Robust Multi-Source Network Tomography using Selective Probes

Akshay Krishnamurthy
Computer Science Department
Carnegie Mellon University
akshaykr@cs.cmu.edu

Aarti Singh
Machine Learning Department
Carnegie Mellon University
aartisingh@cmu.edu

***Abstract*—Knowledge of a network's topology and internal characteristics such as delay times or losses is crucial to maintain seamless operation of network services. Network tomography is a useful approach to infer such knowledge from end-to-end measurements between nodes at the periphery of the network, as it does not require cooperation of routers and other internal nodes. Most current tomography algorithms are single-source methods, which use multicast probes or synchronized unicast packet trains to measure covariances between destinations from a single vantage point and recover a tree topology from these measurements. Multi-source tomography, on the other hand, uses pairwise hop counts or latencies and consequently overcomes the difficulties associated with obtaining measurements for single-source methods. However, topology recovery is complicated by the fact that the paths along which measurements are taken do not form a tree in the network.**

Motivated by recent work suggesting that these measurements can be well-approximated by tree metrics, we present two algorithms that use selective pairwise distance measurements between peripheral nodes to construct a tree whose end-to-end distances approximate those in the network. Our first algorithm accommodates measurements perturbed by additive noise, while our second considers a novel noise model that captures missing measurements and the network's deviations from a tree topology. Both algorithms provably use $O(p \text{ polylog } p)$ pairwise measurements to construct a tree approximation on p end hosts. We present extensive simulated and real-world experiments to evaluate both of our algorithms.

I. Introduction

Knowledge of a network's topology and internal characteristics such as delay times and losses is crucial to maintaining seamless operation of network services. Yet typical networks of interest are incredibly large and decentralized so that these global properties are not directly available, but rather must be inferred from a small number of indirect measurements. Network tomography [1], [2] is a promising approach that aims to gather such knowledge using only end-to-end measurements between nodes at the periphery of a network without cooperation from core routers. Designing algorithms that reliably and accurately recover network characteristics from these measurements is an important research direction.

Most current methods focus on single source network tomography; they use similarity of delay or similarity of loss measurements from a single source to multiple nodes, caused by shared path segments, to infer a tree topology between the source and end nodes. The assumption of a tree topology is justified under the premise of shortest path routing from the source to each end node. These procedures either rely on infrequently deployed multicast probes ([3], [4], [5], [6]) or use a series of back-to-back unicast probes ([7], [8], [9], [10], [11]) that need to be carefully coordinated making the method sensitive to packet re-orderings and asynchrony between end nodes.

Multiple source network tomography is an alternative approach that uses measurements between pairs of end nodes that form an additive metric on a graph. Several network measures such as end-to-end delay, loss, or hop counts between pairs of end nodes form an (approximate) additive metric, as a path measurement is the sum of the measure along links constituting the path. It is possible to learn such metrics using light-weight probes such as hop counts extracted from packet headers [12] or pings. If the given measurements form an additive metric on an acyclic or tree graph, a variety of methods can be used to reconstruct the underlying structure [11], [13], [14]. However, typically, the underlying graph is not an exact tree as peering links between different network providers introduce cycles and violate the tree assumption.

Given the size and complexity of the Internet, the practicality of any network tomography algorithm should be evaluated not only by its noise tolerance and robustness to violations of any modeling assumptions, but also by its probing complexity (the number of probes needed as a function of the number of end hosts in the network). State-of-the-art methods for both single- and multi-source network tomography typically suffer in at least one of these directions. Many methods do not optimize and/or provide rigorous guarantees on the number of probes needed to recover the underlying graph structure, while others are not guaranteed to be robust to noisy measurements. Moreover, to the best of our knowledge, no method, with the exception of [15], [16], consider violations of the assumption that the underlying topology is a tree. In this paper, we address all of these deficiencies.

Specifically, we present two algorithms that use selective light-weight probes to construct a weighted tree whose path lengths provide a faithful representation of the pairwise measurements between end hosts in the network. While the additional nodes in the produced tree need not correspond to hidden network elements, such a representation enables distance approximations between unmeasured hosts, closest

neighbor/server selection, and topology-aware clustering all of which can improve performance of network services.

Motivated by recent work [15] showing that internet latency and bandwidth can be well *approximated* by path lengths on trees, our algorithms are designed to construct tree graphs and consequently exact tree metrics. However, we introduce two models to capture violations of the tree-metric assumption: (a) an *additive noise* model, where all measurements are corrupted by additive subgaussian noise, resulting in small deviations from the tree metric properties, and (b) a *persistent noise* model in which a fraction of the measurements are arbitrarily corrupted. The persistent noise model also captures the effects of missing measurements due to packet drops or unresponsive nodes. Even under these noise models, our algorithms have provable guarantees about correctness and probing complexity.

Our contributions can be summarized as follows:

1) We present algorithms for the multi-source network tomography problem that improve on existing work in at least one of two regards: our algorithms have provable correctness guarantees in the presence of noisy measurements, which can capture violations of the tree-metric assumption, and, by intelligent use of light-weight probes, they come with provable bounds on probing complexity.
2) Our first algorithm addresses the additive noise model. It uses $O(pl\log^2 p)$ pairwise measurements in the presence of noise and $O(pl\log p)$ measurements in the absence of noise, where p is the number of end hosts in the network and l is the maximum degree of any node, to construct a tree that accurately reflects the measurements. As our guarantees hold even for highly unbalanced tree structures, this improves on existing work [10], [11] that requires balanced-ness restrictions.
3) Under the persistent noise model, our second algorithm uses $O(pl\log^2 p)$ pairwise measurements to construct a tree approximation, even when a fixed fraction of the measurements are arbitrarily corrupted. Robustness to persistent noise, however, comes at the cost of requiring some balanced-ness of the underlying tree.

This paper is organized as follows. Section II discusses related work and comparisons to our algorithms. We provide background definitions and formally specify the multi-source tomography problem in section III. Our first algorithm that uses selective pairwise measurements to recover an unrooted, unbalanced tree topology is presented in section IV-A, along with an analysis of its probing complexity and tolerance to additive noise corrupting the measurements. In Section IV-B, we present our main algorithm, RISING (Robust Identification using Selective Information of Network Graphs) and analyze its robustness to persistent noise as well as its probing complexity. We validate the proposed algorithms using simulations as well as real Internet measurements from the King [17] and IPlane datasets [18] in section V and conclude in section VI. Due to space constraints, several proofs are deferred to supplementary material available online [19].

II. Related Work

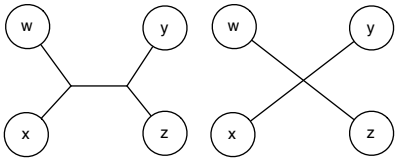
Initial work towards mapping the Internet topology was based on injecting TTL (Time-to-Live)-limited probe packets called `traceroutes` that record the exact path traversed by the packet [20], [21]. These `traceroute`-based approaches require routers to insert information into the packet header, and therefore they fail in the presence of uncooperative network elements. In particular, anonymous routers [22] and router aliases [23] do not augment packet headers, and firewalls as well as network address translation (NAT) boxes simply block `traceroute` packets.

Among the various algorithms for single-source tomography, two recent methods are particularly relevant to our work: the DFS-ordering algorithm of Eriksson et. al. [10] and the work of Ni et. al. [11]. The first provably uses $O(pl\log p)$ probes to recover a balanced l-ary tree topology; however, the authors make no claims about the correctness of the algorithm in the presence of noisy measurements. Ni et. al. present the Sequential Logical Topology (SLT) algorithm, that uses $O(pl\log p)$ ($O(pl\log^2 p)$ under additive noise) probes to recover balanced l-ary trees while also guaranteeing correct recovery of the topology when measurements are corrupted by additive noise. Our first algorithm improves on the work of Ni et. al. by relaxing the balanced-ness assumption while maintaining the same probing complexity.

In multi-source tomography, a number of algorithms ([24], [25], [26]) find Euclidean or non-Euclidean embeddings that accurately reflect the measurements. While some of these algorithms have strong probing complexity guarantees ([24]), they do not capture the inherent hierarchical structure of the network and thus may be less useful than algorithms that recover tree or more intuitive models. In addition to the embedding-based algorithms, the work of Rabbat and Nowak [27] casts the multi-source tomography problem as a set of statistical hypothesis test that differentiates topological structure between two senders and two receivers. While their approach is algorithmically more straightforward, they only identify the presence of a shared link between the senders and the receivers and cannot distinguish all possible topological configurations between four end hosts as we can.

If the measurements formed an additive tree metric, then a host of algorithms could be used to build a tree representation [13], [14], [28], some coming with probing complexity bounds. However, the tree metric assumption does not hold in practice, and as shown in [15], network measurements such as latency and bandwidth only *approximate* additive tree metrics. It is consequently important for us to design algorithms that are robust to violations of the tree metric properties.

Sequoia ([15]) is one algorithm designed for this purpose. Unfortunately, it comes with no guarantees on correctness in the presence of these violations, and while it seems to use only a limited number of probes in practice, it lacks probing complexity bounds. In this paper, we build on this line of work by designing an algorithm with theoretical guarantees on correctness and probing complexity. Another method that



(a) Quartet (b) Star

Fig. 1. Possible structures for four leaves in a tree. If $d(w,x)+d(y,z) < d(w,y)+d(x,z) = d(w,z)+d(x,y)$ then structure and labeling is that of (a). If $d(w,x)+d(y,z) = d(w,y)+d(x,z) = d(w,z)+d(x,y)$ then structure is a star (b).

addresses more general graph structures, beyond trees, was proposed recently in [16]. However, this method also does not attempt to optimize the probing complexity.

Our work, and network tomography in general, have strong connections to the task of learning the structure of latent variable graphical models and to problems in phylogenetic inference. For example, in [13] and [29], algorithms are proposed to learn tree-structured graphical models using pairwise empirical correlations obtained from measurements of variables associated with leaf nodes. Under this setup, the correlations form an exact, rather than approximate, tree metric. Moreover, due to the different measurement model, this work does not explicitly optimize the number of pairwise measurements used. Our first algorithm is indeed based on [13] and hence we call it PEARLRECONSTRUCT.

In phylogenetics, the task of learning an evolutionary tree using genetic sequence data from several extant species is closely related to the single-source tomography problem. Several algorithms, such as the neighbor-joining algorithm [11], [30], [31] have been applied to both problems. Also see [3], [32], and [33] for more details. To the best of our knowledge, the algorithms we propose are novel and do not exist in the phylogenetics literature.

III. BACKGROUND AND PROBLEM FORMULATION

Let $\mathcal{X} \triangleq \{x_i\}_{i=1}^p$ denote the end hosts in a network and let $d : \mathcal{X} \times \mathcal{X} \to \mathbb{R}^+$ be a function representing the true distances between the nodes, so that $d(x_i, x_j)$ is the distance, as measured in the network, between the hosts x_i and x_j.

Our work focuses on distance functions d that form approximate additive tree metrics. Specifically, let $\mathcal{T} = (\mathcal{V}, \mathcal{E}, c)$ be a weighted tree with vertices $\mathcal{V}$, edges $\mathcal{E}$ and weights c, for which $\mathcal{X}$ is the set of leaves. To avoid identifiability issues, our focus will be on **minimal** trees, for which each internal node has degree ≥ 3 and each edge has strictly positive weight. An additive tree metric on $\mathcal{X}$ is a function $d_{\mathcal{T}}$ such that $d_{\mathcal{T}}(x_i, x_j) \triangleq \sum_{(x_k,x_l)\in \text{Path}(x_i,x_j)} c(x_k, x_l)$, that is the distance between two points is the sum of the edge weights along the *unique* path between them. A useful property of additive tree metrics is the **four-point condition**:

Definition 1. *A metric $(\mathcal{X}, d)$ satisfies the **four-point condition** (4PC) if for any set of points $w, x, y, z \in \mathcal{X}$ ordered such that $d(w,x)+d(y,z) \leq d(w,y)+d(x,z) \leq d(w,z)+d(x,y)$, $d(w,y)+d(x,z) = d(w,z)+d(x,y)$.*

The 4PC is related to the *quartet test*, a common technique for resolving tree structures (Indeed, there are a host of quartet-based algorithms for phylogenetic inference, for example [34]). The quartet test is used to identify the structure between any 4 leaves in a tree using only the pairwise distances between those leaves. It is easy to see that any four leaves either form a structure like that in Figure 1(a) or a star (Figure 1(b)), and using the 4PC we can identify not only which structure but also the correct labeling of the leaves (See Figure 1 for more details).

Any metric that satisfies the four-point condition is a tree metric for some tree. Unfortunately, latency and hop counts in real networks do not exactly fit into this framework, but only approximate tree metrics [15]. One characterization of this approximation is the 4PC-ϵ condition which requires $d(w,z)+d(x,y) \leq d(w,y)+d(x,z)+2\epsilon\min\{d(w,x),d(y,z)\}$ for some parameter ϵ instead of the equality in Definition 1. Metrics for which ϵ values are low can be well approximated by tree metrics, and empirical studies showing that real network measurements satisfy 4PC-ϵ for low ϵs motivates the use of this model.

In this work, we take a more statistical approach and instead assume that $d(x_i, x_j) = d_{\mathcal{T}}(x_i, x_j) + g(x_i, x_j)$ where the function g models the networks deviations from a tree metric. This approach allows us to not only formally state the multi-source network tomography problem but also to make rigorous guarantees about the performance of our algorithms. We focus on two models for these deviations:

1) **Additive Noise Model** – In this model, $g(x_i, x_j)$ is drawn from a subgaussian with σ^2 as a scale factor[1]. The small perturbation model studied in single source network tomography (See for example [11]) is similar to this as subgaussian noise is bounded, with high probability, by a small constant (depending on σ^2). This model captures the inherent randomness in certain types of measurements, such as latencies. Under this formulation we allow each measurement to be observed several (n) times.
2) **Persistent Noise Model** – Here $g(x_i, x_j) = 0$ with probability q, independent of all other x_i and x_j, and with probability $1-q$, $g(x_i, x_j)$ is arbitrary (or adversarially) chosen. We believe this is a reasonable model of how the measurements do not exactly form a tree metric, due to violations caused by peering links, unresponsive nodes or missing measurements. To more accurately model violations of tree metric assumptions, multiple request for a measurement all reveal the same (possibly incorrect) value, so we only obtain one sample of each measurement. To the best of our knowledge, there are no other efforts to study this noise model.

While [15] capitalized on the fact that $\sim 80\%$ of the quartets satisfy 4PC with a small perturbation ϵ, we also note that

[1]A random variable X is **subgaussian** with scale factor σ^2 if $\mathbb{P}(\exp(tX)) \leq \exp(\sigma^2 t^2/2)$. This family encompasses both gaussian and bounded random variables.

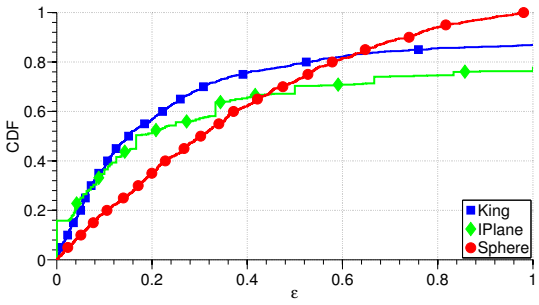


Fig. 2. CDFs of ϵ values in the 4PC-ϵ condition for two real world datasets (King [17] and IPlane datasets [18]) along with a dataset of points drawn uniformly from the surface of a sphere, where geodesic distance defines the metric.

$\sim 20\%$ of the quartets do not satisfy the 4PC even with $\epsilon = 1$, which corresponds to triangle inequality violations (See Figure 2 where we plot the CDF of ϵ values for two real-world datasets). We attempt to address both of these phenomena with our two noise models: additive noise to capture the small deviations from 4PC and persistent noise to capture the larger perturbations. While in this paper, we addresses these two types of noise separately, our second algorithm can be modified to handle both types of noise simultaneously. To keep the exposition simple, we defer that case and all detailed proofs to a longer version of the paper.

We are now prepared to formally specify our problem:

Problem 1. *Given a metric space $(\mathcal{X}, d)$ equipped with a metric $d = d_{\mathcal{T}} + g$ for some tree $\mathcal{T}$, recover $\mathcal{T}$ and $d_{\mathcal{T}}$ while minimizing the number of measurements of d.*

In this paper, we develop algorithms for this problem under the assumption that g corresponds to one of the models above. Before we present our algorithms, we define several quantities that appear in our algorithms and the subsequent analysis. For any tree $\mathcal{T}$, let $\text{lvs}(\mathcal{T})$ denote the set of leaf nodes of $\mathcal{T}$ and let $\deg(\mathcal{T})$ denote the maximum degree of the tree. For convenience will we define $l \triangleq \deg(\mathcal{T})$.

For any three nodes x, y, and z in a tree $\mathcal{T}$ let $\text{ancestor}(x, y, z)$ be the unique node that is the shared common ancestor of x, y and z. This node is the unique point along which the paths between x, y and z intersect in $\mathcal{T}$ and distances to this point can be computed by (where $a = \text{ancestor}(x, y, z)$):

$$d_{\mathcal{T}}(x, a) \triangleq \frac{1}{2}(d_{\mathcal{T}}(x, y) + d_{\mathcal{T}}(x, z) - d_{\mathcal{T}}(y, z)) \tag{1}$$

To avoid propagation of additive noise in ancestor computations, we only use distances between true leaf nodes (nodes in $\mathcal{X}$). To compute the ancestor and associated distances between three nodes x, y, z, some of which may not be leaves, we use a *surrogate* leaf node for each non-leaf node in the computation. A surrogate leaf node for x is one for which x is on the path between that leaf and both y and z. The restriction to minimal trees guarantees the existence of surrogate leaf nodes.

Algorithm 1 PEARLRECONSTRUCT($\mathcal{X}$, d, γ)

Initialize T_3 as a star tree on x_1, x_2, x_3
for $i = 4 \dots p$ **do**
  $T_i = \text{PearlAdd}(x_i, T_{i-1}, d, \gamma)$
end for
return T_p

Algorithm 2 PEARLADD(x_i, T_{i-1}, d, γ)

$T_c = T_{i-1}$
while $|\text{lvs}(T_c)| > 2$ **do**
  Choose a subtree T_{out} such that:
  $\frac{|\text{lvs}(T_c)|}{\deg(T_c)+1} < |\text{lvs}(T_c) \setminus \text{lvs}(T_{out})| < \frac{|\text{lvs}(T_c)|\deg(T_c)}{\deg(T_c)+1}$.
  $r \leftarrow$ parent of T_{out} in T_c
  Let $T_{sub} \neq T_{out}$ be any other subtree of T_c rooted at r and choose $x_k \in \text{lvs}(T_{sub})$, $x_j \in \text{lvs}(T_{out})$.
  $y \leftarrow \text{ancestor}(x_i, x_j, x_k)$, compute $d(x_i, y)$, $d(x_j, y)$, and $d(x_k, y)$, using surrogates as needed.
  If $d(x_j, y) + \gamma/2 < d(x_j, r)$, then $T_c \leftarrow T_{out} \cup \{r\}$
  If $d(x_k, y) + \gamma/2 < d(x_k, r)$, then $T_c \leftarrow T_{sub} \cup \{r\}$
  Otherwise $T_c \leftarrow T_c \setminus \{T_{sub} \cup T_{out}\}$
end while
if $|T_c| = 1$ **then**
  Attach x_i to T_c with edge length $d(x_i, y)$.
else
  T_c has two nodes r and r'. Choose leaves x_k and x_j such that r is on the path between x_k and r', and r' is on the path between x_j and r.
  $y \leftarrow \text{ancestor}(x_i, x_k, x_j)$.
  If $|d(x_k, y) - d(x_k, r)| < \gamma/2$, then attach x_i to r.
  If $|d(x_j, y) - d(x_j, r')| < \gamma/2$, then attach x_i to r'.
  Otherwise, insert y between r and r' (with edge weights $d(x_k, y) - d(x_k, r)$ and $d(x_j, y) - d(x_j, r')$) and attach x_i to y with edge weight $d(x_i, y)$.
end if
return T_{i-1} updated to include x_i.

IV. ALGORITHMS

In this section we describe our algorithms for multi-source network tomography and present our theoretical guarantees on correctness and probing complexity. Our first algorithm, PEARLRECONSTRUCT addresses the additive noise model while our second, RISING addresses the persistent model.

A. Additive Noise

The idea behind our first algorithm is to construct the tree $\mathcal{T}$ by iteratively attaching the leaves. To add leaf x_i, we perform an *intelligent* search to find a pair of nodes x_j, x_k such that the distance between x_i and $\text{ancestor}(x_i, x_j, x_k)$ is minimized. This information, along with the fact that x_i is not in the same subtree as either x_j or x_k (which we also determine), tell us how to add x_i to the tree.

Our search is intelligent in that we choose x_j and x_k to rule out large portions of the tree at every step. Specifically, by choosing a point with fairly balanced subtrees (known as

the *pearl point*), we can determine which of these subtrees x_i belongs to and focus our search to a subtree that is a fraction of the original size, using a constant number of measurements. Formally, for any directed instance of a tree $\mathcal{T}$, the pearl point is the internal node in a tree for which the number of leaves below that node is between $|\text{lvs}(\mathcal{T})|/(\deg(\mathcal{T})+1)$ and $|\text{lvs}(\mathcal{T})|\deg(\mathcal{T})/(\deg(\mathcal{T})+1)$. As we show, using the pearl point results in a strong upper bound on the number of measurements used while ensuring correctness of the algorithm.

PEARLRECONSTRUCT is related to the algorithm in [13], the Sequential Logical Topology (SLT) algorithm of [11], and the Sequoia algorithm of [15]. Our intelligent search parallels that of [13], but by using triplet tests rather than quartet tests and by incorporating slack into our search, PEARLRECONSTRUCT is robust to additive noise while their algorithm is not. On the other hand, the SLT algorithm is robust to noise, but they do not begin their search at the pearl point of the tree, and thus their probing complexity guarantees only hold for balanced trees, while our guarantees are more general. The Sequoia algorithm also adopts some of the same ideas, but since their search is based on heuristics, they do not provide upper bounds on the number of probes used.

Our algorithms have a parameter γ that is a lower bound on the edge weights in the true tree $\mathcal{T}$. This parameter helps us distinguish two nodes separated by a short edge in the presence of noise. Similar parameters have been used in existing tree reconstruction algorithms that are robust to additive noise [11].

Pseudocode for PEARLRECONSTRUCT is shown in Algorithms 1 and 2. We now present our theoretical guarantees for PEARLRECONSTRUCT; note that proofs of some technical lemmas are deferred to the supplementary materials [19].

Theorem 1. *Let $(\mathcal{X}, d)$ be an metric with $|\mathcal{X}| = p$ where $d = d_{\mathcal{T}} + g$ for a tree $\mathcal{T}$ with minimum edge length $\geq \gamma$ and where $g(x_i, x_j)$ is a subgaussian random variable with scale factor $\leq \sigma^2$. Let $\{d^{(i)}\}_{i=1}^n$ be samples of d. Define the sample distance metric $\hat{d}$ where $\hat{d}(x_j, x_k) \triangleq \frac{1}{n}\sum_{i=1}^n d^{(i)}(x_j, x_k)$. If*

$$n > 18\frac{\sigma^2}{\gamma^2}\log(2p^2/\delta) \tag{2}$$

then with probability $\geq 1-\delta$, PEARLRECONSTRUCT *on input $(X, \hat{d}, \gamma)$, recovers $\mathcal{T}$ and $d_{\mathcal{T}}$.*

Proof: First, we consider the noiseless scenario. In the supplementary material [19]we show that PEARLRECONSTRUCT on input $(X, d_{\mathcal{T}})$ deterministically recovers $\mathcal{T}$. Our proof of this Lemma follows that of [13]. Specifically, we show that adding node x_i to T_{i-1} results in not only the correct structure but also the correct distances between $x_1, \ldots x_i$. We arrive at the result by iterative applications of this argument.

In noisy setting, we can no longer deterministically guarantee correct recover of $\mathcal{T}$, but instead require a probabilistic analysis. In the algorithm, we choose three nodes x_i, x_j and x_k and compute distances between these nodes and $y \triangleq \text{ancestor}(x_i, x_j, x_k)$. We need to be able to correctly determine if y lies between the root r and x_j, between r and x_k, or elsewhere in the tree. We therefore seek to bound $|\hat{d}(x_k, y) - d(x_k, y)|$ and $|\hat{d}(x_j, y) - d(x_j, y)|$.

To arrive at these bounds, we first derive concentration inequalities for the directly observed measurements. Specifically, by application of the Gaussian Tail Inequality and the union bound we have that with probability $\geq 1-\delta$:

$$|\hat{d}(x_i, x_j) - d(x_i, x_j)| \leq \sqrt{\frac{2\sigma^2\log(2p^2/\delta)}{n}} \tag{3}$$

for all leaves x_i, x_j, $i, j \in [p]$. Using this bound along with Equation 1, immediately reveals that the distance in the estimated tree between any two nodes deviates from the correct distance by at most $\frac{3}{2}\sqrt{\frac{2\sigma^2\log(2p^2/\delta)}{n}}$.

In order for the algorithm to work, we need to ensure that we can identify when the ancestor node y equals the root node r, in spite of the deviations. If:

$$\gamma > 3\sqrt{\frac{2\sigma^2\log(2p^2/\delta)}{n}} \tag{4}$$

then with high probability we will not confuse the nodes y and r, since distances to each node only deviate by half that. Inverting Equation 4 yields the bound on n in the theorem. ■

Theorem 2. PEARLRECONSTRUCT *uses $O(pl\log^2 p)$ pairwise measurements, where $l \triangleq$ deg$(\mathcal{T})$.*

Proof: We study the `add` procedure. By Lemma 1 in [13], we know that for any T_c there exists a subtree T_{out} for which:

$$\frac{|\text{lvs}(T_c)|}{\deg(T_c)+1} < |\text{lvs}(T_c) \setminus \text{lvs}(T_{out})| < \frac{|\text{lvs}(T_c)|\deg(T_c)}{\deg(T_c)+1}$$

Let $l_c = \deg(T_c)$. The fact that $|\text{lvs}(T_c) \setminus \text{lvs}(T_{out})| < \frac{|\text{lvs}(T_c)|l_c}{l_c+1}$ means that $|\text{lvs}(T_{out})| \geq \frac{|\text{lvs}(T_c)|}{l_c+1}$. Writing T_c^i to denote T_c after i iterations of the loop, we see that no matter how the search proceeds, $|\text{lvs}(T_c^i)| \leq \frac{l_c}{l_c+1}|\text{lvs}(T_c^{i-1})|$.

Thus the number of iterations required to place x_i in T_{i-1} is at most $\log_{\frac{l_c+1}{l_c}}(i-1) \leq l_C\log(i-1)$ [2]. Since each loop iteration uses a constant number of pairwise distance measurements, l_c is upper bounded by l the maximum degree of $\mathcal{T}$, and we call the `add` at most p times, we see that the probing complexity is $O(pl\log p)$ in the absence of noise.

Finally, recall from Theorem 1 that if n is $O(\log p)$ we can guarantee exact recover of the tree. We must therefore observe each measurement $O(\log p)$ times and including this multiplicative factor results in the stated bound. ■

B. Persistent Noise

For the persistent noise model, we propose a divisive algorithm; it recursively partitions the leaves into groups corresponding to subtrees of $\mathcal{T}$. Each partitioning step identifies one internal node in the tree, and by repeated applications of our algorithm, we identify all internal nodes that satisfy certain properties (detailed in Theorem 3).

A top-down partitioning algorithm allows us to use voting schemes that are robust to persistent noise. Specifically, we

[2] since $\log(1+1/l_c) < 1/l_c$

Algorithm 3 RISING($\mathcal{X}, d, m$)

Randomly choose $M \subset \mathcal{X}$ with $|M| = m$
For $x_i, x_j \in M$, compute $s(x_i, x_j) = \max_{x_k \in M} |\{x_{k'} \in M : d(x_i, x_k) - d(x_j, x_k) = d(x_i, x_{k'}) - d(x_j, x_{k'})\}|$
Run Single Linkage Clustering using s as similarities to partition M into a set of clusters $\mathcal{C}$ with $|\mathcal{C}| = 3$.
for $x_i \in \mathcal{X} \setminus M$ **do**
  VOTE(x_i, $\mathcal{C}, d$)
end for
Initialize T with 1 node r
for $C \in \mathcal{C}$ **do**
  $T_{sub} \leftarrow$ SPLIT($C, \mathcal{X} \setminus C, d, m$).
  Choose clusters $C_1, C_2 \in \mathcal{C} \setminus C$
  weight(r, root(T_{sub})) $\leftarrow$ EDGELENGTH(C_1, C_2, T_{sub}, d)
end for
return T

Algorithm 4 SPLIT($\mathcal{S}$, $\mathcal{Y}, d, m$)

Randomly choose $M \subset \mathcal{S}$ with $|M| = m$
For each $x_k \in M$, draw $Z(k)$ randomly from $\mathcal{Y}$.
For $x_i, x_j \in M$, compute $s(x_i, x_j) = |\{x_k \in M : d(x_i, x_k) - d(x_j, x_k) = d(x_i, x_{Z(k)}) - d(x_j, x_{Z(k)})\}|$.
Run Single Linkage Clustering using s as similarities to partition M into clusters $\mathcal{C}$ with $|\mathcal{C}| = 2$
for $x_i \in \mathcal{S} \setminus M$ **do**
  VOTE(x_i, $\mathcal{C} \cup \{\mathcal{Y}\}, d$)
end for
Initialize T with 1 node r
for $C \in \mathcal{C}$ **do**
  $T_{sub} \leftarrow$ SPLIT($C, \mathcal{Y} \cup (\mathcal{S} \setminus C), d, m$).
  Choose $C' \in \mathcal{C} \setminus C$
  weight(r, root(T_{sub})) $\leftarrow$ EDGELENGTH(C', $\mathcal{Y}$, T_{sub}, d)
end for
return T

identify groups of nodes by repeatedly performing quartet or triplet tests and deciding on the structure agreed on by the majority. However, to ensure that these groups are sufficiently large, we require a balancedness condition:

Definition 2 (Balance Factor)**.** *We say that $\mathcal{T}$ has **balance factor** η if there exists a node r such that for all internal nodes h (including r), with subtrees $T_1(h), \ldots, T_k(h)$ directed away from r, $\eta \triangleq \max_h \frac{\max_i |lvs(T_i(h))|}{\min_i |lvs(T_i(h))|}$.*

To identify a single internal node r our algorithm randomly samples a subset of the leaves, forms a clustering of this subset, and then places each remaining leaf into one cluster. After recursively partitioning each cluster, we compute edge lengths using a voting scheme. In the clustering phase, we compute a similarity function s on the sampled leaves where $s(x_i, x_j)$ is large if the two leaves belong in the same subtree of $\mathcal{T}$, viewed with r as the root. We partition the sampled nodes into two clusters in most cases (to find the first split we partition into three). Each of these clusters is comprised of leaves from one or more subtrees rooted at r, but the leaves from any of the subtree are contained wholly in one cluster.

Algorithm 5 VOTE(x, $\mathcal{C}, d$)

Let $C_1, C_2, C_3 \in \mathcal{C}$
$VC_1, VC_2, VC_3 \leftarrow 0$
for $n \in \{1, \ldots, \min_{C \in \mathcal{C}} |C|\}$ **do**
  Choose $x_1 \in C_1, x_2 \in C_2, x_3 \in C_3$.
  $VC_i \leftarrow VC_i + 1$ if x pairs with x_i w.r.t. the other two.
  If x_i, x_1, x_2, x_3 form a star, ignore this vote.
end for
Place x in C_i where $VC_i = \text{argmax}\{VC_1, VC_2, VC_3\}$

Algorithm 6 EDGELENGTH(C_1, C_2, T_{sub}, d)

$C_L \leftarrow$ leaves in one subtree of T_{sub}
$C_R \leftarrow$ leaves in another subtree of T_{sub}
for $n \in \{1, \ldots \min\{m, |C_1|, |C_2|, |C_L|, |C_R|\}$ **do**
  Draw $w \in C_1, x \in C_2, y \in C_l, z \in C_R$
  Record $\frac{1}{2}d(w, y) + d(x, z) - d(w, x) - d(y, z)$
end for
Return the most frequently occuring recorded value

Once we have clustered the sampled nodes, we use voting to determine the group assignments for the remaining nodes. To place a node x_i, we compute quartet structures (See Figure 1) between x_i and x_j, x_k, x_l (each from different clusters) and record which node x_i paired with in the quartet test. We place x_i into the cluster that most commonly paired with x_i.

The computations required to find the initial partition of leaves are slightly different from those required for subsequent splits. To highlight these differences, we present pseudocode for recovering the first partition in Algorithm 3 and for subsequent partitions in Algorithm 4. These algorithms rely on two subroutines which we show in Algorithms 5 and 6.

Before presenting our theoretical guarantees, we remark that while our results analyze RISING in the presence of only persistent noise, with slight modifications the algorithm can be made robust to both persistent and additive noise. The main change would involve incorporating slack into the quartet tests, much like we have done in PEARLRECONSTRUCT. The analysis for this modified algorithm would incorporate the techniques used in Theorem 1 (specifically concentration of subgaussian random variables) into our current proofs. However, for clarity of presentation, our analysis guarantees the correctness of RISING under only persistent noise.

Theorem 3. *Let $(\mathcal{X}, d)$ be a metric where $d \triangleq d_{\mathcal{T}} + g$ for a tree $\mathcal{T}$ with bounded balance factor η and where g is from the persistent noise model with probability of an uncorrupted entry $\geq q$ with $q^6 > C_{\eta,l}$. Then with probability $\geq 1 - 1/p$, every execution of* RISING *and* SPLIT*, with parameter m, will correctly identify an internal node provided that:*

$$m > c_{\eta,l} \frac{\log(pm)}{(q^6 - C_{\eta,l})^2} \tag{5}$$

where $1/2 \leq C_{\eta,l} < 1, c_{\eta,l}$ are constants depending on η and l.

Remark In the absence of noise, we can choose m to be a function of $|\mathcal{S}|$, the subset of leaves passed into the SPLIT routine. However, with noise, m must be $\Omega(\log p)$ and if $\mathcal{S}$ is too small for this, then $\mathcal{S}$ cannot be further resolved, and thus $\log p$ limits the resolution to which the structure can be resolved.

Remark In the supplementary material [19], we give a precise characterization of $C_{\eta,l}$, which plays a critical role in RISING's robustness to noise. While $C_{\eta,l} < 1$ for all values of η and l, it grows with these quantities. Specifically, the minimum value for $C_{\eta,l}$ is $1/2$, which happens when $\eta = 1$ and $l = 2$.

Our proof strategy is to analyze each phase – sampling, clustering and voting. Here, we outline that analysis of each section; we defer all proof details to the supplementary material [19]. In the sampling phase we use concentration inequalities to show that with high probability the balance factor η is not significantly perturbed. This result is necessary for the clustering phase of the algorithm, which is only guaranteed to succeed if the balance factor is sufficiently small.

In the clustering phase, we show that if q, the probability of an uncorrupted entry is sufficiently large, then the Single Linkage algorithm will identify clusters that correspond to the subtrees (or groups of subtrees) of the internal node we hope to recover. While we do not make any guarantees about grouping the leaves associated with two different subtrees, we remark that this does not affect our subsequent analysis and that these subtrees will be separated in later calls to SPLIT, allowing us to accommodate l-ary trees.

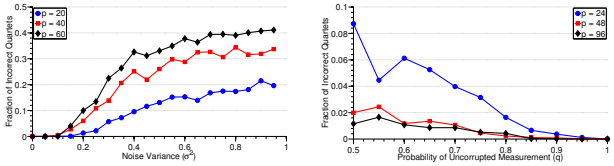
The similarity function used by Single Linkage is the quantity s defined in Algorithms 3 and 4. The analysis for this phase involves showing that $s(x_i, x_j)$ for two leaves that belong in the same subtree is always greater than $s(x_i, x_k)$ and $s(x_j, x_k)$ for every node x_k that does not lie in the subtree. This implies that we will merge clusters containing x_i and x_j before we merge either of these clusters with one containing a node that does not belong in the subtree. Applying this argument to each pair of leaves proves that Single Linkage will correctly identify the clusters.

The condition that η is bounded ensures that, for m large enough, these quantities are well-separated in the absence of noise. With noise, ensuring this gap exists with high probabily requires a condition on m that is subsumed by Equation 5, and that the noise is not too excessive, i.e. $q^4 > C_{\eta,l}$.

For the voting phase, we claim that any single round of voting is correct with probability q^6, that is if every voting node has uncorrupted measurements. Again using concentration inequalities we can show that if $q^6 > C_{\eta,l}$, then for m large enough we will be able to place a node into the correct cluster. This results in the condition on m in Equation 5.

Finally, via union bounds, we apply these arguments to every internal node in $\mathcal{T}$ that meets the conditions on m.

We note that we do not explicitly prove the correctness of EDGELENGTH but similar techniques to the ones we use above can be used to make this claim.



(a) PEARLRECONSTRUCT (b) RISING

Fig. 3. Noise Thresholds for PEARLRECONSTRUCT and RISING.

Theorem 4. *On trees with bounded balance factor,* RISING *uses $O(pml \log p)$ measurements where l is the maximum degree of the tree $\mathcal{T}$.*

Proof: We will analyze each level of the tree in turn. Since η is bounded, there are $O(\log p)$ levels of the tree.

At each level, let $\mathcal{C}$ be the set of all groups we are trying to split at this level, that is each $C \in \mathcal{C}$ is the set of nodes passed in as the first parameter to SPLIT, or in the case of the first call, $\mathcal{C}$ just contains one set with all of the nodes. For each group $C \in \mathcal{C}$ let p_C denote the number of nodes in C and let m_C denote the value of the parameter m which can be a function of $|C|$ [3].

For each cluster C, we require $m_C(m_C+1)/2$ measurements between sampled nodes and, in SPLIT, an additional m_C measurements from the set $\mathcal{Y}$. In the voting phase, we vote on $p_C - m_C$ nodes and for each node we require $m_C + 1$ measurements to the sampled nodes and to one node in $\mathcal{Y}$. Putting this together, we have that at any level, we use:

$$\sum_{C \in \mathcal{C}} \frac{m_C(m_C+1)}{2} + m_C + (p_C - m_C)(m_C+1) \quad (6)$$

$$\leq \sum_{C \in \mathcal{C}} p_C(m_C+1) \leq p(m+1) \quad (7)$$

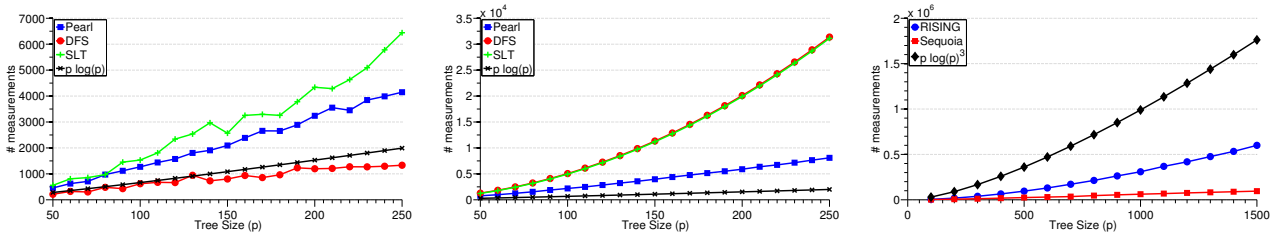
as long as $m_C > 1$ for all C, and where $m \triangleq m_p$ is the value of m passed into the call to RISING, i.e. it is the largest value of m across all calls to RISING and SPLIT. Here we used that $\sum_{C \in \mathcal{C}} p_C = p$. Thus we see that regardless of the balancedness of the tree, at each level we use $O(pm)$ measurements, and as described above, there are $O(\log p)$ levels resulting in a measurement complexity of $O(pm \log p)$. The factor of l arises because each call to SPLIT only splits the subtrees of a node into two groups; it may take up to l calls to recover each internal node.

Lastly, we can compute edge lengths using $O(m)$ measurements. Since this is dominated by the above bounds, we ignore this dependence. ■

V. EXPERIMENTS

We perform several experiments on simulated and real-world topologies to assess the validity of our theoretical results and to demonstrate the performance of our algorithms. We study how increasing noise affects our algorithms ability to correctly recover the topology and also how the number of measurements used compares to related algorithms.

[3]Specifically $m = m(|C|)$ can be any increasing function of $|C|$



(a) Probing Complexity on random Trees (b) Probing Complexity on Caterpillar Trees (c) Probing Complexity of RISING

Fig. 4. Measurements used as a function of p for PEARLRECONSTRUCT, RISING, DFS Ordering [10], SLT [11], and Sequoia [15]

A. Simulations

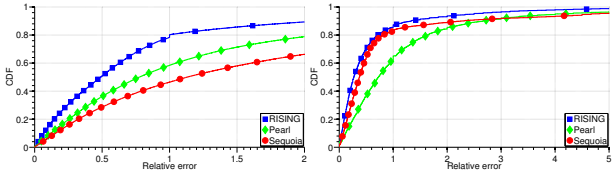
In simulations, we demonstrate how our algorithms tolerate noise, how this tolerance scales with p, and additionally how the number of measurements used scales with p. For these experiments, we generate tree topologies and obtain pairwise distances by computing unweighted path lengths along the tree to represent hop counts in a network. We then perturb this pairwise distance matrix with additive or persistent noise and run our algorithms on this perturbed matrix. We assess the correctness of our algorithms by computing the fraction of quartets for which the structure in the reference tree matches that in the algorithm's output.

For RISING, in simulations we always choose $m = \log^2 |\mathcal{S}|$ (even with noise), which as mentioned, satisfies the conditions of Theorem 3 in the absence of noise. For our real world experiments, we use $m = \log p$.

Our first experiment studies how PEARLRECONSTRUCT and RISING perform in the presence of noise. In Figures 3(a) and 3(b) we plot the fraction of incorrect quartets averaged over 20 trials for PEARLRECONSTRUCT and RISING respectively, as a function of the noise for different values of p. In Figure 3(a) we verify three properties of PEARLRECONSTRUCT: (a) in the absence of noise, it deterministically recovers the true topology as predicted by Lemma 4.1, (b) as the noise variance increases, PEARLRECONSTRUCT becomes less accurate, (c) on larger topologies, PEARLRECONSTRUCT requires lower noise variance. This last properties follows from Equation 2 since if n is constant (we took $n = 1$ for these experiments), we require $\sigma^2 = O(\frac{1}{\log p})$ in order to guarantee successful recovery, and this upper bound decreases with p.

For RISING, in Figure 3(b), we observe the opposite phenomenon; larger topologies can tolerate more persistent noise. This matches our bounds in Theorem 3, which allows q to approach a constant as $m, p \to \infty$. As before, we also observe that in the absence of noise, we deterministically recover the underlying topology, although we note that we used balanced binary trees for these experiments. For highly unbalanced trees, we cannot make this deterministic guarantee.

To assess the probing complexity of our algorithms, we record how many measurements each algorithm uses as a function of p, in the absence of noise. These plots are shown in Figure 4. As is noticeable in Figure 4(a), the probing complexity for PEARLRECONSTRUCT appears to be $O(p \log p)$. We also show the probing complexity for the DFS Ordering algorithm of Eriksson et al [10] and the Sequential Logical Topology (SLT) algorithm [11], both of which are single-source tomography methods with provable $O(p \log p)$ complexity on balanced trees. The trees used here are randomly generated, and we see that the SLT algorithm performs worse that PEARLRECONSTRUCT, while DFS Ordering seems to use a constant multiplicative factor fewer probes.



(a) Relative Error on King (b) Relative Error on iPlane

Fig. 5. CDF of relative error on King (a) and iPlane (b) datasets.

However, in the worst case, PEARLRECONSTRUCT enjoys considerable advantage over both SLT and DFS Ordering as can be seen in Figure 4(b). In this experiment, we used highly unbalanced trees and we see that the probing complexity of both SLT and DFS Ordering scale at $O(p^2)$, while PEARLRECONSTRUCT continues to scale at $O(p \log p)$.

In Figure 4(c), we compare RISING to the Sequoia algorithm of [15]. While Sequoia comes with no guarantees about correctness or probing complexity, it appears to use very few measurements in practice. RISING on the other hand appears to use a multiplicative factor of $\log p$ more probes than Sequoia, which we confirmed empirically. However, as we show in our real world experiments, Sequoia is less robust to noise, which demonstrates the need to use additional measurements to overcome noise. We also emphasize that RISING comes with guarantees on correctness in the presence of noise while Sequoia does not.

B. Real World Experiments

In addition to verifying our theoretical results, we are interested in assessing the practical performance of our algorithms on real world data. We use two network measurement data sets: the King dataset [17] of pairwise latencies and a dataset of hop counts between PlanetLab [35] hosts measured using iPlane [18]. We selected a 500-node subset of the 1740-node King dataset. The iPlane dataset consists of 193 end hosts.

We ran three algorithms, PEARLRECONSTRUCT, RISING, and Sequoia, on both datasets and plot the distribution of *relative error* values for each algorithm. Given the constructed tree metric $(X, \hat{d})$ and the true metric (X, d), we measure

Dataset	Hosts	Total	Pearl	RISING	Sequoia
King	500	125250	8321	43608	42599
iPlane	194	18721	2480	12309	11574

TABLE I
Measurements used on real world data sets

relative error for each pairwise distance as $\frac{|\hat{d}(x_i,x_j)-d(x_i,x_j)|}{d(x_i,x_j)}$. This quantity reflects how well the tree metric approximates the true distances in the network. These plots are shown in Figures 5(a) and 5(b). We see that on both datasets, RISING outperforms both Sequoia and PEARLRECONSTRUCT, with substantial improvements on the King dataset. PEARLRECONSTRUCT performs moderately well on both datasets.

Lastly, we recorded the number of measurements used by the algorithms on the two datasets in Table I. Note that Sequoia can be used to build many trees where the recovered pairwise distances is the median distance across all trees. To ensure a fair comparison, we build several trees so that Sequoia and RISING use a similar number of measurements. However, even with several trees, RISING performs better than Sequoia.

VI. CONCLUSION

In this paper we study the multi-source network tomography problem. We develop two algorithms, with theoretical guarantees, to construct tree metrics that approximate measurements between end hosts in a network. We also demonstrate the effectiveness of these algorithms on real world datasets.

There are several directions for future work. One restriction with the RISING algorithm is the balancedness requirement and it is an open problem to design an algorithm that is robust to persistent noise with correctness guarantees even for unbalanced trees. Another interesting direction is to approximate pairwise measurements with general graphs rather than trees, using knowledge of real-world network structures to avoid identifiability issues. We look forward to exploring both of these lines of work.

ACKNOWLEDGMENTS

The authors would like to thank Mark Crovella and Brian Eriksson for helpful comments. This research is supported in part by AFOSR under grant FA9550-10-1-0382 and NSF under grant IIS-1116458. AK is supported in part by an NSF Graduate Research Fellowship.

REFERENCES

[1] Y. Vardi, "Network Tomography: Estimating Source-Destination Traffic Intensities from Link Data," *Journal of the American Statistical Association*, 1996.
[2] R. M. Castro, M. J. Coates, G. Liang, R. Nowak, and B. Yu, "Network Tomography: Recent Developments," *Statistical Science*, 2004.
[3] S. Bhamidi, R. Rajagopal, and S. Roch, "Network Delay Inference from Additive Metrics," *Random Structures and Algorithms*, 2010.
[4] N. G. Duffield, J. Horowitz, and F. Lo Prestis, "Adaptive Multicast Topology Inference," *Proceedings IEEE INFOCOM 2001*, 2001.
[5] N. Duffield and F. L. Presti, "Network Tomography from Measured End-to-End Delay Covariance," *IEEE/ACM Transactions on Networking*, 2004.
[6] N. G. Duffield, J. Horowitz, F. Lo Presti, and D. Towsley, "Multicast Topology Inference from Measured End-to-End Loss," *IEEE Transactions on Information Theory*, 2002.
[7] M. J. Coates, R. M. Castro, and R. D. Nowak, "Maximum Likelihood Network Topology Identification from Edge-Based Unicast Measurements," *ACM SIGMETRICS*, 2002.
[8] Y. Tsang, M. Yildiz, P. Barford, and R. Nowak, "Network Radar: Tomography from Round Trip Time Measurements," in *ACM SIGCOMM Internet Measurements Conference*, 2004.
[9] N. Duffield, F. L. Presti, V. Paxson, and D. Towsley, "Network loss tomography using striped unicast probes," *IEEE/ACM Transactions on Networking*, 2006.
[10] B. Eriksson, G. Dasarathy, P. Barford, and R. Nowak, "Toward the Practical Use of Network Tomography for Internet Topology Discovery," *Proceedings IEEE INFOCOM*, 2010.
[11] J. Ni, H. Xie, S. Tatikonda, and Y. R. Yang, "Efficient and Dynamic Routing Topology Inference From End-to-End Measurements," *IEEE/ACM Transactions on Networking*, 2010.
[12] B. Eriksson, P. Barford, R. Nowak, and M. Crovella, "Learning Network Structure from Passive Measurements," *ACM SIGCOMM Interent Measurement Conference*, 2007.
[13] J. Pearl and M. Tarsi, "Structuring Causal Trees," *Journal of Complexity*, 1986.
[14] J. J. Hein, "An Optimal Algorithm to Reconstruct Trees from Additive Distance Data," *Bulletin of Mathematical Biology*, 1989.
[15] V. Ramasubramanian, D. Malkhi, F. Kuhn, M. Balakrishnan, A. Gupta, and A. Akella, "On the Treeness of Internet Latency and Bandwidth," *ACM SIGMETRICS*, 2009.
[16] A. Anandkumar, A. Hassidim, and J. Kelner, "Topology Discovery of Sparse Random Graphs With Few Participants," *ACM SIGMETRICS*, 2011.
[17] K. P. Gummadi, S. Saroiu, and S. D. Gribble, "King : Estimating Latency between Arbitrary Internet End Hosts," in *SIGCOMM IMW*, 2002.
[18] "IPlane: An Information Plane for Distributed Services." [Online]. Available: http://iplane.cs.washington.edu/
[19] A. Krishnamurthy and A. Singh, "Robust Multi-Source Network Tomography Using Selective Probes (Supplementary Materials)." [Online]. Available: http://cs.cmu.edu/~akshaykr/files/tomo_app.pdf
[20] B. Donnet, P. Raoult, T. Friedman, and M. Crovella, "Deployment of an Algorithm for Large-Scale Topology Discovery," *IEEE Journal of Selected Areas in Communications, Special Issue on Sampling the Internet*, pp. 2210–2220, 2006.
[21] N. Spring, R. Mahajan, and D. Wetherall, "Measuring ISP Topologies with Rocketfuel," in *Proceedings of ACM SIGCOMM*, 2002.
[22] B. Yao, R. Viswanathan, F. Chang, and D. Waddington, "Topology Inference in the Presence of Anonymous Routers," in *IEEE INFOCOM*, 2003.
[23] M. H. Gunes and K. Sarac, "Resolving IP Aliases in Building Traceroute-based Internet Maps," *IEEE/ACM Transactions on Networking*, vol. 17, pp. 1738–1751, December 2009.
[24] B. Eriksson, P. Barford, and R. Nowak, "Estimating Hop Distance Between Arbitrary Host Pairs," in *IEEE INFOCOM*, 2009.
[25] P. Francis, S. Jamin, C. Jin, Y. Jin, D. Raz, Y. Shavitt, and L. Zhang, "IDMaps: A Global Internet Host Distance Estimation Service," *IEEE/ACM Transactions on Networking*, 2001.
[26] F. Dabek, R. Cox, F. Kaashoek, and R. Morris, "Vivaldi: A Decentralized Network Coordinate System," in *ACM SIGCOMM*, 2004.
[27] M. Rabbat and R. Nowak, "Multiple source, multiple destination network tomography," in *in Proc. of IEEE Infocom*, 2004.
[28] L. Reyzin and N. Srivastava, "On the Longest Path Algorithm for Reconstructing Trees from Distance Matrices," *Information Processing Letters*, 2007.
[29] M. J. Choi, V. Y. F. Tan, A. Anandkumar, and A. S. Willsky, "Learning Latent Tree Graphical Models," *Journal of Machine Learning Research*, 2011.
[30] N. Saitou and M. Nei, "The Neighbor-Joining Method: A New Method for Reconstructing Phylogenetic Trees," *Molecular Biology and Evolution*, 1987.
[31] O. Gascuel and M. Steel, "Neighbor-Joining Revealed," *Molecular Biology and Evolution*, 2006.
[32] C. Daskalakis, E. Mossel, and S. Roch, "Phylogenies without branch bounds: Contracting the short, pruning the deep," *SIAM J. on Discrete Mathematics*, 2011.
[33] J. Felsenstein, *Inferring Phylogenies*. Sinauer Associates, 2004.
[34] V. Ranwez and O. Gascuel, "Quartet-based phylogenetic inference: improvements and limits." *Molecular Biology and Evolution*, 2001.
[35] "PlanetLab: An Open Platform for Developing, Deploying and Accessing Planetary-Scale Services." [Online]. Available: http://www.planet-lab.org/

APPENDIX

Lemma 4.1. *Let (X, d) be a tree metric on $\mathcal{T}$ with $|X| = p$. Then* PEARLRECONSTRUCT *on input (X, d) recovers $\mathcal{T}$.*

Proof: We start with T_3, the tree on leaves x_1, x_2 and x_3. Every *minimal* tree on 3 leaves has the same structure as T_3, so we know this is correct. Moreover, since $d(x_i, y)$ for $i \in \{1, 2, 3\}$ and $y = \text{ancestor}(x_1, x_2, x_3)$ is given by Equation 1, we know that the edge weights in T_3 are also correct.

We now analyze the `add` procedure, showing that it correctly places x_i into the tree so that T_i is the correct minimal tree on $x_1, \ldots, x_i$ with the correct edge weights. We proceed by case analysis: for any root r with subtrees T_{out} and T_{sub}, it must be that either x_i belongs in T_{out}, T_{sub} or in $T_c \setminus \{T_{sub} \cup T_{out}\}$. For any $x_k \in T_{sub}, x_j \in T_{out}$, if x_i belongs in T_{out}, then it must be the case that $d(x_j, y) < d(x_j, r)$ or else the shared common ancestor between x_i, x_j, and x_k could not possibly lie in T_{out}. Similarly, if x_i belongs in T_{sub} then it must be that $d(x_k, y) < d(x_j, r)$. Finally, if x_i lies in neither subtree, then $\text{ancestor}(x_i, x_j, x_k) = r$.

In each case, we update T_c so that it still contains the location where x_i should be added. Since we choose T_{sub} and T_{out} to be non-empty subtrees, the size of T_c decreases on every iteration, so the algorithm must eventually exit the `while` loop.

When this happens, $|T_c| \leq 2$ and T_c contains the location of x_i. If $|T_c| = 1$, then the only place to add x_i is as a child of the node in T_c. This case only happens if $\text{ancestor}(x_i, x_j, x_k) = r$ in the last iteration of the while loop, so the distance $d(x_i, y)$ is the correct edge weight for the new edge.

If $|T_c| = 2$, then we use two additional leaves to determine how to place x_i. Case analysis reveals that our procedure correctly places x_i into T_c. Thus, we conclude that the `add` procedure correctly update T_{i-1} to contain x_i.

By iteratively applying this analysis, we arrive at the claim. ∎

Lemma 4.2 (Sampling)**.** *Let $\mathcal{T}$ have balance factor η and maximum degree k. Then in all iterations of* RISING *and* SPLIT*, with probability $\geq 1 - 2/pk$, the sampled subtree of $\mathcal{T}$ with leaf set M has balance factor:*

$$\hat{\eta} \leq 2\eta + 1$$

as long as $m \geq 4(1 + (k-1)\eta)^2 \log(pk)$.

Proof: In this proof we will simultaneously work with all of the recursive calls of RISING. Since each call recovers one internal node, and there can be no more than p internal nodes in $\mathcal{T}$, we can enumerate the calls from 1 to p. Each call operates on a subset of leaf nodes and we will refer to the tree induced by those leaves as T^s for the sth call.

For fixed s, define the random variables $Z_{ij}, i \in [m], j \in [k]$[4], which takes value 1 if the ith leaf sampled belongs in T_j^s, the jth subtree of r (the root of T^s). Further define $\hat{T}_j^s$ to be the sampled version of T_j^s, that is the tree T_j^s restricted to only the leaves in M. Notice that $\mathbb{E}[Z_{ij}] = |\text{lvs}(T_j^s)|$ and that $|\text{lvs}(\hat{T}_j^s)| = \sum_{i=1}^m Z_{ij}$. By Hoeffding's inequality we have that:

$$\mathbb{P}\left(|\frac{1}{m}|\text{lvs}(\hat{T}_j^s)| - \frac{|\text{lvs}(T_j^s)|}{|\text{lvs}(T^s)|} > \epsilon\right) \leq 2\exp\{-2m\epsilon^2\} \tag{8}$$

$$\tag{9}$$

for any single $j \in [k], s \in [p]$. We would like to do this across all calls to SPLIT, and for each subtree in any of the calls. We take a union bound across all internal nodes and all subtrees, and then rewrite to introduce dependence on the balance factor η, noting that $|\text{lvs}(T_{(k)}^s)| \leq \eta |\text{lvs}(T_{(1)}^s)|$ for any internal node [5]. This gives us that:

$$\frac{1}{m}|\text{lvs}(\hat{T}_{(1)}^s)| \geq \frac{|\text{lvs}(T_{(1)}^s)|}{|\text{lvs}(T^s)|} - \sqrt{\frac{\log(2pk/\delta_1)}{2m}} \tag{10}$$

$$\frac{1}{m}|\text{lvs}(\hat{T}_{(k)}^s)| \leq \frac{\eta|\text{lvs}(T_{(1)}^s)|}{|\text{lvs}(T^s)} + \sqrt{\frac{\log(2pk/\delta_1)}{2m}} \tag{11}$$

Note that since we have established concentration inequalities for all subtrees, we know that the new balance factor $\hat{\eta}$ depends only on the lower bound for the smallest subtree size and the upper bound for the largest subtree size. Now let $m = c\log(pk)$ and set $\delta_1 = 2/pk$. With these settings we have:

$$\frac{1}{m}|\text{lvs}(\hat{T}_{(1)}^s)| \geq \frac{|\text{lvs}(T_{(1)}^s)|}{|\text{lvs}(T^s)|} - \sqrt{\frac{1}{c}} \tag{12}$$

$$\frac{1}{m}|\text{lvs}(\hat{T}_{(k)}^s)| \leq \frac{\eta|\text{lvs}(T_{(1)}^s)|}{|\text{lvs}(T^s)|} + \sqrt{\frac{1}{c}} \tag{13}$$

The new balance factor is the ratio of these two quantities. To find the worst case $\hat{\eta}$, we need to maximize with respect to $|\text{lvs}(T_{(1)}^s)|$. It is easy to verify that the maximum is achieved at the smallest possible size for $T_{(1)}^s$, and given a balance factor of η, we have that $|\text{lvs}(T_{(1)}^s)| \geq \frac{|\text{lvs}(T^s)|}{1+(k-1)\eta}$, achieved when the remaining subtrees are all of the same size. Plugging in this value for $|\text{lvs}(T_{(1)}^s)|$ we have:

$$\hat{\eta} \leq \frac{\frac{\eta}{1+(k-1)\eta} + \sqrt{\frac{1}{c}}}{\frac{1}{1+(k-1)\eta} - \sqrt{\frac{1}{c}}} \tag{14}$$

Now as long as $c \geq (1+(k-1)\eta)^2$, this quantity is guaranteed to be positive and if $c = 4(1 + (k-1)\eta)^2$, then some algebra shows that:

$$\hat{\eta} \leq 2\eta + 1 \tag{15}$$

∎

Lemma 4.3 (Clustering)**.** *Suppose that the probability of an uncorrupted entry $q^4 > C_{\hat{\eta},k}$ and :*

$$m > c_{\hat{\eta},k}\frac{\log(m^2/\delta_2)}{(q^4 - C_{\hat{\eta},k})} \tag{16}$$

[4] we use $[m]$ to denote $\{1, \ldots, m\}$

[5] we use $T_{(1)}^s, \ldots, T_{(k)}^s$ to denote the subtrees of T^s in increasing sorted order by number of leaves

for constants $C_{\hat{\eta},k} < 1, c_{\hat{\eta},k}$ that depend on $\hat{\eta}$ and the max degree k. Then with probability $\geq 1 - \delta_2$, Single Linkage clustering on M using $s(x_i, x_j)$ as the similarity between x_i and x_j partitions M such that either each subtree is entirely contained in one cluster $C \in \mathcal{C}$, or if a subtree is split across clusters, those clusters contain no nodes from other subtrees.

Remark While we have suppressed dependence on $\hat{\eta}$ in Lemma 4.3, we note that a critical condition for correctness is that $\hat{\eta} = O(1)$. This condition ensures that single linkage clustering completely groups any individual subtrees of $\mathcal{T}$ before merging it with any other subtree and is required for our algorithms to be robust to noise.

Proof: The proofs for RISING and SPLIT are almost identical. We tailor our proof to the former, noting where modifications need to be made for the latter.

Our strategy is to lower bound the quantity $s(x_i, x_j)$ for any pair of leaves x_i, x_j that belong to the same subtree and to upper bound $s(x_i, x_k)$ if x_i and x_k do not belong to the same subtree. Under the conditions on q, we show that this lower bound exceeds the upper bound and this guarantees that one subtree will be fully contained in any cluster before any two subtrees are merged. This means that either a subtree is fully contained in a cluster or if it is split across clusters, no nodes from other subtrees are in these clusters.

To assist in our analysis we use the following notation. Let $s^*(x_i, x_j)$ be the value of $s(x_i, x_j)$ in the absence of noise. Let G_{ij} be the group of nodes x_k that all have the same $d(x_i, x_k) - d(x_j, x_k)$ value and that achieve the maximum in the computation of $s(x_i, x_j)$. In particular, this means $s^*(x_i, x_j) = |G_{ij}|$. As above, we write $\hat{T}_i$ to be the ith subtree of r, restricted to the leaves in M. Define $\hat{T}_{(1)}, \ldots, \hat{T}_{(k)}$ to be the subtrees ordered by increasing number of leaves. Finally define $\kappa^T_{min} \triangleq \sum_{i=1}^{k-1} |\text{lvs}(\hat{T}_{(i)})|$ and $\kappa^A_{min} \triangleq \sum_{i=k-1}^{k} |\text{lvs}(\hat{T}_{(i)})|$. κ^T_{min} is a lower bound on $s^*(x_i, x_j)$ for x_i, x_j in the same subtree and κ^A_{min} is an upper bound on $m - s^*(x_i, x_k)$ for x_i, x_k in different subtrees.

We now lower bound $s(x_i, x_j)$ for x_i, x_j in the same subtree. In the presence of noise, any node $x_t \in G_{ij}$ remains in G_{ij} as long as $d(x_i, x_t)$ and $d(x_j, x_t)$ are not corrupted, which occurs with probability at least q^2. Thus:

$$\mathbb{E}[s(x_i, x_j)] \geq q^2 s^*(x_i, x_j)$$

Since each x_t contributes to $s(x_i, x_j)$ independently and since there are $|G_{ij}|$ nodes x_t, we can use Hoeffding's Inequality, coupled with a union bound, to show that with probability $\geq 1 - \delta_{c1}$:

$$s(x_i, x_j) \geq q^2 s^*(x_i, x_j) - m\sqrt{\frac{\log(m^2/\delta_{c1})}{2\kappa^T_{min}}} \tag{17}$$

for all pairs i, j that belong in the same subtree. This is our lower bound.

For SPLIT, we analogously define $G_{ij} \triangleq \{k : d(x_i, x_k) - d(x_j, x_k) = d(x_i, x_{Z(k)}) - d(x_j, x_{Z(k)})\}$ and we require that four measurements are uncorrupted. The above argument, tailored to this scenario gives (with probability $\geq 1 - \delta_{c1}$):

$$\mathbb{E}[s(x_i, x_j)] \geq q^4 s^*(x_i, x_j) \tag{18}$$

$$s(x_i, x_j) \geq q^4 s^*(x_i, x_j) - m\sqrt{\frac{\log(m^2/\delta_{c1})}{2\kappa^T_{min}}} \tag{19}$$

For the upper bound, we can see that a node can contribute to $s(x_i, x_k)$ if it contributes to $s^*(x_i, x_k)$ and it uses no corrupted measurements or if it does not contribute to $s^*(x_i, x_k)$ and it contains a corrupted measurement. For the first case, we will assume pessimistically that all of the nodes $x_t \in G_{ik}$ contribute to $s(x_i, x_k)$. For the latter, we again perform a worst case analysis where we assume any $x_t \notin G_{ij}$ for which either $d(x_i, x_t)$ or $d(x_k, x_t)$ are corrupted contributes to $s(x_i, x_k)$. Thus any x_t contributes with probability $1 - q^2$. If we write $s_2(x_i, x_k)$ to denote the number of nodes $x_t \notin G_{ij}$ that could contribute to $s(x_i, x_k)$, then by the same techniques as above, we arrive at the following upper bound:

$$\begin{aligned} \mathbb{E}[s_2(x_i, x_k)] &\leq (1 - q^2)(m - s^*(x_i, x_k)) \\ s_2(x_i, x_k) &\leq (1 - q^2)(m - s^*(x_i, x_k)) \\ &+ m\sqrt{\frac{\log(m^2/\delta_{c2})}{2\kappa^A_{min}}} \end{aligned}$$

Where the second statement holds with probability $\geq 1 - \delta_{c2}$

In order to ensure success of our clustering algorithm, we need the lower bound for $s(x_i, x_k)$ to be larger than the upper bound for $s(x_i, x_k)$.

Setting $\delta_2 \triangleq \delta_{c1} = \delta_{c2}$, we can now bound q as:

$$\begin{aligned} q^2 &\geq \frac{m}{m + s^*(x_i, x_j) - s^*(x_i, x_k)} + \sqrt{\frac{1}{2}\log(m^2/\delta_2)} \\ &\times \left[\frac{s^*(x_i, x_j)\sqrt{1/\kappa^T_{min}} + (m - s^*(x_i, x_k))\sqrt{1/\kappa^A_{min}}}{m + s^*(x_i, x_j) - s^*(x_i, x_k)}\right] \end{aligned}$$

For this inequality to hold, we require that $s^*(x_i, x_j) \geq s^*(x_i, x_k)$, but this is always the case since $s^*(x_i, x_j) - s^*(x_i, x_k) \geq |\text{lvs}(\hat{T}_{(1)})|$, i.e. the size of the smallest subtree.

To better illustrate the dependence between q and the various parameters of the problem, we simplify the expression using the following bounds (which are straightforward to verify):

$$\kappa^T_{min} \geq \frac{m\hat{\eta}}{1 + (k-1)\hat{\eta}} \tag{20}$$

$$\kappa^A_{min} \geq \frac{2m\hat{\eta}}{1 + (k-1)\hat{\eta}} \tag{21}$$

$$|\text{lvs}(\hat{T}_{(1)})| \geq \frac{m}{1 + (k-1)\hat{\eta}} \tag{22}$$

Using this bounds we arrive at the following lower bound on q^2:

$$q^2 \geq \frac{1 + \frac{1+\sqrt{2}}{2}\sqrt{\frac{\log(m^2/\delta_2)(1+(k-1)\hat{\eta})}{m\hat{\eta}}}}{1 + \frac{1}{1+(k-1)\hat{\eta}}} \tag{23}$$

Specifically, this means that the constant $C_{\hat{\eta},k}$ and $c_{\hat{\eta},k}$ in the Lemma are:

$$C_{\hat{\eta},k} = \frac{1+(k-1)\hat{\eta}}{2+(k-1)\hat{\eta}} \quad (24)$$

$$c_{\hat{\eta},k} = \frac{(1+\sqrt{2})(1+(k-1)\hat{\eta})^{3/2}}{2\sqrt{\hat{\eta}}(2+(k-1)\hat{\eta})} \quad (25)$$

Plugging in these constants and reorganizing the expression results in Equation 16. Both constants depend on both $\hat{\eta}$ and k, however notice that $C_{\hat{\eta},k} < 1$ and both $C_{\hat{\eta},k}$ and $C_{\hat{\eta},k}$ are smaller for $\hat{\eta}$ close to 1. Thus we see that it is easier to cluster more balanced trees.

The analysis for SPLIT is the same, except that we require q^4 to be greater than the right hand side of above lower bound on q^2. Since this dependence is worse than the one for RISING, we use this expression in our result. ∎

Lemma 4.4 (Voting)**.** *Suppose that $q^6 > C_{\hat{\eta},k}$. Then with probability $\geq 1-\delta_3$ the voting phase of* RISING *and* SPLIT *correctly partition the leaves into their subtrees as long as*

$$m > c_{\hat{\eta},k}\frac{\log(p/\delta_3)}{(q^6 - C_{\hat{\eta},k})^2} \quad (26)$$

for some constants $c_{\hat{\eta},k}, C_{\hat{\eta},k}$ that depends on $\hat{\eta}$ and k.

Proof: The voting procedure works by taking one node from each cluster in $\mathcal{C}$ and computing the quartet between those three nodes and the node we are trying to place, x_i. Suppose that x_i belongs in cluster C^*; then it must be the case that $C^* \in \mathcal{C}$ or there exists some $C' \in \mathcal{C}$ such that $C^* \subset C'$. This latter case can happen if we merge two subtrees in the clustering phase.

Since $\mathcal{C}$ always has cardinality 3 in Algorithm 5, when we draw one node from each of the three clusters one of two things can happen: If we draw a node from C^* then in the absence of noise, this quartet would correctly vote that x_i belongs in the cluster C'. If on the other hand, we draw a node from $C' \setminus C^*$, then in the absence of noise this quartet would vote that x_i forms a star. Our analysis must consider both of these scenarios.

Specifically, let Z_i be the indicator that the ith quartet test correctly voted that x_i belongs in C'. We perform $z \triangleq |\hat{T}_{(1)}|$ rounds of voting and by application of a Hoeffding's Inequality and a union bound:

$$\mathbb{P}\left(\frac{|C^*|}{|C'|}q^6 - \frac{1}{z}\sum_{i=1}^{z} Z_i > \epsilon\right) \leq \exp\{\frac{1}{c}m\epsilon^2\} \quad (27)$$

$$Z \triangleq \frac{1}{z}\sum_{i=1}^{z} Z_i > \frac{|C^*|}{|C'|}q^6 - \sqrt{\frac{c\log(p/\delta_3)}{m}} \quad (28)$$

for each $x_i \in \mathcal{X} \setminus M$ for some constant c that depends on $\hat{\eta}$ and k ($c = 1/|\text{lvs}(\hat{T}_{(1)})| \leq 1+(k-1)\hat{\eta}$) . We see that with probability δ_3, the fraction of correct votes is bounded from below as long as $m = \omega(\sqrt{\log p})$ so that the second expression $\to 0$ as $p \to \infty$.

We will need a similar concentration bound on the number of votes that form a star. Define W_i to be the indicator that the ith quartet test correctly forms a star. By a similar argument we see that with probability $\geq 1-\delta_3$:

$$W \triangleq \frac{1}{z}\sum_{i=1}^{z} W_i \geq \frac{|C'|-|C^*|}{|C'|}q^6 - \sqrt{\frac{c\log(p/\delta_3)}{m}} \quad (29)$$

for all $x_i \in \mathcal{X} \setminus M$.

To guarantee that we place x_i correctly, we will pessimistically assume that every vote not for C' and not for a star will vote for the same $C \in \mathcal{C}$, $C \neq C'$. Thus the fraction of votes for C is $1-Z-W$ and we require that $Z > 1-Z-W$. Some algebra shows that this is true if:

$$q^6 > \frac{|C'|}{|C'|+|C^*|} + 3\sqrt{\frac{c\log(p/\delta_3)}{m}} \quad (30)$$

Inverting this equation gives us the lower bound on m in the Lemma. The constant $C_{\hat{\eta},k}$ is exactly $\frac{|C'|}{|C'|+|C^*|} \leq \frac{1+(k-1)\hat{\eta}}{2+(k-1)\hat{\eta}}$ which is the same as the constant in Lemma 4.3. ∎

A. Recovering One Split

Each time we call RISING or SPLIT we attempt to recover one internal node of the tree. In terms of dependence on m, we showed above that as long as m is sufficiently large, the sampling phase will result in a new balance factor $\hat{\eta}$ that is not too different from the original balance factor η and that Single Linkage will produce clusters that reflect the subtrees. Combining the bounds on m from all three phases, we have the following lower bound on m:

$$m > c_{\hat{\eta},k}\frac{\log(pm^2/\delta)}{(q^6 - C_{\hat{\eta},k})^2} \quad (31)$$

And the restrictions on the probability of an uncorrupted entry arise from the clustering and voting phases, but the voting phase's condition is more stringent. We therefore need $q^6 > C_{\hat{\eta},k}$

Finally, we require that the balance factor of the tree $\eta = O(1)$ so that $\hat{\eta}$ will also be a constant for large enough m with high probability.

Putting these conditions together, we can characterize the dependence on m and p under which successful recovery of a single split is possible. Specifically, we have that if $m = \Omega(\log(p/\delta))$, then with probability $\geq 1-\delta$ (where $\delta \triangleq \delta_1 + \delta_2 + \delta_3$), we correctly recover one internal node.

B. Recovering All Splits

There are at most p internal nodes in the tree. To recover all of these nodes with probability $1-o(1)$, we set each $\delta_i = O(1/p)$, and again characterize the dependence between m and p. In the sampling phase, we require that $m = \omega(\log p)$ to ensure that $\hat{\eta}$ does not grow with p. In clustering, we similarly require $m = \omega(\log(m^2 p))$. Finally, in the voting phase, we see that $m = \omega(\log(p))$. These lower bounds determine conditions for successful recovery of the entire tree.