Efficient Beacon Placement for Network Tomography

Ritesh Kumar and **Jasleen Kaur**
Department of Computer Science
University of North Carolina at Chapel Hill
{*ritesh, jasleen*}*@cs.unc.edu*

Abstract— **Recent interest in using tomography for network monitoring has raised the fundamental issue of whether it is possible to use only a small number of probing nodes (beacons) for monitoring all edges of a network in the presence of dynamic routing. Past work has shown that minimizing the number of beacons is NP-hard, and has provided approximate solutions that may be fairly suboptimal. In this paper, we use a two-pronged approach to compute an efficient beacon set: (i) we formulate the need for, and design algorithms for, computing the set of edges that can be monitored by a beacon under all possible routing states; and (ii) we minimize the number of beacons used to monitor all network edges. We show that the latter problem is NP-complete and use an approximate placement algorithm that yields beacon sets of sizes within $1 + ln(|E|)$ of the optimal solution, where E is the set of edges to be monitored. Beacon set computations for several Rocketfuel ISP topologies indicate that our algorithm may reduce the number of beacons yielded by past solutions by more than 50%.**

I. Introduction

The last two decades have witnessed an exponential growth of the Internet in terms of its infrastructure, its traffic load, as well as its commercial usage. Today, the growth of the world's economy depends heavily on the connectivity, reliability, and quality of service provided by Internet Service Providers (ISPs). The ability to monitor the health of their networks is essential for ISPs to provide good service to customers. Consequently, there is significant interest in developing network monitoring infrastructures that allow ISPs to monitor their network links.

A key consideration in the design of monitoring infrastructures is to develop low-cost solutions. In particular, the idea of placing and operating sophisticated monitors at all nodes in a network is not cost-efficient. Instead, there has been significant recent interest in relying on tomographic techniques that use only a few probing nodes (beacons) for monitoring the health of all network links [1], [2], [3], [4], [5], [6], [7]. A key challenge is to find a *small* set of beacons that is guaranteed to be able to monitor all network links, even with *dynamically-changing* IP routes. Two recent efforts have focused on the problem of finding the smallest beacon sets for a network [4], [7]. These, however, do not adequately meet the above challenge—the beacon set of [4] is not robust to changes in IP routes, and the beacon set proposed in [7] can be quite large for real ISP topologies (Sections II and V). In this paper, we present beacon placement strategies that meet both aspects of the above challenge.

Our approach relies on a two-pronged methodology. First, we define the concept of a *deterministically monitorable edge set* (DMES) of a beacon as the set of edges that can be monitored by the beacon under all possible route configurations. We present efficient graph-theoretic algorithms for computing the DMES of all candidate beacons for a given network. Second, we consider the problem of finding the minimum number of beacons such that the union of their DMES covers all network edges. We show that this is an NP-complete problem. We then use an approximate solution that yields beacon sets of sizes within $1 + ln(|E|)$ of the optimal solution, where E is the set of network edges. Finally, we prove and exploit additional properties of beacons that help in improving the computational efficiency of our algorithm. Our experimental results with several real ISP topologies obtained from the Rocketfuel project [8] illustrate that our beacon placement strategy yields beacon sets that are $50 - 70\%$ smaller than those yielded by [7].

The rest of this paper is organized as follows. In Section II, we formulate the problem of beacon placement and discuss past work. In Section III, we define and compute DMES. Section IV discusses beacon set minimization. Section V presents experimental results with Rocketfuel topologies. We conclude in Section VII.

Notations and Assumptions. We model a network as an undirected graph $G(V, E)$, where V is the set of network nodes and E is the set of links (or edges)—in Section VI, we extend our analysis to directed graphs as well. We use the terms network and graph interchangeably. We assume that G is connected (there exists a path from any node to

any other node) and that all routes are simple (acyclic). Finally, we say that two physical paths between a pair of nodes are *distinct*, if they differ in even one of the edges traversed.

II. Problem Formulation

In a tomographic network monitoring infrastructure, each network link is monitored by a special probing node, referred to as a *beacon*.[1] The basic idea behind most tomographic setups is fairly simple: the beacon sends a pair of nearly-simultaneous probes to the two end-nodes of the link, only one of which traverses the link. Each end-point sends back a response to the beacon—this may be implemented using ICMP echo messages. The results of the probes can then be used to infer properties of the link. For instance, if the objective is to measure link delays, then the difference in round-trip times of the two probes can be used as an estimate. If the objective is to simply detect link transmission failures, the success and failure of the two probes may be used as reasonable estimators.

Note that, in general, a beacon is capable of monitoring several network links. A set of beacons that can be collectively used to monitor *all* the links of a network is referred to as a *beacon set*. A central issue in the design of a monitoring infrastructure is that of *beacon placement*—which network nodes should be used to construct a beacon set? Two requirements guide the design of a good beacon placement strategy:

• *Minimizing the number of beacons.*

One of the prime motivations for using tomography for network monitoring is to reduce the cost of the monitoring infrastructure. However, even a tomographic infrastructure involves the development, installation, debugging, operation, and maintenance of specialized software/hardware on each beacon. In order to minimize the cost of doing so, it is important that the number of beacons used to monitor all links of a given network are minimized.

• *Robustness to routing dynamics.*

Routing state in many networks responds to changes in traffic patterns and link loads, as well as to link failures. Since Internet traffic conditions are highly dynamic, the default IP routes in a given network may change at relatively small time-scales. A monitoring infrastructure, therefore, should not assume a specific routing configuration in order to assign a beacon to a given link. More generally, a beacon set should be able to monitor all network links, independent of the current route configuration.

In this paper, we focus on the problem of beacon placement that meets the above requirements. Specifically, our objective is to: *minimize the number of beacons required to deterministically monitor all the links of a given network, even in the presence of dynamism in IP routes.*

[1]Some applications of tomography may require multiple beacons to monitor a given link [9].

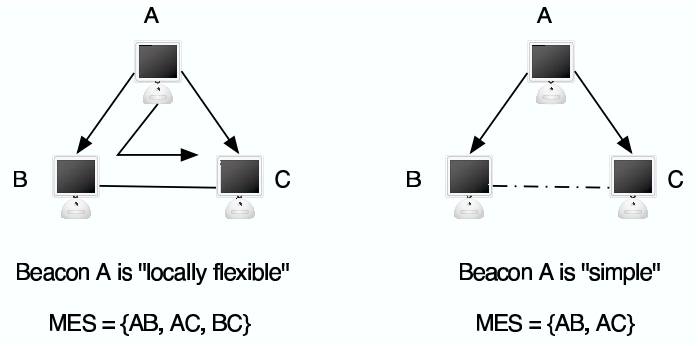

Fig. 1. Simple vs. Locally-flexible Beacons

A. Past Work

A fundamental step in finding the smallest beacon set for a network is to first enlist the edges that can be monitored by each candidate beacon, referred to as the *monitorable edge set* (MES) of the beacon. Note that the union of MES of all beacons in a beacon set is equal to the set of all network edges. In general, the larger is the average MES size in a beacon set, the smaller is the beacon set.[2] Below, we briefly discuss two beacon placement schemes that have been proposed in recent literature, which differ in their assumptions about which links comprise the MES of a beacon.

• *Simple Beacons*: In [4], the authors assume that the MES of a beacon consists of all links that can be reached by the beacon—which are links that lie on its IP routing tree.[3] In order to monitor a link in its MES, the beacon—henceforth referred to as a "simple" beacon—sends probes to the end-points of the link, along the default IP paths to those end-points. The authors demonstrate that the problem of minimizing the size of the beacon set with such beacons, is NP-hard, and provide a placement strategy that produces a beacon set no larger than $1 + log|E|$ times the optimal beacon set. Unfortunately, since the authors assume that all links within the routing tree of a beacon be-

[2]Perhaps the largest MES (and smallest beacon set) that can be envisioned is when a *single* beacon monitors *all* the links of a network—this is feasible, for instance, in a network which supports source-routing [10]. In such a network, a beacon can precisely specify the path traversed by its probes, and hence can probe the end-points of any network link. However, this strategy relies on the availability of source-routing support at *all* network nodes, which is the not the case with a majority of current networks [4].

[3]The IP routing tree of a node refers to the tree, rooted at the node, which is formed by the links that lie on the default IP routes from that node to each of the other nodes in the network.

long to its MES, their strategy is not robust to changes in routing trees and works only for networks with static routes.

• *Locally-flexible Beacons*: In [7], the authors consider beacons that have a greater flexibility in selecting the paths taken by the probes. Specifically, the beacons—henceforth referred to as "locally-flexible" beacons—are capable of selecting the *first* link (outgoing link from beacon) on which a probe to any destination is transmitted. A probe can, therefore, be sent to a destination either along the current IP route to the destination, or along one of the current IP route from any immediate neighbor to the same destination (Figure 1).[4] Furthermore, the authors do not assume static routing state and define the MES of a beacon to consist of links that, irrespective of what current routes are, can always be monitored. The authors do not provide a mechanism to compute such an MES for a beacon, but show that even if these sets are known, the beacon set minimization problem is NP-hard. The authors instead suggest an alternative beacon-placement strategy which, unfortunately, could result in fairly large beacon sets for current network topologies (see Section V).

To summarize, existing beacon placement strategies[5] are either not robust to routing dynamics or are inefficient in minimizing the number of beacons. In this paper, we build on past work to address these limitations by using a two-pronged approach:

1. **Deterministic MES Computation:** In order to be insensitive to routing dynamics, for each candidate beacon, we determine the set of edges—referred to as its Deterministic MES (DMES)—that can be monitored by it under *all* possible routing configurations.
2. **Beacon Set Minimization:** In order to minimize cost, we address the problem of finding the smallest beacon set.

In the following two sections we present our abstractions and methodologies for achieving these two steps for both simple and locally-flexible beacons.

[4]The authors in [7] implicitly assume that the default IP route from any neighbor to the said destination will not go through the beacon node. This assumption may get violated when a path through the beacon has a smaller cost that any other physical path between a neighbor and the destination.

[5]An orthogonal problem of beacon placement for detecting *multiple link failures* that occur simultaneously has been considered in [11]. In general, it is not possible to detect all cases of simultaneous link failures in a given network. In [11], the authors restrict their attention to those simultaneous link failures that can be detected in the absence of any limitations on the number of beacons and probes. They then provide efficient algorithms for minimizing the number of beacons and probes needed for detecting these failures. Like [4], this work assumes "simple" beacons and uses the IP routing tree in the beacon set computation and, hence, is applicable only to networks with non-dynamic routes. As part of future work, we hope to use our formulations from this paper to extend the work in [11] to other beacon types and to networks with dynamic routing.

III. Deterministically Monitorable Edge Sets

The first key problem we need to solve is to find the set of edges that can be monitored by a beacon, independent of the routing configuration. This is formally captured in the following definition.

Definition 1: An edge is said to be *deterministically monitorable* by a beacon if the beacon can monitor it under all possible route configurations. The *Deterministically Monitorable Edge Set* (DMES) of a beacon is the set of all deterministically monitorable edges associated with that beacon.

In what follows, we consider both simple and locally-flexible beacons and present algorithms for computing their DMES. Below, we consider both simple and locally-flexible beacons and present methodologies for computing their DMES. We assume throughout that a beacon of either type is able to monitor all edges directly connected to it, irrespective of the routing configuration. Thus, all edges incident on a node u belong to its DMES. Lemma 1 establishes a crucial property of a deterministically monitorable edge (DME).

Lemma 1: If a beacon u has the ability to reach, under all possible route configurations, one of the end-points of an edge e through that edge, then u can deterministically monitor e.

Proof: Let the two nodes on the link e be x and y such that the probe packet from u to y traverses in the direction $x->y$. Now, given a path p of a probe to x, consider the following properties of such a path;

• p can always be extended by e to give p', a path of a probe to y. Thus, p cannot traverse y because if we extended p by e, the path doesn't remain simple and hence is no more valid.

• all paths from u to y have e has the last link in the path.

Let us consider the following probe results.

• The probe to y is successful. This means that e is up. This is because *all* paths for the probe to y go through e at the end.

• The probe to y fails but the probe to x is successful. This means that e is down. The probe to x being successful means that there was a path to y through e after the successful path to x. Since this path couldn't be taken, it implies that link e was down.

• The probes to both x and y fail. This leads to uncertainty as there could be faults in the paths to both x and y.

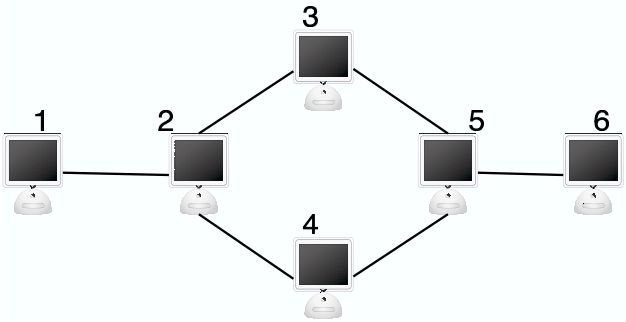

Fig. 2. The DMES may not a connected graph.

To measure link delays we require probes from u to both x and y to be successful. Assuming that the default routing policy is monotonic implies that the path of the probe from u to y was exactly the same path as that of the probe from u to x except the last edge e. The difference in the round trip times for both these probes gives us the round trip delay of the link e.

Thus we have proved that given the above properties of an arbitrary path p, we can monitor the link e for link failure and under assumptions of routing monotonicity, link delay.

■

It is computationally expensive to enumerate *all* paths of a probe from a beacon to a particular node. We will make a simplifying assumption which we believe is valid at least for inter-domain networking. We will assume that *no physically available path in the network is prohibited as a network path*. This allows us to use graph search algorithms like depth first search to infer about the above mentioned properties of paths in an efficient manner. The algorithms 1 and 2 and their correctness proofs should give the reader further insights into how this can be done.

A. DMES for Simple Beacons

Theorem 1: Let u be a simple beacon and let $S(v)$ be the set of all distinct physical paths from u to another node v. The link $l(v)$ is deterministically monitorable by u if for *all* paths p in $S(v)$, $l(v)$ is the last edge on p. The DMES of u is the set of all such edges $l(v)$ for all nodes $v \in V$.

Proof: Since all paths from the beacon u to v have $l(v)$ as the last edge, the current IP route from u to v (which takes one of these paths) ends in the edge $l(v)$. From Lemma 1, therefore, beacon u is able to monitor the link $l(v)$. ■

Note that a DMES yielded by Theorem 1 has no more structure than an arbitrary edge set. In particular, the DMES need not form a connected sub-graph; Figure 2 illustrates that the DMES of node 1 includes the edges 1-2 and 5-6. We now present an efficient algorithm for computing the DMES for simple beacons.

Algorithm 1: Computing DMES of a simple beacon u.

```
Initialize S to be an empty set;
For all edges l neighboring u
    include l in S;
For all nodes v in V
    do a depth first search from v;
    (we get a set of forests each
    connected to v by one or more
    edges)
    if u lies in a forest connected
      to v by only a single edge e
          include the edge e in S;
S is the DMES for u;
```

Proof: (Proof of correctness) Consider a depth first tree (along with its back edges) constructed from the node v. If we consider all forests rooted at the immediate neighbors of v, then these might connect to v via one or more edges. Separating forests this way helps us to isolate all possible paths from the beacon u to the node v. Any probe packet from u to v is entirely confined to paths in the forest containing u. Now, if the beacon u lies in a forest which connects to v via only one edge, all paths from u to v have to cross this edge at the end of the path. However, if u lies in a forest which is connected to v via two or more edges, then there exist at least two distinct paths from the beacon u to the node v which end in different edges to the node v. This means that the edges are not deterministically monitorable from u (Theorem 1). ■

Time Complexity: The cost of computing the DMES of a simple beacon is essentially that of running a depth first search (DFS) algorithm at every node in the network. Since the time complexity of running a depth first search on $G(V, E)$ is $\theta(|E| + |V|)$, the time complexity of Algorithm 1 is $\theta(|V|(|E| + |V|))$.

Note that the DMES for multiple beacons can be computed in parallel. After running DFS on a node v, we can add an incident edges of v to the DMES of all beacons that belong to the forest rooted at the edge, if there are no more edges connecting that forest to v. Since the number of potential beacons is bounded by $|V|$, and the time complexity of depth first search is $\theta(|E| + |V|)$, the time complexity for the parallel DMES computation algorithm is the same as above. Hence, we can calculate the DMES of *all* nodes in $G(V, E)$ in $\theta(|V|(|E| + |V|))$ time.

B. DMES for Locally-flexible Beacons

Theorem 2: Let u be a locally-flexible beacon and E_u be the set of edges directly connected to u. For each edge $i \in E_u$, let $S_i(v)$ be the set of all paths from u to any other node v, that start with the edge i. A link $l_i(v)$ is deterministically monitorable from u if for all paths in $S_i(v)$, $l_i(v)$ is the last edge. The DMES of u is the set of all deterministically-monitorable edges $l_i(v)$, for all $v \in V$ and all $i \in E_u$.

Proof: Since locally-flexible beacons can select the outgoing link on which to transmit a probe, we need to consider only those paths to v which start from a specific edge in E_u, to see if there is a common ending edge. Thus, even if u has paths to v which end with different edges, if all paths to v that start from u with edge i end with a common edge $l_i(v)$, u has the control over the ability to reach v through $l_i(v)$. From Definition 1 and Lemma 1, therefore, the common edge is deterministically monitorable. ■

Below, we present an algorithm for computing the DMES for locally-flexible beacons.

Algorithm 2: Computing DMES of a locally-flexible beacon u.

```
Initialize S to be an empty set;
For all edges i neighboring u
    include i in S;
    remove i from E;
For all nodes v in V
    do a depth first search from v;
    (we get a set of forests each
    connected to v by one or more
    edges)
    if one of u's neighbors lies in
      the forest connected
      to v by a single edge e
          include the edge e in S;
S is the DMES for u;
```

Proof: (Correctness) The proof is similar to that for Algorithm 1. Let u_i be the neighbor connected to u through i. The forest containing u_i also contains all paths from u to v that start in i. This is because, if there was another path from u to v through i, v and u_i would have been connected via a path which would be captured in the depth first search. Conversely, consider any simple path from u_i to v. Since Algorithm 2 removes i from E, adding i at the start of the path still retains the "simple" property of the path. Such a path is a valid path from u to v starting with edge i. Since we removed u's neighboring edges from E, one could argue that some paths might be missing. However, this cannot be true because no simple paths from v to u would transit u in the middle of the path. Hence the forests obtained by removing the edges neighboring u are representative of all paths from u's neighbors to v. ■

Time Complexity: The cost of this algorithm is that of running a depth first search on each node and for each depth first search run checking if any of the neighbors of u are in a singly connected forest. Thus, if the degree of u is k, the time complexity of the algorithm is $\theta(|V|(|E| + |V| + k))$. Since k is bounded by $|V|$, the time complexity is $\theta(|V|(|E| + |V|))$. Note that, unlike simple beacons, DMES can not be computed in parallel for multiple nodes because for each beacon we customize the graph $G(V, E)$ (removal of neighboring edges) specific to the beacon before doing all the depth first searches. The complexity of computing DMES for *all* nodes in $G(V, E)$ is, therefore, $\theta(|V|^2(|E| + |V|))$.

IV. Beacon Set Minimization

The second key problem—of minimizing the beacon set for a network—is formally stated below:

Beacon Minimization Problem (BMP). Let D_u be the DMES associated with a node $u \in V$. Then the *beacon-minimization problem* is to find the smallest set of beacons, $B \subseteq V$, such that $\bigcup_{b \in B} D_b = E$.

Theorem 3: The Beacon Minimization Problem is NP-complete.

Proof: Let the graph under consideration be $G(V, E)$. Let S be the set $\{D_v : v \in V\}$. Since every node can deterministically monitor at least its neighboring edges, $\bigcup_{v \in V} D_v = E$. Also, $D_v \subseteq E$. To find the smallest beacon set we need to find $B \subseteq S$ such that $\bigcup_{D_v \in B} D_v = E$ and $|B|$ is minimized. This is the the same as the classic Minimum Set Cover problem (MSCP) [12]. Thus, there is a one-to-one correspondence between BMP and MSCP, by using the concept of deterministically monitorable edge sets. The Minimum Set Cover Problem is known to be NP-Complete [12], [13]; this implies that BMP is NP-complete as well. ■

Fortunately, MSCP has a pruning-based approximate solution—below, we adapt the pruning algorithm and use heuristics from the literature to establish optimality bounds for it.

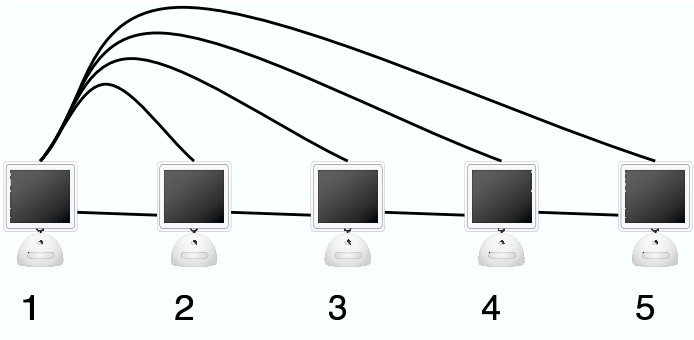

Fig. 3. Optimality of the "pruning" algorithm depends a lot on the order of selection of nodes.

Algorithm 3: Find the beacon set for completely monitoring a graph $G(V, E)$.

```
Initialize B to be an empty set;
Initialize E' = E;
while E' is not empty
    Select* node u from V not in B;
    E' = E' - the DMES of u;
    Include u in B;
B is the beacon set;
```

It is straightforward to see that the algorithm returns a valid beacon set. This is because every edge that was eliminated from E' could be deterministically monitored by some node that was included in the beacon set.

The efficacy of the "pruning" algorithm in minimizing the size of the beacon set depends on the order of selection (the *ed operation in Algorithm 3) of nodes. For instance, consider the topology in Figure 3. The optimal beacon set (with locally-flexible beacons) for this topology contains just node 1. However, the "pruning" algorithm will lead to a non-optimal beacon set if it selects any node other than node 1 as its first beacon. In fact, selecting the nodes in the order 5, 4, 3, 2 and 1 causes the "pruning" algorithm to select all the nodes in the graph for the beacon set.

There exists a known heuristic for the MSCP pruning-based solution that ensures that the size of the solution is within a bound of the optimal [14]. The heuristic maps to the following node-selection rule (* in above algorithm) for BMP. Select that node for a beacon whose DMES has the maximum overlap with the current pruned graph. Specifically, if E' is the current set of edges of the pruned graph then we choose the node v such that $|D_v \cap E'|$ is maximum. This heuristic results in provable [14] bounds of optimality of the beacon set as: $\frac{|B(heuristic)|}{|B(optimal)|} = 1 + ln|E|$.

A. Further Optimizations

We next establish additional monitoring-related properties of networks that let us further optimize the computation of the minimal beacon set. The concept of *node arity* in an undirected graph $G(V, E)$, defined in [7], is useful for this discussion. Below, we restate the definition from [7] in a slightly different manner.

Definition 2: *(Node Arity)* The arity of a node, u, *with respect to another node*, v, is defined as the number of distinct paths that exist between the two nodes such that each of these paths starts from a unique outgoing edge from u. The arity of a node u is defined as the maximum of arities of u with respect all nodes of the graph.

Using the terminology of [7], we call a node "high arity" if the node's arity is more than one. Note that since $G(V, E)$ is assumed to be connected, there is at least one path from every node to every other node (assuming $|V| > 1$). Hence, the arity of a node is always greater than or equal to one. Also, since the maximum number of distinct paths (with a unique outgoing edge) from u to v can not be more than the degree of u, the arity of a node is bounded by its degree. Algorithm 4 in the Appendix is an efficient way for finding whether a node in $G(V, E)$ is high arity or not. Our algorithm is based on the insight that if a node v has arity one, all forests generated in the depth-first search from v will be connected to v by a single edge.

A.1 Single Arity Networks

It is interesting to study the Beacon Placement Problem for a graph $G(V, E)$ that contains only single arity nodes. We show the optimal beacon set for such a graph is a singleton set, and can contain any one node of the graph.

Lemma 2: A graph $G(V, E)$ with no high arity nodes is a tree.

Proof: Since the graph is connected, the nodes present in the graph have an arity of at least one. Since there are no high arity nodes in graph, the arity of all nodes is one. Suppose there is a cycle in G. Then, any two distinct nodes in the cycle have separate paths to each other from their different outgoing edges. Thus, the nodes in the cycle are high arity, which is a contradiction. Hence, the graph cannot contain cycles. This implies that $G(V, E)$ is a tree. ■

Lemma 3: A tree can be completely monitored by just one beacon instantiated on any one of its nodes.

Proof: Since the graph is a tree, there is one unique path from any node to any other node. Let us consider an ar-

bitrary node u in the tree. Consider the leaves of the tree as rooted at u. The edges which connect the leaves to the tree can be monitored by probing the leaf and its parent. Hence, all edges connecting the leaves to the tree can now be monitored. Now consider the paths from the root u to any other node. This path will not contain any edges which connect the leaves to the tree. Now, probing for any other edge would require probes to be sent to the nodes on its two sides. Since none of these nodes are leaves, we can conclude that none of the probing paths will contain edges connecting the leaves to the tree. Hence, we can eliminate the leaf nodes and their edges to the tree from the graph for further analysis. Now we have a new set of leaf nodes and we recursively apply the logic above to this till we eliminate all edges from the tree. Hence, the tree can be completely monitored by having any one node as the beacon. ■

Theorem 4: A network with no high arity nodes can be monitored by a single beacon on any node in the network.

Proof: Follows from Lemma 2 and Lemma 3. ■

This theorem implies that a minimum beacon set can be computed trivially and optimally for any single-arity network, without using the "pruning" algorithm presented earlier.

A.2 Relationship between Optimal Beacon Sets and High Arity Node Sets

We next show that for networks that require use of the pruning algorithm, it is possible to substantially reduce the search space for a beacon set. Specifically, in Theorem 5 we show that we can eliminate all single arity nodes from our consideration. Lemma 4 is used to prove the theorem.

Lemma 4: A graph which has at least one high arity node cannot be completely monitored by a single simple or locally-flexible beacon placed on a single arity node.

Proof: Consider a graph $G(V, E)$, which has a high arity node x. Also consider a single beacon on a single arity node b. Since x is a high arity node, there exists a node y, to which x has multiple paths with different outgoing edges from x and multiple incoming edges to y. Thus, x and y are part of a cycle. b cannot be on this cycle as otherwise it would be high arity. Note that b cannot deterministically monitor any edges in this cycle. This is because for any edge in the cycle, b has multiple paths to one of its end-points, which end in a different edge. Hence, b cannot monitor the entire graph. ■

Theorem 5: An optimal beacon set, when beacons are simple or locally-flexible, is a subset of the set of high arity nodes.

Proof: Let B be an optimal beacon set of graph $G(V, E)$. Let $b \in B$ be a single arity node. Since, B is optimal, removing b from B causes at least one edge $e \in E$ to be not deterministically monitorable by B. Let x and y be the two nodes on either side of e and assume that e is traversed when a probe packet is sent from b to y (and hence not traversed when sent to x). Now, consider a depth first search from y. Consider the forests sprouting from the edges adjacent to y. Since e is deterministically monitorable from b, b lies in the forest, F_e, sprouting from the edge e. Also, F_e is connected to y by only e and no other edge (otherwise, e wouldn't have been deterministically monitorable by b). Since, e was exclusively monitorable by only b, there are no other beacons in F_e.

Now consider the following two cases:

- *F_e has at least one high arity node.* Let h be a high arity node in F_e. Since e is the only edge connecting the forest to the rest of the graph, h must be high arity with respect to a node in the forest itself. Hence from Lemma 4, b cannot monitor the entire forest. Also, no other beacon outside F_e can completely monitor F_e, as all paths from such beacons must traverse e; just like b, such beacons can not deterministically guarantee the last edge in their paths to nodes in the cycle formed by h (Lemma 4). Since b should be able to monitor the entire forest as it is the sole beacon in it, we reach a contradiction. It follows that all nodes in the forest must be single arity.
- *F_e has only single arity nodes.*
If F_e contains only single arity nodes then all edges in the forest can be monitored by a single beacon outside the forest. This is because F_e is a tree, and is reachable only through e from a beacon outside F_e. Any path from an outside beacon to any node in F_e has a unique sub-path after crossing e. Hence, all edges within F_e can be monitored by it. This implies that b is a redundant beacon, and removing b from B doesn't effect the monitorable edge coverage of the beacon set B. This contradicts the definition of B as an optimal beacon set.

Thus, there can not exist a single arity node in B. Hence, we have proved that B has to contain only high arity nodes. ■

Theorem 5 lets us reduce the set of potential beacons used in Algorithm 3 to the set of high arity nodes. This can lead to substantial computational savings. For instance, we show in Section V (and Figure 4), that the number and fraction of single arity nodes in current ISP topologies can be quite high.

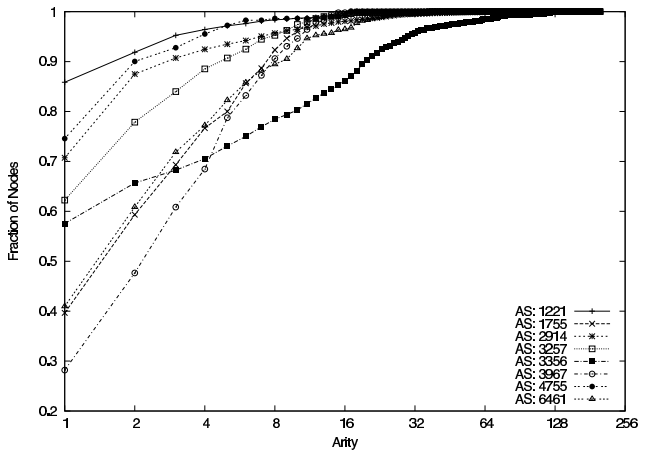

Fig. 4. Cumulative distribution of node arities for eight Rocketfuel topologies.

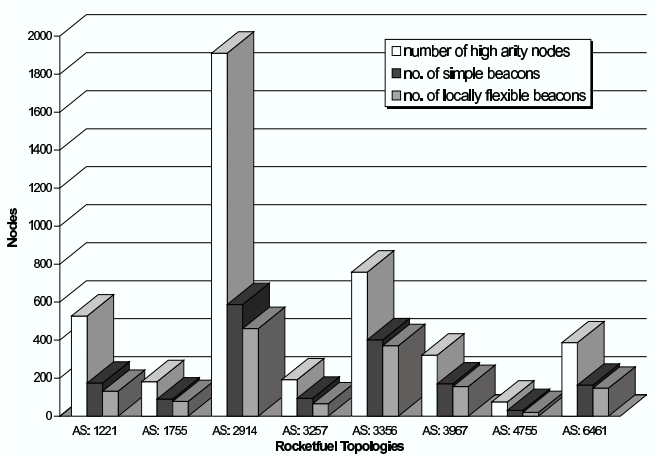

Fig. 5. Histogram of beacon set sizes yielded by different algorithms for the Rocketfuel topologies.

Please note that a network consisting of only single arity nodes is an exception to the above theorem.

In [7] the authors have shown that the set of high arity nodes in a graph is a beacon set—though potentially a non-optimal set—when beacons are locally-flexible. We have much strengthened this result by showing that the optimal beacon set is always a subset of the set of high-arity nodes (even with simple beacons). Not surprisingly, our pruning algorithm is able to find smaller beacon sets for all topologies. In order to numerically evaluate the efficacy of our formulations, we next present results of beacon set computations on a few real ISP topologies.

V. Experimental Results

In this section, we compute and compare the beacons sets yielded for several current ISP topologies: (i) by the beacon placement solution with locally-flexible beacons suggested in [7]; (ii) by our beacon placement algorithms for simple beacons; and (iii) by our algorithms for locally-flexible beacons. We refer to the resultant beacon sets as B_{HA}, B_S, and B_{LF}, respectively. We have implemented these algorithms in Java and have run these on eight major ISP topologies obtained from the Rocketfuel project at the University of Washington [8]. For each of the eight topologies, we analyze the distribution of node arities and calculate the sizes of beacon sets yielded by the three solutions.

Node Arities. The distribution of node arity for the eight topologies is plotted in Figure 4. We observe that:

1. The distribution of node arities are quite different for different ISPs, indicating that ISP topologies can be quite diverse in their topological structure. In particular, some ISP topologies have a long-tailed arity distribution, indicating that only a handful of nodes have significant redundancy in the manner in which they connect to the rest of the network. For most topologies, a majority of nodes have arities within 20, although we some nodes can have arities higher than 150.
2. The fraction of single arity nodes in the ISP topologies varies from less than 30% to more than 85%. It is important to note that, for every other node in the network, a single arity node has only one local edge that can be used to reach it. Single arity nodes, therefore, are not robust to failures of local links. We find that for most topologies, more than half of the nodes have a single arity.

A large fraction of single-arity nodes also implies that the optimizations proposed in Section IV-A to enable fast computation of beacon sets, can result in substantial savings.

It is important to observe that Rocketfuel ISP topologies are subject to inference errors. In particular, [15] demonstrates that the inclusion of links that do not exist and the omission of links that are actually present can inflate path diversity in these inferred topologies. This limits the accuracy of node arities computed above.

Beacon Set Sizes. It is important to mention that the Rocketfuel topologies for an ISP may not be connected (possibly due to lack of data about some links). Thus, some of the topologies we analyze have multiple (independent) connected components. More importantly, some of the components consist of only single-arity nodes (such components have a tree structure). For a fair comparison with the previous work in [7], which does not apply to single-arity networks, we ignore such components when computing beacon sets.[6] For any ISP topology, we add up the sizes of beacon sets computed for each of the remaining compo-

[6]This eliminates only a small fraction of nodes from each of the ISP topologies.

nents to get the total beacon set sizes—$|B_{HA}|$, $|B_S|$, and $|B_{LF}|$—for the three solutions being compared. Figure 5 plots the histograms of these beacon set sizes for the eight topologies. We observe that:

1. *Beacon sets with locally-flexible beacons.* Our beacon placement solution for locally-flexible beacons reduces the beacon set sizes yielded by [7] by $50-70\%$. More importantly, we find that some major ISP topologies can be completely monitored, independent of routing state, using less than a hundred locally-flexible beacons. This is an encouraging observation as it suggests that a tomography-based monitoring infrastructure may not be infeasible even for major ISP topologies.

2. *Beacon sets with simple beacons.* Even with simple beacons, our beacon placement solution reduces the beacon set sizes of [7] by $40-70\%$. This suggests that it may be feasible to design a simpler monitoring infrastructure that does not require that network nodes use different transmission rules for probe packets.

This conclusion is further supported by the comparison of beacon set sizes yielded by our solution for simple vs. locally-flexible beacons, which indicates that locally-flexible beacons may not yield significant gains for many major ISP topologies.

VI. Incorporating Half-Duplex Links

For brevity concerns, we have used an undirected graph model to present our algorithms and proofs. In practice, networks may connect any two nodes using a pair of half-duplex links, rather than a single full duplex link. A network monitoring infrastructure should ideally be capable of monitoring both half-duplex links as separate entities. Below, we illustrate that our solutions can be extended to such networks with slight modifications.

We change our network model by replacing each undirected link by two half-duplex links between adjacent nodes. More formally, if $G(V, E)$ is the undirected graph, we derive $G'(V', E')$ from G as follows:

- $V' = V$;
- for all $e \in E$, add $e_1 = x \rightarrow y$ and $e_2 = y \rightarrow x$ to E', where x and y are the nodes connected by e in G.

The following definitions need to be seen in the light of the new model.

- *Node Arity:* The definition of node arity remains the same except that when we talk about the outgoing edges of a node, we are talking about the links directed away from the node.
- *Path:* A path now indicates paths connected by links in the same direction. This also holds for the definition of physically connected paths.
- *DME:* The definition of a DME should now be seen under the light of the new definition of paths.

The algorithm to find the DMES of a beacon changes slightly. We do a depth first search on node v to find all possible paths from the beacon u to node v. In the undirected graph this didn't matter as the set of paths from beacon u to node v is the same as that from node v to beacon u. However, for a directed graph, before we do a depth first search from v, we need to reverse the graph so that at the end we are speaking of paths from u to v. However, since all directed links are paired with another link in the opposite direction, we don't actually need to reverse the graph to find the forest of nodes. We should only be concerned with the edge which finally gets included in the DMES of the beacon. This edge should be the edge going into node v and hence is the edge opposite to which connected v to the forest.

The pruning algorithm requires no changes since it is completely abstracted away from the beacon capabilities and network routing mechanisms used. To argue about the applicability of our optimizations to the new model we first present an observation for directed graphs.

Lemma 5: The existence of a high arity node in the directed graph as constructed in Section VI implies that we have a cycle. Conversely, a cycle implies the existence of a high arity node.

Proof:

Let u be high arity with respect to v. This means that we have distinct paths p_1 and p_2 from u to v both of which start from different outgoing edges of u. Since every directed link is paired with an oppositely directed link between the same nodes, we have paths p_1' and p_2' from v to u. Let us assume that we chose a v such that the first edges in paths p_1' and p_2' differ. Now paths p_1 and p_2' (and also paths p_1' and p_2) combine to form a cycle. Now consider that we have a cycle in the graph. We break the cycle along any two paths p_1 and p_2. Like the argument above, we would have a reverse path for both p_1 and p_2 which makes the nodes at which we broke the cycle, high arity. ■

The above keeps our single arity network optimization intact. Having no high arity nodes means that our network has no cycles. This is even stronger than a directed acyclic graph as, because of the way links are connected, any physical loop in the network becomes a cyclic path. The graph looks like a tree except that all edges are actually paired.

The above also keeps the optimization of choosing just the high arity nodes for the beacon set, intact. In the proof,

a high arity node is said to imply a cycle which we have already proved above.

VII. Future Work

There are several ways in which our work can be extended. We briefly discuss a few below.

In this paper, we consider two kinds of beacons: simple and locally-flexible. An important component of our future work is to generalize the notion of DMES for other kinds of beacons. For example, the beacons could form an overlay and use routing-tunnels to increase their DMES and reduce the beacon set size. We plan to explore the trade-off between beacon complexity and the beacon set size for monitoring realistic network topologies.

Our formulations assume that all high-level policies do not prohibit the use of a certain physical path between two network nodes. While this is a reasonable assumption for networks that are operated by a single autonomous entity, this may not be true when the network considered consists of multiple Autonomous Systems. In particular, multi-homed *stub* networks typically do not provide transit to traffic arriving from one of their ISPs and destined to a different ISP. We plan to extend our network model to incorporate such networks and compute beacon sets for large internetworks like the Internet.

An interesting extension of our framework is for monitoring infrastructure needed to monitor only a subset of all network links. For instance, an ISP may be interested in monitoring only its backbone or peering links. One approach for finding a beacon set for this scenario would be to create an abstraction in which some network nodes are collapsed to create a new network that contains only the relevant edges. The key challenges then would be in deciding where to install the beacons (as a single node in the collapsed graph may represent several nodes from the original network). We plan to explore this problem as part of future work.

Another interesting direction in which our work can be extended in by finding tree-like subgraphs in the network and use assigning one beacon to every such subgraph. The probes and/or routers would now need to be configured so that probe packets generating within a subgraph remain confined to the subgraph. This might be helpful in reducing the number of probes required, and the distance they travel, for monitoring all the edges of a large network.

References

[1] CAIDA, "Skitter," http://www.caida.org/tools/ measurement/skitter.

[2] M. Coates, A. Hero, R. Nowak, and B. Yu, "Internet tomography," *IEEE Signal Processing Magazine*, May 2002.

[3] K. Claffy, T.E. Monk, and D. McRobb, "Internet tomography," *Nature*, January 1999.

[4] Y. Bejerano and R. Rastogi, "Robust monitoring of link delays and faults in ip networks," in *Proceedings of the 2003 ACM INFOCOM*, April 2003.

[5] N.G. Duffield and F. Lo Presti, "Multicast inference of packet delay variance at interior network links," in *INFOCOM (3)*, 2000, pp. 1351–1360.

[6] N.G. Duffield, J. Horowitz, F. Lo Presti, and D. Towsley, "Network delay tomography from end-to-end unicast measurements," *Lecture Notes in Computer Science*, vol. 2170, pp. 576–??, 2001.

[7] J.D. Horton and A. Lopez-Ortiz, "On the number of distributed measurement points for network tomography," in *Proceedings of the 2003 ACM SIGCOMM conference on Internet measurement*, October 2003, pp. 204–209.

[8] N. Spring, R. Mahajan, and D. Wetherall, "Measuring isp topologies with rocketfuel," in *Proceedings of ACM/SIGCOMM '02*, August 2002.

[9] A. Shriram and J. Kaur, "Identifying bottleneck links using distributed end-to-end available bandwidth measurements," *First ISMA Bandwidth Estimation Workshop*, December 2003.

[10] Postel et. al., "Rfc 791: Internet protocol," *Request for Comments*, pp. 17–18, 1999.

[11] H.X. Nguyen and P. Thiran, "Active measurement for multiple link failures diagnosis in ip networks," in *Proceedings of Passive and Active Measurement Workshop (PAM)*, April 2004.

[12] T.H. Cormen, C. Stein, R.L. Rivest, and C.E. Leiserson, *Introduction to Algorithms*, McGraw-Hill Higher Education, 2001.

[13] M.R. Garey and D.S. Johnson, *Computers and Intractability: A Guide to the Theory of NP-Completeness*, W. H. Freeman & Co., 1979.

[14] T.H. Cormen, C.E. Leiserson, and R.L. Rivest, "Theorem 37.4, minimum set cover," *Introduction to Algorithms*, 1999.

[15] R. Teixeira, K. Marzullo, S. Savage, and G.M. Voelker, "In search of path diversity in isp networks," in *Proceedings of the ACM SIGCOMM Internet Measurement Conference*, October 2003.

Appendix

Algorithm 4: Finding out if a node, u in $G(V, E)$ is high arity or not.

```
Do a depth first search from u;
if there is a back edge to u from
  any other node
    declare u high arity;
else
    declare u single arity;
```

Proof: (Proof of Correctness) Consider the case when u is high arity. Then there exists a node (say v) to which there are more than one paths from u with different outgoing edges from u. In a depth first search from u, one of these edges will be selected first for traversal. In the depth first search run, consider that we are on the first edge from u which has a path to v, with an alternating path to the same node v as defined above. Because of the alternating path, there exists a cycle with nodes u and v on it. Since the edge on the alternating path outgoing from u is not yet traversed, it will become a back edge and will be detected as outlined in the algorithm.

Now consider that node u is not a high arity node. Also assume that we detect a back edge in the way outlined in the algorithm. Since the back edge was to node u itself, it means that node u is part of a cycle with two outgoing edges participating in the cycle. Thus, there is at least one node in the network (from the same cycle) which has two paths to u each path having a different outgoing edge from u. This means that u is a high arity node. This is a contradiction. Hence if u is an arity one node then one cannot detect a back edge as described in the algorithm.

■

Time Complexity: The complexity of depth first search is $\theta(|E| + |V|)$. Detecting a back edge involves going through all the neighbors of u. If the degree of u is k, then the cost for checking for a back edge is $\theta(k)$. Since k is bounded by $|E|$, the time complexity of our algorithm is $\theta(|E| + |V|)$.