Network Performance Anomaly Detection and Localization

Paul Barford
University of Wisconsin
pb@cs.wisc.edu

Nick Duffield
AT&T Labs–Research
duffield@research.att.com

Amos Ron
University of Wisconsin
amos@cs.wisc.edu

Joel Sommers
Colgate University
jsommers@colgate.edu

***Abstract*—Detecting the occurrence and location of performance anomalies (*e.g.*, high jitter or loss events) is critical to ensuring the effective operation of network infrastructures. In this paper we present a framework for detecting and localizing performance anomalies based on using an active probe-enabled measurement infrastructure deployed on the periphery of a network. Our framework has three components: an algorithm for detecting performance anomalies on a path, an algorithm for selecting which paths to probe at a given time in order to detect performance anomalies (where a path is defined as the set of links between two measurement nodes), and an algorithm for identifying the links that are causing an identified anomaly on a path (*i.e.*, localizing). The problem of detecting an anomaly on a path is addressed by comparing probe-based measures of performance characteristics with performance guarantees for the network (*e.g.*, SLAs). The path selection algorithm is designed to enable a tradeoff between ensuring that all links in a network are frequently monitored to detect performance anomalies, while minimizing probing overhead. The localization algorithm is designed to use existing path measurement data in such a way as to minimize the number of paths necessary for additional probing in order to identify the link(s) responsible for an observed performance anomaly. We assess the feasibility of our framework and algorithms by implementing them in ns-2 and conducting a set of simulation-based experiments using several different network topologies. Our results show that our method is able to accurately detect and localize performance anomalies in a timely fashion and with lower probe and computational overheads than previously proposed methodologies.**

I. Introduction

One of the primary responsibilities of network operators is to ensure that their infrastructures provide a level of performance and robustness that satisfies the commitments specified in their service level agreements (SLAs) with customers. A standard approach for addressing this responsibility is to monitor the behavior of the infrastructure and identify unusual or anomalous activity that either directly indicates or indirectly implies that the network is no longer behaving satisfactorily. Monitoring and detecting meaningful events in huge infrastructures with high bandwidth links where normal traffic behavior can be highly dynamic presents serious challenges.

Prior work on detecting network anomalies has focused primarily on techniques for distinguishing significant deviations from normal (*e.g.*, [1], [2]) or on specific aspects of unwanted behavior such as malicious attacks (*e.g.*, [3], [4]). These studies have been based almost exclusively on passive measurements, which cannot be used for monitoring key metrics in SLAs including delay or jitter.

In this paper we present a framework for monitoring, detecting and localizing performance anomalies in a network. Our framework assumes that a set of active measurement systems is deployed around the edges of the network (*e.g.*, implemented directly in routers as in [5]) and that probes sent between these systems can cover all links of interest in the network. The first component of our framework is an algorithm for detecting performance anomalies on an individual path, where a path is defined as the set of links between two measurement nodes. Our algorithm uses active probes between measurement nodes in conjunction with performance thresholds specified in SLAs. When a threshold is exceeded, an alarm is raised.

The second component of our framework is a path selection algorithm for performance anomaly *detection* that determines which path(s) should be probed at a given point in time. The objectives of this algorithm are to ensure that all links in the network are frequently probed in order to quickly detect anomalies, while limiting probing overhead. Our algorithm is iterative and associates weights with links such that the weight w_i associated with a link i increases over time if it is not on a path chosen for probing during a given time interval. This increase in weight results in an increased probability that a path containing that link will be probed in the near future. Whereas previous approaches, *e.g.* [6], [7], have employed techniques that ensure full coverage of all links of the network at each probing interval, this may come at an unacceptable cost of high probing overhead on network links used by multiple probing paths. Our methodology enables a tradeoff between the volume of probe traffic and the frequency of probing on individual links, and by extension the timeliness of detection.

The third component of our framework is a *localization* algorithm that enables efficient identification of the link that is responsible for the anomalous behavior. This algorithm is triggered when an anomaly is detected on a path. Our approach is to iteratively select additional paths to probe that will maximally increase information regarding the location of an anomaly. We show that conditioning the selection of a new path to probe such that it overlaps half of the current set of links with uncertain state optimally increases, on average, the information about the location of an observed anomaly.

We demonstrate and evaluate our framework by implementing our algorithms in ns-2. We use a set of synthetic topologies plus a set of topologies derived from traceroute measurements from the Rocketfuel [8] project as the basis for our experiments. We generate random performance anomalies in our experiments and assess the capability of our algorithms to detect and identify the offending links. Our results show that our algorithms are able to detect and localize the anomalies accurately, efficiently and in a timely fashion. We also show that while other path selection algorithms can introduce significant probing overhead on network links, our algorithm enables quick performance anomaly detection while limiting probing overhead. We believe that our results highlight the promise of our framework and its potential utility in operational deployments.

II. Related Work

Detecting and localizing network performance anomalies is an important problem for operational networks and has received substantial attention in prior work. Past efforts have largely focused on using passively collected data to detect, and possibly locate, performance anomalies; see, *e.g.*, [9]. For example, Huang *et al.* considered the problem of detecting general performance problems in a network using only passive packet measurements [10].

Bejerano and Ragosti describe a methodology for actively monitoring all the links of a network in an efficient way [6]. They show that the problem of selecting a minimal set of paths such that all links of a network are monitored is an instance of the well-known NP-hard *minimal set covering problem*, for which they provide a greedy heuristic solution. This work (and other, similar work, *e.g.*, [7]) assumes that per-path probing overhead is negligible, so that links that are probed by more than one path as part of the set covering do not suffer high probing overhead. However, many probing algorithms can produce 10s to 100s of kb/s of traffic or more per path, making this assumption problematic in operational networks. Our approach for selecting network paths differs from prior work in that we allow a tradeoff to be made between detection timeliness and potential overhead on network links.

Dhamdhere *et al.* considered the detection and localization of reachability faults in [7] based on the insights of [6] as well as work by Duffield [11]. In the latter work, Duffield describes a tomographic method for inferring the location of problematic links when considering binary network performance characteristics (*i.e.*, either a link exhibits a problem or it does not). Dhamdhere *et al.* extend the work of Duffield to an interdomain setting and attempt to improve the localization technique by leveraging additional network measurements. A number of other fault localization methods may be found in, *e.g.*, [12]–[17]. In general, these prior methods employ models (*e.g.*, prior fault probabilities in a Bayesian framework [15]) or external (not probe-based) measurement data (*e.g.*, route advertisements [7]) in order to better hypothesize the location of an observed fault. Our work differs from these prior methodologies in that we develop an algorithm for selecting additional paths for probing in order to empirically localize a performance anomaly that has been detected.

A great deal of prior work has focused on developing active probe-based methods for measuring performance characteristics of individual end-to-end paths. A repository of a number of popular tools for end-to-end measurement can be found in [18]. We use SLAm [19] in this study since it enables performance characteristics associated with SLAs to be measured accurately with a single probe stream. Our framework does not require the use of SLAm and could be used with other tools, if desired.

III. Framework Terminology and Approach

We consider the network under study as a directed multigraph $G = (V, E)$. We consider a *directed* network because we may want to label the edge with values (*e.g.*, failure status, congestion) that depend on the direction of traffic flow. We consider a *multigraph* because we may want to distinguish multiple links between two nodes (*e.g.*, composite links in which an IP layer link is realized by operating several physical links in parallel). Along with G comes a set of routes R, which for each ordered pair $(u, v) \subset V$ specifies a subset $R(u, v) \subset R$ of paths that join from u to v. We assume there is at least one route $r(u, v)$ from each u to each v. We also assume that the set R does not change during detection and localization. In § VII we discuss the issue of routing changes. Furthermore, we assume for simplicity that there is only one route from any node u to v.

A subset $W \subset V$ of nodes will be the measurement nodes. Each measurement node can act as a source or sink of probes, although it need not perform both functions. In this paper we assume that each measurement node is a *leaf node* in the network. That is, there is only one edge incident on a measurement node. We assume there is some subset $F \subset E$ of edges under study, in the sense that when a problem affecting packet transmission occurs on a link in F, we wish to become aware that an anomaly exists and determine its location. Finally, we assume for simplicity that performance anomalies are *persistent*.

The general approach of our framework is to proceed in two phases. During the first phase, we probe the network using the paths between the measurement nodes in W to detect performance anomalies. In this paper, we assume an intra-domain network setting in which the routes between the nodes in W are known, and in particular that the links traversed (covered) by those paths are known. During the detection phase, the main goal is to quickly detect anomalies on any link(s) residing on all paths between nodes in W. That is, our probing paths must *cover* (over some time period) all links in F. A key constraint is that of probing bandwidth. For example, it may not be possible to *simultaneously* probe all $O(N^2)$ paths between the W measurement points at a rate fast enough to detect events at all (small) timescales of interest. Furthermore, given a modest per-link probing bandwidth constraint, it may not even be possible to probe over a reduced set of paths that cover all the links of the network since there may be multiple probe streams crossing a single link.

Once an anomaly has been detected, we initiate a localization phase in an attempt to identify the link(s) responsible for the observed anomaly. In contrast to prior work, we assume that all localization is done through active probing. That is, we do not employ *external* information such as routing updates or device SNMP traps in order to narrow down the hypothesized location of a performance anomaly. Thus, the goal is to, as quickly as possible, determine the link(s) on which the performance anomaly is located while limiting the bandwidth used for additional probing.

Finally, we assume that for both the detection and localization phases time proceeds through a series of intervals, the length t of which is matched to the probing methodology employed. For example, t might be set such that a reasonable bound on measurement accuracy can be obtained.

IV. Performance Anomaly Detection

In active probing of a network to detect performance anomalies there is an inherent tension between attempting to maximize coverage over network links on the one hand, and minimizing the load due to probing on the other. In considering how to balance these desiderata, we first consider three strategic extremes.

A. Detection Strategies

1) Complete Path Coverage: The simplest, yet most intrusive strategy is to choose *all* measurement nodes in W and probe over this $O(N^2)$ set of paths. Depending on topology, this approach may result in excessive probing of some links.

2) Minimal Edge Coverage: A less intrusive strategy is to find a minimal set N of measurement nodes such that there is a set R' of paths $r(u, v)$ between nodes u, v in N with the property that each edge in F lies on at least one such path. The idea here is that an anomaly on any link in F will be evident in at least one measurement between nodes in N.

One approach to finding this set of paths is, following the approach of [6], to compute a minimal set covering. The standard greedy approximation is well known to be within a log-factor of the optimal [20]. For example, in Figure 1a, a greedy set covering approach might result in the following paths: $(A, E, G, H, D), (B, F, H, G, C), (A, E, F, B)$ (plus the reverse paths). Note however, that the links $(A, E), (B, F), (G, H)$ are covered by *two* probing paths. Depending on the probing methodology and operational constraints, this load may not be acceptable. Furthermore, for large networks the set covering computational overhead can be quite costly.

Another approach to furnishing a minimal set may come from the *path*-tomographic work of [21]. Given an incidence matrix A of edges on a set of paths Q, *i.e.*, $A_{ie} = 1$ iff path i traverses edge e, then the rows of A may not be of full rank (*i.e.*, rows are not linearly independent). In this case, we find the minimal spanning set of rows (= paths) $Q' \subset Q$. It would appear that the (set of endpoints of) paths in Q' provide minimal path coverage of all the edges traversed by paths in Q, though we have not yet verified this formally.

3) Complete Edge Coverage: A different strategy is to find the minimal set N of measurement nodes such that there is a set R' of paths $r(u, v)$ between nodes u, v in N such that if there is an anomaly on any edge or set of edges in F the identity of the anomalous link(s) can be determined from measurements on R'. Here we think of examples when the problem on a link is a 0-1 variable (lack of connectivity, performance measure exceeds threshold), and can be inferred by standard performance tomography methods.

Clearly, minimal edge coverage is necessary in order to detect anomalies on any edge. However, complete edge coverage can be viewed as wasteful in the sense that for a given problematic edge, some of the measurements may be not be needed to locate the link exhibiting an anomaly.

B. Detection Tactics

We now present an algorithm for determining which paths to probe during the detection phase. In principle, knowledge of the routing and topology could be used to find the exact minimal set of paths. However, topology may change (rarely) and so may routing (often). Hence some mechanism is needed to dynamically adapt the set of measurement paths to underlying changes in the network. One approach may be to precompute the routing changes due to a set of possible link failures (*i.e.*, compute minimum set coverings for several routing scenarios), and to employ a larger set of measurement paths in order to provide illumination of edges that would no longer be covered by measurements under rerouting under just minimal edge coverage. However, this may be computationally infeasible, particularly in large networks. This problem may even apply for minimal edge coverage even without the requirement of robustness under rerouting.

An opposite approach to heavyweight optimization is to adopt a computationally simple algorithm that nevertheless generates a set of covering paths that is not much larger than the optimal set. It is this approach that we take.

To each link $i \in F$ in our network, we assign an *importance weight* W_i. As our algorithm proceeds, a *current link weight* w_i is calculated. Each path r in the network is weighted according to the sum of the current weights of the links that comprise it. Initially, the current link weights are set to 0.

We start by sorting all possible paths according to their weight, and probe on k paths selected randomly from the K paths with the largest weights. For example, if $k = 1$ and the largest weight of all paths is $\bar{W}$, we choose one path randomly from all paths K that have weight $\bar{W}$. Once a path (or set of paths) is chosen, we send probes along each of those paths in an effort to detect performance anomalies.

Once probing during the current time interval has completed, we set to 0 the current weights on all links of paths that were checked. For links that were not probed, the current weight w_i is updated as $w_i = min(W_i, w_i + \frac{W_i}{(N-1)/k})$, where N is the total number of paths in the network that can be probed. Thus, the current weight w_i of a link varies between 0 and the importance (maximum) weight for that link W_i.

The heuristic here is that by visiting a certain link during an experiment, we reduce the weight of all paths that utilize that link. However, the weight of those paths rises as time proceeds. Note that the importance weight W_i of a given link may be set based, *e.g.*, on knowledge of important traffic flows, in order to induce more or less frequent probing. For simplicity in our experiments we set these importance weights to 1. Note also that since at each time interval we greedily choose the k highest weighted paths for probing, our algorithm is effectively a greedy solution to the minimum set covering problem. The key difference is that the computations are divided over a series of time intervals, thus reducing overhead at each interval. In § VI we quantify this reduction.

Note that this algorithm does not by itself guarantee that, during a given probing interval, links are covered *at most once*. In particular, with $k > 1$, it is indeed possible for links to be probed by multiple measurement nodes simultaneously during a given probing interval. An important variant of the above path selection methodology that we consider in our evaluation is to select k paths with the highest weights in a given time interval that do not have any (unidirectional) links in common.

C. Detecting Anomalies on an End-to-end Path

Network anomaly detection is based on the idea of identifying significant deviations from the normal state. Thus, our challenge is to develop a method for establishing a normal state of an end-to-end path with active probes, and then to detect when the conditions on that path are sufficiently different from normal to raise an alarm.

Once a path has been selected, a series of probe packets is sent along the path (according to some probing methodology), over a given time interval t. The duration of t should be sufficient to identify an anomalous condition on the path. In general, the specific value of t will depend on the probing methodology and desired tradeoffs between speed and accuracy of anomaly detection. In this paper, we use SLAm for probing [19] and set t to 10 seconds in our experiments below.

The result of probing a single path is a set of one-way measurements. Our goal is to label the links of a path as being in an anomalous state (*e.g.*, "condition red"), nearly in an anomalous state (*e.g.*, "condition yellow"), or functioning normally (*e.g.*, "condition green"). Similar to Nguyen and Thiran [15], our approach is to require a user to specify thresholds for what is anomalous (maximum values for delay/jitter/loss) and nearly anomalous (relative to maximum) for measurement on a given path. This provides an implicit definition of normal for each path characteristic and has important implications for deployment such as obviating the need to apply complex signal analysis to measurements. We argue that this approach is highly relevant to the target measured values for which explicit thresholds in SLAs already exist.

V. Performance Anomaly Localization

Once a performance anomaly has been detected on a path during the detection phase, the localization phase is initiated in order to "drill down" and find a link or set of links responsible for the observed anomaly. To illustrate the problem, we consider again the topology in Figure 1a. Even in such a simple topology, a set of paths that covers all links during the detection phase may not enable immediate localization of an observed anomaly. For example, given probing along the paths $(A, E, G, H, D), (B, F, H, G, C), (A, E, F, B)$ (plus the reverse paths) and an anomaly observed only on path (A, E, G, H, D), we can immediately eliminate the links $(A, E), (G, H)$ since they lie on anomaly-free paths. However, *we cannot narrow the location of the anomaly any further than* $\{(E, G), (D, H)\}$ *without additional probing or information.*

A. Localization Strategy

Given that a performance anomaly has been detected, the minimal edge coverage and complete edge coverage strategies described above lead naturally to a strategy that we call *conditional complete edge coverage*. Here we condition on an anomaly having been detected on some maximal subset R'' of measurement paths (maximal in that it contains all potentially problematic paths). We may then attempt to find the minimal set of nodes N' such that there is a set of paths Q between nodes in N' for which measurements on Q suffice to locate the problematic link(s) within R''.

Measurement can then be done on one of the set of paths in Q. The result of this probing can be used to condition a subsequent set of possible measurements. Each additional measurement gives rise to a new version of the conditioning problem. Note that this occurs whether or not the path measurement finds an anomaly; an outcome of no problem observed on a newly measured path may provide sufficient information to locate the remaining anomalous links.

B. Path Selection during Localization

How should paths be selected from the set Q for localizing a performance anomaly? In a general sense, given the current state of knowledge of the network (*e.g.*, some set of paths exhibits an anomaly) the next measurement path should be chosen in order to maximize the mutual information between the current and new knowledge. Finding a global maximum across a network may be difficult in practice; here we concentrate on a single path. Specifically, suppose a single path exhibits an anomaly. We aim to select an intersecting path in order to maximize the mutual information. In this paper we consider a simple state-counting approach to locating a single anomalous link on a path. We have also considered a more complex analysis of this problem that also allows for an anomaly on a new path chosen for measurement during the localization process. We intend to report on the latter analysis in the future. Our analysis in both cases shows that we should choose the second path to partially intersect the first path, rather than, for example, maximizing the path overlap. When anomalies are rare, half the first path should be overlapped by the second. A key to analysis is first conditioning on the existence of an anomaly on the first path.

1) Path Selection Analysis: State Counting on a Single Path: Consider a path of n links which we label $1, 2, \dots, n$. Suppose that end-to-end measurement along the path has indicated an anomaly on the path. For simplicity we assume that only one link is problematic. This is reasonable when the probability of any link anomaly is very small. If we assume links are independently problematic, then conditioned on *at least* one link being problematic, it is overwhelmingly likely that *only one* link is problematic. We also assume each link has an equal chance to exhibit an anomaly.

We consider a state of the path to specify for each link whether it is problematic or not. Conditioned on the path being problematic, in the above model there are n possible states (one for each link to be problematic). The number of possible states represents our uncertainty in the location of the anomaly. The aim of localization measurement is to minimize this uncertainty.

Now suppose we supplement the initial measurement that revealed a performance anomaly with a second measurement from the source along the subpath $(1, 2, \dots, m)$ for $m < n$. How should we best choose m? The idea is to choose m in order to minimize the number of possible states compatible with the information received in the second measurement. Here we do this is an average sense: we minimize the average number of possible states after the second measurement. We assume that the anomaly is persistent: the second measurement is subject to the same link problems as the first.

Suppose the original problem were in subpath $(1, 2, \dots, m)$. This happens with probability $p_- = m/n$ since links are equally likely to exhibit an anomaly. Assuming the anomaly lies on one of the first m links, there are $N_- = m$ possible states remaining. In the alternative case, the anomaly was in the subpath $(m+1, \dots, n)$, with probability $p_+ = (n-m)/n$, corresponding to $N_+ = (n-m)$ states. Hence the average number of possible states after the second measurement is $p_-N_- + p_+N_+ = \frac{m^2+(n-m)^2}{n}$ which is minimized when $m = n/2$. Thus the second measurement should encompass half the problematic path.

C. Localization Tactics

We now describe the algorithm used during the localization phase. Suppose that during the most recent detection phase interval we probe the paths $r_i, i \in 1..k$. Call this set of paths D. Assume we observe a performance anomaly on at least one path; call this set of paths $D' \subseteq D$. Initially, the set of paths in D' constitutes the current hypothesized location of the anomaly. We denote the hypothesized location of the anomaly as H. Note that as we narrow the location of the observed anomaly, links will be removed from H. Thus, we more generally consider H to be a set of partial paths, or *path segments*. Now, if $k > 1$, we may be able to immediately narrow the location of an observed anomaly. First, if we assume there is a single anomaly, we can reduce H to the set of path segments common to all paths in H. (That is, if there is only one anomaly, we assume that we observe the same anomaly on all problematic paths.) Next, we can remove links on which we did not observe an anomaly (*i.e.,* links on paths in $D - D'$) from path segments in H.

At this point, we employ the path selection analysis from above to select a set of paths for additional probing in order to further reduce H. That is, a path to probe for localization is chosen such that the overlap between the new path and a path segment $h_i \in H$ is as close as possible to half the length of h_i. Furthermore, if H consists of more than one path segment, we may probe more than one path during a given interval in the localization phase *as long as the paths do not overlap*.

Depending on the result of probing along the new path(s) (*i.e.,* a performance anomaly is observed or not), the set of path segments in H will decrease. If H has been reduced to one segment consisting of a single link, the localization algorithm terminates. Otherwise, a new overlapping path is chosen such that it has not yet been probed during the localization phase. If all paths that overlap path segments in H have been probed, the localization algorithm terminates and reports Has the best possible estimate of the anomaly location. Note that depending on probe monitor placement and network topology, it may not be possible to localize a given anomaly to a single link.

VI. Evaluation

A. Experiment Setup

We constructed simulation experiments using ns-2 [22] in order to evaluate the detection and localization algorithms described above. We modified ns-2 to implement the SLAm probing methodology described in Sommers *et. al* [19]. Currently, the SLAm implementation in ns-2 only includes the per-path probing methodologies described in [19] and does not yet include any distributional inference capabilities.

The focus of our experiments was to evaluate the path selection algorithms for the detection and localization phases as described above in a variety of increasingly complex topological settings. We explicitly do not focus on the accuracy characteristics of SLAm or indeed any particular one-way probing methodology. For this reason, we did not experiment with a variety of link bandwidths or propagation delays. The link bandwidths in each topology are 100 Mb/s with one-way delays of 10 milliseconds. In addition, rather than creating network performance anomalies *endogenously* using a range of background traffic conditions, we *explicitly* introduced them, *i.e.,* through the introduction of an ns-2 `ErrorModel` element for temporarily inducing packet loss or through temporary increase of the one-way link delay. All errors are injected in one direction for a given link. For simplicity, we use a network-wide loss rate threshold for detecting a performance anomaly, set at 0.5%, and focus solely on loss rate anomaly detection and localization. Finally, we note that in our current study we assume a centralized controller that chooses paths for probing. In future work we intend to investigate architectures for distributed coordination among measurement nodes.

We focus on a variety of topological settings for our experiments. We used 15 topologies: 5 synthetic HOT-like topologies created using the Orbis topology generator [23], and 10 topologies derived from the Rocketfuel project [8].

Four of these topologies are shown in Figure 1, and summary statistics for all topologies considered in our evaluation are given in Table I. For each topology, we consider all leaf nodes to have SLAm measurement capability (both sending and receiving probes). Among these measurement nodes, all paths are considered as candidates on which to initiate measurement. Furthermore, we assume that an anomaly can occur on *any* link in a given topology and that we wish to detect anomalies on *all* links of the network. To compute paths, we calculated shortest paths (using number of hops for simplicity) between all pairs of measurement nodes at the start of a simulation.

An individual simulation experiment proceeds with probing in the detection regime. For this phase, k paths are probed at a given time; paths with highest sum of link weights are chosen for probing, as described in § IV-B. After each group of k paths has been probed, link weights are updated. Each path is probed for t seconds. The value of t can be chosen based on the probing methodology, and need not be a fixed value. For simplicity in our experiments, we chose $t = 10$. At random time t_a, a persistent performance anomaly is created (*i.e.*, loss is induced by a packet dropping element) on a given link, which should result in an observable performance anomaly. For our simulations, we set the magnitude of the performance anomaly to be a loss rate of 1%, which should result in detection at some point in the future. In this paper we create only one anomalous link at a time, and assume that once a link has been corrupted, it persists in that state.

Once the performance anomaly has been detected at time $t_d \geq t_a$, the localization algorithm described in § V is invoked. In this paper, we assume that only one anomaly occurs at a time. Once the anomaly has been localized to the degree possible, the simulation terminates.

TABLE I
SUMMARY OF TOPOLOGIES CONSIDERED IN EVALUATION.

Topology	**Interior Nodes**	**Leaf Nodes**	**Total Nodes**	**Monitor Path Length** μ (σ)	**Node Degree** μ (σ)
Synthetic Topologies					
HOT 1	5	16	21	2.458 (0.266)	2.381 (11.748)
HOT 2	8	31	39	2.888 (0.452)	2.205 (13.746)
HOT 3	17	49	66	3.806 (1.240)	2.333 (10.718)
HOT 4	93	246	339	4.562 (1.147)	2.690 (19.078)
HOT 5	181	502	683	4.752 (1.269)	2.630 (27.799)
Rocketfuel-derived Topologies					
AS1299	17	14	31	3.692 (1.408)	2.839 (5.206)
AS3300	21	22	43	4.017 (1.206)	2.605 (6.435)
AS4323	40	11	51	4.400 (1.453)	6.314 (44.020)
AS1239	33	19	52	4.129 (1.004)	3.231 (7.750)
AS209	33	25	58	3.907 (0.806)	3.724 (13.572)
AS3549	51	10	61	3.467 (0.342)	15.934 (197.729)
AS3356	46	17	63	3.250 (0.262)	9.048 (104.562)
AS2914	47	23	70	4.403 (1.009)	3.171 (7.390)
AS701	48	35	83	3.235 (0.305)	5.277 (51.934)
AS7018	35	80	115	3.615 (0.753)	2.574 (16.352)

B. Results

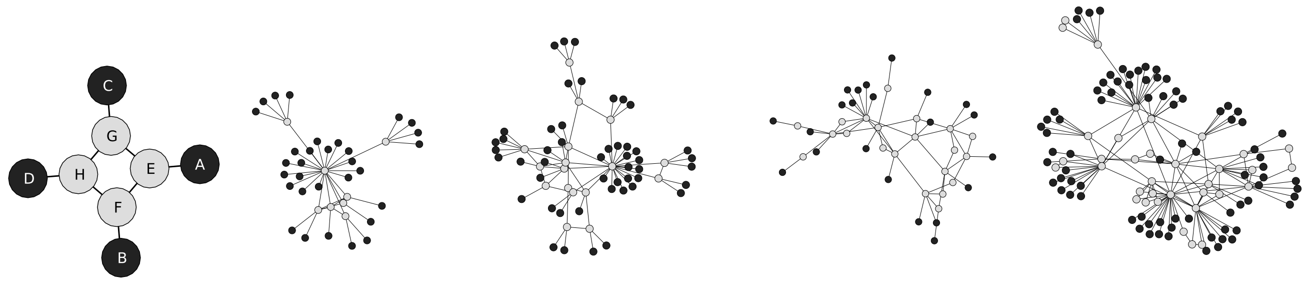
In the results below, we use a series of 100 simulation experiments started using different random seeds. We first discuss results related to the detection phase, followed by results related to the localization phase.

We first note that there were no false positives in performance anomaly detection or localization (*i.e.*, link(s) misidentified as the location of an anomaly), and there were no false negatives (*i.e.*, failure to identify the link(s) responsible for an anomaly). Given that our experiments were performed in simulation, this result should not be surprising. Nonetheless, this result is in sharp contrast to prior work (including simulation studies) in which both false positives and false negatives have been reported. The key difference is that we employ probing to localize anomalies whereas other approaches have used additional information (*e.g.*, routing updates, prior fault probabilities, *etc.*) that may simply not lead to correct inference of the location of the anomaly. While additional probing has some cost associated with it, our results clearly demonstrate the benefits of this approach.

1) Detection: For the detection phase, we examine three metrics: *detection time*, *link load*, and *network coverage*. We report detection time in terms of the number of probing intervals. For link load, we report mean and standard deviation of the number of probe streams incident on links that are probed during each time interval. We also consider the distribution of link load across the topology. Network coverage considers the fraction of links in F that are probed during a given time interval. For this metric, we report on the mean and standard deviation of coverage over all detection time intervals.

We considered three basic variants of path selection for the detection phase. In the first, we choose k paths for each interval, allowing these paths to possibly overlap (*i.e.*, link load may be greater than 1, coverage $\leq$ 1). In the second, we choose k top-weighted paths that do not overlap (*i.e.*, link load is at most 1, coverage $\leq$ 1). In the third, we compute a minimum set cover (*i.e.*, link load may be greater than 1, coverage = 1). For the first two path selection variants, we used values of k of $1, 2, 4, 6, 8, 10$ as well as the extreme $k = |R|$. For the case in which we allow paths to overlap, this results in probing *all* ($O(N^2)$) paths in network (*i.e.*, link load $\geq$ 1, coverage = 1). For the case in which we do not allow paths to overlap during a given detection probing interval, this results in choosing a set of non-overlapping, maximally-weighted paths in R (*i.e.*, link load at most 1, coverage $\leq$ 1). Below, we denote the overlapping case as N^2 and the non-overlapping case as k_{max}.

To begin, we show in Table II results for the two topologies HOT-3 and Rocketfuel-derived AS7018. The table shows the mean and standard deviation of detection time, link load, and network coverage for a range of values of k. We show results for the case in which we allow paths probed during detection to have overlapping links, and for the case in which we disallow overlap. For each topology, we see that increasing the number of paths probed during a given probing interval decreases the detection time. In particular, we observe that for the cases in which we cover *all* links in each probing interval (*i.e.*, $k = N^2$ and the minimum set cover), detection time is 0, meaning that the anomaly is detected during the same interval in which it occurs. However, we see that this rapid detection comes at a cost: for the N^2 case, the mean probing overlap on the link is extremely high, and for the minimum set covering case,



(a) Simple topology for illustrative purposes. (b) HOT topology 2. (c) HOT topology 3. (d) Rocketfuel AS3300 topology. (e) Rocketfuel AS7018 topology.

Fig. 1. Selected topologies used for simulation experiments. For each topology, leaf (measurement-capable) nodes are dark and standard packet forwarding elements are in light gray.

the mean load is greater than 1, *i.e.*, on average, each link is covered by more than one probe stream. For the case in which we choose as many non-overlapping paths as possible (k_{max}) in each probing interval, we see that while link load is (by definition) at most 1, the detection time does not rise substantially. From these results, we can clearly observe the potential problem of blindly choosing links that overlap. In the results below, we focus on on choosing non-overlapping paths during the detection phase, as well as the minimum set covering path selection.

Figure 2 expands on these results by showing, for a set of representative topologies, detection times for a range of values of k. For this figure, we only consider non-overlapping paths. As with Table II, we see a rapid decrease in detection time as we increase the number of paths probed during each interval. We also see that for the case k_{max}, detection time is often close to zero.

Table III shows detection results for all topologies, for the case of choosing as many high-weighted, non-overlapping paths as possible (k_{max}) and for a minimum set covering. We observe that although detection times for k_{max} are somewhat topology dependent, in general the mean detection time is much less than 1. Topology also affects link load when choosing a minimum set covering for anomaly detection probing. The table shows that, in general, average link load is less than 2. We show, in Figure 3, the distribution of link load for minimum set coverings for all topologies in order to examine this issue more carefully. We see that between 60–85% of links only have 1 probe stream traversing them, but there are links on which there are several probe streams. For nearly all of the topologies we consider, the load on some links introduced by using a greedy minimum set covering may be too large to consider using this algorithm in practice.

Finally, we examine processing overhead during the detection phase. Table IV shows the mean and standard deviation of processing time required to compute the set of paths to probe during detection for the k_{max} (no-overlap) and minimum set covering algorithms over 100 runs on a modern server. For this experiment we compute the minimum set covering for *each* interval, which is normally not required unless the routing topology has changed. Thus, this computational cost can be amortized over a series of probing intervals. Nevertheless, we see from the table that the cost of computing the minimum set covering is extremely high compared with calculating a maximal subset of high-weighted, non-overlapping paths. In particular, the cost of the minimum set covering is about a factor of 40 greater for the rather small HOT-2 topology; this

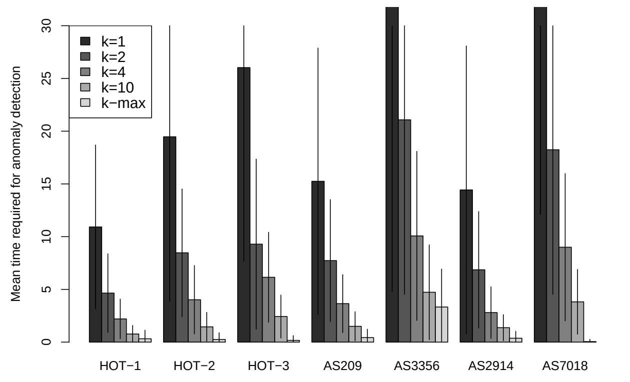


Fig. 2. Representative results of performance anomaly detection time for $k = (1, 2, 4, 10, k_{max})$ paths simultaneously probed during the detection phase. The *no-overlap* variant of the detection path selection algorithm is used for these results. Bars indicate mean detection time and lines show one standard deviation above and below the mean.

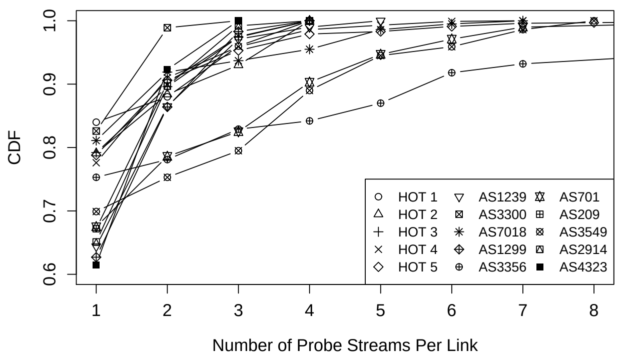


Fig. 3. CDF of link loads when using a minimum set covering during the detection phase.

TABLE II
DETECTION PHASE RESULTS FOR THE HOT-3 AND ROCKETFUEL AS7018 TOPOLOGIES WITH A RANGE OF VALUES FOR k.

Topology	**k**	**Detection Time** μ (σ)	**Link Load** μ (σ)	**Network Coverage** μ (σ)	**Detection Time** μ (σ)	**Link Load** μ (σ)	**Network Coverage** μ (σ)
		Link overlap allowed			**No link overlap allowed**		
HOT-3	1	26.030 (4.285)	1.000 (0.000)	0.028 (0.033)	26.030 (4.285)	1.000 (0.000)	0.028 (0.035)
	2	13.720 (3.343)	1.379 (0.225)	0.043 (0.040)	9.290 (2.840)	1.000 (0.000)	0.058 (0.022)
	4	9.010 (2.647)	1.828 (0.268)	0.067 (0.049)	6.150 (2.066)	1.000 (0.000)	0.113 (0.017)
	N^2 / k_{max}	0.000 (0.000)	58.130 (0.000)	1.000 (0.000)	0.170 (0.671)	1.000 (0.000)	0.887 (0.030)
	min set cover	0.000 (0.000)	1.338 (0.000)	1.000 (0.000)			
AS7018	1	36.200 (4.090)	1.000 (0.000)	0.020 (0.026)	36.200 (4.909)	1.000 (0.000)	0.020 (0.026)
	2	25.060 (4.113)	1.417 (0.193)	0.028 (0.026)	18.240 (3.703)	1.000 (0.000)	0.039 (0.017)
	4	13.950 (3.266)	1.849 (0.253)	0.043 (0.037)	9.000 (2.643)	1.000 (0.000)	0.075 (0.014)
	N^2 / k_{max}	0.000 (0.000)	102.910 (0.000)	1.000 (0.000)	0.050 (0.468)	1.000 (0.000)	0.928 (0.053)
	min set cover	0.000 (0.000)	1.396 (0.000)	1.000 (0.000)			

TABLE III
DETECTION PHASE RESULTS FOR ALL TOPOLOGIES FOR THE k_{max} AND SET COVER PATH SELECTION ALGORITHMS. NOTE: FOR THE k_{max} CASE, LINK LOAD IS ALWAYS 1; FOR THE MINIMUM SET COVER CASE, DETECTION TIME IS ALWAYS 0 AND NETWORK COVERAGE IS ALWAYS 1.

	Detection Time μ (σ)	**Network Coverage** μ (σ)	**Link Load** μ (σ)
Topology	k_{max} **(No overlap)**		**Minimum set cover**
HOT-1	0.320 (0.903)	0.762 (0.046)	1.320 (0.000)
HOT-2	0.250 (0.801)	0.841 (0.056)	1.395 (0.000)
HOT-3	0.170 (0.671)	0.887 (0.030)	1.338 (0.000)
HOT-4	0.120 (0.571)	0.909 (0.032)	1.377 (0.000)
HOT-5	0.060 (0.239)	0.924 (0.001)	1.409 (0.000)
AS1299	0.310 (0.784)	0.737 (0.052)	1.525 (0.000)
AS3300	0.220 (0.645)	0.818 (0.051)	1.185 (0.000)
AS4323	0.210 (0.675)	0.736 (0.058)	1.462 (0.000)
AS1239	0.520 (0.957)	0.664 (0.045)	1.529 (0.000)
AS209	0.480 (0.839)	0.710 (0.047)	1.451 (0.000)
AS3549	1.700 (1.378)	0.470 (0.032)	1.973 (0.000)
AS3356	3.210 (1.904)	0.360 (0.039)	2.267 (0.000)
AS2914	0.460 (0.863)	0.694 (0.047)	1.460 (0.000)
AS701	1.340 (1.348)	0.537 (0.024)	1.917 (0.000)
AS7018	0.050 (0.468)	0.928 (0.051)	1.396 (0.000)

disparity grows dramatically with topology size.

TABLE IV
PROCESSING TIME COMPARISON FOR k_{max} AND MINIMUM SET COVERING PATH SELECTION. TABLE ENTRIES SHOW MEAN AND STANDARD DEVIATION OVER 100 RUNS; VALUES ARE ALL IN SECONDS. EXPERIMENTS RUN ON AN OTHERWISE IDLE SYSTEM WITH A DUAL-CORE INTEL XEON CPU RUNNING AT 2.4 GHZ.

Topology	k_{max} **(No overlap)** μ (σ)	**Minimum set cover** μ (σ)
HOT-1	0.001147 (0.000144)	0.023271 (0.002110)
HOT-2	0.004275 (0.000112)	0.159257 (0.007673)
HOT-3	0.011065 (0.000340)	0.634245 (0.007915)
HOT-4	0.356962 (0.009711)	120.318191 (1.654599)
HOT-5	1.530194 (0.041411)	1914.196850 (29.703459)

2) Localization: For the localization phase, we focus on the time required to localize an observed performance anomaly, and the number of paths probed during the localization phase. As with the results for detection, we report the time required for localization in terms of the number of probing intervals. Note that because we may probe more than one (non-overlapping) path during a given localization interval (as described in §V-C), the number of paths probed during the localization phase may be greater than the time required for localization. Finally, we note that in all experiments, the localization algorithm succeeded to the extent possible given measurement node placement and the topology.

Table V shows results for localizing an anomaly that was observed during the detection phase. First, we observe that the time required for localization is somewhat higher for the detection phase case of k_{max} than for the choice of a minimum set cover during detection. The basic reason is that for the minimum set cover, more paths are probed during each probing interval, thus there is greater opportunity for narrowing the set of hypothesized anomalous path segments immediately after an anomaly has been detected, but prior to probing during localization. From this result, we observe some benefit to over-probing links during detection with a minimum set cover. However, we also see that localization times are still rather small when using k_{max} during the detection phase. Furthermore, the probing and computational costs of a minimum set cover may simply not be acceptable.

TABLE V
LOCALIZATION PHASE RESULTS FOR ALL TOPOLOGIES FOR THE k_{max} AND MINIMUM SET COVER PATH SELECTION ALGORITHMS.

	Localization Time μ (σ)	**Paths Probed** μ (σ)	**Localization Time** μ (σ)	**Paths Probed** μ (σ)
	k_{max} **(No overlap)**		**Minimum set cover**	
HOT-1	2.340 (0.690)	3.340 (0.690)	1.710 (0.675)	2.420 (0.955)
HOT-2	2.320 (0.700)	3.410 (0.826)	1.680 (0.714)	2.300 (1.005)
HOT-3	3.040 (1.045)	5.050 (1.470)	2.140 (0.817)	3.320 (1.158)
HOT-4	3.420 (1.079)	5.820 (1.522)	2.560 (1.896)	3.850 (1.998)
HOT-5	3.370 (1.182)	5.730 (1.667)	3.180 (1.092)	5.360 (1.544)
AS1299	3.040 (0.959)	4.230 (1.109)	1.750 (0.758)	2.230 (1.094)
AS3300	3.470 (1.010)	5.280 (1.369)	2.620 (1.074)	3.580 (1.394)
AS4323	3.290 (1.009)	5.070 (1.370)	1.770 (0.842)	1.950 (1.080)
AS1239	3.420 (1.018)	4.990 (1.320)	1.820 (0.878)	2.440 (1.213)
AS209	3.190 (0.958)	5.020 (1.330)	2.000 (0.882)	2.650 (1.187)
AS3549	2.570 (0.733)	3.540 (0.735)	1.250 (0.791)	1.240 (0.788)
AS3356	2.500 (0.736)	3.600 (0.851)	1.040 (0.493)	1.080 (0.697)
AS2914	3.310 (0.958)	5.170 (1.304)	1.890 (0.882)	2.360 (1.188)
AS701	2.880 (0.691)	4.530 (0.907)	1.280 (0.703)	1.360 (0.933)
AS7018	2.560 (0.810)	4.070 (1.128)	1.980 (0.787)	3.000 (1.138)

Finally, we compare the path selection algorithm for localization described in § V-B with two other approaches. In the first approach, we choose a path that has the highest number of overlapping links on a path segment in the set of hypothesized anomalous links ("longest match"). In the second approach, we choose a path that has the smallest number of overlapping links on a path segment in the set of hypothesized anomalous links (but with at least one overlapping link) ("shortest match"). Figure 4 compares the mean number of paths required to localize an anomaly for these three approaches (half overlap, longest overlap, and shortest overlap) for 7 representative topologies. In general, we observe that the mean and variability of the number of paths required for anomaly localization increase for the shortest and longest match path selection variants, though the results are somewhat topology dependent.

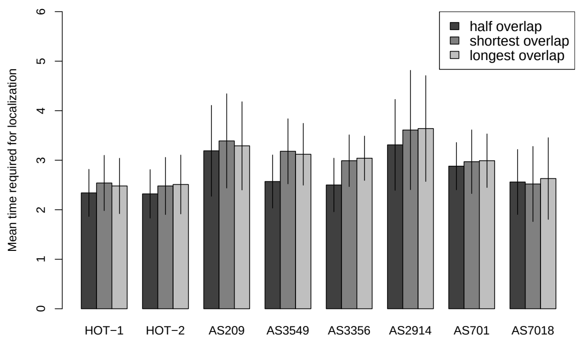


Fig. 4. Representative results comparing path selection policies during localization. Bars indicate mean number of paths required for localization. Lines indicate one standard deviation above and below the mean.

VII. Summary and Conclusions

In this paper we present a framework for identifying and localizing performance anomalies in a network. In contrast to prior work on network-wide anomaly detection, our framework is based on using active probes to measure behavior on origin-destination paths and simplifies the problem of raising alarms by using SLA performance guarantees as thresholds. We describe an algorithm for selecting the paths over which probes will be sent that is designed to ensure that all links in a network are probed in a fair and parsimonious fashion. We also describe a localization algorithm that identifies the link that is causing the anomaly to be identified efficiently and quickly. Using a set of simulations, we show that our set of algorithms is effective at detecting and localizing performance anomalies.

In this work we assume a centralized control point has up-to-date topological information, and can use that information to make decisions about which paths to probe. In the future we intend to examine distributed algorithms for network-wide monitoring. We also intend to broaden our work to include different kinds of performance anomalies, and to perform experiments in a laboratory testbed. In our experiments, we have assumed that routes do not change. If a route changes during the detection phase for a path between monitor stations u and v *and* we detect a performance anomaly between those two measurement hosts, we can set the hypothesized set of problematic links to include the path segments *both before and after the routing change*. The localization phase can then proceed as normal. If no anomaly occurred between stations u and v but the route changed, we can update the weights only on links that were part of the route both before and after the change (*i.e.*, set their current weights to zero). Thus, we will be likely to choose the parts of the path that changed in a near iteration of the detection phase. We leave further examination of this issue for future work.

References

[1] P. Barford, J. Kline, D. Plonka, and A. Ron, "A Signal Analysis of Network Traffic Anomalies," in *Proceedings of ACM SIGCOMM Internet Measurement Workshop*, November 2002.

[2] A. Lakhina, M. Crovella, and C. Diot, "Diagnosing Network-wide Traffic Anomalies," in *Proceedings of ACM SIGCOMM '04*, August 2004.

[3] A. Hussain, J. Heidemann, and C. Papadopoulos, "A Framework for Classifying Denial of Service Attacks," in *Proceedings of ACM SIGCOMM '03*, August 2003.

[4] M. Roesch, "Snort - Lightweight Intrusion Detection," in *Proceedings of USENIX LISA '99*, November 1999.

[5] "Cisco IOS IP SLAs," http://www.cisco.com/go/ipsla, 2008.

[6] Y. Bejerano and R. Rastogi, "Robust Monitoring of Link Delays and Faults in IP Networks," in *Proceedings of IEEE INFOCOM '03*, April 2003.

[7] A. Dhamdhere, R. Teixeira, C. Dovrolis, and C. Diot, "NetDiagnoser: Troubleshooting network unreachabilities using end-to-end probes and routing data," in *Proceedings of ACM CoNEXT '07*, December 2007.

[8] N. Spring, R. Mahajan, and D. Wetherall, "Measuring ISP Topologies with Rocketfuel," in *Proceedings of ACM SIGCOMM '02*, August 2002.

[9] M. Steindera and A. Sethi, "A Survey of Fault Localization Techniques in Computer Networks," *Science of Computer Programming*, vol. 53, pp. 165–194, 2004.

[10] P. Huang, A. Feldmann, and W. Willinger, "A Non-intrusive, Wavelet-based Approach to Detecting Network Performance Anomalies," in *Proceedings of ACM Internet Measurement Workshop '01*, November 2001.

[11] N. Duffield, "Network tomography of binary network performance characteristics," *IEEE Transactions on Information Theory*, vol. 52, pp. 5373–5388, 2006.

[12] S. Kandula, D. Katabi, and J. Vasseur, "Shrink: A tool for failure diagnosis in IP networks," in *Proceedings of ACM SIGCOMM MineNet Workshop*, August 2005.

[13] R. Kompella, J. Yates, A. Greenberg, and A. Snoeren, "IP fault location via risk modeling," in *Proceedings of NSDI '05*, May 2005.

[14] ——, "Detection and Localization of Network Black Holes," in *Proceedings of IEEE INFOCOM '07*, May 2007.

[15] H. Nguyen and P. Thiran, "The Boolean Solution to the Congested IP Link Location Problem: Theory and Practice," in *Proceedings of IEEE INFOCOM '07*, May 2007.

[16] M. Natu and A. Sethi, "Efficient Probing Techniques for Fault Diagnosis," in *Proceedings of the IEEE Conference on Internet Monitoring and Protection (ICIMP '07)*, July 2007.

[17] Y. Zhao, Y. Chen, and D. Bindel, "Scalable deterministic overlay network diagnosis," in *Proceedings of ACM SIGCOMM '06*, August 2006.

[18] CAIDA, "Internet Tools Taxonomy," http://www.caida.org/tools/taxonomy, 2008.

[19] J. Sommers, P. Barford, N. Duffield, and A. Ron, "Accurate and Efficient Network-wide SLA Compliance Monitoring," in *Proceedings of ACM SIGCOMM '07*, August 2007.

[20] P. Slavík, "A tight analysis of the greedy algorithm for set cover," in *STOC '96: Proceedings of the twenty-eighth annual ACM symposium on Theory of computing*, 1996, pp. 435–441.

[21] D. Chua, E. Kolaczyk, and M. Crovella, "Efficient estimation of end-to-end network properties," in *Proceedings of IEEE INFOCOM '05*, 2005.

[22] "The network simulator ns-2," http://www.isi.edu/nsnam/ns/.

[23] P. Mahadevan, C. Hubble, D. Krioukov, B. Huffaker, and A. Vahdat, "Orbis: Rescaling Degree Correlations to Generate Annotated Internet Topologies," in *Proceedings of ACM SIGCOMM '07*, August 2007.