Network Tomography Based on Additive Metrics

Jian Ni  Sekhar Tatikonda
Department of Electrical Engineering, Yale University, New Haven, CT, USA

August 31, 2008

Abstract

Inference of the network structure (e.g., routing topology) and dynamics (e.g., link performance) is an essential component in many network design and management tasks. In this paper we propose a new, general framework for analyzing and designing routing topology and link performance inference algorithms using ideas and tools from phylogenetic inference in evolutionary biology. The framework is applicable to a variety of measurement techniques. Based on the framework we introduce and develop several polynomial-time distance-based inference algorithms with provable performance. We provide sufficient conditions for the correctness of the algorithms. We show that the algorithms are consistent (return correct topology and link performance with an increasing sample size) and robust (can tolerate a certain level of measurement errors). In addition, we establish certain optimality properties of the algorithms (i.e., they achieve the optimal $l_\infty$-radius) and demonstrate their effectiveness via model simulation.

network tomography, routing topology inference, link performance estimation, additive metrics, neighbor-joining.

1 Introduction

Network tomography (network inference) \cite{7, 11, 29} is an emerging field in communication networks which studies the estimation and inference of the network structure and dynamics (e.g., routing topology, link performance, traffic demands) based on indirect measurements when direct measurements are unavailable or difficult to collect. As modern communication networks (e.g., the Internet, wireless communication networks) continue to grow in size, complexity, and diversity, scalable and accurate network inference algorithms and tools will become increasingly important for many network
design and management tasks. These include service provision and resource allocation, traffic engineering, network monitoring, application design, etc.

In network monitoring, such tools can help a network operator obtain routing information and network internal characteristics (e.g., loss rate, delay, utilization) from its network to a set of other collaborating networks that are separated by non-participating autonomous networks. If the performance of a certain portion of the network experiences sudden, dramatic changes, it can be an indication of failures or anomalies occurred in that portion of the network.

In application design, such tools can be particularly useful for peer-to-peer (P2P) style applications where a node communicates with a set of other nodes (called peers) for file sharing and multimedia streaming. For example, a node may want to know the routing topology to other nodes so that it can select peers with low or no route overlap to improve resilience against network failures (e.g., [1]). As another example, a streaming node using multi-path may want to know both the routing topology and link loss rates so that the selected paths have low loss correlation (e.g., [2]).

So far there are two primary approaches to infer the routing topology and link performance of a communication network. An internal-assisted approach uses tools based on measurements or feedback messages of the internal nodes (e.g., routers). Such an approach is limited as today’s communication networks are evolving towards more decentralized and private administration. For example, a common approach to infer the routing topology in the Internet is to use traceroute. However, an increasing number of routers in the Internet will block traceroute requests due to privacy and security concerns. These routers are known as anonymous routers [30] and their existence makes the routing topology inferred by traceroute inaccurate. In addition, administrators of different networks normally will not reveal or share their link-level measurement data for us (e.g., end hosts) to derive the link performance.

Not depending on extra cooperation from the internal nodes (except the basic packet forwarding functionality), a network tomography approach utilizes end-to-end packet probing measurements (such as packet loss and delay measurements) conducted by the end hosts to infer the routing topology and link performance. Under a network tomography approach, a source node will send probes to a set of destination nodes. The basic idea is to utilize the correlations among the observed losses and delays of the probes at the destination nodes to infer the routing topology and link performance from the source node to the destination nodes. Due to its flexibility and reliability, network tomography has attracted many recent studies. Both
multicast probing based approaches (e.g., [6], [13], [19], [22], [23]) and unicast probing based approaches (e.g., [10], [12], [14], [26], [28]) have been investigated. The main challenges of existing approaches and techniques include

- **computational complexity**;
- **information fusion**: how to fuse information from different measurements to achieve the best estimation accuracy;
- **probing scalability** (especially under unicast probing);
- **node dynamics**: how to handle dynamic node joining and leaving efficiently.

In this paper we propose a new, general framework for designing and analyzing topology and link performance inference algorithms using ideas and tools from phylogenetic inference in evolutionary biology. The framework is built upon additive metrics. Under an additive metric the path metric (path length) is expressed as the summation of the link metrics (link lengths) along the path. The basic idea is to use (estimated) distances between the terminal nodes (end hosts) to infer the routing tree topology and link metrics. Based on the framework we introduce and develop several computationally efficient inference algorithms with provable performance.

The advantages of our framework are summarized as follows.

- The framework is applicable to a variety of measurement techniques, including multicast probing, unicast probing, and traceroute probing. Since a linear combination of different additive metrics is still an additive metric, the framework can flexibly fuse information available from different measurements to achieve better estimation accuracy.

- Based on the framework we can design, analyze, and develop distance-based inference algorithms that are **computationally efficient** (polynomial-time), **consistent** (return correct topology and link performance with an increasing sample size), and **robust** (can tolerate a certain level of measurement errors).

We organize the paper as follows. In Section 2 we describe the network model and the inference problem. In Section 3 we introduce additive metrics on trees, and we discuss how to construct additive metrics and compute/estimate the distances between the terminal nodes from end-to-end
measurements. In Section 4 we introduce the neighbor-joining (NJ) algorithm for constructing binary trees from distances. In Section 5 we propose a rooted version of the NJ algorithm and extend it to general trees. In Section 6 we demonstrate the effectiveness of the inference algorithms via model simulation. In Section 7 we extend our framework to infer the routing topology and link performance from multiple source nodes to a single destination node. We summarize the paper in Section 8.

2 Network Model and Inference Problem

Let \( \mathcal{G} = (\mathcal{V}, \mathcal{E}) \) denote the topology of the network, which is a directed graph with node set \( \mathcal{V} \) (end hosts, internal switches and routers, etc.) and link set \( \mathcal{E} \) (communication links that join the nodes). For any nodes \( i \) and \( j \) in the network, if the underlying routing algorithm returns a sequence of links that connect \( j \) to \( i \), we say \( j \) is reachable from \( i \). We call this sequence of links a path from \( i \) to \( j \), denoted by \( P(i, j) \). We assume that during the measurement period, the underlying routing algorithm determines a unique path from a node to another node that is reachable from it.

Hence the physical routing topology from a source node to a set of destination nodes is a (directed) tree. From the physical routing topology, we can derive a logical routing tree which consists of the source node, the destination nodes, and the branching nodes (internal nodes with at least two outgoing links) of the physical routing tree (e.g., \([6],[13],[23]\)). Notice that a logical link may comprise more than one consecutive physical links, and the degree of an internal node on the logical routing tree is at least three. An example is shown in Fig. 1. For simplicity we use routing tree to express logical routing tree unless otherwise noted.

Suppose \( s \) is a source node in the network, and \( D \) is a set of destination nodes that are reachable from \( s \). Let \( T(s, D) = (\mathcal{V}, \mathcal{E}) \) denote the routing tree from \( s \) to nodes in \( D \), with node set \( \mathcal{V} \) and link set \( \mathcal{E} \). Let \( U = s \cup D \) be the set of terminal nodes, which are nodes of degree one (e.g., end hosts).

Every node \( k \in \mathcal{V} \) has a parent \( f(k) \in \mathcal{V} \) such that \( (f(k), k) \in \mathcal{E} \), and a set of children \( c(k) = \{j \in \mathcal{V} : f(j) = k\} \), except that the source node (root of the tree) has no parent and the destination nodes (leaves of the tree) have no children. For notational simplification, we use \( e_k \) to denote link \( (f(k), k) \).

Each link \( e \in \mathcal{E} \) is associated with a performance parameter \( \theta_e \) (e.g., success rate, delay distribution, utilization, etc.). The network inference problem involves using measurements taken at the terminal nodes to infer

1. the topology of the (logical) routing tree;
(2) link performance parameters $\theta_e$ of the links on the routing tree.

We want to point out that the network inference problem is similar to the *phylogenetic inference problem* in evolutionary biology. The phylogenetic inference problem is to determine the evolutionary relationship among a set of species. Such relationship is often represented by a phylogenetic tree, in which the terminal nodes represent extant species and the internal nodes represent extinct common ancestors of the extant species. Many methods have been developed to reconstruct phylogenetic trees from biological information (e.g., biomolecular sequence data) observed at the terminal nodes (e.g., [15], [25]). The mathematical models of the two problems are very similar, except that in the network inference problem we can control and observe the source node, while in the phylogenetic inference problem the information of the source node (the common ancestor of all species) is lost. We will use ideas and tools from phylogenetic inference to analyze and solve the network inference problem.
3 Additive Metrics on Trees

The tool that we will use to analyze and solve the network inference problem is the so-called additive tree metric \[25\], or additive metric for short. We consider trees with internal node degree at least three. Notice that all (logical) routing trees have such property.

**Definition 1** \( d: V \times V \rightarrow \mathbb{R}^+ \) is an additive metric on \( T = (V, E) \) if

\begin{align*}
(a) & \quad 0 < d(e) < \infty, \quad \forall e = (i, j) \in E; \\
(b) & \quad d(i, j) = d(j, i) = \begin{cases}
\sum_{e \in P(i,j)} d(e), & i \neq j; \\
0, & i = j.
\end{cases}
\end{align*}

\( d(e) \) can be viewed as the length of link \( e \), and \( d(i, j) \) can be viewed as the distance between nodes \( i \) and \( j \). Basically, an additive metric associates each link on the tree with a finite positive link length, and the distance between two nodes on the tree is the summation of the link lengths along the path that connects the two nodes.

Suppose \( T(s, D) = (V, E) \) is a routing tree with source node \( s \) and destination nodes \( D \). Let \( d(E) = \{d(e) : e \in E\} \) denote the link lengths of \( T(s, D) \) under additive metric \( d \).

Remember \( U = s \cup D \) is the set of terminal nodes on the tree. Let \( d(U^2) = \{d(i, j) : i, j \in U\} \) denote the distances between the terminal nodes.

Buneman \[5\] showed that the topology and link lengths of a tree are uniquely determined by the distances between the terminal nodes under an additive metric.

**Theorem 1** There is a one-to-one mapping between \( (T(s, D), d(E)) \) and \( (U, d(U^2)) \) under any additive metric \( d \) on \( T(s, D) \).

From Theorem \[1\] we know that we can recover the topology and link lengths of a routing tree if we know \( d(U^2) \). In addition, if there is a one-to-one mapping between the link performance parameters and link lengths (which will be clear in Section \[5.1\]), then we can recover the link performance parameters from the link lengths. The challenges are:
(1) Constructing an additive metric for which we can derive/estimate $d(U^2)$ from measurements taken at the terminal nodes. We will address this issue in this section.

(2) Developing efficient and effective algorithms to recover the topology and link lengths from the (estimated) distances between the terminal nodes. We will address this issue in Sections 4, 5.

3.1 Construct Additive Metrics

A source node can employ different probing techniques, e.g., multicast probing and unicast probing, to send probes (packets) to a set of destination nodes. For multicast probing, when an internal node on the routing tree receives an packet from its parent, it will duplicate the packet and send a copy of the packet to all its children on the tree. Therefore, the packets received by different destination nodes have exactly the same network experience (loss, delay, etc.) in the shared links.

For a (multicast) probe sent by source node $s$ to the destination nodes in $D$, we define a set of link state variables $Z_e$ for all links $e \in E$ on the routing tree $T(s, D)$. $Z_e$ takes value in a state set $Z$. The distribution of $Z_e$ is parameterized by $\theta_e$, e.g.,

$$P(Z_e = z) = \theta_e(z), \quad \forall z \in Z. \tag{1}$$

The transmission of a probe from $s$ to nodes in $D$ will induce a set of outcome variables on the routing tree. For each node $k \in V$, we use $X_k$ to denote the (random) outcome of the probe at node $k$. $X_k$ takes value in an outcome set $X$. By causality the outcome of the probe at node $k$ (i.e., $X_k$) is determined by the outcome of the probe at node $k$’s parent $f(k)$ (i.e., $X_{f(k)}$) and the link state of $e_k$ (i.e., $Z_{e_k}$):

$$X_k = g(X_{f(k)}, Z_{e_k}). \tag{2}$$

Assumption 1 The link states are independent from link to link (spatial independence assumption) and are stationary during the measurement period (stationarity assumption).

Proposition 1 Under the spatial independence assumption that the link states are independent from link to link,

$$X_V \triangleq (X_k : k \in V) \tag{3}$$
is a Markov random field (MRF) on \( T(s,D) \). Specifically, for each node \( k \in V \), the conditional distribution of \( X_k \) given other random variables \( (X_j : j \neq k) \) on \( T(s,D) \) is the same as the conditional distribution of \( X_k \) given just its neighboring random variables \( (X_j : j \in f(k) \cup c(k)) \) on \( T(s,D) \).

Proof For notational simplification, we use \( p(x_A) \) to represent \( P(X_k = x_k : k \in A) \) for any subset \( A \subseteq V \). First we prove by induction that

\[
p(x_V) = p(x_s) \prod_{k \in V \setminus s} p(x_k | x_{f(k)}).
\]

Equation (4) is clearly true for any tree with \(|V| = 1\) or \(|V| = 2\). Assume (4) is true for any tree with \(|V| \leq n\). Now consider a tree \( T \) with \(|V| = n + 1\).

Let \( i \) be a leaf node of \( T \), then by (2) and the spatial independence assumption we have

\[
p(x_V) = p(x_i | x_{V \setminus i})p(x_{V \setminus i})
= p(g(x_{f(i)}, z_{e_i}) | x_{V \setminus i})p(x_{V \setminus i})
= p(g(x_{f(i)}, z_{e_i}) | x_{f(i)})p(x_{V \setminus i})
= p(x_i | x_{f(i)})p(x_{V \setminus i}).
\]

(5)

\( X_{V \setminus i} \) is defined on \( T' = (V \setminus i, E \setminus (f(i), i)) \), a tree with \( n \) nodes. By induction assumption

\[
p(x_{V \setminus i}) = p(x_s) \prod_{k \in V \setminus V \setminus s} p(x_k | x_{f(k)}).
\]

Substituting it into (4) we have shown that Equation (4) holds for \( T \) with \(|V| = n + 1\). By induction argument, Equation (4) is true for any tree.

Now for any \( k \in V \), from (4) we have

\[
p(x_V) = \left( p(x_k | x_{f(k)}) \prod_{j \in c(k)} p(x_j | x_k) \right) \cdot q(x_{V \setminus k}),
\]

\[
p(x_{V \setminus k}) = \sum_{x_k} \left( p(x_k | x_{f(k)}) \prod_{j \in c(k)} p(x_j | x_k) \right) \cdot q(x_{V \setminus k}),
\]

8
where \( q(x_{V \setminus k}) \) is a function that does not depend on \( x_k \). Then

\[
p(x_k | x_{V \setminus k}) = \frac{p(x_V)}{p(x_{V \setminus k})} = \frac{p(x_k | x_{f(k)}) \prod_{j \in c(k)} p(x_j | x_k)}{\sum_{x_k} \left( p(x_k | x_{f(k)}) \prod_{j \in c(k)} p(x_j | x_k) \right)} = \frac{p(x_f(k)) p(x_k | x_{f(k)}) \prod_{j \in c(k)} p(x_j | x_k)}{\sum_{x_k} \left( p(x_f(k)) p(x_k | x_{f(k)}) \prod_{j \in c(k)} p(x_j | x_k) \right)}
\]

Therefore \( X_V \) is a Markov random field on \( T(s, D) \). □

For an MRF \( X_V = (X_k : k \in V) \) on \( T(s, D) \), we can construct an additive metric as follows. For each link \((i, j) \in E\), we define an \( M \times M \) (assume \(|X| = M\)) forward link transition matrix \( P_{ij} \) and an \( M \times M \) backward link transition matrix \( P_{ji} \) with entries

\[
P_{ij}(x_i, x_j) = P(X_j = x_j | X_i = x_i), \quad P_{ji}(x_j, x_i) = P(X_i = x_i | X_j = x_j),
\]

\( x_i, x_j \in X \).

If the link transition matrices are invertible so that \(|P_{ij}| \triangleq |\det(P_{ij})| > 0\), not equal to a permutation matrix (a matrix with exactly one entry in each row and each column being 1 and others being 0) so that \(|P_{ij}| < 1\), and there exists a node \( i \in V \) with positive marginal distribution, then we can construct an additive metric \( d_0 \) with link length (e.g., [4], [9]):

\[
d_0(e) = -\log |P_{ij}| - \log |P_{ji}|, \quad \forall e = (i, j) \in E.
\]

(6)

For any pair of terminal nodes \( i, j \in U \), the distance between \( i \) and \( j \) under additive metric \( d_0 \) can be computed by

\[
d_0(i, j) = -\log |P_{ij}| - \log |P_{ji}|, \quad i, j \in U.
\]

(7)

We can construct other additive metrics based on the specific network inference problem. We use link loss inference as the example. Additive metrics based on link utilization inference and link delay inference can be found in [21].

Example 1 (Link Loss Inference): For this example, the link state variable
\(Z_e\) is a Bernoulli random variable which takes value 1 with probability \(\alpha_e\) if link \(e\) is in \textit{good state} and the probe can go through the link, and takes value 0 with probability \(1 - \alpha_e = \bar{\alpha}_e\) if the probe is lost on the link (e.g., [3]). \(\alpha_e\) is called the \textit{success rate} or \textit{packet delivery rate} of link \(e\), and \(\bar{\alpha}_e\) is called the \textit{loss rate} of link \(e\).

The outcome variable \(X_k\) is also a Bernoulli random variable, which takes value 1 if the probe successfully reaches node \(k\). Since the probe is sent by the source node \(s\), we have \(X_s \equiv 1\). It is clear that for link loss inference

\[
X_k = X_{f(k)} \cdot Z_{ek} = \prod_{e \in \mathcal{P}(s,k)} Z_e. \tag{8}
\]

If \(0 < \alpha_e < 1\) for all links, then we can construct an additive metric \(d_l\) with link length

\[
d_l(e) = -\log \alpha_e, \quad \forall e \in E. \tag{9}
\]

Notice that there is a one-to-one mapping between the link length and link success rate, hence we can derive the link success rates from the link lengths, and vice versa.

Under the spatial independence assumption that the link states are independent from link to link, we have

\[
P(X_i = 1) = \prod_{e \in \mathcal{P}(s,i)} Z_e = 1 = \prod_{e \in \mathcal{P}(s,i)} \alpha_e,
\]

\[
P(X_j = 1) = \prod_{e \in \mathcal{P}(s,j)} Z_e = 1 = \prod_{e \in \mathcal{P}(s,j)} \alpha_e,
\]

\[
P(X_i X_j = 1) = \prod_{e \in \mathcal{P}(s,ij)} Z_e \prod_{e \in \mathcal{P}(ij,i)} Z_e \prod_{e \in \mathcal{P}(ij,j)} Z_e = 1 = \prod_{e \in \mathcal{P}(s,ij)} \alpha_e \prod_{e \in \mathcal{P}(ij,i)} \alpha_e \prod_{e \in \mathcal{P}(ij,j)} \alpha_e,
\]

where \(ij\) is the \textit{nearest common ancestor} of \(i\) and \(j\) on \(T(s,D)\) (i.e., the ancestor of \(i\) and \(j\) that is closest to \(i\) and \(j\) on the routing tree). For example, in Fig. 1b), the nearest common ancestor of destination nodes 4 and 5 is node 2, and the nearest common ancestor of destination nodes 4 and 6 is node 1.

Therefore, the distances between the terminal nodes, \(d_l(U^2)\), can be computed by

\[
d_l(i,j) = \log \frac{P(X_i = 1)P(X_j = 1)}{P^2(X_i X_j = 1)}, \quad i, j \in U. \tag{10}
\]
3.2 Estimation of Distances

As in (7) and (10), if we know the pairwise joint distributions of the outcome variables at the terminal nodes, then we can construct an additive metric and derive $d(U^2)$. In actual network inference problems, however, the joint distributions of the outcome variables are not given. We need to estimate the joint distributions based on measurements taken at the terminal nodes. Specifically, the source node will send a sequence of $n$ probes, and there are totally $n$ outcomes $X^{(t)}_V = (X^{(t)}_k : k \in V)$, $t = 1, 2, ..., n$, one for each probe.

For the $t$-th probe, only the outcome variables $X^{(t)}_{U_i} = (X^{(t)}_k : k \in U = s \cup D)$ at the terminal nodes can be measured and observed. We can estimate the joint distributions of the outcome variables using the observed empirical distributions, which will converge to the actual distributions almost surely if the link state processes are stationary and ergodic during the measurement period.

Suppose $s$ sends a sequence of $n$ probes to (a subset of) destination nodes in $D$. For any probed node $i$, let $X^{(t)}_i$ be the measured loss outcome of the $t$-th probe at node $i$, with $X^{(t)}_i = 1$ if node $i$ successfully receives the probe and $X^{(t)}_i = 0$ otherwise.

We use the empirical distributions of the outcome variables to estimate the distances. For a Bernoulli random variable $X$ (as in link loss inference), the empirical probability that $X$ takes value 1 is just the sample mean $\bar{X}$ of the samples $X^{(1)}, ..., X^{(n)}$:

\[
\hat{P}(X = 1) = \bar{X} = \frac{1}{n} \sum_{t=1}^{n} X^{(t)}. \tag{11}
\]

We can construct explicit estimators for the distances in (10) as follows (we use over $d$ to represent estimated distances):

\[
\hat{d}(i, j) = \log \frac{\bar{X}_i \bar{X}_j}{\bar{X}_i \bar{X}_j}. \tag{12}
\]

$\bar{X}$ is the maximum likelihood estimator (MLE) of $P(X = 1)$ for the samples.
where

\[
X_i = \frac{1}{n} \sum_{t=1}^{n} X_i^{(t)}, \\
\bar{X}_j = \frac{1}{n} \sum_{t=1}^{n} X_j^{(t)}, \\
\bar{X}_i \bar{X}_j = \frac{1}{n} \sum_{t=1}^{n} X_i^{(t)} X_j^{(t)}. 
\]

We can derive exponential error bounds for the distance estimators in (12) using Chernoff bounds [20].

**Proposition 2** For any pair of nodes \(i, j \in U\), a sample size of \(n\) (number of probes to estimate \(\hat{d}_{l(i,j)}\)), and any small \(\epsilon > 0\):

\[
\mathbb{P}\{ |\hat{d}_{l(i,j)} - d_{l(i,j)}| \geq \epsilon \} \leq e^{-c_{ij}(\epsilon)n}
\]

(13)

where \(c_{ij}(\epsilon)\)'s are some constants.

### 3.3 Other Additive Metrics and Information Fusion

We can also construct additive metrics and compute/estimate the distances between the terminal nodes using (end-to-end) unicast packet pair probing or traceroute probing, as described in [21]. A nice property of additive metrics is that a linear combination of several additive metrics is still an additive metric. In order to fuse information collected from different measurements, we can construct a new additive metric using a linear (convex) combination of additive metrics \(d_1, d_2, ..., d_k\):

\[
d = a_1 d_1 + a_2 d_2 + ... + a_k d_k,
\]

s.t. \(a_1 + a_2 + ... + a_k = 1\).

\[(14)\]

The estimated distance between terminal nodes \(i, j \in U\) under the new additive metric can be easily computed:

\[
\hat{d}(i, j) = a_1 \hat{d}_1(i, j) + a_2 \hat{d}_2(i, j) + ... + a_k \hat{d}_k(i, j).
\]

In practice we can select the coefficients empirically based on the current network state or to minimize the variance of the estimator \(\hat{d}\).
4 Neighbor-Joining Algorithm

We have described how to construct additive metrics and estimate the distances between the terminal nodes via end-to-end packet probing measurements. In this section we introduce the neighbor-joining (NJ) algorithm, which is considered the most widely used algorithm for building binary phylogenetic trees from distances (e.g., [10], [24], [27]).

**Definition 2** A distance-based tree inference algorithm (or distance-based algorithm for short) takes the (estimated) distances between the terminal nodes of a tree as the input and returns a tree topology and the associated link lengths. The input distances $\hat{d}(U^2)$ satisfy:

\[
\begin{align*}
\hat{d}(i, j) &\geq 0, \quad \text{with equality if and only if } i = j, \\
\hat{d}(i, j) &= \hat{d}(j, i).
\end{align*}
\]

**Definition 3** Two or more nodes on a tree are called neighbors (siblings), if they are connected via one internal node (if they have the same parent) on the tree.

The NJ algorithm is an agglomerative algorithm. The algorithm begins with a leaf set including all destination nodes. In each step it selects two leaf nodes that are likely to be neighbors, deletes them from the leaf set, creates a new node as their parent and adds that node to the leaf set. The whole process is iterated until there is only one node left in the leaf set, which will be the child of the root (source node).

To avoid trivial cases, we assume $|D| \geq 2$.

---

**Algorithm 1: Neighbor-Joining (NJ) Algorithm for Binary Trees**

**Input:** Estimated distances between the nodes in $U = s \cup D, \hat{d}(U^2)$.

1. $V = \emptyset, E = \emptyset$.

2.1 For any pair of nodes $i, j \in D$, compute

\[
\hat{Q}(i, j) = \sum_{k \in U} \hat{d}(i, k) + \sum_{k \in U} \hat{d}(j, k) - (|U| - 2)\hat{d}(i, j).
\]

2.2 Find $i^*, j^* \in D$ with the largest $\hat{Q}(i, j)$ (break the tie arbitrarily).

Create a node $f$ as the parent of $i^*$ and $j^*$. 

13
\[ D = D \setminus \{i^*, j^*\}, \quad U = U \setminus \{i^*, j^*\} \]
\[ V = V \cup \{i^*, j^*\}, \quad E = E \cup \{(f, i^*), (f, j^*)\}. \]

2.3 Compute the link lengths from the distances:
\[ \hat{d}(f, i^*) = \frac{1}{|U|} \sum_{k \in U} [\hat{d}(k, i^*) + \hat{d}(i^*, j^*) - \hat{d}(k, j^*)] / 2, \] (16)
\[ \hat{d}(f, j^*) = \frac{1}{|U|} \sum_{k \in U} [\hat{d}(k, j^*) + \hat{d}(i^*, j^*) - \hat{d}(k, i^*)] / 2. \] (17)

2.4 For each \( k \in U \), compute the distance between \( k \) and \( f \):
\[ \hat{d}(k, f) = \frac{1}{2} [\hat{d}(k, i^*) - \hat{d}(f, i^*)] + \frac{1}{2} [\hat{d}(k, j^*) - \hat{d}(f, j^*)]. \] (18)

\[ D = D \cup f, \quad U = U \cup f. \]

3. If \(|D| = 1\), for the \( i \in D \): \( V = V \cup \{i\}, \quad E = E \cup (s, i). \)
Otherwise, repeat Step 2.

**Output:** Tree \( \hat{T} = (V, E) \), and link lengths \( \hat{d}(e) \) for all \( e \in E \).

The NJ algorithm has several nice properties:
- it is computationally efficient, with a polynomial-time complexity \( O(N^3) \) for (binary) trees with \( N \) terminal nodes;
- it returns the correct tree topology and link lengths if the input distances are additive (i.e., if the input distances are derived from an additive metric without estimation errors);
- it is robust: it achieves the optimal \( l_\infty \)-radius among all distance-based algorithms for binary trees.

The \( l_\infty \)-radius notation was introduced by Atteson [3].

**Definition 4** For a distance-based algorithm, we say it has \( l_\infty \)-radius \( r \), if for any tree \( T \) associated with any additive metric \( d \), whenever the input distances between the terminal nodes, \( d(U^2) \), satisfy:
\[ ||\hat{d}(U^2) - d(U^2)||_\infty \leq \max_{i,j \in U} |\hat{d}(i,j) - d(i,j)| \]
\[ < r \min_{e \in E} d(e), \] (19)
the algorithm will return the correct topology of \( T \).
An algorithm with larger $l_\infty$-radius is more robust, because it can tolerate more estimation errors. [3] showed that no distance-based algorithm has $l_\infty$-radius larger than $\frac{1}{2}$ via an example, and proved that the NJ algorithm in fact achieves the optimal $l_\infty$-radius for binary trees.

**Theorem 2** The NJ algorithm achieves the optimal $l_\infty$-radius $\frac{1}{2}$ for binary trees.

It is not straightforward to extend the NJ algorithm for general (non-binary) trees. Since most routing trees in communication networks are not binary, we are motivated to design algorithms that can handle general trees.

## 5 Rooted Neighbor-Joining Algorithm

### 5.1 Binary Trees

We first present an algorithm which can be viewed as a rooted version of the NJ algorithm for binary trees. To avoid trivial cases, we assume $|D| \geq 2$.

---

**Algorithm 2: Rooted Neighbor-Joining (RNJ) Algorithm for Binary Trees**

**Input:** Estimated distances between the nodes in $U = s \cup D$, $\hat{d}(U^2)$.

1. $V = \{s\}$, $E = \emptyset$.

For any pair of nodes $i, j \in D$, compute

$$\hat{\rho}(i, j) = \frac{\hat{d}(s, i) + \hat{d}(s, j) - \hat{d}(i, j)}{2}. \tag{20}$$

2.1 Find $i^*, j^* \in D$ with the largest $\hat{\rho}(i, j)$ (break the tie arbitrarily).

Create a node $f$ as the parent of $i^*$ and $j^*$.

$D = D \setminus \{i^*, j^*\}$,

$V = V \cup \{i^*, j^*\}$, $E = E \cup \{(f, i^*), (f, j^*)\}$.

2.2 Compute:

$$\hat{d}(s, f) = \hat{\rho}(i^*, j^*), \tag{21}$$

$$\hat{d}(f, i^*) = \hat{d}(s, i^*) - \hat{\rho}(i^*, j^*), \tag{22}$$

$$\hat{d}(f, j^*) = \hat{d}(s, j^*) - \hat{\rho}(i^*, j^*). \tag{23}$$
2.3 For each $k \in D$, compute:

\[ \hat{d}(k, f) = \frac{1}{2} [\hat{d}(k, i^*) - \hat{d}(f, i^*)] + \frac{1}{2} [\hat{d}(k, j^*) - \hat{d}(f, j^*)], \]

\[ \hat{\rho}(k, f) = \frac{1}{2} [\hat{d}(s, k) + \hat{d}(s, f) - \hat{d}(k, f)] \]

\[ = \frac{1}{2} [\hat{\rho}(k, i^*) + \hat{\rho}(k, j^*)]. \]

$D = D \cup f$.

3. If $|D| = 1$, for the $i \in D$: $V = V \cup \{i\}$, $E = E \cup (s, i)$.

Otherwise, repeat Step 2.

**Output:** Tree $\hat{T} = (V, E)$, and link lengths $\hat{d}(e)$ for all $e \in E$.

The major difference between the NJ algorithm and the RNJ algorithm is the selection of the score function: the NJ algorithm uses the $\hat{Q}$ function defined in (15), which has no simple interpretation; while the RNJ algorithm uses the $\hat{\rho}$ function in (20), which has a simple interpretation that we will explain next.

For any pair of nodes $i, j \in D$, remember $ij$ is their nearest common ancestor on $T(s, D)$. Under additive metric $d$, we know

\[ \rho(i, j) = \frac{d(s, i) + d(s, j) - d(i, j)}{2} = d(s, ij) \] (24)

is the distance from the root (source node $s$) to $ij$. It is not hard to verify that a pair of nodes $i^*, j^*$ with largest $\rho(i, j)$ must be neighbors (siblings) on the tree. $\hat{\rho}(i, j)$ in (20) is the estimated distance from the root to $ij$ computed from the input distances. If the input distances are close to the true additive distances, then we would expect that the two nodes selected in Step 2.1 of Algorithm 2 are indeed neighbors.

We provide a sufficient condition for Algorithm 2 to return the correct tree topology. From the condition we can establish several nice properties of the algorithm.

**Lemma 1** For binary trees, a sufficient condition for Algorithm 2 to return the correct tree topology is:

\[ \forall i, j, k \in D \text{ s.t. } ij < ik \]

\[ \Rightarrow \hat{\rho}(i, j) > \hat{\rho}(i, k), \] (25)
where \( ij < ik \) means that \( ij \) is descended from \( ik \).

**Proof** We prove the lemma by induction on the cardinality of \( D \).

1. If \( |D| = 2 \), then clearly Algorithm 2 will return the correct tree topology.
2. Assume Algorithm 2 returns the correct tree topology under condition (25) for \( |D| \leq N \). Now consider \( |D| = N + 1 \).

**Claim 1.** \( i^*, j^* \) found in Step 2.1 which maximize \( \hat{\rho}(i, j) \) are siblings (neighbors).

If \( i^* \) and \( j^* \) are not siblings, then there exists \( k \in D \) such that either \( i^*k \) or \( j^*k \) is descended from \( i^*j^* \). Under condition (25), this implies either

\[
\hat{\rho}(i^*, k) > \hat{\rho}(i^*, j^*)
\]

or

\[
\hat{\rho}(j^*, k) > \hat{\rho}(i^*, j^*)
\]

a contradiction to the maximality of \( \hat{\rho}(i^*, j^*) \).

**Claim 2.** Condition (25) is maintained over the nodes in \( D \) after Step 2.

After Step 2, \( i^*, j^* \) are deleted from \( D \) and \( f \) is added to \( D \) as a new leaf node. Since \( i^*, j^* \) are siblings and \( f \) is their parent, we know that for any \( i \in D \),

\[
if = ii^* = ij^*.
\]

Therefore, \( \forall i, j \in D \) s.t. \( ij < if \), we have \( ij < ii^* \) and \( ij < ij^* \), which implies

\[
\hat{\rho}(i, j) > \hat{\rho}(i, i^*)
\]

and

\[
\hat{\rho}(i, j) > \hat{\rho}(i, j^*)
\]

hence

\[
\hat{\rho}(i, j) > \hat{\rho}(i, f) = \frac{1}{2} [\hat{\rho}(i, i^*) + \hat{\rho}(i, j^*)].
\]

Similarly, \( \forall i, k \in D \) s.t. \( if < ik \), we can show \( \hat{\rho}(i, f) > \hat{\rho}(i, k) \).

From claims 1 and 2, we know that after one iteration of Step 2, Algorithm 2 will correctly find out a pair of siblings, and condition (25) is maintained for the new set of leaf nodes in \( D \). Then \( |D| \) is decreased by 1. By induction assumption, the algorithm will return the correct topology of the remaining part of the tree. This completes our proof of the lemma. \( \square \)

**Proposition 3** For binary trees, Algorithm 2 will return the correct tree topology and link lengths if the input distances \( \hat{d}(U^2) \) are additive.
Proof If the input distances are additive, then $\hat{\rho}(i, j)$ and $\hat{\rho}(i, k)$ are the actual distances from $s$ to $ij$ and $ik$ under an additive metric. In this case, if $ij$ is descended from $ik$, since link lengths are positive, we have $\hat{\rho}(i, j) > \hat{\rho}(i, k)$, hence condition (25) holds. Then by Lemma 1, Algorithm 2 will return the correct tree topology. In addition, under additive distances it is clear that the link lengths computed in Step 2.2 of Algorithm 2 are correct.

In practice, the distances between the terminal nodes are estimated from measurements taken at the terminal nodes. The estimated distances may deviate from the true additive distances due to measurement errors. Nevertheless, we will show that if the estimated distances are close enough to the true distances, then Algorithm 2 will return the correct tree topology. In addition, Algorithm 2 achieves the optimal $l_{\infty}$-radius among all distance-based algorithms.

Proposition 4 The RNJ algorithm (Algorithm 2) achieves the optimal $l_{\infty}$-radius $\frac{1}{2}$ for binary trees, i.e., for any binary tree associated with any additive metric $d$, whenever the input distances $\hat{d}(U^2)$ satisfy:

$$\max_{i,j \in U} |\hat{d}(i, j) - d(i, j)| < \frac{1}{2} \min_{e \in E} d(e), \quad (26)$$

Algorithm 2 will return the correct tree topology.

Proof Using Lemma 1 we only need to show that condition (26) implies condition (25). Let

$$\Delta = \min_{e \in E} d(e)$$

be the minimum link length on the tree. If $ij \prec ik$, i.e., if $ij$ is descended from $ik$, since link lengths $\geq \Delta$, we have

$$\rho(i, j) - \rho(i, k) \geq \Delta.$$
Then from (20), (24), (26) we have:

\[ \hat{\rho}(i, j) - \hat{\rho}(i, k) \geq (\hat{\rho}(i, j) - \hat{\rho}(i, k)) - (\rho(i, j) - \rho(i, k) - \Delta) \]

\[ \geq \Delta - \frac{1}{2}|\hat{d}(s, j) - d(s, j)| - \frac{1}{2}|\hat{d}(i, j) - d(i, j)| \]

\[ - \frac{1}{2}|\hat{d}(s, k) - d(s, k)| - \frac{1}{2}|\hat{d}(i, k) - d(i, k)| \]

\[ > \Delta - \frac{1}{4}\Delta - \frac{1}{4}\Delta - \frac{1}{4}\Delta - \frac{1}{4}\Delta \]

\[ > 0. \]

Hence condition (26) indeed implies condition (25). Since (25) is a sufficient condition for Algorithm 2 to return the correct tree topology, (26) is also a sufficient condition for Algorithm 2 to return the correct tree topology.

5.2 General Trees

\( \rho(i, j) \) is the distance from the root to the nearest common ancestor of nodes \( i \) and \( j \). For a general routing tree with positive link lengths, we have several observations of the \( \rho \) function.

- If nodes \( i \) and \( j \) are neighbors on the tree, then for any other node \( k \) on the tree we have

\[ \rho(i, j) \geq \rho(i, k). \] (27)

- If nodes \( i \) and \( j \) are neighbors on the tree, then for any other node \( k \) that is also a neighbor of \( i \) and \( j \) we have

\[ \rho(i, j) = \rho(i, k) \] (28)

because \( ij = ik \).

- If nodes \( i \) and \( j \) are neighbors on the tree, then for any other node \( k \) that is not a neighbor of \( i \) and \( j \) we have

\[ \rho(i, j) \geq \rho(i, k) + \Delta \] (29)

(where \( \Delta \) is the minimum link length) because \( ij \) is descended from \( ik \) and they are separated by at least one link.
Therefore, we can determine whether a group of nodes are neighbors on the tree from knowledge of the \( \rho \) function under an additive metric.

To extend the RNJ algorithm (Algorithm 2) for general trees, after we find out two nodes \( i^* \) and \( j^* \) that are likely to be neighbors in Step 2.1, we need to find out other nodes that are likely to be neighbors of \( i^* \) and \( j^* \) based on \( \hat{\rho} \) computed from the input distances. We use the following threshold neighbor criterion:

**Threshold Neighbor Criterion.**
Suppose \( i^* \) and \( j^* \) are neighbors on the tree. Node \( k \) will be chosen as a neighbor of \( i^* \) and \( j^* \) if and only if

\[
\hat{\rho}(i^*, j^*) - \hat{\rho}(i^*, k) \leq t \tag{30}
\]

for some threshold \( t > 0 \).

Based on observations (28) and (29), and since \( \hat{\rho} \) is an estimator of \( \rho \) with possible estimation errors, we use the middle point \( \frac{\Delta}{2} \) as the threshold. Later we will show that such a threshold enables the algorithm to achieve the optimal \( l_\infty \)-radius \( \frac{1}{4} \) for general trees if the threshold criterion (30) is used in the algorithm (see the proof of Proposition 7).

---

**Algorithm 3: Rooted Neighbor-Joining (RNJ) Algorithm for General Trees**

**Input:** Estimated distances between the nodes in \( U = s \cup D, \hat{d}(U^2) \); estimated minimum link length \( \Delta > 0 \).

1. \( V = \{s\}, E = \emptyset \).
   
   For any pair of nodes \( i, j \in D \), compute
   
   \[
   \hat{\rho}(i, j) = \frac{\hat{d}(s, i) + \hat{d}(s, j) - \hat{d}(i, j)}{2}. \tag{31}
   \]

2.1 Find \( i^*, j^* \in D \) with the largest \( \hat{\rho}(i, j) \) (break the tie arbitrarily).
   
   Create a node \( f \) as the parent of \( i^* \) and \( j^* \).
   
   \( D = D \setminus \{i^*, j^*\} \),
   
   \( V = V \cup \{i^*, j^*\}, E = E \cup \{(f, i^*), (f, j^*)\} \).
2.2 Compute:

\[ \hat{d}(s, f) = \hat{\rho}(i^*, j^*), \]
\[ \hat{d}(f, i^*) = \hat{d}(s, i^*) - \hat{\rho}(i^*, j^*), \]
\[ \hat{d}(f, j^*) = \hat{d}(s, j^*) - \hat{\rho}(i^*, j^*). \]  

(32)

(33)

(34)

2.3 For every \( k \in D \) such that \( \hat{\rho}(i^*, j^*) - \hat{\rho}(i^*, k) \leq \frac{\Delta}{2} \):

\[
D = D \setminus k, \\
V = V \cup k, \\
E = E \cup (f, k).
\]

Compute:

\[ \hat{d}(f, k) = \hat{d}(s, k) - \hat{\rho}(i^*, j^*). \]  

(35)

2.4 For each \( k \in D \), compute:

\[
\hat{d}(k, f) = \frac{1}{2} \left[ \hat{d}(k, i^*) - \hat{d}(f, i^*) \right] + \frac{1}{2} \left[ \hat{d}(k, j^*) - \hat{d}(f, j^*) \right], \\
\hat{\rho}(k, f) = \frac{1}{2} \left[ \hat{d}(s, k) + \hat{d}(s, f) - \hat{d}(k, f) \right] \\
= \frac{1}{2} \left[ \hat{\rho}(k, i^*) + \hat{\rho}(k, j^*) \right]. 
\]

\[
D = D \cup f. 
\]

3. If \( |D| = 1 \), for the \( i \in D \): \( V = V \cup \{i\}, E = E \cup (s, i) \).

Otherwise, repeat Step 2.

Output: Tree \( \hat{T} = (V, E) \), and link lengths \( \hat{d}(e) \) for all \( e \in E \).

---

**Lemma 2** Let \( \Delta \leq \min_{e \in E} d(e) \) be the input parameter. A sufficient condition for Algorithm 3 to return the correct tree topology is:

\[
\forall i, j, k \in D \text{ s.t. } ij \prec ik \Rightarrow \hat{\rho}(i, j) - \hat{\rho}(i, k) > \frac{\Delta}{2}, \\
\forall i, j, k \in D \text{ s.t. } ij = ik \Rightarrow |\hat{\rho}(i, j) - \hat{\rho}(i, k)| \leq \frac{\Delta}{2}. 
\]

(36)

**Proof** We outline the proof, which is similar to the proof of Lemma 1. There are three key observations:
(1) Under condition (36), $i^*, j^*$ found in Step 2.1 of Algorithm 3 are siblings.

(2) Under condition (36), $k$ will be selected in Step 2.3 if and only if it is a sibling of $i^*$ and $j^*$.

(3) Condition (36) is maintained over the nodes in $D$ after Step 2.

The lemma then follows by induction on the cardinality of $D$.

**Proposition 5** For general trees, Algorithm 3 will return the correct tree topology and link lengths if the input distances $\hat{d}(U^2)$ are additive.

**Proof** The proof is similar to the proof of Proposition 3.

In practice the input distances may deviate from the true additive distances due to measurement errors. Again we can show that if the input distances are close enough to the true additive distances, then Algorithm 3 will return the correct tree topology.

**Proposition 6** For a general tree with additive metric $d$, if the input parameter

$$\Delta \leq \min_{e \in E} d(e)$$

and the input distances $\hat{d}(U^2)$ satisfy:

$$\max_{i,j \in U} |\hat{d}(i,j) - d(i,j)| < \frac{\Delta}{4}, \quad (37)$$

then Algorithm 3 will return the correct tree topology.

**Proof** The proof is similar to the proof of Proposition 4. We can show that condition (37) implies condition (36), then the proposition follows by Lemma 2.

If the input parameter $\Delta = \min_{e \in E} d(e)$, then Proposition 6 says that the RNJ algorithm has $l_\infty$-radius $\frac{1}{4}$ for general trees.

**Corollary 1** The RNJ algorithm (Algorithm 3) has $l_\infty$-radius $\frac{1}{4}$ for general trees when $\Delta = \min_{e \in E} d(e)$. 

22
We have the following conjecture.

**Conjecture 1** No distance-based algorithm has $l_\infty$-radius greater than $\frac{1}{4}$ for general trees. If this is true, then the RNJ algorithm (Algorithm 3) achieves the optimal $l_\infty$-radius $\frac{1}{4}$ for general trees when $\Delta = \min_{e \in E} d(e)$.

We can show that no distance-based algorithm has $l_\infty$-radius greater than $\frac{1}{4}$ if the threshold (neighbor) criterion (30) is used in the algorithm.

**Proposition 7** If the threshold criterion (30) is used, then no distance-based algorithm has $l_\infty$-radius greater than $\frac{1}{4}$ for general trees.

**Proof** Suppose $\mathcal{A}$ is a distance-based algorithm with $l_\infty$-radius $r$ in which the threshold criterion (30) is used. Let $\Delta = \min_{e \in E} d(e)$. Therefore, for any tree $T$ associated with any additive metric $d$, if the input distances $\hat{d}(U^2)$ satisfy:

$$\max_{i,j \in U} |\hat{d}(i,j) - d(i,j)| < r\Delta, \quad (38)$$

then $\mathcal{A}$ will return the correct topology of $T$.

Suppose $i^*$ and $j^*$ are neighbors on $T$, and $k$ is a neighbor of them. Then we have $\rho(i^*, j^*) = \rho(i^*, k)$. Under condition (38) we know

$$\hat{\rho}(i^*, j^*) - \hat{\rho}(i^*, k) < 2r\Delta. \quad (39)$$

Since the threshold criterion (30) is used, we need to have

$$\hat{\rho}(i^*, j^*) - \hat{\rho}(i^*, k) < 2r\Delta \leq t \quad \Rightarrow \quad r \leq \frac{t}{2\Delta}$$

(39)

to correctly add $k$ as a neighbor of $i^*$ and $j^*$.

Now suppose $k'$ is not a neighbor of $i^*$ and $j^*$. Then we have $\rho(i^*, j^*) - \rho(i^*, k') \geq \Delta$. Under condition (38) we know

$$\hat{\rho}(i^*, j^*) - \hat{\rho}(i^*, k') > \Delta - 2r\Delta.$$  

Since the threshold criterion (30) is used, we need to have

$$\hat{\rho}(i^*, j^*) - \hat{\rho}(i^*, k') > \Delta - 2r\Delta \geq t \quad \Rightarrow \quad r \leq \frac{1}{2} - \frac{t}{2\Delta} \quad (40)$$

(40)

to correctly not add $k'$ as a neighbor of $i^*$ and $j^*$.

Combining (39) and (40) we have

$$r \leq \min \left( \frac{t}{2\Delta}, \frac{1}{2} - \frac{t}{2\Delta} \right) \quad \Rightarrow \quad r \leq \frac{1}{4} \quad (41)$$

(41)

where the upper bound $\frac{1}{4}$ of $r$ is achieved with the threshold $t = \frac{\Delta}{2}$. \qed
5.3 Complexity and Consistency

The computational complexity of the RNJ algorithm is $O(N^2 \log N)$ for a routing tree with $N$ destination nodes. We now show the consistency of the RNJ algorithm for general trees (Algorithm 3), and a similar result holds for binary trees.

Let $\hat{T}_n$ be the inferred tree topology returned by the RNJ algorithm with a sample size $n$ (number of probes to estimate the distances between the terminal nodes). Let

$$P_n = \Pr\{\hat{T}_n = T\}$$

denote the probability of correct topology inference of the RNJ algorithm.

**Proposition 8** Let $\Delta \leq \min_{e \in E} d(e)$ be the input parameter of the RNJ algorithm. If

$$\Pr\{|d(i, j) - d(i, j)| \geq \frac{\Delta}{4}\} \leq e^{-c_{ij}(\Delta)n}, \forall i, j \in U,$$  \hspace{1cm} (42)

where $n$ is the sample size and $c_{ij}(\Delta)$ is a constant, then for a routing tree with $N$ terminal nodes:

$$P_n \geq 1 - N^2 e^{-c(\Delta)n}.$$  \hspace{1cm} (43)

**Proof** By Proposition 6 we have

$$P_n \geq \Pr\left\{ \bigcap_{i,j \in U} |\hat{d}(i, j) - d(i, j)| < \frac{\Delta}{4}\right\}$$

$$= 1 - \Pr\left\{ \bigcup_{i,j \in U} |\hat{d}(i, j) - d(i, j)| \geq \frac{\Delta}{4}\right\}$$

$$\geq 1 - \sum_{i,j \in U} e^{-c_{ij}(\Delta)n}$$

$$\geq 1 - N^2 e^{-c(\Delta)n}$$

where $C(\Delta) = \min_{i,j \in U} c_{ij}(\Delta)$. \hfill \Box

**Proposition 9** If the input distances $\hat{d}(U^2)$ are consistent (i.e., they converge to the true distances in probability in the sample size) and the RNJ algorithm returns the correct tree topology, then the link lengths returned by the RNJ algorithm are consistent.
If we use the distance estimators in (12), since they satisfy condition (42) (by Proposition 2) and are consistent, by Proposition 8, the probability of correct topology inference of the RNJ algorithm goes to 1 exponentially fast in the sample size. If the inferred topology is correct, then by Proposition 9, the returned link lengths are also consistent. For network inference problems where there is a one-to-one mapping between the link performance parameters and the link lengths (e.g., (9)), the link lengths returned by the RNJ algorithm provide consistent estimators for the link performance parameters (e.g., success rates).

6 Simulation Evaluation

In addition to analysis, we demonstrate the effectiveness of the NJ algorithm and the RNJ algorithm via model simulation. For each experiment, we first randomly generate the tree topology and select the link success rates in a certain range. The source node then sends a sequence of probes to the destination nodes. The destination nodes measure the loss outcomes of the probes. We consider both random binary trees and general trees.

The distances between the terminal nodes are estimated from the empirical distributions of the observed outcomes at the destination nodes as in (12). We then apply both inference algorithms to infer the tree topology and link success rates using the estimated distances between the terminal nodes.

We compare the inferred tree topology with the true tree topology. If the inferred topology is correct, then we further compare the inferred link success rates $\hat{\alpha}_e$’s with the true link success rates $\alpha_e$’s. Specifically, for each link $e$, we compute the relative error of the inferred link success rate (compared with the true link success rate) as follows:

$$
\epsilon_e = \left| \frac{\hat{\alpha}_e - \alpha_e}{\alpha_e} \right|
$$

and we calculate the average relative error among all links on the tree:

$$
\epsilon_E = \frac{1}{|E|} \sum_{e \in E} \epsilon_e.
$$

Each experiment is repeated 100 times. For each inference algorithm, we compute the fraction of correctly inferred trees among all 100 trials.

\[2\] Like the RNJ algorithm, we extend the NJ algorithm for general trees using a similar threshold neighbor criterion as in (30).
Figure 2: Binary trees: fraction of correctly inferred trees.

Figure 3: Binary trees: average relative error of inferred link success rates.
Figure 4: General trees: fraction of correctly inferred trees.

Figure 5: General trees: average relative error of inferred link success rates.
(which can be viewed as the probability of correct topology inference of the algorithm), as well as the average value of $\epsilon_E$ among the correctly inferred trees. The results are shown in Figs. 2-5. The $x$ axis is in log scale, i.e., it is $\log_2 n$ for a sample size of $n$ probes. As we expect from our analysis, the NJ algorithm and the RNJ algorithm are consistent: the fraction of correctly inferred trees of both algorithms goes to 1 (exponentially fast) as we increase the sample size, and the average relative error of the inferred link success rates goes to 0 with an increasing sample size.

For binary trees, we observe that the NJ algorithm and the RNJ algorithm have very similar performance; while for general trees, the RNJ algorithm has a clear advantage over the NJ algorithm in terms of the fraction of correctly inferred trees, implying that the RNJ algorithm is more accurate for topology inference of general trees.

We conduct experiments for trees with different sizes and ranges of link success rates, and we observe the same pattern of the results.

7 Multiple-Source Single-Destination Network Inference

In this section we study the network inference problem of estimating the routing topology and link performance from multiple source nodes to a single destination node, in contrast to the single-source multiple-destination network inference problem we have addressed in the previous sections.

Again we assume that during the measurement period, the underlying routing algorithm determines a unique path from a node to another node that is reachable from it. Therefore, the physical routing topology from a set of source nodes to a destination node forms a reversed directed tree. From the physical routing topology, we can derive a logical routing tree which consists of the source nodes, the destination node, and the joining nodes (internal nodes with at least two incoming links) of the physical routing tree. Each internal node on the logical routing tree has degree at least three, and a logical link may comprise more than one physical links. An example is shown in Fig. 6.

Let $r$ be a destination node (receiver) in the network, and $S$ be a set of source nodes that will communicate with $r$. Let $T(S, r) = (V, E)$ denote the (logical) routing tree from nodes in $S$ to $r$, with node set $V$ and link set $E$. Let $U = S \cup r$ be the set of terminal nodes, which are nodes with degree one.
Each node $k \in V$ has a child $c(k) \in V$ such that $(k, c(k)) \in E$, and a set of parents $f(k) = \{j \in V : c(j) = k\}$, except that the destination node has no child and the source nodes have no parents.

For notational simplification, we use $e_k$ to denote link $(k, c(k))$. Each link $e_k$ is associated with a performance parameter $\theta_k$ (e.g., success rate, delay distribution, utilization, etc.) that we want to estimate. The network inference problem involves using measurements taken at the terminal nodes to infer

1. the topology of the (logical) routing tree;
2. link performance parameters $\theta_e$ of the links on the routing tree.

### 7.1 Reverse Multicast Probing

Similar to multicast probing from a source node to a set of destination nodes, we can have reverse multicast probing from a set of source nodes to a single destination node. We illustrate the idea of reverse multicast using Fig. 6(b) as the example.
Under a reverse multicast probing, source nodes 4 and 5 will send a packet (probe) to their child node 2. Node 2 may receive both packets, or one of them, or none of them (because of packet loss). If node 2 receives at least one packet from its parents, it will combine (e.g., concatenate) the packets and sends the combined packet (as a probe) to its child node 1. Otherwise, node 2 will send nothing. Similarly, source node 3 will send a packet to its child node 1. Node 1 combines the packets received from its parents (if any) and sends the combined packet to the destination node \( r \). The whole process is like the reverse process of multicasting a probe from node \( r \) to the other nodes on the routing tree.

For a probe sent from the source nodes in \( S \) to the destination node \( r \), we define a set of link state variables \( Z_e \) for all links on the routing tree \( T(S, r) \). Using link loss inference as the example, \( Z_e \) is a Bernoulli random variable which takes value 1 with probability \( \alpha_e \) if the probe can go through link \( e \), and takes value 0 with probability \( 1 - \alpha_e = \bar{\alpha}_e \) if the probe is lost on the link.

For each node \( k \) on the routing tree, we use \( X_k \) to denote the (random) outcome of the probe sent from node \( k \) observed by the destination node \( r \). For link loss inference, \( X_k \) takes value 1 if \( r \) successfully receives the probe sent from node \( k \), and takes value 0 otherwise. It is clear that for any source node \( i \),

\[
X_i = Z_{e_{i1}} \cdot X_{e(i)} = \prod_{e \in P(i, r)} Z_e. \tag{44}
\]

If \( 0 < \alpha_e < 1 \) for all links, then we can construct an additive metric \( d_l \) with link length

\[
d_l(e) = -\log \alpha_e, \quad \forall e \in E. \tag{45}
\]

For any pair of source nodes \( i, j \in S \), let \( ij \) denote their nearest common descendant on \( T(S, r) \) (i.e., the descendant of both nodes \( i \) and \( j \) that is closest to \( i \) and \( j \) on the routing tree). For example, in Fig. 6(b), the nearest common descendant of source nodes 4 and 5 is node 2, and the nearest common descendant of source nodes 3 and 4 is node 1.

Under the spatial independence assumption that the link states are in-
dependent from link to link, for any pair of source nodes \(i\) and \(j\), we have

\[
\begin{align*}
\mathbb{P}(X_i = 1) &= \prod_{e \in P(i,r)} \alpha_e, \\
\mathbb{P}(X_j = 1) &= \prod_{e \in P(j,r)} \alpha_e, \\
\mathbb{P}(X_iX_j = 1) &= \prod_{e \in P(i,ij)} \alpha_e \prod_{e \in P(j,ij)} \alpha_e \prod_{e \in P(ij,r)} \alpha_e.
\end{align*}
\]

Therefore, the distances between the terminal nodes, \(d_l(U^2)\), can be computed by \(\mathbb{P}(X_r = 1) = 1\):

\[
d_l(i, j) = \log \frac{\mathbb{P}(X_i = 1)\mathbb{P}(X_j = 1)}{\mathbb{P}^2(X_iX_j = 1)}, \quad i, j \in U. \tag{46}
\]

We can see that the mathematical model of a reverse multicast probing on a routing tree (with multiple source nodes and a single destination node) is similar to the mathematical model of a multicast probing on a routing tree (with a single source node and multiple destination nodes). Therefore, the additive-metric framework can be directly applied to analyze and solve the multiple-source single-destination network inference problem. Specifically, we can construct additive metrics, estimate the distances between the terminal nodes from end-to-end measurements, and apply the distance-based algorithms to infer the routing tree topology and the link performance metrics.

### 7.2 Passive Network Monitoring in Wireless Sensor Networks

Although the current Internet does not support reverse multicast probing because internal nodes (routers) do not combine packets sent from different source nodes to a destination node, reverse multicast can be deployed in wireless networks (e.g., [17], [18]) for efficiently data collecting.

A typical scenario in wireless sensor networks for data collecting is as follows. A base station (a receiver) will first propagate an *interest message* into the network via flooding or constrained/directional flooding. An interest message could be a query message which specifies what the base station wants (e.g., temperature statistics). A node, when first receives the interest message from another node, will set that node as its *child* and forward the interest message to its own neighbors excluding its child. Hence the interest
propagation procedure serves both to disseminate the interest message, and to set up a reverse path from each node to the base station.

When a sensor node which has the data of interest (a source node) receives the interest message, it can send the data back to the base station using the reverse path (i.e., it sends the data to its child). Assume each source node has a unique ID (e.g., the geographical location of the node). The data sent by a source node to the base station also include the source node’s ID so the base station knows from where it receives the data.

If each node selects only one node as its child, i.e., if there is a unique path from a node to the base station, then we know that the routing topology (undirected version) from the source nodes to the base station is a tree. We call it a data collecting tree. Each internal node on the tree only needs to maintain the information of a set of parents that it will receive data from, and a child that it will send data to.

Suppose directed diffusion [17] is applied on the data collecting tree, under which an internal node will aggregate (e.g., combine, compress, code, etc.) the data sent from its parents and then send the aggregated data to its child. Then this process is like a reverse multicast probing process as we described in Section 7.1. Using the algorithms we have developed in this paper, the base station can infer: (1) the topology of the data collecting tree; (2) the link performance (e.g., packet delivery rate) of every link on the data collecting tree.

There are several advantages for the base station to do network inference based on the collected data from the sensor nodes. First, the (internal) sensor nodes do not need to measure and infer the link performance which can save their resources; while normally the base station has sufficient battery and computation power so it is competent for the network inference task. Second, this is a passive network monitoring framework so no extra probing traffic is generated. In addition, since the inference is based on real data transmission, the inferred link performance metrics are more accurate and meaningful.

8 Conclusion

In this paper we address the network inference problem of estimating the routing topology and link performance in a communication network. We propose a new, general framework for designing and analyzing network inference algorithms based on additive metrics using ideas and tools from phylogenetic inference. The framework is applicable to a variety of measure-
ment techniques. Based on the framework we introduce and develop several distance-based inference algorithms. We provide sufficient conditions for the correctness of the algorithms. We show that the algorithms are computationally efficient (polynomial-time), consistent (return correct topology and link performance with an increasing sample size), and robust (can tolerate a certain level of measurement errors). In addition, we establish certain optimality properties of the algorithms (i.e., they achieve the optimal $l_\infty$-radius) and demonstrate their effectiveness via model simulation. The framework provides powerful tools that enable us to infer and estimate the structure and dynamics of large-scale communication networks.

References

[1] D. G. Andersen, H. Balakrishnan, M. F. Kaashoek, R. Morris, “Resilient Overlay Networks,” *Proc. SOSP 2001*, October 2001.

[2] D. Antonova, A. Krishnamurthy, Z. Ma, R. Sundaram, “Managing a Portfolio of Overlay Paths,” *Proc. NOSSDAV 2004*, Kinsale, Ireland, June 2004.

[3] K. Atteson, “The Performance of Neighbor-Joining Methods of Phylogenetic Reconstruction,” *Algorithmica*, vol. 25, pp. 251-278, 1999.

[4] D. Barry and J. A. Hartigan, “Asynchronous Distance Between Homogeneous DNA Sequences,” *Biometrics*, vol. 43, pp. 261-276, June 1987.

[5] P. Buneman, “The Recovery of Trees from Measures of Dissimilarity,” *Mathematics in the Archaeological and Historical Sciences*, Edinburgh University Press, pp. 387-395, 1971.

[6] R. Caceres, N. G. Duffield, J. Horowitz, D. Towsley, “Multicast-Based Inference of Network-Internal Loss Characteristics,” *IEEE Transactions on Information Theory*, vol. 45, no. 7, pp. 2462-2480, Nov. 1999.

[7] R. Castro, M. Coates, G. Liang, R. Nowak, B. Yu, “Network Tomography: Recent Developments,” *Statistical Science*, vol. 19, no. 3, pp. 499-517, 2004.

[8] R. Castro, M. Coates, R. Nowak, “Likelihood Based Hierarchical Clustering,” *IEEE Transactions on Signal Processing*, vol. 52, no. 8, pp. 2308-2321, Aug. 2004.
[9] J. T. Chang, “Full Reconstruction of Markov Models on Evolutionary Trees: Identifiability and Consistency,” *Mathematical Biosciences*, vol. 137, pp. 51-73, 1996.

[10] M. Coates and R. Nowak, “Network Loss Inference using Unicast End-to-End Measurement,” *Proceedings of ITC Conference on IP Traffic, Modelling and Management*, Monterey, CA, Sep. 2000.

[11] M. Coates, A. O. Hero III, R. Nowak, B. Yu, “Internet Tomography,” *IEEE Signal Processing Magazine*, vol. 19, no. 3, pp. 47-65, May 2002.

[12] M. Coates, R. Castro, M. Gadhiok, R. King, Y. Tsang, R. Nowak, “Maximum Likelihood Network Topology Identification from Edge-Based Unicast Measurements,” *Proc. ACM Sigmetrics 2002*, Jun. 2002.

[13] N. G. Duffield, J. Horowitz, F. Lo Presti, D. Towsley, “Multicast Topology Inference From Measured End-to-End Loss,” *IEEE Transactions on Information Theory*, vol. 48, no. 1, pp. 26-45, Jan. 2002.

[14] N. G. Duffield, F. Lo Presti, V. Paxson, D. Towsley, “Network Loss Tomography Using Striped Unicast Probes,” *IEEE/ACM Transactions on Networking*, vol. 14, no. 4, pp. 697-710, Aug. 2006.

[15] J. Felsenstein, *Inferring Phylogenies*, Sinauer, New York, 2004.

[16] O. Gascuel and M. Steel, “Neighbor-Joining Revealed,” *Molecular Biology and Evolution*, vol. 23, no. 11, pp. 1997-2000, 2006.

[17] C. Intanagonwiwat, R. Govindan, D. Estrin, J. Heidemann, F. Silva, “Directed Diffusion for Wireless Sensor Networking,” *IEEE/ACM Transactions on Networking*, vol. 11, no. 1, pp. 2-16, February 2003.

[18] Y. Mao, F. R. Kschischang, B. Li, S. Pasupathy, “A Factor Graph Approach to Link Loss Monitoring in Wireless Sensor Networks,” *IEEE Journal on Selected Areas in Communications*, vol. 23, no. 4, pp. 820-829, April 2005.

[19] J. Ni and S. Tatikonda, “A Markov Random Field Approach to Multicast-Based Network Inference Problems,” *2006 IEEE International Symposium on Information Theory*, Seattle, July 2006.

[20] J. Ni and S. Tatikonda, “Explicit Link Parameter Estimators Based on End-to-End Measurements,” *45th Allerton Conference on Communication, Control, and Computing*, Monticello, Illinois, September 2007.
[21] J. Ni, H. Xie, S. Tatikonda, Y. R. Yang, “Network Routing Topology Inference from End-to-End Measurements,” *Proc. IEEE Conference on Computer Communications (INFOCOM)*, Phoenix, Arizona, April 2008.

[22] F. L. Presti, N. G. Duffield, J. Horowitz, D. Towsley, “Multicast-Based Inference of Network-Internal Delay Distributions,” *IEEE/ACM Transactions on Networking*, vol. 10, no. 6, pp. 761-775, Dec. 2002.

[23] S. Ratnasamy and S. McCanne, “Inference of Multicast Routing Trees and Bottleneck Bandwidths using End-to-end Measurements,” *Proc. IEEE INFOCOM 1999*, Mar. 1999.

[24] N. Saitou and M. Nei, “The Neighbor-Joining Method: A New Method for Reconstruction of Phylogenetic Trees,” *Molecular Biology and Evolution*, vol. 4, no. 4, pp. 406-425, 1987.

[25] C. Semple and M. Steel, *Phylogenetics*, volume 22 of Mathematics and Its Applications Series, Oxford University Press, 2003.

[26] M. Shih, A. O. Hero III, “Hierarchical Inference of Unicast Network Topologies Based on End-to-End Measurements,” *IEEE Transactions on Signal Processing*, vol. 55, no. 5, pp. 1708-1718, May 2007.

[27] K. Tamura, M. Nei, S. Kumar, “Prospects for Inferring Very Large Phylogenies by Using Neighbor-Joining Method,” *Proc. of the National Academy of Sciences*, vol. 101, no. 30, pp. 11030-11035, July 2004.

[28] Y. Tsang, M. Coates, R. Nowak, “Network Delay Tomography,” *IEEE Transactions on Signal Processing*, Special Issue on Signal Processing in Networking, vol. 51, no. 8, pp. 2125-2136, Aug. 2003.

[29] Y. Vardi, “Network Tomography: Estimating Source-Destination Traffic Intensities from Link Data,” *Journal of the American Statistical Association*, vol. 91, no. 433, pp. 365-377, 1996.

[30] B. Yao, R. Viswanathan, F. Chang, D. Waddington, “Topology Inference in the Presence of Anonymous Routers,” *Proc. IEEE INFOCOM 2003*, pp. 353-363, Apr. 2003.