The Design Space of Probing Algorithms for Network-Performance Measurement

Aaron D. Jaggard[*]
Rutgers University
adj@dimacs.rutgers.edu

Swara Kopparty[†]
Harvard University
skopparty@post.harvard.edu

Vijay Ramachandran[‡]
Colgate University
vijayr@cs.colgate.edu

Rebecca N. Wright[§]
Rutgers University
rebecca.wright@rutgers.edu

ABSTRACT

We present a framework for the design and analysis of probing methods to monitor network performance, an important technique for collecting measurements in tasks such as fault detection. We use this framework to study the interaction among numerous, possibly conflicting, optimization goals in the design of a probing algorithm. We present a rigorous definition of a probing-algorithm design problem that can apply broadly to network-measurement scenarios. We also present several metrics relevant to the analysis of probing algorithms, including probing frequency and network coverage, communication and computational overhead, and the amount of algorithm state required. We show inherent tradeoffs among optimization goals and give hardness results for achieving some combinations of optimization goals. We also consider the possibility of developing approximation algorithms for achieving some of the goals and describe a randomized approach as an alternative, evaluating it using our framework. Our work aids future development of low-overhead probing techniques and introduces principles from IP-based networking to theoretically grounded approaches for concurrent path-selection problems.

[*]Partially supported by NSF grants 0751674, 0753492, and 1101690. Work done in part while Visiting Assistant Professor of Computer Science, Colgate University. Current affiliation: Formal Methods Section (Code 5543), U.S. Naval Research Laboratory.

[†]Work done in part during the 2010 DIMACS/DIMATIA U.S./Czech International REU Program while supported by NSF grants 0753492 and 1004956.

[‡]Partially supported by NSF grant 0753061 and by the DIMACS special focus on Algorithmic Foundations of the Internet. Work done in part while visiting DIMACS, Rutgers University.

[§]Partially supported by NSF grants 0753061 and 1101690.

SIGMETRICS'13, June 17–21, 2013, Pittsburgh, PA, USA.

Categories and Subject Descriptors

C.2.3 [**Computer-Communication Networks**]: Network Operations—*Network monitoring*; C.4 [**Performance of Systems**]: Measurement techniques; F.2.3 [**Analysis of Algorithms and Problem Complexity**]: Tradeoffs among Complexity Measures

General Terms

Algorithms, Measurement

Keywords

Network-performance analysis; probing algorithms; probing metrics and complexity measures; design-space tradeoffs; hardness results; randomized probing; coupon-collector's problem

1. INTRODUCTION

Operators must monitor performance of their networks for many reasons, *e.g.*, providers may want to ensure that customers' service-level agreements (SLAs) are being fulfilled [22], or administrators of a datacenter may want to detect and diagnose abnormalities affecting latency-critical applications [20]. The measurement data used to analyze performance or to detect anomalies can be inferred from observing existing traffic, or they can be collected from analyzing the properties of test packets (or probes) injected into the network. Both types of methods have benefits and drawbacks. For example, the former, more passive, type of measurement is lower cost but restricts analysis to data that happens to be available. Probing can give more current and accurate information about the state of the network; but, realizing the benefits of probing without incurring substantial overhead involves important design decisions, because the injected test traffic can consume network resources. Thus, the design of low-overhead probing methods is an important, ongoing area of research.

However, the "correct" definition of "low-overhead" is unclear, because overhead of a probing method can be described in many ways, *e.g.*: the raw amount of additional probing traffic, the distribution of probing traffic among nodes and links in the network, the number of monitoring stations involved, the amount of state required to coordinate probing, *etc*. Optimizing any one of these measures may come at the detriment of another. More importantly, there may be tradeoffs among these overhead-minimization goals and the quality of the measurements obtained or the scope of the network that can be monitored properly. We demonstrate several such tradeoffs in this paper.

Our goal in this paper is neither to posit a single "correct" notion of overhead nor is it to propose a single probing method that achieves the "correct" balance among overhead and measurement quality. Instead, we seek to build on work that investigates tradeoffs in probing-method design (*e.g.*, [3, 5, 18]) by beginning a thorough exploration of the space of probing algorithms and the various tradeoffs inherent in that space. A good deal of research has studied how to infer performance characteristics from end-to-end measurements (the field of network tomography, *e.g.*, [8, 10]), and how to use network measurements to diagnose traffic anomalies (*e.g.*, [9]). However, research that studies the design and impact of the measurement technique itself is more sparse and generally considers a small set of optimization goals at a time. In this paper, we attempt to understand the relationships among a broad set of optimization goals in a more general sense. Towards this end, we focus here on unicast probing along traffic-routable paths that is used to collect end-to-end measurements; we present, and work with, an abstraction of the probing-design problem that is general enough to capture the problem of designing a probing strategy for a variety of tasks, including anomaly detection or tomography. This allows us to investigate the tradeoffs in the design of probing techniques.

1.1 A motivating example

Probing can be used for detecting performance anomalies by recording transmission failures, slow speeds, *etc.*, as probe packets travel along paths in the network. It is unreasonable to probe all possible paths all the time; this would detect anomalies quickly, but it could unreasonably burden the network with probing traffic (and even exacerbate problems that it might be used to detect). Instead, consider a minimum set-covering approach (a variant of that used in [5]): Precompute (or approximate) a minimum-size subset of paths that includes all the links of interest, and then probe only these paths at repeated time intervals. After the precomputation phase, the probing can be decentralized with little state (because source nodes for probing packets need only keep track of the destinations of precomputed probing paths); in addition, it is guaranteed to measure every link of interest, resulting in reasonably fast anomaly detection. However, as shown by Barford *et al.* [3], this procedure can create unnecessary load on links, probing them from multiple sources in the same time interval. And, as we show in Sec. 6, because finding a minimal set of paths is NP-complete, it may be unreasonable to perform the computation (even though a $O(\log n)$-approximation exists) whenever the network topology or routing changes.

Alternately, Barford *et al.* [3] propose an algorithm that balances the tradeoff among two important goals: ensuring that each network link is probed "often enough" (parameterized by an importance value I_ℓ for each link ℓ) and ensuring that the the link load is kept low (measured by the number of concurrent probing streams transiting a link). Their experimental results show an improvement in load over the minimal set-covering technique. The algorithm selects a subset of paths to probe in each time interval based on a dynamic weight that helps track which links were probed in previous intervals; in this way, paths that contain links that have not been probed in some time (relative to their importance) will be chosen, but links will tend not to be "over-probed." More formally, initialize the dynamic weight w_ℓ of each link ℓ to be 0, and let the weight of a path be the sum of the weights of the links it comprises. Then in each timestep, for fixed parameters k and K, k paths selected randomly from the K paths of largest weight are probed. For the next timestep, link weights are updated in the following manner: the weight of every link just probed is set to 0; the weight of every other link is updated to $w_\ell' = \min\left(I_\ell, w_\ell + \frac{I_\ell}{(N-1)/k}\right)$, where N is the total number of paths that can be probed.

Although this algorithm achieves a balance among the two important goals mentioned above, it comes at the expense of other algorithm properties that might be desirable; in particular, there are properties that the set-covering approach has that this algorithm does not. First, this algorithm requires some additional, constantly updated, state, because the dynamic link weights used for path selection must be adjusted at every timestep. Second, it requires some amount of centralization or communication among the nodes sending probing packets, because the number of paths selected and the updates to link weights must be coordinated; nodes cannot meet the algorithm's specification by independently deciding when to send probing packets. Third, we show that there are inputs for which this algorithm may never probe some particular links. (We give such an example in Sec. 7.1.2.)

It is not clear whether the combinations of properties achieved by these two different approaches are mutually exclusive or not. More generally, it is not clear from previous work what, if any, tradeoffs are inherent in the design of probing algorithms. This paper begins to address these types of questions by identifying algorithmic properties of interest and investigating their relationships.

1.2 Our contributions

In this paper, we rigorously develop a framework for analyzing the design space of probing algorithms. We give a formal definition of an abstraction for the probing-algorithm design problem (Sec. 3) that is general enough to capture various goals of unicast probing techniques used for end-to-end measurements. Applications of our results include the design of network-tomography or anomaly-detection algorithms. We highlight assumptions about inputs to the problem, often taken for granted in previous work, that are relevant to the difficulty of solving the problem. Our framework also includes different types of metrics (Sec. 4) that can be used to evaluate probing-path selection; these metrics correspond to realistic optimization goals and constraints that network operators may have.

We use this framework to demonstrate some inherent tradeoffs in the design space of probing algorithms (Sec. 5). Specifically, there exist networks such that algorithms that perform well with respect to some metrics will perform badly with respect to other metrics. Further, for numerous combinations of optimization goals, it is computationally intractable to find a sequence of paths that achieves that combination of goals, even approximately (Sec. 6). Finally, we consider a randomized approach (Sec. 7) as an alternative, evaluating it using the properties defined in our framework.

2. RELATED WORK

Barford *et al.* [2] were among the first to examine tradeoffs in probing-strategy design. They introduce the concept of marginal utility to probing-node selection for topology discovery. They find that adding nodes to the set of probing sources has quickly diminishing utility (*i.e.*, provides little additional topology information) beyond the second node, while adding nodes to the set of probing destinations is much more helpful. Because that work is focused on topology discovery, issues of constant monitoring overhead are not considered.

Bejerano and Rastogi [5] address the problem of low-overhead probing for anomaly detection with a two-phase approach, separately considering the number of source nodes involved in probing and the cost of probing traffic resulting from that choice. They show that the optimization problem in each phase is NP-hard but

admits approximation based on well-known algorithms for Minimum Set Cover and Minimum Vertex Cover. Their work does not consider the relationship between the two phases' optimization goals, nor does it consider how minimizing overall network load (a sum over all paths chosen) might impact individual links in terms of load. (We examine versions of both of these tradeoffs in Sec. 5.) It also does not consider probing frequency (in that all chosen paths are continually used for monitoring). Further, their formulation is more specific than ours, in that its measurement strategy sends a probe from the same source node to both endpoints of a link; thus the network-coverage requirements of the selected nodes and paths differ slightly from our setting.

Nguyen *et al.* [18] present a variant of the probing-design problem with a polynomial-time solution: they consider how to design a probing strategy, by choosing the frequency at which paths are probed, that minimizes the total number of probing packets needed to detect two different types of reachability failures; by allowing different coverage requirements of probes based on the type of failure being detected, they are able to reduce their optimization problem to linear programming. Their setting is more specific than most previous work and our work because of the coverage relaxation; similar toBejerano and Rastogi [5], Nguyen *et al.* use a network-wide definition of overhead without considering per-link impacts.

As discussed above in Sec. 1.1, Barford *et al.* [3] propose an algorithm for anomaly detection that attempts to balance the frequency at which links are probed with the per-link load imposed by probes. That work does not use network-wide overhead metrics, such as the total number of paths or packets, as optimization goals. In addition to the algorithm for anomaly detection, Barford *et al.* propose a probing strategy for localization of the anomaly.

Song *et al.* [23] develop the NetQuest framework and apply it to additive performance metrics (such as delay, in contrast to non-linear metrics such as failure). They use Bayesian experimental design to (centrally) choose the paths that would be probed in a way that would maximize the benefit to the experiment (according to a chosen metric). They also consider the problem, distinct from what we consider here, of how to best infer information about the network from a limited set of measurements.

As we do here, Breslau *et al.* [7] provide a theoretical treatment of problems relevant to network monitoring, but they focus on novel facility-location problems. Their work, like ours, includes properties of IP networks in the definition of an optimization problem requiring the choice of paths to cover network resources. For example, sets of paths are included in the problem input to account for a preselection of forwarding paths by some underlying network-routing mechanism.

There are several relevant theoretically grounded results about difficult path-selection problems. However, many of these problems do not account for routing issues. For example, the Multiple Edge Disjoint Paths problem [11] seeks a maximum-cardinality subset of given source-destination pairs such that a directed path can be assigned to each pair with no two pairs' paths sharing an edge in common. This problem is NP-hard in general. Although it resembles the problem of finding a maximum-cardinality set of non-overlapping probing paths in a network, there is an important difference: in the network-probing setting, there is often only one possible directed path to assign to any source-destination pair (*e.g.*, the IP-forwarding path from the source to the destination computed by some routing protocol). We show in Sec. 6 that, unfortunately, this restriction does not make the problem easy.

Parekh and Segev [19] describe the Path Hitting problem, which has a direct correlation to minimum covering in our setting. Assuming that each path $p \in \mathscr{D}$ is associated with a cost c_p, and saying that $p \in \mathscr{H}$ hits $p' \in \mathscr{D}$ if p and p' share at least one edge, the Path Hitting problem is: Given two sets of paths $\mathscr{D}$ and $\mathscr{H}$ in an undirected graph G, find a minimum-cost subset of $\mathscr{H}$ whose members collectively hit those of $\mathscr{D}$. The NP-complete Minimum Set Cover problem reduces to Path Hitting, and Path Hitting is a special case of Minimum Set Cover, implying the same approximation results for Path Hitting as for Set Cover. We use a similar reduction to show a hardness result for one of our probing-design problem variants (see Sec. 6).

3. FORMALIZING THE PROBLEM

The following is our definition of the abstract problem of probing-algorithm design for a network. The output corresponds to an implementation of selecting paths to probe over time. Like the problem definitions in [7]—but unlike routing-agnostic formulations, *e.g.*, the Maximum Edge Disjoint Paths problem [11], in which any set of links can be chosen to form paths between source-destination pairs—we assume that the set of possible probing paths is somehow constrained (*e.g.*, by some underlying routing system) and thus forms part of the input to the problem.

DEFINITION 3.1 (PROBING ALGORITHM). *For an undirected network $G=(V,E)$ and a set of paths $\mathscr{P} \subseteq 2^E$, define a (possibly randomized) algorithm $f : \mathbb{N} \to 2^{\mathscr{P}}$ that, for each discrete timestep $t = 1,2,\ldots$, selects a set of paths from $\mathscr{P}$. We call f a* probing algorithm for $(G,\mathscr{P})$ *(or just a* probing algorithm*), and we think of $f(t)$ as the set of paths that will be probed at time t.*

Of course, without any additional requirements, the problem is uninteresting; thus, at various points below, we impose different additional restrictions on f and $\mathscr{P}$, occasionally requiring additional parameters as input to the design problem. Such additional requirements might also be needed to ensure that a probing algorithm is suitable for a particular application or that it achieves certain other goals that may also be desired. For example, consider the problem of designing a sequence that guarantees measurement across some of the network's links: here, we may assume that the input specifies a subset of links $F \subseteq E$ and may require that every link in F appears in at least one path in $\cup_t f(t)$. Sec. 4 presents five categories of metrics and optimization goals that we use to vary the problem definition in this way.

Specifying the set $\mathscr{P}$ as part of the input is motivated by the possibility of having a predefined subset of *probing nodes* $R \subseteq V$ that are the allowed sources and destinations of the probing messages (or *probes*). If $u \in R$ may send a probe to $v \in R$, the path that this takes may be determined by the underlying routing system; this path (but not necessarily *all* paths from u to v in G) would then be included in $\mathscr{P}$.

Some additional requirements on the structure of $\mathscr{P}$ that are of interest below (but which are not, in general, assumed to hold) are: (1) $\mathscr{P}$ contains a path between every ordered pair of distinct nodes in $R \times R$; (2) $\mathscr{P}$ contains only one path for each ordered probing-node pair (u,v) (but the path for (v,u) need not be its reverse);[1] (3) $\mathscr{P}$ corresponds to destination-based IP routing, *i.e.*, the various paths in $\mathscr{P}$ must not be inconsistent with their being parts of routing sink trees (in particular, if $\mathscr{P}$ contains multiple paths with destination $v \in R$ that also include the node $u \in V$, then all of these paths coincide between u and the destination v); (4) $\mathscr{P}$ corresponds to paths that are consistent with a *coherent* cost function [12] used

[1]This assumption precludes multipath routing and load balancing due to traffic engineering, but it permits our analysis to assume, with certainty, which links are covered by a given probe.

as the basis for route selection.[2] We will explicitly highlight when any of these assumptions hold and when those assumptions impact our results.

A variation of this problem (*e.g.*, similar to that considered in [5]) is one in which the probing nodes R are not predefined but must be chosen, perhaps to optimize some metric. This is a special case of the problem above: let $R = V$, let $\mathscr{P}$ contain viable probing paths between all ordered pairs of nodes, and additionally require that the union of source and destination nodes over all sets of paths output by the algorithm (*i.e.*, the target R) meets the desired constraints.

In some cases, it makes sense to perform some precomputation on the input (like the first step of the two-phase approach in [5]) to limit $\mathscr{P}$ to a selected subset of paths from the original network as a starting point for designing a probing algorithm. One useful property (that we use in some analyses below) for such a set of paths is the following.

DEFINITION 3.2 (MINIMAL SET OF PATHS). *A set of paths $\mathscr{P}$ is* minimal *with respect to a set of edges $F \subseteq E$ if every path in $\mathscr{P}$ must be probed to cover all edges in F;* i.e.*, every $P \in \mathscr{P}$ traverses at least one link that does not appear in any of the other paths in $\mathscr{P}$.*

This is not a strong property: given any set of paths whose union covers all the edges in F, we may simply go through the paths in some arbitrary order and discard any whose edges all appear in previously seen paths. It does not guarantee that the resulting set is of *minimum* possible size; however, it lets us assume that we can always produce a minimal set and therefore decouple the problem of finding the "best" (*e.g.*, minimum-sized) minimal set from the problem of designing an algorithm to probe all the paths in a given minimal set.

4. METRICS AND OPTIMIZATION GOALS

We have five categories of metrics by which we analyze probing algorithms. Different applications may lead to a focus on different combinations of these (and perhaps other) metrics.

4.1 Number of probes or probing nodes used

This family of metrics corresponds to optimization goals that appear most often in previous work to represent low overhead or resource minimization, *e.g.*, [5, 18].

Let $X = \cup_t f(t)$ be the set of paths probed[3] by algorithm f, and let $N \subseteq R$ be the set of probing nodes used by the sequence; these are endpoints of paths in X. (Note that some previous work, *e.g.*, [2, 5], has separately categorized sources and destinations of probing paths.) Then, $|X|$ (the number of probing paths, which is the number of distinct probes needed) and $|N|$ (the number of probing nodes) serve as natural metrics for f, with the corresponding possible optimization goals: (1) require that f minimize $|X|$; and/or (2) require that f minimize $|N|$. We explore the interaction among these goals in Secs. 5 and 6.1.

4.2 Probing frequency

This family of metrics formalizes the flexibility in probing-algorithm design explored in [3, 18], *i.e.*, that dividing probes over time can reduce overhead while retaining effectiveness as long as measurements are collected "often enough." The following ways to characterize the frequency at which links are probed attempt to make this notion precise so that we can better describe the tradeoffs between frequent and infrequent probing. Of course, it is trivial to minimize the delay between measurements if no additional restrictions are placed on the algorithm: just probe every path at every timestep. We investigate the difficulty of combining this goal with others in Sec. 6.

Probing probability: Assume there exists a parameter k that describes a length of time (in number of timesteps). We can then measure the minimum probability, over all time windows of length k (intervals of consecutive timesteps of length k), that a given link is probed by f. For a corresponding optimization goal, assume that each link $\ell \in E$ is assigned a parameter p_ℓ, and require that, for every time window of length k and for every link ℓ, f probes ℓ with probability bounded below by p_ℓ. A simpler version of this goal (especially appropriate for deterministic f) is to ensure that a subset of links is probed with some regularity (or at all), *i.e.*, there exists a subset of "links of interest" $F \subseteq E$ such that $p_\ell = 1$ for $\ell \in F$ and $p_\ell = 0$ for $\ell \notin F$. We note that certain hardness proofs below require setting $F = E$.

Probing delay: We can measure the (average, maximum, *etc.*) time between probes of a given link. For an optimization goal, assume that every link ℓ has a parameter k_ℓ; we can then require that, for all links ℓ and for all times t, the expected time it will take to next probe ℓ after t is bounded above by k_ℓ. A simpler condition on f would be to require that the expected time to probe every link is bounded (*i.e.*, finite).

4.3 Load on links

In order to better understand the tradeoffs between additional probing and other goals, we want to characterize the effect of generated probing traffic on network resources. For example, an algorithm that probes on all paths in every timestep will minimize measurement delay on all links covered but uses a lot of network traffic.

The following two metrics focus on per-link effects of probing traffic (in contrast to the more global measures in Sec. 4.1 and in [5, 18]). We can measure, for some time parameter k and for every link $\ell \in E$: (1) the (expected, maximum, *etc.*) number of times that probes traverse ℓ over all time windows of length k; (2) the probability that ℓ is probed more than once (or more than some threshold L_ℓ) in a time window of size k. Analogously, we can impose as a requirement on a probing algorithm that, given parameters k and per-link load limits L_ℓ, the expected number of times ℓ is probed in any time window of length k is at most L_ℓ.

We note that it is also trivial to minimize per-timestep link loads if no additional restrictions are placed on the algorithm: simply iterate through probing paths, probing one at each timestep. This, of course, may probe many links with low frequency; if link load must be kept at a minimum while simultaneously increasing probing frequency, the design problem becomes difficult (see Sec. 6).

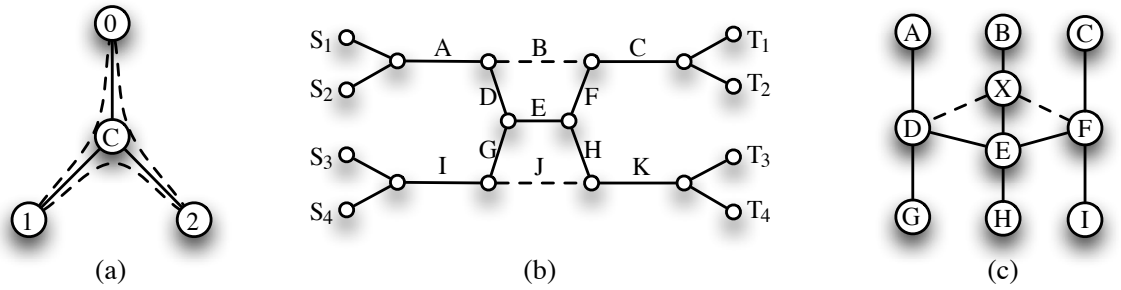
4.4 Load on nodes

The probing nodes, *i.e.*, source and destination nodes for messages sent to acquire measurements, incur some computational load for having to generate and process probe traffic. Thus, we may be interested in evaluating how many requests a probing algorithm imposes on probing nodes.

More precisely, we can measure the number of probe messages for which each node $r \in R$ is a source or destination in a time window of some size k; as a design goal, we may require that an algorithm involve each node r as a probing source or destination at

[2] *I.e.*, each directed edge is assigned a cost—with negative-cost edges allowed as long as all directed cycles have positive cost—and the lowest-cost directed path is chosen for routing.

[3] For a randomized algorithm f, we take this to be the set of paths that f might possibly probe.



(a) (b) (c)

Figure 1: Example networks illustrating various tradeoffs among probing-algorithm design goals.

most n_r times in a time window of size k, given input parameters n_r (for each node r) and k. We investigate this type of goal in Sec. 6.2.

There may also be some computational load incurred by non-probing nodes while forwarding probing packets. This load may not be accurately captured by link-load metrics used in previous work; *e.g.*, in a star topology, the central node is involved in the communication of a probing packet for every link, even if it is not a probing node itself. Thus, we may want to measure the total load on non-probing nodes and perhaps try to bound it when designing an algorithm.

It is possible to transform networks so that link-load metrics may capture load on nodes: expand each node v into a pair of connected nodes (i, j) such that all paths through v now traverse i then j (and never in reverse), connecting i and j to the expanded versions of the neighbors of v in the original network; then, the number of probing packets traversing v corresponds to the number traversing the link (i, j) in the transformed network.

4.5 State complexity

Different probing algorithms may require different amounts or kinds of internal state to be implemented, as discussed in Sec. 1.1. The algorithms that we consider will typically use some combination of fixed-length values, *e.g.*, constants and counters, but they might also keep track of dynamic-length values, *e.g.*, sets of probing paths (or sequences of such sets). These might be global values, or they may be stored at each network node, stored for each network link, or stored for each probing path. We are able to obtain at least partial comparisons among different algorithms' levels of complexity and implementation difficulty by considering the amount and type of state required by their implementations.

5. TRADEOFFS

In this section, we use three example networks (shown in Fig. 1) to illustrate tradeoffs among some of the probing-algorithm design goals discussed in Sec. 4. In particular, we demonstrate that optimization of one metric may come at the expense of another metric. Although the examples are small, toy networks, it is possible to produce networks of larger size or of more realistic topologies that have similar properties. However, even these small examples are enough to show the existence of networks on which it may be impossible to choose a probing strategy that achieves multiple optimization goals.

5.1 Probing frequency and number of probing paths used

Our first example is shown in Fig. 1(a), a star with four nodes. Here, the possible probing paths (the set $\mathscr{P}$) are all three of the paths between two degree-1 vertices, shown as dashed lines. Suppose we place a link-load bound of one probing message per timestep for every link. Then, at most one path in $\mathscr{P}$ can be probed in any timestep, and it is impossible to probe every link in every timestep. However, probing any two distinct paths from this set will cover all three links in the network over two timesteps; this can be done using two different approaches. The first approach probes all three paths (at different times), *e.g.*, sending a probe from i to $i+1$ (all indices considered modulo 3) when time $t \equiv i \mod 3$. The second approach selects any two of the paths (which necessarily have a link in common) and then alternates between which one is used to probe (*e.g.*, sending a probe from node 2 to node i when time $t \equiv i \mod 2$).

Under each of these approaches, every link in the network is probed at least once every two timesteps, and, in each timestep, some link is not probed. In the first approach, every link is uniformly not probed every third timestep; in the second approach, one link is probed every timestep while the other two are probed at alternating timesteps. Uniformity of probing frequencies across all links (which may be desirable if, *e.g.*, the different network links are of equal importance) comes at the cost of using all three possible probing paths. In the second approach, the set of paths that used for probing is smaller (size 2).

This example easily generalizes to a star on $2k+2$ nodes (whose degree-1 nodes we label $0, 1, \ldots, 2k$). With $2k+1$ degree-1 vertices, the algorithm can probe every link $2k$ out of every $2k+1$ timesteps if uniformity is desired. At the opposite extreme, the algorithm may probe $1/3$ of the links in every timestep (if $2k+1$ is also a multiple of 3) and the other $2/3$ of the links in half of the timesteps. Both of these approaches still ensure that no link is probed less frequently than every other timestep and that no link carries more than one probe per timestep.

5.2 Balancing load and frequency among various links

Our second example, shown in Fig. 1(b), illustrates the potential benefit that an increase in the link-load limit on just a single link can have with respect to the load imposed on other links, even while also requiring a particular probing frequency for links in the network.

Note that, in the diagram in Fig. 1(b) for this example network, the links, not nodes, are labeled; furthermore, assume that B and J represent groups of links (*i.e.*, paths containing many links). In this network, consider the probing paths S_1ABCT_1, $S_2ADEFCT_2$, $S_3IGEHKT_3$, and S_4IJKT_4. Suppose that links A, C, I, K all should be probed at every timestep, while the links in B, J, and the remaining links in the network need to be probed much less frequently. If link E has a link-load limit of 1 probe per timestep, then in order to probe A, C, I, K, we must probe one of B or J in every timestep,

exceeding its target probing frequency and unnecessarily imposing probing load. By increasing the limit on link E to 2, the paths $ADEFC$ and $IGEHK$ can sometimes be used simultaneously instead, equalizing the burden among E, the links in B, and the links in J. Thus, allowing E to occasionally have a load of 2 can reduce the number of extra[4] probes on links over some number of timesteps.

5.3 Number of probing paths and number of probing nodes

Our third example, shown in Fig. 1(c) (ignoring for now the dashed links), shows a network for which the set of paths minimizing the number of probing paths used has a different size than the set of paths minimizing the number of probing nodes used. Let $\mathscr{P}$ contain the paths ADG, $BXEH$, CFI, $ADEH$, $CFEH$, DEF; the node-minimizing set uses all of these paths except DEF, involving nodes A, B, C, G, H, I; the path-minimizing set uses only ADG, $BXEH$, CFI, DEF requiring the use of probing nodes A, B, C, D, F, G, H, I.

We note that, in this example, $\mathscr{P}$ does not contain a probing path between every pair of nodes (*e.g.*, between A and C). We may modify this example by adding the dashed links and adding to $\mathscr{P}$ a single probing path between every pair of nodes—all paths except DEF, $ADEH$, $CFEH$ use the new dashed links through X. The node-minimizing and path-minimizing sets are the same as before; in particular, different sets minimize the number of probing nodes and the number of number of probing paths.

However, in the modified example, the paths in $\mathscr{P}$ no longer form a routing tree (*i.e.*, the paths $CFEH$ and $IFXEH$ both have destination H and contain intermediate node F but different subpaths from F to H). We conjecture that there is a set of probing paths covering all network links that minimizes both the number of paths needed to do this and the number of probing nodes that must be used to do this, under the following assumptions: (1) the union of all probing paths covers all links in the underlying network; (2) there is a probing path between every two probing nodes; and, (3) for every probing node, the probing paths to that node form a routing tree.

6. HARDNESS AND APPROXIMATION

6.1 Minimizing number of probes/nodes used

Suppose we want to design a probing algorithm that covers a target set of links $F \subseteq E$ in the network, but does so either by using as few probing paths or by using as few probing nodes as possible. The results below demonstrate that these probing-design problems are computationally difficult.

THEOREM 6.1. *Given an instance of the probing-algorithm design problem along with a target set of links $F \subseteq E$ to probe, computing a set of probing paths of minimum size that covers F is NP-complete.*

PROOF. It is straightforward to show that the problem is in NP. To show NP-hardness, we give a reduction from Minimum Set Cover (MSC) that is a modification of the reduction from MSC to the Path Hitting problem in [19]. Begin with an instance of MSC: there is a universe of elements U and subsets of those elements $S_1, S_2, \ldots, S_n$; our goal is to choose a minimum-size collection C of subsets such that the union of the subsets selected for C includes all the elements in U.

Create a probing network based on the MSC input as follows: Let G be a bipartite graph with vertices $x_1, \ldots, x_m$ and $y_1, \ldots, y_m$, where $m = |U|$. For each $i = 1, \ldots, m$, add an edge between x_i and y_i, and let F consist of these edges. These edges correspond to the elements of U, which we assume can be ordered $1, \ldots, m$. For each subset S_j, create an ordered list of indices $k_1, \ldots, k_{S_j}$ that correspond to the order on U. Then, for every adjacent pair of elements k_a, k_{a+1} in the ordered list, create an edge from y_{k_a} to $x_{k_{a+1}}$. Thus there is a path connecting the corresponding (x_i, y_i) edges in G for each the subsets given in the MSC problem, each of which is added to the set of probing paths $\mathscr{P}$. With this construction, a minimum-size set of probing paths selected from $\mathscr{P}$ that covers F maps directly to a minimum-size set cover. □

The above argument shows that designing a probing algorithm that minimizes the number of probing paths used to cover a target set F of network links is computationally intractable. A similar argument can be used to show that minimizing the number of probing nodes is, too: In the reduction above, simply modify G so that, for any path from x_i to y_j corresponding to a subset in the MSC problem, there is an extra vertex adjacent to x_i and an extra vertex adjacent to y_j that serve as the probing source and destination nodes for that path; furthermore, these extra vertices should be unique for each subset in the MSC problem. Then, the number of probing nodes used to probe F corresponds exactly with the number of subsets that would cover U.

We note that our reduction does not construct a set of probing paths consistent with a coherent cost function. The hardness of finding minimizing sets when $\mathscr{P}$ is consistent with a coherent cost function remains an open question.

Because these probing-design problems are special cases of Minimum Set Cover, the standard MSC approximation algorithms apply to finding probe- and node-minimizing sets of probing paths (just as they do to the related Path Hitting problem [19]).

6.2 Limiting work of probing nodes

Suppose we want to limit the load of each probing node to one probe per timestep; in other words, we insist that each probing node send or receive at most one probing packet per timestep. This would mean the maximum number of paths we could probe per timestep is $\lfloor |R| / 2 \rfloor$, although achieving this upper bound might depend on the topology and other constraints. To minimize the number of timesteps needed to probe all links with this node-load constraint, we want the algorithm f to decompose the probing paths into a minimum-length sequence of sets such that no two paths in a set share a probing node. Unfortunately, this design problem is computationally intractable.

THEOREM 6.2. *Designing a probing algorithm that minimizes the number of timesteps in which all possible links are probed while limiting the probing-node load to one probe per timestep is NP-complete.*

PROOF. To simplify the proof, we use the decision version of the probing-design problem in the theorem statement, *i.e.*, designing an algorithm in which the number of timesteps is bounded by some parameter k. The reductions between the decision and optimization versions of the problem are straightforward. It is also easy to see that the problem is in NP.

To show NP-hardness, we give a straightforward reduction from Minimum Edge Coloring (MEC), which is NP-complete [14]. Begin with any instance of MEC: the input is a graph $G = (V, E)$ and

[4] *I.e.*, the sum over all links of the number of probes traversing the link minus the number of probes that are required to traverse that link based on its target frequency.

a parameter k, and we are to decide if it is possible to partition E into k disjoint sets $E_{1,\ldots,k}$ such that for each set E_i, no two edges in E_i share a common endpoint. Now, let G be the probing network; let the set R of probing nodes be equal to V, and let the set $\mathscr{P}$ of probing paths be equal to E (so that each probing path consists of a single edge between nodes in the network). Then there is a sequence of sets of probing paths of length k covering the entire network in which each probing node is a source or destination of at most one probing path in each set if and only if the original graph G is k-edge-colorable. □

We note that this reduction creates an instance of the probing-design problem in which the constructed probing-path set $\mathscr{P}$ has no link overlap among paths; in particular, it is minimal.

6.3 Minimizing probing delay with a restriction on link load

As noted in Sec. 4.2, it is trivial to minimize the number of timesteps between probes along every network link if we do not consider any other metrics (simply by probing all paths at every timestep). If we additionally bound the number of probes that may traverse a link in any timestep, then it becomes interesting to ask whether it is possible to probe along all paths in a given probing-path set using at most k timesteps. (In particular, if the probing-path set is minimal, we would need to probe every path to cover all links of interest with delay at most k).

6.3.1 Basic hardness result

We will now show that answering the above question is computationally intractable. We start with the following definition:

DEFINITION 6.3 (L-STRONG k-COLORABLE HYPERGRAPH). *A hypergraph $\mathscr{H}$ is* L-strongly k-colorable *if there is a k-coloring of the vertices of $\mathscr{H}$ such that no edge of $\mathscr{H}$ has more than L vertices of any one color.*

The L-strong k-colorability problem for hypergraphs is computationally difficult to solve exactly and to approximate. The following theorem provides reductions, in both directions, between this problem and the probing-design problem considered in this subsection (minimizing probing delay subject to a link-load restriction). This result and the reductions in its proof are used to justify the computational intractability of exactly or approximately solving the probing-design problem.

THEOREM 6.4. *Polynomial-time reductions exist between the following two problems: (1) deciding whether a hypergraph is L-strongly k-colorable; and (2) deciding, for a given probing-design-problem input, whether all the probing paths in $\mathscr{P}$ can be probed within k timesteps without imposing a load of more than L on any link in any timestep.*

PROOF. *Reduction from (2) to (1):* Consider a probing problem consisting of a graph $G=(V,E)$ and the set $\mathscr{P}$ of probing paths. Construct a hypergraph $\mathscr{H}$ as follows: The vertices of $\mathscr{H}$ correspond (bijectively) to the elements of $\mathscr{P}$. For each edge $e \in G$, add as an edge in $\mathscr{H}$ the set of those vertices in $\mathscr{H}$ that correspond to the paths in $\mathscr{P}$ that include the edge e. Viewing the colors of vertices in $\mathscr{H}$ as assignments of elements of $\mathscr{P}$ to timesteps, the colorings of $\mathscr{H}$ with i colors in which j is the maximum number of vertices in any one edge with a single color bijectively correspond to the assignments of elements of $\mathscr{P}$ to i timesteps in which the maximum load on any edge in any timestep is j. In particular, deciding L-strong k-colorability of $\mathscr{H}$ decides whether $\mathscr{P}$ can be probed in k timesteps without an edge load exceeding L in any timestep.

Reduction from (1) to (2): Conversely, consider a hypergraph $\mathscr{H}=(V_{\mathscr{H}},E_{\mathscr{H}})$ and fixed values of k and L; we wish to decide whether $\mathscr{H}$ is L-strongly k colorable. Fix an ordering $\hat{e}_1,\ldots,\hat{e}_m$ of the edges of $\mathscr{H}$. Construct a network with m disjoint links $e_1,\ldots,e_m$; for the purposes of our construction, make these directed (we will ignore these directions when probing). For each vertex $v \in V_{\mathscr{H}}$ that is contained in at least one hypergraph edge, let ℓ_v be the number of hypergraph edges that contain v, and add to the network the ℓ_v+1 nodes s_v, t_v, and $\{x_{(v,i)}\}_{i=1}^{\ell_v-1}$.

Add to the network $2\ell_v$ links as follows: Let the hyperedges that contain v be (in order) $\{\hat{e}_{i_j}\}_{j=1}^{\ell_v}$. Add a link from s_v to the tail of e_{i_1}, and add a link from the head of $e_{i_{\ell_v}}$ to t_v. For $j=1,\ldots,\ell_v-1$, add a link from the head of e_{i_j} to $x_{(v,i_j)}$ and a link from $x_{(v,i_j)}$ to the tail of $e_{i_{j+1}}$.

Once this network is constructed, let the set $\mathscr{P}$ of probing paths contain the $|V_{\mathscr{H}}|$ paths that were implicitly constructed above; the probing path corresponding to a hypergraph vertex v is the (undirected) path whose links are: the link from s_v to the tail of e_{i_1} (where i_1 is as above for this particular v), the link e_{i_1}, the link from the head of e_{i_1} to $x_{(v,i_1)}$, the link from $x_{(v,i_1)}$ to the tail of e_{i_2}, *etc.*, until the link from the head of $e_{i_{\ell_v}}$ to t_v. Note that only one path in $\mathscr{P}$ goes through each network node $x_{(v,i)}$ and the two links incident upon it.

If we have an assignment of probing paths to timesteps $1,\ldots,t$, we may view that as a t-coloring of the vertices of the original hypergraph; vertex v is assigned the color i if and only if its corresponding probing path is probed in timestep i. If the maximum number of probing paths that traverse the network link e_j in any timestep is M, then $\hat{e}_j$ has M, but no more, vertices of one color. (If there are vertices in $\mathscr{H}$ that did not belong to any edge, these can be colored arbitrarily without affecting this. Similarly, if there are empty edges in $\mathscr{H}$, there will be some isolated links in the network; however, these are not part of any probing path.) In particular, an assignment of probing paths to k distinct timesteps such that no link ever carries more than L probes in a timestep is possible if and only if $\mathscr{H}$ was L-strongly k-colorable. Note that the resulting network was constructed with $2\,|E_{\mathscr{H}}| + \sum_{v\in E_{\mathscr{H}}} d(v)$ nodes (where $d(v)$ is the number of hyperedges containing v) and $|E_{\mathscr{H}}| + 2\sum_{v\in E_{\mathscr{H}}} d(v)$ links. □

Because hypergraph L-strong k-colorability is NP-complete [1], Thm. 6.4 shows that minimizing probing delay subject to a link-load restriction is NP-complete as well. We note that the result applies to the general probing-design problem; it is possible that assumptions on the structure of the probing-path set $\mathscr{P}$ may admit a feasible solution.

6.3.2 Hardness of approximation

Approximating a minimum-delay sequence of probes subject to a link-load restriction is also computationally difficult. Numerous results (*e.g.* [15]) have established the hardness of approximate hypergraph coloring. For each reduction in the proof of Thm. 6.4, note that a solution of size k' in one problem corresponds to a solution of size k' in the other problem. Thus, approximation results for hypergraph coloring also carry over to the probing-design setting.

6.3.3 Link load at most one

Consider the special case of restricting the per-link probing load to one message per timestep. Given the hardness results above, one might attempt to use a greedy bin-packing approach to approximate an optimal selection of probing paths; intuitively, the idea is

to "fill" a timestep with the maximum number of concurrent, disjoint probing paths possible before proceeding to the next timestep.

Unfortunately, it turns out that the component problem in this approach, *i.e.*, finding the maximum number of concurrent, disjoint probes, is also computationally intractable.

THEOREM 6.5. *Deciding whether it is possible to probe at least k paths from a given set $\mathscr{P}$ simultaneously when links can carry at most 1 probing message is NP-complete.*

PROOF. It is obvious that the probing decision problem is in NP. To show NP-hardness, we give a straightforward reduction from Independent Set. Given a graph with $|V|$ vertices and $|E|$ edges (sorted in an arbitrary order), construct a network as follows: Create $|V|$ disjoint paths, each with $3|E|$ links. Iterate through the edges of the graph ($i = 0, \ldots, |E|-1$). For each edge i, let x and y be its endpoints in the graph, and identify the $(3 \cdot i + 2)^{\text{th}}$ edges in the paths corresponding to x and y so that these paths use the same edge in the $(3 \cdot i + 2)^{\text{th}}$ position. Thus, the paths corresponding to x and y may be probed in the same timestep if and only if x and y are not adjacent in the original graph. In particular, the original graph has an independent set of size k if and only if at least k of the paths in the network constructed can be probed in the same timestep without sending more than one probe across any single link. □

7. RANDOMIZED PROBING

Our analysis thus far has dealt with deterministic approaches to probing. Because, as we have shown, it is computationally difficult even to approximate an optimal probing algorithm for certain combinations of goals, we now turn instead to randomized probing algorithms. Our focus here will not be on absolute guarantees but on identifying what we can expect (on average) from a baseline class of randomized algorithms.

From an analytic perspective, the class of randomized algorithms we consider can be guaranteed to provide finite expected delay[5] in detecting a network abnormality. Additionally, we can bound the delay more precisely with more assumptions on the probing probabilities used in the algorithm. This is in contrast to previous work (*e.g.*, [3]) that used randomized probing approaches.

However, randomized path selection makes it difficult to disentangle the causes of poor algorithm performance. The probing-design problem is different than problems considered in existing analytical work on concurrent path selection in networks (*e.g.*, [11]) because probes must follow paths determined by an underlying routing system on the network. For example, because it is reasonable to assume that we can only probe the whole paths that appear in the path set (and not their proper subpaths that do not appear in the set), we do not have link- or node-level choice over what gets probed. If we pick paths randomly, it is difficult to say if link- or node-level load is high because certain links or nodes appear on many paths, or whether a particular run of the algorithm was "unlucky" in its random choices. Thus, in order to get more insight into how these randomized algorithms perform with respect to the metrics introduced in Sec. 4, we present simulations of the performance of algorithms from this class.

7.1 Analytic results

A member of the class of randomized algorithms that we consider is described by a set of probabilities $\{g_t(P)\}_{t \geq 1, P \in \mathscr{P}}$. Here, t ranges over all possible timesteps, and P ranges over all probing paths; $g_t(P)$ is the probability that P is probed at time t. (From such a description, we can obtain the description of a probing algorithm in the sense of Def. 3.1.) In particular, the decisions to probe paths are made independently at each timestep, so nodes do not need to retain state. (We discuss below some possibilities for using state, the nature of which may serve as another metric.)

We now investigate the metric of expected probing delay (or expected probing frequency) under various assumptions about the sequence of path-probing probabilities used by the algorithm.

7.1.1 Lower-bounded probabilities

To begin, assume that the path-probing probabilities are strictly positive for all timesteps and for all paths in the probing-path set. We then obtain the following result.

THEOREM 7.1. *If there exists some ε such that $0 < \varepsilon < 1$ and $g_t(P) \geq \varepsilon$ for all timesteps t and for all paths P, then the expected probing delay for any path is finite (in particular, bounded above by $1/\varepsilon^2$).*

PROOF. Let $E_t(P)$ be the expected number of timesteps following timestep t until path P is probed. We may write this as:

$$E_t(P) = 1 \cdot g_{t+1}(P) + 2 \cdot (1 - g_{t+1}(P))(g_{t+2}(P)) + 3(1 - g_{t+1}(P))(1 - g_{t+2}(P))(g_{t+3}(P)) + \ldots \quad (1)$$

In the above expression, we know that all the $g_t(P)$ probabilities are bounded below by ε; thus, every $1 - g_t(P)$ term is bounded above by $1 - \varepsilon$. Furthermore, because each $g_t(P)$ is a probability, we know that it is bounded above by 1. Using these bounds (and the fact that $\varepsilon < 1$), we may bound the expression in (1) as:

$$\begin{aligned} E_t(P) &\leq 1 \cdot 1 + 2 \cdot (1-\varepsilon)(1) + 3(1-\varepsilon)^2(1) + \ldots \\ &= \sum_{i=0}^{\infty} i \cdot (1-\varepsilon)^{i-1} \\ &= \frac{1}{\varepsilon^2}. \end{aligned}$$

Because this calculation is independent of the starting timestep t and the choice of path P, this means that the expected delay for any path P in the network is finite and bounded above by $1/\varepsilon^2$. □

By extension, for finite networks, the above result implies that the time until all paths in the network are probed, regardless of the starting timestep, is also finite.

If we further assume that $g_t(P) = \varepsilon$ for every timestep t and every path P, then we can rewrite the expression in (1) exactly as:

$$\begin{aligned} E_t(P) &= 1 \cdot \varepsilon + 2 \cdot (1-\varepsilon) \cdot \varepsilon + 3 \cdot (1-\varepsilon)^2 \cdot \varepsilon + \cdots \\ &= \sum_{i=0}^{\infty} i \cdot (1-\varepsilon)^{i-1} \cdot \varepsilon \\ &= \frac{1}{\varepsilon}. \end{aligned}$$

This means that, no matter the initial timestep t or path P, the number of expected number of timesteps to probe path P is $1/\varepsilon$. Because we know that each link in the network is covered by some path in the probing-path set, we can be confident that the expected probing delay of any link is bounded by $1/\varepsilon$.

It is important that the path-probing probabilities are nonzero; without this, as we discuss in the next section, it is possible that some links in a network might be forever ignored by the probing algorithm.

[5] Recall from Sec. 4.2 that the *probing delay* of a link is the number of timesteps between consecutive probes of that link. In the randomized setting, there may not be a fixed pattern describing when a link is probed; thus, we focus on the expected probing delay given the sequence of path-probing probabilities used by the algorithm.

7.1.2 Example algorithm with links ignored

The randomized algorithm in [3] probes k paths selected randomly from the K paths of largest dynamic weight, where the dynamic weight of a path is the sum of the dynamic weights of constituent links. Initially, the dynamic weight w_ℓ of each link ℓ is 0. In each timestep after probing, weights are updated as follows: the weight of every link just probed is set to 0; the weight of every other link is updated to $w_\ell' = \min\left(I_\ell, w_\ell + \frac{I_\ell}{(N-1)/k}\right)$, where N is the total number of paths that can be probed and I_ℓ is a link-specific importance parameter. Here, we provide an example in which the algorithm never probes some particular links.

Let $k = K = 1$. Consider a network with three disjoint paths P_1, P_2, and P_3, and let $I_\ell = 1$ for each link ℓ in the network. Let each path P_i have length L_i (number of links), such that $L_1 < L_2 < L_3$ and $L_2 > L_1 \cdot (N-1)/k$. Suppose P_3 is probed in some timestep. All links in P_3 have their dynamic weight set to 0, making P_3 have path weight 0. The maximum dynamic weight of links in P_1 is 1, so the maximum dynamic weight for P_1 is L_1. The minimum dynamic weight for each link in P_2 is $k/(N-1)$ (in the case that P_2 was probed in the previous timestep), so the minimum path weight for P_2 is $L_2 \cdot k/(N-1)$. However, because $L_2 > L_1 \cdot (N-1)/k$, $L_2 \cdot k/(N-1) > L_1$, and as a result P_2 is guaranteed to be probed in the next timestep. Similarly, for the following timestep, since $L_3 > L_2$ and consequently $L_3 \cdot k/(N-1) > L_1$, P_3 will be probed next. Thus, once P_3 is probed, P_2 and P_3 will be alternately probed in the following time steps, and P_1 will never be probed.

7.1.3 State complexity and probing frequency

In Sec. 4.2, we considered the optimization goal of requiring that our probing algorithm, for every time window of length k and for every link ℓ, probes ℓ with probability bounded below by p_ℓ, which is a per-link parameter.

We note that achieving a probability $p_\ell = 1$ is impossible with a "purely probabilistic algorithm" of the type described above (meaning that all paths are probed with probabilities strictly less than 1). In other words, to *guarantee* that certain links are probed for every window of length k, it is not enough to leave probing completely to chance.

To address this, we can use a modification to the randomized approach such as the following: Each timestep, probe a random set of paths combined with the set of paths not probed within the last k timesteps (for some parameter k). This ensures that some "catch-up" probing occurs to account for random selection of paths that were probed in the previous k timesteps.

The downside to this workaround is that additional state is required by the probing algorithm at each node; in particular, source nodes are required to maintain the last time that each potential destination was probed. We note that, without synchronization, using this "last time probed" value may not provide an actual guarantee of timely measurements; thus, additional global coordination may be necessary to achieve stronger algorithm guarantees.

7.1.4 Restriction to one probing path per timestep

Suppose we wish to restrict the number of probing paths per timestep to 1. A randomized probing algorithm with this restriction may proceed as follows: in each timestep, choose one of the paths from the probing-path set according to some probability distribution. Such an algorithm enforces very low overhead (in terms of the metrics of link load, node load, and number of probing nodes and paths used), perhaps at the cost of expected probing delay.

To analyze the expected probing delay of such an algorithm, we relate the probing algorithm to the classic *coupon-collector's problem* [17]: There are n different coupons, each of which can be drawn with equal probability; if, every day, one coupon is drawn and then returned to the pool of coupons, in how many days will all the coupons have been drawn at least once? The answer is known to be $O(n \log n)$ days.

This result can be fashioned into a probing-frequency result. If we have n paths, and, in each timestep, probing each path is equally likely, then the coupon-collector's result states that $O(n \log n)$ timesteps are needed before every path is probed at least once.

An extension on the coupon collector's result is the case in which coupons are drawn with differing probabilities. That is, every day a coupon is drawn with replacement (as above), however, each coupon i has probability p_i of being drawn (instead of equal probabilities $1/n$ as in the original version). In the probing-algorithm context, we still assume that there are n different paths, but each path i is probed in a given timestep with probability p_i. Berenbrink and Sauerwald [6] show that the bound on the number of timesteps needed to probe every path is $O(\log \log n \cdot \sum_{i=1}^{n} i \cdot 1/p_i)$.

To find a bound on this value, let p_e be the minimum of the probabilities p_i. We use that $\sum_{i=1}^{n} i \cdot 1/p_e = 1/p_e \cdot H_n$, where H_n is the nth harmonic number. Given the asymptotics for the harmonic numbers, we have the bound that $1/p_e \cdot H_n = O(1/p_e \cdot \log n)$. Thus, the final upper bound on expected probing delay would be $O(\log \log n \cdot 1/p_e \log n)$.

7.2 Simulation results

In this section, we present empirical analysis of the basic randomized probing algorithm discussed above with an expected delay bound $1/\varepsilon$, for some path-probing probability ε ($0 < \varepsilon < 1$).

In this basic approach, we assume the existence of this global probability parameter p and, in each timestep, each path is probed with probability p. (Below, we show the effect of different values of p on the algorithm's performance metrics.) To simulate and analyze the behavior of this algorithm, we wrote a set of simulation procedures in Python 2.7; our routines use data structures and algorithms in the NetworkX [13] graph-theoretic library to represent the network and to compute a network routing (used to induce a probing-path set for a network).

The simulation inputs (each a network topology and set of probing paths) include some network topologies that are randomly generated[6] and others taken from the Internet Topology Zoo repository [16]. To compute path-set variations for each network topology, we randomly selected a subset of the network's nodes as probing nodes (potential source and destination nodes for probing traffic); we then computed shortest-path routes on the entire network and filtered out paths that did not start or end at one of the probing nodes in the selected subset. These variations were created using probing-node subsets ranging in size from 33–100% of the whole set of network nodes. In all, the simulation results that we present capture performance analysis of 6515 inputs to the basic randomized algorithm.

7.2.1 Probing delay

It is obvious that the higher the probability parameter p used to determine whether a path is probed or not in a given timestep, the more frequently network links will be probed and the shorter the amount of time between probes of that link (*i.e.*, the link's probing delay). However, because a given link may appear on any number of the potential probing paths, depending on the underlying routing system, the exact relationship between probability and link probing delay is less clear.

[6] We used the fast version of the Erdős-Rényi graph generator [4] implemented in NetworkX [13].

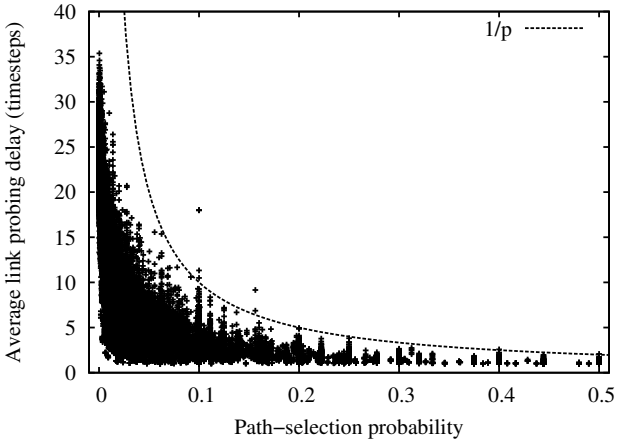


Figure 2: Relationship between the number of timesteps between probes of a link (averaged over all links) and the path-selection probability used for probing.

The basic randomized approach we tested avoids maintaining state for coordinating probing-path selection across nodes in a given timestep. We ran over 40 trials for each input combination of a network and a path-set, using different path-selection probabilities across those trials (ranging from near 1% to 50%). Each trial involved running 50 timesteps of the basic randomized probing algorithm. In each timestep, each probing node independently decided to send a probe to each possible destination with the trial's assigned path-selection probability p.

Figure 2 shows the results of this simulation. The horizontal axis shows the path-selection probability p for the trials, while the vertical axis shows the average link probing delay for that trial. This delay was computed by averaging, over all links, the mean amount of time between probes of a link during that trial. Even with links appearing in multiple probing paths, the probing frequency for a link seems to be correlated with $1/p$, where p is the path-selection probability; however, as expected from the results of Sec. 7.1.1, $1/p$ does serve as a rough upper bound for probing delay.

7.2.2 Link load

Given that each path is probed or not probed independently of all the other probing decisions, it is possible that the basic randomized algorithm might probe all of the probing paths in a single timestep. While this is a worst-case scenario in terms of link load, there is still a reasonable chance that the simple approach produces unacceptably (or at least undesirably) high load on some links.

Thus, we attempt to quantify the effect of path-probing probability on link load, again realizing that the appearance of links on multiple probing paths chosen independently at random makes the connection potentially unclear. Figure 3 shows the results of analyzing link load from the simulation trials described above in Sec. 7.2.1. The horizontal axes of both plots show the path-selection probability p, while the vertical axes show the average load on links (computed by averaging, over all links, the mean load over the 50 timesteps in the trial).

Figure 3(a) shows results from simulation trials where p was chosen from the set $F = \{0.1, 0.2, 0.3, 0.4, 0.5\}$, regardless of topology. As expected, load grows with increased path-selection probability; moreover, the results suggest that it can grow to unacceptably high levels. For any given probability in F, trials showed a variety of average-load levels, mostly scattered uniformly throughout the range between 0 and $60 \cdot p$.

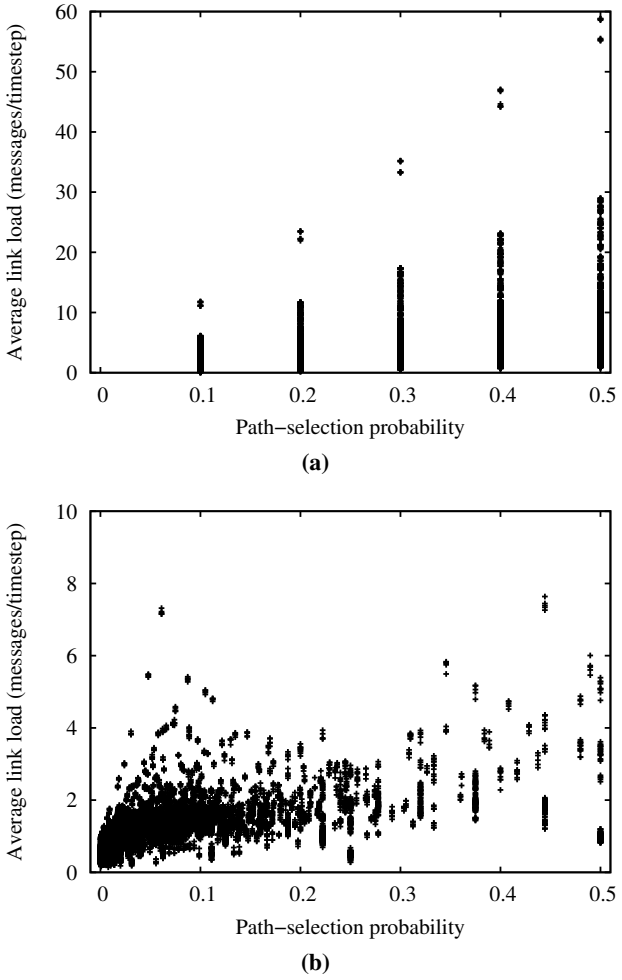


Figure 3: Relationship between the probing node on links (averaged over all links) and the path-selection probability used for probing, when the probability is chosen (a) without examining the network topology or (b) after examining the network topology.

On the other hand, Fig. 3(b) shows the load results from simulation trials where p was set to range between two topology-dependent values: In particular, in a given network $G = (V, E)$ with a given probing-path set $\mathscr{P}$, compute for each link $e \in E$ the fraction f_e of paths in $\mathscr{P}$ that contain e; then, the selection probability p for this second set of trials ranged between $\min_{e \in E} f_e$ and $\max_{e \in E} f_e$. This set of topology-dependent probabilities produced a significant improvement in link load.

(Note that, although the horizontal axes of the two plots in Fig. 3 span the same range of values, the vertical axis of Fig. 3(b) spans a smaller range of values than that of Fig. 3(a), reflecting an improvement in link load.)

Still, the randomized approach does not compare well with probing a subset of paths produced by using the standard $O(\log n)$-approximation-factor minimum-set-cover (MSC) approximation algorithm [21]. In this algorithm, a subset of probing paths $S \subseteq \mathscr{P}$ is chosen in the following manner:

- Initialize $S = \{\}$;

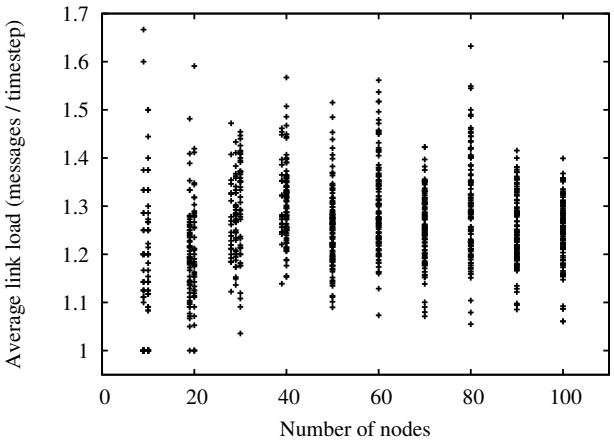


Figure 4: Relationship between average link load in a minimum set cover and the number of nodes, for trials involving the randomly generated graphs.

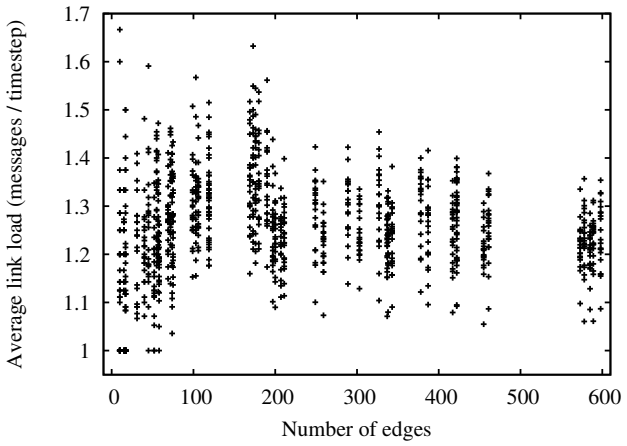


Figure 5: Relationship between average link load in a minimum set cover and the number of edges, for trials involving the randomly generated graphs.

- Iteratively pick the path $P \in \mathscr{P} - S$ that covers the most uncovered links, and add that path P to S;
- Terminate when all links are covered.

Figures 4–6 show the average link load that would be incurred by probing the entirety of the MSC set of paths S in a single timestep. (Each data point represents the average link load for one of the randomly generated inputs; recall that each input is a combination of a network topology and a probing-path set.) Doing so would give a probing delay of 1 (because every edge appears in the set cover), with an average load of no more than 1.7. The figures show that there is no strong relationship between average link load for the minimum set cover with respect to the number of nodes, number of links, or size of the probing-node subset for any of the randomly generated inputs.

Of course, the set-cover approach comes with its own downsides, as discussed before: the approximation algorithm requires iteratively selecting paths covering the most yet-uncovered edges, which can be a costly pre-computation for large networks.

The randomized approach can achieve some level of balance without the pre-computation: Simulation results point to a number of inputs where a path-selection probability of 0.2 leads to an average load of about 2 with average delay also about 2.

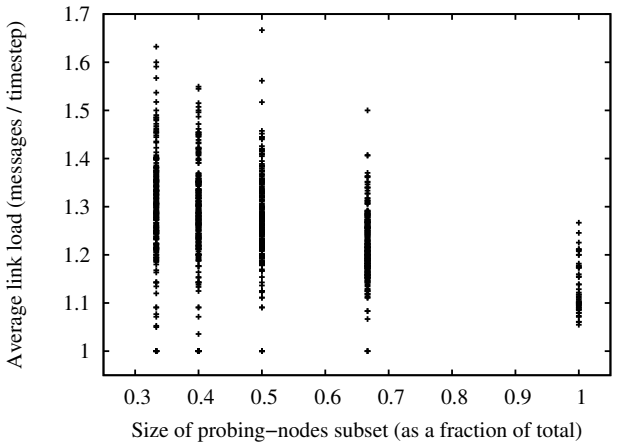


Figure 6: Relationship between average link load in a minimum set cover and the fraction of nodes chosen for the probing-node subset, for trials involving the randomly generated graphs.

8. CONCLUSIONS AND FUTURE WORK

In this paper, we have introduced a formal framework for evaluating probing strategies for network-performance measurement. We have formalized metrics that capture the performance of such algorithms with respect to various desiderata. This identifies areas for improving existing approaches. We have also identified some formal tradeoffs among different desiderata and have showed that achieving certain combinations are computationally intractable (even to approximate). Our analysis includes both deterministic and randomized approaches to network probing.

The framework and results that we present here suggest numerous directions for future work, as follows:

In terms of analysis, this framework should be applied to probing algorithms that are currently in use in order to assess their strengths and weaknesses and to gain insight into their suitability for various applications. At the same time, our theoretical analysis should be extended to identify additional fundamental tradeoffs between different metrics (including three or more metrics taken together).

In terms of algorithms, as this framework is used to study existing algorithms in settings of current interest, it will identify the need for new algorithms to meet certain performance goals suggested by the metrics we identify here. Such algorithms will need to be developed. This process will also make the development of active-probing algorithms more rigorous.

The development of this framework will also promote the identification of other desiderata for probing algorithms. Those, in turn, will suggest new metrics for evaluating the performance of algorithms. Those metrics will need to be applied to existing and new algorithms; of particular interest, their relationships to other metrics (correlations and tradeoffs) will need to be evaluated.

9. ACKNOWLEDGMENTS

We thank Joel Sommers for the many valuable conversations that helped motivate and shape this paper, and we thank the referees for their helpful feedback.

10. REFERENCES

[1] G. Agnarsson and M. M. Halldórsson. Strong colorings of hypergraphs. In *Proc. Approximation and Online Algorithms (WAOA'04)*, pages 253–266, Bergen, Norway, Sept. 2004. Springer-Verlag LNCS 3351.

[2] P. Barford, A. Bestavros, J. Byers, and M. Crovella. On the marginal utility of network topology measurements. In *Proc. ACM IMW'01*, pages 5–17, Nov. 2001.

[3] P. Barford, N. Duffield, A. Ron, and J. Sommers. Network performance anomaly detection and localization. In *Proc. IEEE INFOCOM 2009*, pages 1377–1385, Apr. 2009.

[4] V. Batagelj and U. Brandes. Efficient generation of large random networks. *Phys. Rev. E*, 71(3):1–5, 2005.

[5] Y. Bejerano and R. Rastogi. Robust monitoring of link delays and faults in IP networks. *IEEE/ACM Trans. Net.*, 15(5):1092–1103, Oct. 2006.

[6] P. Berenbrink and T. Sauerwald. The weighted coupon collector's problem and applications. In *Proc. COCOON'09*, pages 449–458, July 2009.

[7] L. Breslau, I. Diakonikolas, N. G. Duffield, Y. Gu, M. Hajiaghayi, D. S. Johnson, H. J. Karloff, M. G. C. Resende, and S. Sen. Disjoint-path facility location: Theory and practice. In *Proc. SIAM ALENEX'11*, pages 60–74, Jan. 2011.

[8] M. Coates, A. Hero, R. Nowak, and B. Yu. Internet tomography. *IEEE Sig. Proc. Mag.*, 19(3):47–65, May 2002.

[9] A. Dhamdhere, R. Teixeira, C. Dovrolis, and C. Diot. NetDiagnoser: Troubleshooting network unreachabilities using end-to-end probes and routing data. In *Proc. ACM CoNEXT'07*, Dec. 2007.

[10] N. Duffield. Network tomography of binary network performance characteristics. *IEEE Trans. Inf. Theory*, 52(12):5373–5388, Dec. 2006.

[11] T. Erlebach. Approximation algorithms for edge-disjoint paths and unsplittable flow. In E. Bampis, K. Jansen, and C. Kenyon, editors, *Efficient Approximation and Online Algorithms*, pages 97–134. Springer-Verlag, 2006.

[12] T. G. Griffin, F. B. Shepherd, and G. Wilfong. The stable paths problem and interdomain routing. *IEEE/ACM Trans. Net.*, 10(2):232–243, Apr. 2002.

[13] A. A. Hagberg, D. A. Schult, and P. J. Swart. Exploring network structure, dynamics, and function using NetworkX. In *Proceedings of the 7th Python in Science Conference (SciPy2008)*, pages 11–15, Pasadena, CA USA, Aug. 2008.

[14] I. Holyer. The NP-completeness of edge-coloring. *SIAM J. Comput.*, 10(4):718–720, 1981.

[15] S. Khot. Hardness results for approximate hypergraph coloring. In *Proc. ACM STOC'02*, May 2002.

[16] S. Knight, H. X. Nguyen, N. Falkner, R. Bowden, and M. Roughan. The internet topology zoo. *IEEE JSAC*, 29(9):1765–1775, Oct. 2011.

[17] R. Motwani and P. Raghavan. *Randomized Algorithms*, Sec. 3.6.1, pages 57–59. Cambridge Univ. Press, Aug. 1995.

[18] H. X. Nguyen, R. Teixeira, P. Thiran, and C. Diot. Minimizing probing cost for detecting interface failures: Algorithms and scalability analysis. In *Proc. IEEE INFOCOM 2009*, pages 1386–1394, Apr. 2009.

[19] O. Parekh and D. Segev. Path hitting in acyclic graphs. *Algorithmica*, 52(4):466–486, Dec. 2008.

[20] P. Singh, M. Lee, S. Kumar, and R. R. Kompella. Enabling flow-level latency measurements across routers in data centers. In *Proc. USENIX Hot-ICE'11*, Mar. 2011.

[21] P. Slavík. A tight analysis of the greedy algorithm for set cover. In *Proc. ACM STOC'96*, pages 435–441, May 1996.

[22] J. Sommers, P. Barford, N. Duffield, and A. Ron. Multiobjective monitoring for SLA compliance. *IEEE/ACM Trans. Net.*, 18(2):652–665, Apr. 2010.

[23] H. H. Song, L. Qiu, and Y. Zhang. NetQuest: A flexible framework for large-scale network measurement. *IEEE/ACM Trans. Net.*, 17(1):106–119, Feb. 2009.