set V includes routers and switches, and E is a set of bidirectional edges representing the communication links that connect the network nodes. We denote the bidirectional link between nodes u and v by $\{u, v\}$, and the two directed links comprising it by $\langle u, v\rangle$ and $\langle v, u\rangle$.

In this paper, we consider mesh and tree topologies for the network graph G since both can be found in practical settings. Wide Area Networks (WANs), like Service Provider networks, have general mesh topologies with multiple paths between nodes. However, in Enterprise environments, Ethernet frames are routed along the edges of a tree using the spanning tree protocol [20]. Although mesh topologies are more general, and provide redundancy, tree topologies are cost-effective, simpler to implement, and form an important subset of topologies employed for designing Enterprise and wireless sensor [10] networks. Hence, we carry out our passive monitoring study for both kinds of topologies.

All IP packets in a network originate and terminate at a subset of nodes from V, which we refer to as *edge* nodes. In a Service Provider network, these edge nodes are basically the edge routers that interface with customer networks or possibly other provider networks. In an Enterprise network, edge nodes can be either client hosts, or alternately, application servers like web servers or mail servers. The IP traffic between a pair of nodes traverses the network through a sequence of nodes and links which is dictated by the network topology and routing protocol. For instance, in a Service Provider network, the communication path between a pair of edge nodes can either be a preconfigured MPLS path or the shortest path between the nodes computed using the OSPF protocol. In the Enterprise case (tree topology), the communication path between an edge node pair is unique, and traces the edges of the spanning tree.

We will denote by P, the set of these paths (between edge nodes) in the network. Here each path $p \in P$ is a sequence of directed links that the path traverses. Note that P may not contain paths between every pair of edge nodes. This could happen in MPLS networks when there is no MPLS path provisioned between an edge node pair. Further, in OSPF or Enterprise networks, if there is inadequate traffic between a pair of edge nodes, then we do not include the path between them in P. This is because our passive monitoring approach relies on continuously observing IP packets traversing network paths to detect anomalies, and so for our approach to work, continuous traffic on the paths being monitored is crucial.

The path set P is dynamic, and is recomputed (e.g., by downloading routing tables) in the event of routing or topology changes. We assume that every link in E appears in at least one path in P. At some places in the text, for clarity of discussion, we will represent a path by the sequence of nodes it traverses. The direct mapping to the sequence of directed links between those nodes should be clear from the context. Also, we will frequently apply the set operators $\cap$ and $\cup$ to paths, considering each path to be a set (instead of a sequence) of links.

Now, the traffic on a communication link $\langle u, v\rangle$ corresponds to packets transmitted by u on its interface that is connected to v. Similarly, the traffic on $\langle v, u\rangle$ is due to packet transmissions on an interface of v. In most commercially available routers, packets get delayed or dropped primarily due to queuing at the transmitting or outbound interfaces [20]. Thus, for a (directed) communication link $\langle u, v\rangle$, anomalies like excessive delays or

losses can essentially be traced back to queuing-related delays or losses at u's outbound interface leading to v. Hence, there is a one-to-one correspondence between a link $\langle u, v\rangle \in E$ in the network graph and the (transmitting) interface on node u that is connected to node v; the same holds for $\langle v, u\rangle$.

In the following subsection, we will describe how passive monitoring can be used to identify paths with excessive delays or losses, and subsequently use the path-level information to pinpoint the miscreant links/interfaces.

C. Passive monitoring infrastructure

Our passive monitoring solution consists of a set of passive monitoring devices placed at various points in the network where they passively analyze the traffic to detect and diagnose link-level anomalies.

Passive monitoring devices. A variety of technologies are available for non-intrusively observing network traffic on the links or the link-interfaces (ports) of switches. *Port mirroring* is a method that forwards a copy of each incoming and outgoing packet from one port of a network switch to another port that is connected to a monitoring device. Most of the commercial vendors of network switches, e.g., Cisco, support this capability in their devices. *Network taps* are hardware devices that hook directly onto the network cable and send a copy of the (forward and reverse) traffic that passes through them to one or more other networked devices. For example, Net Optics [21] provides passive taps for real-time monitoring of full-duplex Gigabit copper and fiber links. Our passive monitoring devices consist of these network taps placed on network links, and connected to an off-the-shelf PC or server either directly through regular *network interface cards* (NIC) or via high performance cards like the DAG cards from Endace [22] specially engineered for this purpose. We use the terms "monitors", "taps" and "probes" interchangeably in the text to refer to these passive monitoring devices.

Detecting anomalies using passive probes. The packets observed by passive taps on a path p can be used to detect excessive packet losses or delays for links in path p. Let us first consider the case of packet loss. Let s and t be the two passive taps located at the two ends of path p that monitor p's packets (s is on a link just before p begins and t is on the last link of p). At regular time intervals (e.g., 1, 10 or 30 seconds), both s and t send to a central Network Operations Center (NOC), the number of packets on path p seen at the probes in the most recent time interval[1]. Now, if the difference between the packet counts at s and t is large and exceeds a certain pre-specified threshold (after accounting for packets that may be in transit between s and t), then we can conclude at the NOC that excessive packets are being lost along some links on p. Alternately, probes s and t can ship to the NOC samples of the observed packets on path p, and an inference of excessive loss for p can again be made if there is a large discrepancy in the samples from the two probes. Along similar lines, by associating timestamps with packets, it is possible to detect excessive delays for p by keeping track at the NOC, the difference between packet timestamps averaged over an interval or for sample packets at s and t.

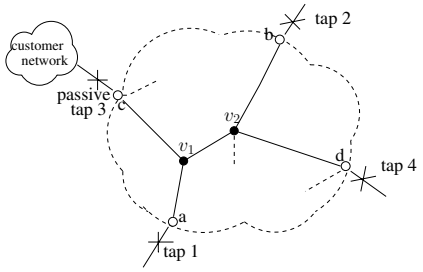

Fig. 1. Example Service Provider network

Based on the above discussion, it follows that we can detect link-level delay and loss anomalies by monitoring paths such that every link belongs to at least one monitored path. This way, if a link were to experience excessive delay or loss, then for monitored paths containing the link, delay or loss thresholds would be violated, and the link-level anomaly would be detected. In a Service Provider network, we place passive taps on links connecting the provider to customer networks. This enables all paths in P between the edge nodes to be monitored, and thus to ensure that every network link indeed belongs to some monitored path. Of course, in order to minimize communication overhead between the probes and the NOC, we should monitor as few paths as possible (while ensuring that all links are covered by the monitored paths). In an Enterprise setting, probes can be placed either on the client and server machines (software agents that snoop on IP packets sent and received), or on the links connecting them to the network.

Figure 1 depicts an example Service Provider network. Nodes a, b, c and d are edge nodes which are connected to customer networks. Suppose that there are 3 bidirectional communication paths in P: path $p_1 = \langle a, v_1, v_2, b\rangle$ between nodes a and b, path $p_2 = \langle c, v_1, v_2, d\rangle$ between c and d, and path $p_3 = \langle a, v_1, v_2, d\rangle$ between a and d. Passive taps are placed on the interfaces of the edge nodes a, b, c and d to the customer networks so that the traffic on the above 3 paths can be passively monitored[2]. However, we can detect link-level anomalies by only monitoring the 2 paths p_1 and p_2, since they cover all the network links. Thus, for instance, if link $\langle v_2, b\rangle$ started dropping excessive packets, then passive taps 1 and 2 would report excessive losses for path p_1, and the link-level anomaly would be detected.

Diagnosing anomalies using passive probes. In the example of Figure 1, while paths p_1 and p_2 suffice to detect link-level anomalies, monitoring them alone may not enable us to determine the identity of the problem link. This is because if,

[1]We assume that the clocks on the passive probes are synchronized using the NTP protocol or a GPS device. For details on accurate measurement of network delay in the presence of clock skew, refer to [23], [24].

[2]For simplicity, we assume that edge node interfaces to customers are well-behaved and do not exhibit anomalies.

say, only path p_1 reports an anomaly, then either of the links $\langle a, v_1\rangle$ or $\langle v_2, b\rangle$ could be the cause of the anomaly. (Note that we can eliminate $\langle v_1, v_2\rangle$ from the list of suspects since p_2 did not report an anomaly.)

Let Q be the subset of paths being monitored by our passive probes. Below, we identify the property that the set Q must satisfy in order to be able to uniquely identify the culprit link in the event that an anomaly is detected. We will initially assume that *the network anomaly is caused by a single link.* We relax this assumption later, and discuss extensions to our monitoring system design schemes to handle multiple (up to k) link anomalies in Section V. Although we must point out here that the likelihood of multiple concurrent anomalous links is low given previous findings of packet loss independence among links due to traffic and path diversity [5], [8].

In our initial development, we will also assume that *a path reports an anomaly if and only if it contains an anomalous link*. This has been referred to as *separable* link performance in [9], and holds if the normal and anomalous link states are well separated. Examples include link failures, delay spikes in links, and loss rate of a wireless link in sensor networks (which is either small or high, but rarely in between [10]). Later, in Section V, we will discuss the ramifications of relaxing this assumption on our anomaly diagnosis procedure.

Now suppose that $Q_1 \subseteq Q$ is the set of paths reporting an anomaly, and $Q_2 = Q - Q_1$ is the set of paths for which there is no anomaly. Then the *single* anomalous link must belong to the set $R = (\cap_{p\in Q_1} p) - (\cup_{p\in Q_2} p)$. Clearly, to be able to uniquely pinpoint the culprit link, R must contain exactly the link. The following theorem gives a necessary and sufficient condition that Q must satisfy to accurately isolate the problem link.

Theorem 1.1 (Anomaly Diagnosis Condition): A set of monitored paths Q is sufficient to diagnose the anomalous link iff for every pair of links (e_1, e_2) in E, there is at least one monitored path in Q that contains exactly one of the two links (that is, a path that *distinguishes* between the links e_1 and e_2). □

Clearly, the above condition is necessary. To see this, suppose that Q does not satisfy the condition, that is, there exists a pair of links (e_1, e_2) in E which is not distinguished by any path in P. This means that every path in Q either contains both e_1 and e_2, or none of them. Now, if one of e_1 or e_2 were to experience an anomaly, then R would contain both e_1 and e_2, and we would not be able to differentiate between the two. The above condition is also sufficient to ensure that R contains only the culprit link. This is because the condition ensures that in case of an anomaly on one link, every other link either belongs to a path in Q_2 (paths with no anomaly) or is absent from some path in Q_1 (paths with anomaly) - thus, the link cannot appear in R.

Let us revisit the example in Figure 1. The path set $Q = \{p_1, p_2\}$ is not sufficient for anomaly diagnosis because the paths in Q do not distinguish between the links in pairs $\{\langle a, v_1\rangle, \langle v_2, b\rangle\}$ and $\{\langle c, v_1\rangle, \langle v_2, d\rangle\}$. Paths p_1 and p_2 either contain both or none of the links in each of the pairs. Thus Q does not satisfy the anomaly diagnosis condition. Now consider the path $p_3 = \langle a, v_1, v_2, d\rangle$. Path p_3 contains exactly one of the two links in each of the above pairs. Thus on adding p_3, the path set $Q = \{p_1, p_2, p_3\}$ distinguishes between all the link pairs and is sufficient for anomaly diagnosis.

Now suppose that there is no communication over the path p_3 in the network, and the only communication paths available between edge nodes are $P = \{p_1, p_2\}$. Thus the original set of paths P does not satisfy the anomaly diagnosis condition and monitoring P or any subset of P will leave some link pairs undistinguished. One way to fix this is to place an additional tap on a link in the path segment between the undistinguished link pairs, and break each path into two new paths. The new path segments contain exactly one of the two links in each pair, and thus all the link pairs are distinguished. For instance, placing an additional passive tap on link $\{v_1, v_2\}$ segments the path p_1 into two path segments $p_3 = \langle a, v_1, v_2\rangle$ and $p_4 = \langle v_2, b\rangle$, and p_2 into two path segments $p_5 = \langle c, v_1, v_2\rangle$ and $p_6 = \langle v_2, d\rangle$. New paths p_3 and p_5 contain exactly one of the links in the pairs $\{\langle a, v_1\rangle, \langle v_2, b\rangle\}$ and $\{\langle c, v_1\rangle, \langle v_2, d\rangle\}$, respectively. Thus the new path set $\{p_1, \ldots, p_6\}$ is sufficient to diagnose the anomalous link. Now, selecting the monitored path set as subset $Q = \{p_1, p_2, p_3, p_5\}$ or $Q = \{p_3, p_4, p_5, p_6\}$ satisfies the anomaly diagnosis condition.

D. Problem Formulation

We build our passive monitoring infrastructure in two phases. As described previously, passive probes at edge node interfaces are required to ensure that all paths in P are monitored, and all link-level anomalies are detected. However, this may not be enough – to diagnose anomalies, we may need to install additional probes in the network core. Therefore, in the *first phase*, we choose a minimal set of links (in the core network) at which to place additional passive probes so that we can accurately diagnose all single-link anomalies. Our goal in this phase is to minimize the installation and operating costs of the probing infrastructure to the largest possible extent. In the *second phase*, we determine a minimal set of paths to monitor in order to detect and diagnose all single-link anomalies. Since the probes need to continuously transmit traffic statistics (e.g., packet counts) and samples to the NOC for each monitored path, communication costs depend directly on the number of monitored paths. Thus, by minimizing the total number of paths, we seek to reduce the impact of our monitoring system on the underlying network.

Below, we formally define the problems of optimal selection of probe locations and monitored path set so that we are able to perform the required detection and diagnosis while incurring minimal monitoring infrastructure costs and communication overhead.

Probe Placement problem. As we saw earlier, placing a probe on some edge e can distinguish a link pair if the edge occurs on some path $p \in P$ in the path segment between the link pair. Such a probe divides p into two segments such that each segment contains exactly one of the two links in the pair, thus

satisfying the anomaly diagnosis condition. The problem of selecting probe locations can thus be stated as follows:

Problem statement 1 *(Probe Placement):* Given a directed graph $G = (V, E)$, set of paths P, and set of directed link pairs L that cannot be distinguished by paths in P, select the smallest subset of (bidirectional) edges F on which to place probes so that every link pair in L is distinguished by some edge in F. □

Note that, above, although we consider every edge in the network as a candidate for the installation of probe equipment, our techniques can be easily adapted to scenarios where only certain links are eligible for placement of probes. Thus, we can readily handle instances where it may not be feasible to install probes on very high-speed links in the core of the network or links in a different Service Provider's network.

Path Monitoring problems. At the end of the first phase, the probes placed on links in F enable additional path segments to be monitored. For example, consider the path p_1 in the network of Figure 1. With a probe on link $\{v_1, v_2\}$, we can additionally monitor the sub-paths $\langle a, v_1, v_2\rangle$ and $\langle v_2, b\rangle$ of path p_1. Let P' denote the set of paths in P plus these additional subpaths.

In the second phase, we select the minimum subset of paths from P' to be monitored for (a) detection and (b) diagnosis of problematic links. We denote these path sets by Q_{det} and Q_{diag}, respectively. The key idea here is that Q_{det} will be much smaller than Q_{diag}, and so during normal operation, we will only monitor the few paths in Q_{det}. Only in the rare event that we detect a network anomaly, will we shift to monitoring the full set of paths in Q_{diag} to identify the faulty link.

To be able to detect excessive link loss or delay, the subset of paths Q_{det} must cover all the links. Thus, the problem of Path Monitoring for Detection can be stated as follows:

Problem statement 2 *(Path Monitoring for Detection):* Given a directed graph $G = (V, E)$, and a set of paths P' that can be monitored, select the minimum subset of paths $Q_{det} \subseteq P'$ such that every link in E belongs to some path in P'. □

To accurately diagnose the problematic interface, the set of monitored paths Q_{diag} must satisfy the anomaly diagnosis condition, and so the problem of Path Monitoring for Diagnosis can be stated as follows:

Problem statement 3 *(Path Monitoring for Diagnosis):* Given a directed graph $G = (V, E)$, and a set of paths P' that can be monitored, select the minimum subset of paths $Q_{diag} \subseteq P'$ that satisfies the anomaly diagnosis condition (see Theorem 1.1). □

Discussion. Observe that the probe locations and monitored paths computed above are very much dependent on the existing path set P, which is dynamic and may change due to temporary events like link failures, or permanent events like addition/deletion of network links, adjustment to link weight values (e.g., for OSPF) or provisioning of new MPLS LSPs. In response to short-lived or transient changes in topology, we simply recompute the paths to monitor Q_{det} and Q_{diag} for the new topology, and instruct the passive probes to start monitoring these new paths. This can be completely automated, and thus, accomplished without incurring much overhead. Only occasionally, when longer term changes to the network topology are made, do we need to consider the more cumbersome task of relocating (the few) probe devices in the network core as well.

E. Our Contributions

For the three problems (Probe Placement, Path Monitoring for Detection, and Path Monitoring for Diagnosis) described in the previous section, we consider both mesh (general graph) and tree network topologies, and make the following contributions in this paper:

• **Algorithms for Probe Placement.** We show that the problem of Probe Placement is NP-hard for both mesh and tree network topologies. The problem for meshes maps naturally to the Set Cover problem, and thus a polynomial-time greedy algorithm gives a solution that is within a logarithmic factor of the optimal. This correspondence to the Set Cover problem also shows that it is hard to get better approximations for general mesh topologies. When the network has a tree topology, we show that a 3-factor approximation is possible.

• **Algorithms for Path Monitoring for Detection.** We show that the problem is NP-hard for both mesh and tree topologies. We develop a logarithmic factor approximation algorithm and show that it is hard to get better approximations for general mesh topologies. For *tree topologies*, we present a 2-approximate solution.

• **Algorithms for Path Monitoring for Diagnosis.** We present a logarithmic factor approximation algorithm for communication networks with general mesh topologies. We also devise a constant factor approximation algorithm for anomaly-diagnosis in *tree topologies*.

II. Probe Placement

In this section, we present schemes for solving the Probe Placement problem for meshes and trees. Recall from Problem Statement 1 (see Section I-D) that the problem is to compute the smallest subset of edges on which to place probes so that every directed link pair in set L is distinguished.

A. Probe Placement for mesh topologies

The Probe Placement problem for mesh topologies is NP-hard. In [25], we show a reduction from the Set Cover problem. The reduction from Set Cover also proves that it is hard to obtain an approximation algorithm for the Probe Placement problem with approximation ratio lower than $\Omega(\log |L| + |E|)$ (using the hardness result from [26] for Set Cover).

We can, however, obtain a *logarithmic factor approximation algorithm* as follows. The main idea is to reduce a Probe Placement problem instance to an instance of Set Cover by mapping each undistinguished directed link pair in L to an element, and each edge (potential probe location) to a subset of link pairs in L that it can distinguish. The problem of

selecting the minimum number of probe locations thus reduces to one of selecting the minimum number of subsets (edges) to cover all the elements (undistinguished link pairs). The greedy algorithm for Set Cover [26] thus yields a logarithmic factor approximation algorithm. In each step, it selects the edge that distinguishes the maximum number of (still undistinguished) link pairs in L, until all the pairs are distinguished.

Note that the above algorithm can be easily modified to handle the case when possible probe locations are restricted to some (and not all) of the edges. We simply consider subsets of link pairs for only the permitted (and not all) edges.

B. Probe Placement for tree topologies

Next, we shift our attention to scenarios where the graph G is constrained to be a tree T and the paths in P are between leaves of the tree. The Probe Placement problem for tree topologies turns out to be NP-hard as well. We show a reduction from Vertex Cover in [25]. In the following, we present a 3-factor approximation algorithm for the problem.

Lazy Probe Placement algorithm. We know from earlier discussion that a link pair can be distinguished by placing a probe anywhere on the path segment between the pair. Thus, in a tree with paths between the leaves, we can move bottom-up without distinguishing a pair until we reach the closest node to the root that is also on a path segment between the pair. Observe that the pair cannot be distinguished further up in the tree. We say that the link pair becomes *ripe* at that point. Our Lazy Placement algorithm delays selecting an edge (probe location) to distinguish a pair until it becomes ripe.

We start with an outline of the algorithm. Essentially, an arbitrary node in T is chosen as root, and the algorithm then proceeds bottom-up in the tree. For each node v, the algorithm iterates over the set of undistinguished link pairs L, and identifies the ripe pairs. Then, it selects some child edge(s) of v to distinguish these ripe pairs. Here the *child edges* of a node v are the edges connecting v to its direct children $child(v) = \{c_1, c_2, \ldots, c_m\}$. At each step, all the link pairs distinguished by the selected edges are removed from the set L.

Now, we give an exact categorization of the ripe pairs and illustrate the choice of child edge(s) made for each of these categories. Observe that all the ripe pairs for a node v, i.e. the pairs in L which cannot be distinguished by probes on edges outside the subtree rooted at v, can be categorized based on the following 4 cases (see Figure 2):

• **case 1:** one link e_1 is in the subtree rooted at c_j (directed downward), for some $c_j \in child(v)$, and the other link e_2 is $\langle v, c_j \rangle$.

• **case 2:** one link e_1 is in the subtree rooted at a child node c_j (directed upward) or $\langle c_j, v \rangle$, for some $c_j \in child(v)$, and the other link e_2 is on the edge connecting v to its parent (directed upward).

• **case 3:** one link e_1 is in the subtree rooted at a child node c_j (directed upward) or $\langle c_j, v \rangle$, and the other link e_2 is $\langle v, c_k \rangle$ for a different child node c_k.

• **case 4:** one link e_1 is in the subtree rooted at c_j (directed upward) or $\langle c_j, v \rangle$, and the other link e_2 is in the subtree rooted at c_k (directed downward), for some $c_j, c_k \in child(v)$.

Now, in cases 1, 2, and 3, the child edge $\{v, c_j\}$ is the only possible choice of probe location that is capable of distinguishing the link pair (e_1, e_2)[3]. So, if any undistinguished link pair in L satisfies case 1, 2, or 3, we add the child edge $\{v, c_j\}$ to the solution. But in case 4, either of the two child edges $\{v, c_j\}$ or $\{v, c_k\}$ can distinguish the pair. The problem of selecting the minimum number of child edges to distinguish the case 4 pairs can be reduced to an instance of the Vertex Cover problem $\bar{G} = (\bar{V}, \bar{E})$ by (1) mapping each child edge $\{v, c_j\}$ to a vertex in $\bar{V}$, and (2) adding an edge to $\bar{E}$ between two vertices corresponding to child edges $\{v, c_j\}$ and $\{v, c_k\}$ if either of them can distinguish a case 4 link pair. We use the (well-known) 2-approximation algorithm for Vertex Cover[4] to find a subset of child edges that distinguish all the case 4 link pairs, and add them to the solution. Further, in this case, we include one more edge in the solution - the edge $\{v, p(v)\}$ connecting the node v to its parent $p(v)$. The addition of this edge is required to ensure a 3-approximation, as explained later.

Algorithm 1 contains a description of the pseudo-code for our algorithm. It iterates over nodes in the tree in a bottom-up order $v_1, \ldots, v_{|V|}$, where $|V|$ is the number of nodes in the tree.

Analysis of the algorithm. It is easy to see that our algorithm is correct, that is, after all nodes are processed, the edges in F are sufficient to distinguish all the link pairs in L. This is because our algorithm maintains the invariant that at the end

Algorithm 1 Lazy Probe Placement algorithm

Input: L, the set of undistinguished link pairs.
Output: F, edges on which to place probes.
Initialize $F = \emptyset$, and $L' = L$.
for $i = 1$ to $|V|$ **do**
  Let $S \subseteq L'$ be the set of ripe (case 1, 2, 3, or 4) pairs for node v_i among the pairs in L'.
  Let $C = \{\{v_i, c_j\}\colon c_j \in child(v_i)\}$ be the set of child edges of v_i.
  Let $C' \subseteq C$ be the edges that distinguish case 1, 2, and 3 pairs in S.
  Delete edges in C' from C, and add them to F.
  Remove the pairs distinguished by C' from S and L'.
  if (S is not empty) **then**
    Use the 2-approximation algorithm for Vertex Cover to select edges $C'' \subseteq C$ to distinguish the case 4 pairs in S.
    Add the parent edge $\{v_i, p(v_i)\}$ to C''.
    Add the selected edges in C'' to F.
    Remove the link pairs distinguished by C'' from L'.
  end if
end for
Return F.

[3]Recall that there is a one-to-one correspondence between link $\langle u, v \rangle$ and u's interface connected to v.

[4]The 2-factor approximation algorithm for Vertex Cover repeatedly adds the (two) end-points of an arbitrary edge to the solution, and then removes them (and all incident edges) from the graph.

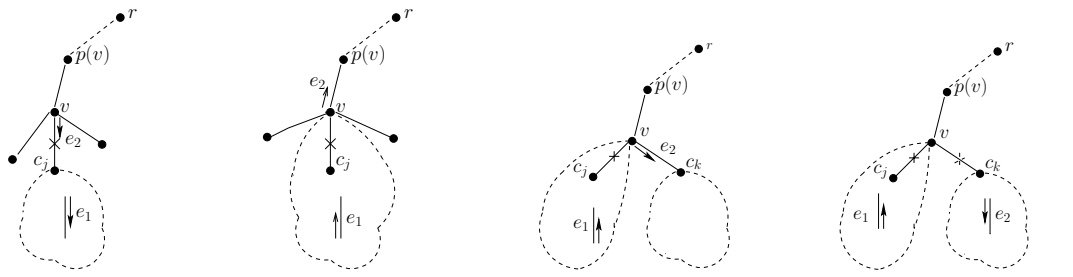

Fig. 2.

of the i^{th} iteration, all link pairs that are ripe for $v_1, \ldots, v_i$ are distinguished by F.

Further, it can be shown that the algorithm has a time complexity of $O(|V| \cdot |L|)$ – thus, it runs in polynomial time. And as stated below, the solution F computed by the algorithm is near-optimal.

Theorem 2.1: Our Lazy Placement algorithm returns a 3-approximate solution, i.e., the size of F is within a factor of 3 of the optimal solution. □

A detailed proof of the above can be found in [25]. The intuition behind the proof is that in each iteration of our algorithm, we make optimal choices for case 1, 2, and 3 ripe pairs. But for case 4 pairs, we select edges using a 2-approximate solution for Vertex Cover plus 1 extra edge to the node's parent, thus yielding a 3-factor approximation of the optimal. Here, selecting the extra edge is necessary to preserve the following invariant: at the end of each iteration i of our algorithm, the remaining undistinguished pairs L' in our solution is always a subset of the pairs in L still remaining to be distinguished by any optimal solution (considering only the edges in subtrees rooted at $v_1, \ldots, v_i$).

III. Path Monitoring for Anomaly Detection

In this section, we investigate the Path Monitoring for Detection problem, which involves selecting the minimum subset of paths Q_{det} from P' such that all the directed links in E are covered. We will refer to this as the Path Cover problem.

A. Path Cover for mesh topologies

The Path Cover problem is NP-hard for mesh topologies. This can be shown using a reduction from the Set Cover problem (see [25] for details). Thus, the lower bound of $\Omega(\log |P'| + |E|)$ applies to the hardness of approximating the Path Cover problem. As before, we can employ the greedy algorithm for Set Cover [26] to get a *logarithmic factor approximation* algorithm for selecting a subset of paths (sets) to cover all the directed links (elements) in the network graph.

B. Path Cover for tree topologies

Unfortunately, restricting the network topology to be a tree T does not make the problem of selecting a minimum path set for anomaly detection any easier. The Path Cover problem for tree topologies is also NP-hard. To prove this, we show a reduction from the following problem in [25], which is known to be NP-hard [27]:

Tree Bridge Connectivity Augmentation: Given an undirected graph $\bar{G} = (\bar{V}, \bar{E})$ and a spanning tree $T_s = (\bar{V}, E_s)$ of $\bar{G}$ with $2B$ leaves, is there a set $A \subseteq \bar{E} - E_s$ of edges such that $|A| = B$ and $(\bar{V}, E_s \bigcup A)$ is 2-edge-connected? □

Luckily, a 2-*factor approximation* to the optimal number of paths needed to cover links is possible for tree topologies. Our approximation algorithm essentially starts by choosing an arbitrary node in T as the root. Then, for each leaf node, it adds to Q_{det} the following two paths from P': the outgoing path that reaches closest to the root, and the incoming path that reaches closest to the root. Note that Q_{det} can be computed in time linear in the size of P'.

We claim that the set of paths Q_{det} thus chosen covers all the links in the tree. To see this, note that if a link is covered by some path $p \in P'$, then for one of the leaf nodes serving as an end-point of p, the link will be on the path segment of p that lies on the path between the leaf node and the root; thus the link will be covered by the closest path to the root from that leaf node. Thus, all the links in the tree will be covered by the selected paths.

Next, we show that the selected paths are very close to the optimal.

Theorem 3.1: The computed path set Q_{det} is a 2-approximation to the optimal Path Cover.

Proof: Let n_1 be the number of leaf nodes in the tree. Then, our scheme selects at most $2n_1$ paths, one outgoing and one incoming for each leaf. Now, any path in the tree can cover at most two directed edges from those incident on the leaf nodes. These edges are - the outgoing edge at the leaf node at which the path starts and the incoming edge at the leaf node at which the path terminates. Thus, at least n_1 paths are required to cover the $2n_1$ directed links incident on the leaf nodes. Our solution has $2n_1$ paths, and so it is a 2-approximation. ■

IV. Path Monitoring for Anomaly Diagnosis

Once an anomaly is detected, we need to identify the responsible link. In this section, we propose approximation algorithms for computing a minimal subset of paths Q_{diag} from P' to diagnose single-link anomalies. Recall that the anomaly diagnosis condition (see Theorem 1.1) requires that for every pair of edges, there is a path in Q_{diag} that contains exactly one of them, and can thus distinguish between them.

Consider the undirected version of the Path Monitoring for Diagnosis problem in which each edge in E is undirected (as opposed to bidirectional). We observe that the undirected version of our problem on star graphs is equivalent to the Minimum Test Collection problem with test size at most 2, which is known to be NP-hard [28]. Based on this, we conjecture that the Path Monitoring problem for directed graphs (including trees) is also NP-hard[5], and so we go on to devise approximation algorithms.

Logarithmic factor approximation algorithm for mesh topologies. We can reduce the Path Monitoring problem to the Set Cover problem by mapping each link pair to an element and each path in P' to the set of link pairs it distinguishes; a path distinguishes all the links in it from the links it does not contain. For example, a path $p = \langle e_1, e_2 \rangle \in P'$, where $e_1, e_2 \in E$ can be reduced to the set $\{(e_1, e_j) | e_j \in E, e_j \notin p\} \cup \{(e_2, e_j) | e_j \in E, e_j \notin p\}$. Also note that these are the only link pairs that the path can distinguish. The greedy algorithm for Set Cover [26] thus yields a logarithmic factor approximation algorithm to compute a set of paths that distinguishes all the link pairs.

Constant factor approximation algorithm for tree topologies. For a tree $T = (V, E)$, our approximation algorithm for computing path set $Q_{diag} \subseteq P'$ that satisfies the anomaly diagnosis condition is as follows.

1. Find the Path Cover Q_{det} for T using the 2-approximation algorithm given in Section III-B. Set $Q_{diag} = Q_{det}$.
2. For each edge $e = \langle u_e, v_e \rangle$, fix a path p_e in the Path Cover that covers this edge. Also denote by s_e and t_e, the end-points of p_e; further, let s_e be the end-point closer to u_e. Now each edge $e = \langle u_e, v_e \rangle$ divides the path p_e into at most three segments: $\langle s_e, u_e \rangle$, $\langle u_e, v_e \rangle$ and $\langle v_e, t_e \rangle$. Among all the paths in P' that pass through e and deviate from p_e (or terminate) in the segment $\langle s_e, u_e \rangle$, choose the one that deviates at a vertex closest to u_e. Call this path $p_{s,e}$. Similarly, choose $p_{t,e}$. Add the chosen paths to Q_{diag}.

The time complexity of the algorithm is $O(|P'| \cdot |E|)$. Further, we can show that the algorithm is correct, that is, every pair of edges is distinguished by some path in Q_{diag}. Consider any pair of edges e and f. If both p_e and p_f pass through both e and f, then we need to show that there is another path in Q_{diag} that distinguishes e and f. The other cases are trivial as one of the paths p_e or p_f will do the job. Assume w.l.o.g. that $\langle v_e, \ldots, u_f \rangle$ is the segment of p_e (or equivalently p_f) lying between e and f. If all the paths in P' that pass through e also pass through f, then there is no way of differentiating the two links. So there must be at least one path in P' that passes through e (or f, but w.l.o.g we can assume it is e) but not through f. Call this path q. Now this path must deviate at some vertex belonging to $\langle v_e, \ldots, u_f \rangle$. This implies that $p_{t,e}$ chosen by the algorithm will differentiate e and f since it deviates at least as early as q. Hence, Q_{diag} distinguishes between all the link pairs.

Finally, we prove that Q_{diag} is very close to the optimal.

Theorem 4.1: The computed path set Q_{diag} is an 8-approximation of the optimal.

Proof: Let OPT denote the optimal solution, and n be the number of vertices in tree T. We show that $|Q_{diag}|$ is at most a constant times n, and $|OPT|$ is at least a constant fraction of n.

Recall from the previous section that the Path Cover Q_{det} contains $2n_1$ paths, where n_1 is the number of leaf nodes in T. Now, for each directed edge e in T, we added at most 2 additional paths ($p_{s,e}$ and $p_{t,e}$, as discussed above) to Q_{diag}. Since the number of directed edges in T is at most $2(n-1)$, we get that the number of paths added to Q_{diag} is at most $4(n-1)$. But, note that for each directed edge incident on a leaf node, only one of $p_{s,e}$ or $p_{t,e}$ exists. Thus, the total number of paths added cannot be more than $4(n-1) - 2n_1$. Hence, after including the paths in Q_{det}, we get that $|Q_{diag}| \leq 4(n-1)$.

Next, we will show that $|OPT| \geq \frac{n+2}{2}$. Let n_1 be the number of degree-1 vertices (essentially the leaf nodes), n_2 be the number of degree-2 vertices, and n_3 be the number of remaining vertices in T[6] . Note that for each leaf node, there should be at least two paths in OPT with the leaf as an end-point, one to cover the outgoing edge and the other to cover the incoming edge. Also, consider a degree-2 vertex, which is essentially an intermediate node in the tree. To distinguish the two directed link pairs passing through it, each such vertex must have at least two paths in OPT that start or end at the vertex. Since $|OPT|$ paths have $2|OPT|$ end-points, from the above arguments, we get that $2|OPT| \geq 2(n_1 + n_2)$. Now, the sum of the degrees in a tree with n vertices is $2(n-1)$ which is at least $n_1 + 2n_2 + 3n_3$. But $n_3 = n - n_1 - n_2$, which gives $n + 2 \leq 2n_1 + n_2 \leq 2(n_1 + n_2) \leq 2|OPT|$. Hence, $|OPT| \geq \frac{n+2}{2}$.

Putting all of the above together, we get $\frac{n+2}{2} \leq |OPT| \leq |Q_{diag}| \leq 4n - 4$, which proves the theorem. ■

V. Extensions

The anomaly diagnosis schemes presented in the previous sections assumed only a single link anomaly and separable link performance (that is, a path reports an anomaly iff it contains an anomalous link). In this section, we discuss the implications of relaxing these 2 assumptions.

A. Multiple Link Anomalies

Our goal is to be able to uniquely pinpoint up to k concurrent link anomalies for a small constant k (e.g., $k \leq 3$). We can achieve this by extending the anomaly diagnosis condition (see Theorem 1.1) as follows.

Theorem 5.1: (*Anomaly Diagnosis Condition – Multiple Links)* A set of monitored paths Q is sufficient to diagnose up to k anomalous links iff for every pair of link subsets (S_1, S_2) of size at most k (that is, $S_i \subseteq E$ and $|S_i| \leq k$), there is at least one monitored path in Q whose intersection with exactly

[5]The reduction for proving NP-hardness for the directed version is still an open problem.

[6]Here, by degree-k vertex, we mean a vertex with k bidirectional edges incident on it.

one of the two link subsets is non-empty (that is, a path that *distinguishes* between the subsets S_1 and S_2). □

Now, if $Q_1 \subseteq Q$ is the set of paths that report an anomaly, and $Q_2 = Q - Q_1$ is the set of paths for which there is no anomaly, then the set R of up to k anomalous links is the one satisfying: (1) $R \cap p \neq \emptyset$, for all $p \in Q_1$, and (2) $R \cap p = \emptyset$, for all $p \in Q_2$. A straightforward scheme to find the culprit link set R is to simply enumerate all link subsets of size less than or equal to k (after eliminating the links that appear in paths in Q_2), and return the one that satisfies (1).

To select the subset $Q_{diag} \subseteq P'$ of paths to monitor for diagnosing multiple link anomalies, we can extend the Path Monitoring algorithm based on Set Cover for mesh topologies that was described in Section IV. Basically, the link subset pairs (S_1, S_2) correspond to elements and each path p in P' corresponds to the set of all subset pairs (S_1, S_2) that it distinguishes – these are the pairs such that p intersects with exactly one of S_1 or S_2. Thus, the greedy algorithm for Set Cover can be used to compute a logarithmic factor approximation to the minimal set of paths that distinguish all the subset pairs.

B. Non-separable Link Performance

In case the normal and anomalous link states are not well-separated, it is possible for there to be *false positives*, that is, a path containing no anomalous links may still report an anomaly. This could happen, for instance, if we set low threshold values for paths. Although, with low thresholds, we can be assured that paths that do not report an anomaly contain no anomalous links, that is, there are no *false negatives*.

In the presence of false positives, the anomaly diagnosis condition of Theorem 5.1 is no longer sufficient to uniquely identify the anomalous links. Here, a simple heuristic to identify the most likely culprits is to enumerate all link subsets (with size $\leq k$) that do not overlap with non-anomalous paths in Q_2 and return the one that intersects with the maximum number of anomalous paths in Q_1. One of our goals for future work is to formulate an appropriate anomaly diagnosis condition for uniquely diagnosing anomalies in the presence of both false positives and negatives.

VI. Summary

In this paper, we proposed techniques for building a low-cost *passive monitoring* infrastructure for diagnosing link-level anomalies from path level-measurements. In order to keep the probe device and network communication costs low, we looked at minimizing (a) the number of passive probes deployed within the network, and (b) the set of paths monitored by the probes. Unfortunately, finding optimal solutions for both of the above problems is NP-hard. Consequently, we developed polynomial-time approximation algorithms for the probe placement and path monitoring problems that achieve close to the best possible approximations for both mesh and tree network topologies. We are working on extending our model and techniques to diagnose multiple-link anomalies when link performance is not separable (that is, there may be false positives or negatives).

References

[1] "CAIDA cooporative association for internet data analysis." [Online]. Available: http://www.caida.org/tools/

[2] R. Castro, M. Coates, G. Liang, R. Nowak, and B. Yu, "Network tomography: Recent developments," *Statistical science*, 2004.

[3] H. X. Nguyen and P. Thiran, "Binary versus analogue path monitoring in IP networks," in *Lecture Notes in Computer Science, Volume 3431*, January 2005, p. 97.

[4] Y. Shavitt, X. Sun, A. Wool, and B. Yener, "Computing the unmeasured: An algebraic approach to internet mapping," in *IEEE INFOCOM*, 2001.

[5] R. Caceres, N. Duffield, J. Horowitz, and D. Towsley, "Multicast-based inference of network internal loss characteristics," *IEEE Transactions on Information Theory*, vol. 45, 1999.

[6] T. Bu, N. Duffield, F. L. Presti, and D. Towsley, "Network tomography on general topologies," in *SIGMETRICS*, 2002.

[7] V. N. Padmanabhan, L. Qiu, and H. J. Wang, "Server-based inference of internet performance," in *IEEE INFOCOM*, 2003.

[8] Y. Chen, D. Bindel, H. Song, and R. H. Katz, "An algebraic approach to practical and scaleable overlay network monitoring," in *SIGCOMM*, 2004.

[9] N. G. Duffield, "Simple network performance tomography," in *IMC*, 2003.

[10] H. Nguyen and P. Thiran, "Using end-to-end data to infer lossy links in sensor networks," in *IEEE INFOCOM*, Barcelona, April 2006.

[11] ——, "Active measurement for multiple link failures: Diagnosis in IP networks," in *PAM 2004*, April 2004.

[12] J. D. Horton and A. López-Ortiz, "On the number of distributed measurement points for network tomography," in *IMC*, 2003.

[13] Y. Bejerano and R. Rastogi, "Robust monitoring of link delays and faults in IP networks." in *IEEE INFOCOM*, 2003.

[14] R. Kumar and J. Kaur, "Efficient beacon placement for network tomography," in *IMC*, 2004.

[15] Y. Breitbart, C. Y. Chan, M. Garofalakis, R. Rastogi, and A. Silberschatz, "Efficiently monitoring bandwidth and latency in IP networks," in *IEEE INFOCOM*, 2000.

[16] F. Li and M. Thottan, "End-to-end service quality measurement using source-routed probes," in *IEEE INFOCOM*, 2006.

[17] C. Chaudet, E. Fleury, I. G. Lassous, H. Rivano, and M.-E. Voge, "Optimal positioning of active and passive monitoring devices," in *CoNEXT*, 2005.

[18] K. Suh, Y. Guo, J. Kurose, and D. Towsley, "Locating network monitors: Complexity, heuristics and coverage," in *IEEE INFOCOM*, March 2005.

[19] G. R. Cantieni, G. Iannaccone, C. Barakat, C. Diot, and P. Thiran, "Reformulating the monitor placement problem: Optimal network-wide sampling," in *Intel Research Technical Report*, February 2005.

[20] J. F. Kurose and K. W. Ross, *Computer Networking: A Top-Down Approach Featuring the Internet Package, 3rd edition*. Addison-Wesley Longman Publishing Co., Inc., 2005.

[21] "Net optics: Network taps and aggregation solutions for passive network access." [Online]. Available: http://www.netoptics.com

[22] "Endace: Accelerated network security." [Online]. Available: http://www.endace.com/

[23] S. Moon, P. Skelley, and D. Towsley, "Estimation and removal of clock skew from network delay measurements," in *IEEE INFOCOM*, 1999.

[24] V. Paxson, "On calibrating measurements of packet transit times," in *Measurement and Modeling of Computer Systems*, 1998.

[25] S. Agrawal, K. V. M. Naidu, and R. Rastogi, "Efficient detection of distributed constraint violations," Bell Labs Technical Memorandum, Tech. Rep. ITD-05-46578D, 2006.

[26] U. Feige, "A threshold of ln n for approximating set cover (preliminary version)," in *STOC*, 1996.

[27] M. R. Garey and D. S. Johnson, *Computers and Intractability; A Guide to the Theory of NP-Completeness*. W. H. Freeman & Co., 1990.

[28] K. M. J. de Bontridder, B. V. Halldórsson, M. M. Halldórsson, C. A. J. Hurkens, J. K. Lenstra, R. Ravi, and L. Stougie, "Approximation algorithms for the test cover problem," in *Math. Program., 98(1-3, Ser. B)*, 2003, pp. 477–491.