ISSN:0975-9646

An inference routing tree topology algorithm for reducing overhead in packet probing

V. Srujana [*1] and Vadlamudi Rajesh [*2]

[*1] *Assistant Professor, DEPT of CSE, KL University, Vaddeswaram , Vijayawada , Andhra Pradesh, India*
[*2] *Student, M.Tech (CSE) ,Vaddeswaram, Green Fields, KL University, Vijayawada , Andhra Pradesh, India*

***Abstract*- Network structure (Topology) and Node dynamics (Links) are essential components of a secure robust network monitoring and application design. Packet probing is an important network measurement technique, used to understand packet delays, paths, and loss. In this paper we propose an RNJ(Route Neighbor Joining) algorithm which is a grouping type algorithm that recovers the tree topology by recursively joining the neighbors on the tree by using the concept of additive metrics which is a general adaptive framework for analyzing and designing routing topology in an efficient scalable network structure. The framework can flexibly fuse information from multiple measurements to achieve better estimation accuracy besides Packet probing. Based on the framework we introduce and develop a polynomial-time based inference algorithms that effectively manages the number of probing packets. In particular, for applications where nodes may join or leave frequently and randomly such as overlay networks, application-layer multicast, and peer-to-peer file systems, we propose a novel sequential topology inference algorithm that significantly reduces the probing overhead and can efficiently handle node dynamics. We demonstrate the effectiveness of the proposed inference algorithms using network simulations.**

Keywords- **Network Measurement, Network Monitoring, Packet probing**

I. INTRODUCTION

Developing the efficient tools to check the network topology and link performance of the node to group of other nodes is an important approach. **Network monitoring** describes the use of a system that constantly monitors a computer network for slow or failing components and that notifies the network administrator (via email, pager or other alarms) in case of outages. It is a subset of the functions involved in network management. It is also help the network operator obtain routing information and network internal characteristics from network and from node to node within the individual networks. While an intrusion detection system monitors a network for threats from the outside, a network monitoring system monitors the network for problems caused by overloaded and/or crashed servers, network connections or other devices. For example, to determine the status of a web server, monitoring software may periodically send an HTTP request to fetch a page. For email servers, a test message might be sent through SMTP and retrieved by IMAP or POP3.Commonly measured metrics are response time, availability and uptime, although both consistency and reliability metrics are starting to gain popularity. The widespread addition of WAN optimization devices is having an adverse effect on most network monitoring tools -- especially when it comes to measuring accurate end-to-end response time because they limit round trip visibility. Status request failures - such as when a connection cannot be established, it times-out, or the document or message cannot be retrieved - usually produce an action from the monitoring system. These actions vary -- an alarm may be sent (via SMS, email, etc.) to the resident system admin, automatic failover systems may be activated to remove the troubled server from duty until it can be repaired, etc. Monitoring the performance of a network uplink is also known as network traffic measurement, and more software is listed there. Here, two approaches are used to infer the network topology and link performance. **Routing topology** is the process of selecting paths in a network along which to send network traffic. Routing is performed for many kinds of networks, including the telephone network (Circuit switching), electronic data networks (such as the Internet), and transportation networks. In packet switching networks, routing directs packet forwarding, the transit of logically addressed packets from their source toward their ultimate destination through intermediate nodes, typically hardware devices called routers, bridges, gateways, firewalls, or switches.

In this work we have introduced different topology networks such as unicast multicast networks in which the concept of probing is used to the networks to infer the topology of the network so that it is easy to understand the structure of the network and it is also useful for the P2P networks where nodes may join and leave to estimate the network topology with regarding to unicast to the end devices on knowing whether destination have occurred or not. On other hand known as *network tomography*[1], utilizes end-to-end packet probing measurements (such as packet loss and delay measurements) conducted by the end hosts and does not require extra cooperation from the internal nodes (except the basic packet forwarding functionality). Under a network tomography approach, a source node will send probes to a set of destination nodes. Internet, unicast network tomography approaches based on back-to-back unicast packet pairs or strings have also been investigated [2]. Two fundamental challenges of network tomography approaches include *computational complexity* and *probing scalability* [3](especially under unicast probing). These limit the number

of destination nodes that a source node can infer. So we have introduced an efficient and inference tree topology algorithm by using RNJ(Route Neighbor Joining)[4] and from this a Novel Sequential algorithm is derived which is computationally efficient.

II. *RELATED WORK*

Grouping algorithm to infer the tree topology is based on shared losses identified at the destination node which was proposed by the Multicast routing tree topology. In computer networking, **multicast** routing tree[5] is the delivery of a message or information to a group of destination computers simultaneously in a single transmission from the source creating copies automatically in other network elements, such as routers, only when the topology of the network requires it. Multicast is most commonly implemented in IP multicast, which is often employed in Internet Protocol (IP) applications of streaming media and Internet television. In IP multicast the implementation of the multicast concept occurs at the IP routing level, where routers create optimal distribution paths for datagram's sent to a multicast destination address. At the Data Link Layer, the *multicast* describes one-to-many distribution functions such as Ethernet multicast addressing, Asynchronous Transfer Mode(ATM) point-to-multipoint virtual circuits (P2MP) or Infiniband multicast. After verifying all the algorithms, with best of our knowledge, the sequential topology inference algorithm proposed in this paper is a first effort to address the issues of dynamic node and probing scalability for network for network routing topology inference. The rooted neighbor-joining (RNJ) algorithm proposed in this paper is also a grouping type algorithm that recovers the tree topology by recursively joining the neighbors on the tree. This agglomerative joining/grouping idea has been used in *clustering* for building cluster trees and in *evolutionary biology* for building phylogenetic trees [6].Unicast routing tree topology inference was studied in Coates *et al.* [7] introduced a *sandwich* probing technique to conduct delay measurements and proposed a Markov chain Monte Carlo procedure to search the most likely tree topologies.

II.NETWORK MODEL AND TREE TOPOLOGY INFERENCE BY PROBING :

When dealing with networking, the terms "network model" and "network layer" used often, **Network models** define a set of network layers and how they interact. 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 (reachable) 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. 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. In this paper, we consider topology inference of logical routing trees and use the *routing tree* to express the *logical routing tree*.

A.Mulicast tree topology inference:

Multicast packets flow along a distribution tree rooted at the source. The receivers form the leaves of the tree, and the links from the edges of the tree. The internal nodes in the tree and the links form the edges of the tree. A packet that is dropped along any link of the distribution tree, is lost by all the downstream receivers in the subtree rooted at the link. The tree structured delivery model thus introduces correlations in the packet losses seen by the different receivers. This loss correlation between receivers can be exploited to infer the topology of the tree that caused the observed loss patterns. The tree inference algorithm described in this attempts to reconstruct this logical tree in a bottom-up fashion using information regarding the loss patterns of the different receivers. Receivers having similar loss patterns are aggregated together and represented by a single node one level higher in the tree. The aggregated nodes can then be regarded as a single node for further aggregation. The entire tree has been reconstructed when all the receivers have been coalesced in this manner into a single tree In order to rebuild the tree shown in figure 1, the algorithm[8] initially begins with a set of individual receivers A,B and C. Information obtained from the loss patterns of the three receivers indicates that A and B are more closely located than A and C or B and C, thus aggregate A and B into a single macro-node (AB). Next, (AB) and C are aggregated to yield the logical tree ((AB)C).

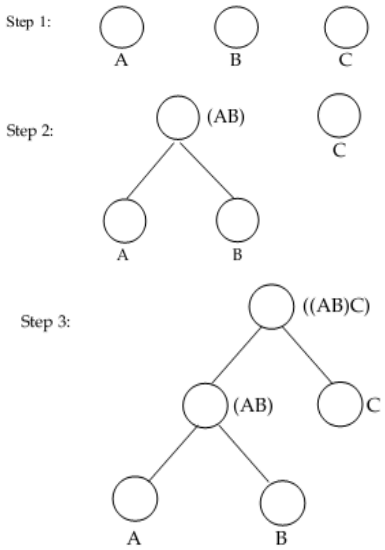

Figure 1: Inference of logical tree

This algorithm reconstructs a 'logical' representation of the multicast tree. A logical representation of a multicast tree is one in which each interior node is merely the closest common ancestor of all downstream receivers in the tree. In reality each branch of the logical tree could consist of a series of links. In order to learn the exact topology of the tree we would have to enlist the help of each intermediate router along the path as is done in the traceroute and mtrace tool which internally uses probing. Our algorithm is based only on end-to-end measurements using only information that is readily available at the end hosts and requires no special router support, as such, reconstructing a logical tree is as accurate as we can get. Under *multicast probing,* when an internal node on the routing tree receives a packet from its parent, it will

send a copy of the packet to all its children on the tree. Hence, the packets of the same probe received by different destination nodes have exactly the same network experience (loss, delay, etc.) in the shared links.

B. Unicast tree topology inference:

However, due to the poor availability of multicast in real-world networks, several projects also studied topology inference based on unicast end-to-end measurements Coates et al. [9] presented a method to capture path delay in unicast routing tree topologies called **sandwich probing**.

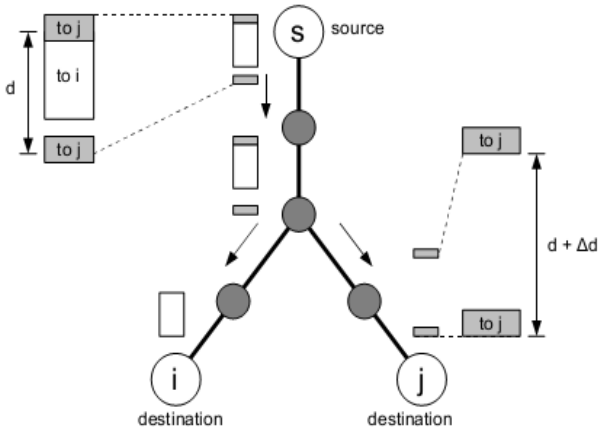

Figure 2: The general idea of sandwich probing

Compared to classic packet delay measurements, sandwich probing eliminates the need for synchronized clocks on the sender and the receiver node because it only measures delay differences. As illustrated in Figure 2 a sender node *s* sends out a sequence of so-called sandwich probes. Each sandwich probe consists of three packets, two small packets destined for receiver *j* separated by a larger packet destined for receiver *i*. The second small packet is expected to queue behind the large one at every inner node of the routing tree (e.g. bridge, switch, etc.). This induces an additional delay Δd between the small packets on the shared links. The longer the common subpath from *s*, the closer *i* and *j* must be in the logical routing tree. Since all sandwich probes originate from a single sender *s*, all inferred logical routing trees will have *s* as their root node.
Under *unicast probing,* the source node sends a string of back-to-back unicast packets to the destination nodes, one packet for each destination node, respectively (to mimic the transmission of a multicast probe). We call it a 1 *packet string probing* if the string size (i.e., number of probed destination nodes) is . Since the back-to-back packets are very close to each other, it is normally assumed that these packets have the same network experience in the shared links just like a multicast probe.

IV.TREE TOPOLOGY INFERENCE BASED ON NEIGHBOR JOINING

Let be a routing tree with source node $T(s,D) = (V,E)$ be a routing tree with source node s and destination nodes D. In an **additive metric** [10] the path metric is expressed as the summation of the metrics along the path. The linear combination of different additive metrics is still an additive metric. We say is an *additive metric* on $T(s,D)$ if
a) $0 < d((e) < \infty, \quad \forall\, e \in E$
b) $d(i,j) = \sum_{e \in p(i,j)} d(e), \ \forall i,j \in v$
$d(e)$can be viewed as the *length* of link e and $d(i,j)$ can be viewed as the *distance* between nodes i and j . Remember $U = s \cup D$ is the set of terminal nodes on the tree. We use $d(U^2) = \{d(i,j): i,j \in U\}$ to denote the distances between the terminal nodes. It is known that the topology and link lengths of a tree are uniquely determined by the distances between the terminal nodes under an additive metric.
Suppose the source node is fixed. For any destination node $i \in D$,let $p(i) = d(s,i)$ denote the *path length* from s to i(under additive metric d).For any pair of destination nodes i,j$\in D$, let $\underline{ij}$denote their *nearest common ancestor* on $T(s,D)$ (i.e., the ancestor of both nodes i and j that is closest to i and j on the routing tree).Let $p(i,j) = d(s,\underline{ij})$ denote the *shared path length* from s to i and (i.e., the distance between and the nearest common ancestor of i and i).
Let $p(s,D) = \{p(i): i \in D\}$ denote the path lengths from s to D nodes in and denote the shared path lengths from to pairs of nodes in . Note that there is a one-to-one mapping between $d(U^2)$ and $p(s,D) \cup p(s,D^2)$. We can recover the topology of the routing tree if we know either $d(U^2)$ or $p(s,D) \cup p(s,D^2)$. The key is to construct an additive metric for which we can derive/estimate $d(U^2)$ or $p(s,D) \cup p(s,D^2)$. from end-to-end measurements.
We first present a topology inference algorithm using (estimated) path lengths and shared path lengths as the input. The algorithm is a grouping type algorithm as in [11] It can be viewed as a rooted version of the widely used *neighbor joining* algorithm for constructing phylogenetic trees from distances. The algorithm begins with a leaf set including all the destination nodes. In each step, it selects a group of nodes that are likely to be *neighbors* (i.e., *siblings*, nodes with the same parent on the tree), 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| ≥2.

Algorithm 1 : RNJ algorithm:
Input: Source s, Destination D, $\hat{p}$(s,D), $\hat{p}$(s,D^2), $\Delta > 0$.

1. V={s}∪ D,E= Ø.

2.1. Find i*, j* ∈ D with the largest $\hat{p}$(i, j) (break the tie arbitarily).Create a node f as the parent of i* and j*

D=D\{i*, j*},

V= V ∪ {f},

E = E ∪ {(f , i*) , (f , j*)}.

(+) $\hat{d}$(f , j*) = $\hat{p}$(i*) - $\hat{p}$(i*, j*),

(+) $\hat{d}$(f , j*) = $\hat{p}$(j*) - $\hat{p}$(i*, j*),

2.2. For every k ∈ D such that $\hat{p}$(i*, j*) - $\hat{p}$(i*, k) ≤ $\frac{\Delta}{2}$

D=D/k,

E = E ∪ (f, k).

(+) $\hat{d}$(f ,k) = $\hat{p}$(k*) - $\hat{p}$(i*,j*),

2.3. For each k∈ D, compute:

$\hat{p}$(k ,f) = $\frac{1}{2}$[$\hat{p}$(k ,i*)+$\hat{p}$(k , j*)].

D = D ∪ f.

(+) $\hat{p}$(f) = $\hat{p}$(i*, j*).

3. If |D| =1, for the k ∈D: E = E ∪ (s ,k).

Otherwise repeat step 2.

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

Note that in algorithm step 2.3, we compute the shared path length between nodes k and f, $\hat{p}$(k , f). Using measurements from only two children of f. if f has more than two children, we could utilize measurements from all of them as follows:

$$\hat{p}(k, f) = \frac{1}{|c(f)|}\sum_{i \in c(f)} \hat{p}(k, f)$$

This modification improves the accuracy of the RNJ algorithm. The computational complexity of the RNJ algorithm is $O(N^2 logN)$ for a routing tree with N destination nodes. Note that the RNJ algorithm only requires (estimated) shared path lengths $\hat{p}$(s,D^2) to infer the tree topology (steps without (+)). If the (estimated) path lengths $\hat{p}(s, D)$), are also available, then the RNJ algorithm can infer the link lengths as well (steps with (+)). If there is a one-to-one mapping between the link performance parameters (e.g., success rate, utilization, delay variance) and the link lengths, as in [1] we can use the link lengths returned by the RNJ algorithm to estimate the link performance parameters.

V.DYNAMIC TREE TOPOLOGY INFERENCE

In practice, the RNJ algorithm (and other existing topology inference algorithms) may have some limitations. First, the focus of previous studies is on a relatively stable set of nodes. In real applications (e.g., P2P applications), the destination nodes that a source node communicates with will often change over time. Hence, the routing tree topology will also change over time. When an existing destination node leaves, it is straightforward to derive the updated routing tree topology. When a new destination node joins, running the RNJ algorithm over the new set of destination nodes to infer the updated routing tree topology is not efficient when the nodes join and leave frequently.

The second limitation is the *probing scalability problem* under unicast probing. The RNJ algorithm requires estimated shared path lengths from the source node to all pairs of the destination nodes as the input. Suppose there are N destination nodes. If multicast probing is available, then the source node can use a $1 \times N$ multicast probing to obtain the required measurements. The probing overhead is $O(N)$. On the other hand, if multicast probing is not supported and N is large, then it is difficult to obtain $\hat{p}$(s,D^2) using a single $1 \times N$ unicast packet string probing without violating the assumption that the string of packets has the same or even positively correlated network experiences in the shared links.

The source node could use back-to-back (unicast) packet pair probings. This requires probing $O(N^{2)})1 \times 2$ The probing overhead is $O(N^{2)})$. If these probings are conducted in parallel, then this will quickly consume the outgoing bandwidth of the source node; while if these probings are conducted in sequence, then it will take a long time to obtain the measurements, and it is likely that the network states (routing topology, link performance metrics) will change during the measurement period, which will violate the stationary assumption. We tested the RNJ algorithm via Internet experiments, and we found that it only has decent accuracy for a small number of destination nodes (less than six). Therefore, poor *probing scalability* of unicast packet pair probing will limit the number of destination nodes that a source node can infer when multicast probing is not supported. We address these issues in this section. We design procedures to add a node to (add_node) and delete a node from (delete_node) a routing tree. These procedures can handle node joining and leaving efficiently and are particularly useful for applications where node dynamics are prevalent. Based on the add_node procedure, we propose a novel sequential topology inference algorithm, which greatly reduces the probing overhead under unicast packet pair probing..

1) Procedure add_node (T, K, j, Δ):

Procedure add_node (T, K, j, Δ) is a recursive procedure that adds a new destination node to the routing tree via an existing node on the tree, with the initial condition that is a sibling or descendant of node . is the (estimated) minimum link length. Let be the parent of on the (old) tree.

add_node(T, K, j, Δ) is a recursive procedure that adds a new destination node j to the routing tree via an existing node k on the tree, with the initial condition that j is a sibling or descendant of node k . Δ is the (estimated) minimum link length. Let $f(k)$ be the parent of k on the (old) tree$T$.

Procedure: add_node (T, K, j, Δ)

IF **is a leaf node on the tree:**$T = (V, E)$

(j *will be a sibling of* k *on the new tree.*)

1. Create a node $p$as the parent of $k$and$j$.

$V = V \cup \{p , j\}$

$E = E \setminus (f(k), k) \cup \{(f(k), p), (p, k), (p, j)\}$.

ELSE k **Suppose has** l **children** $c_1 \ldots\ldots c_l$

2. Select a destination node descended from c_i

3. Measure/estimate and for $\hat{p}(d_1, d_2)$ and $\hat{p}(j, d_i)$ for i =1……..l

4. Find d_{i*} with the largest $\hat{p}(j, d_i)$

Case (a) : $\hat{p}(d_1, d_2) - \hat{p}(j, d_{i*}) \geq \frac{\Delta}{2}$

(will *be a sibling of on the new tree.*)

5. Create a node p as the parent of k and j.

$V = V \cup \{p , j\}$,

$E = E \setminus (f(k), k) \cup \{(f(k), p), (p, k), (p, j)\}$

Case (b) : $\hat{p}(d_1, d_2) - \hat{p}(j, d_{i*}) < \frac{\Delta}{2}$

(j *will be a child of on the new tree.*)

6. $V = V \cup j, E = E \cup (k, j)$

Case (c) : $\hat{p}(j, d_{i*}) - \hat{p}(d_1, d_2) \geq \frac{\Delta}{2}$

(j *will be a sibling or descendant of on the new tree.*)

7. Execute add_node (T, c_i, j, Δ).
By running add_node(T, s, j, Δ), we add a new destination node j to the routing tree T rooted at s.
In step 3 of add_node(T, K, j, Δ), in order to estimate the shared path lengths $\hat{p}(d_1, d_2)$
and $\hat{p}(j, d_i)$ for i =1……..l,s can use a $1 \times (l+1)$ (multicast) probing by sending probes to destination nodes for $j, d_1 \ldots\ldots\ldots\ldots d_l$ alternatively, s can use l+1 (unicast) packet pair probings by sending probes to node pairs $(d_1, d_2), (j, d_1) \ldots\ldots\ldots (j, d_l)$. For any l -ary (balanced) tree with N destination nodes, the depth of the tree is$O(log_l N)$. In the worst case, the add_node procedure needs to be executed $O(log_l N)$ times in order to add a new destination node to the tree. Under unicast packet pair probing, if we apply the add_node procedure to infer the topology of the new tree, we need $O(log_l N)$ packet pair probings, and the computational complexity is $O(log_l N)$. If we apply the RNJ algorithm to infer the topology of the new tree,we need $O(N^2)$packet pair probings, and the computational complexity is $O(N^2 logN)$

2) Procedure delete_node(T, j) :
Procedure delete_node(T, j)deletes a destination node j from routing tree T. It will first remove node j and link $(f(j), j)$ from the tree. If $f(j)$ has only one child left after deleting j, it will then further remove node $f(j)$ and connect the child of $f(j)$ to the parent of $f(j)$, so that the new routing tree maintains the property that each internal node has at least two children.
Procedure: delete_node(T, j)
1. $V = V \setminus j, E = \overline{E} \setminus \cup (f(j), j)$
2. If $f(j)$ has only one child left:
$V = V \setminus f(j)$
$E = E\{ (f(f(j)), f(j), f(j), c))\} \cup (f(f(j), c)$.

A .Sequential Topology Inference Algorithm

For a source node and a set of destination nodes D, we can apply the add_node procedure over the nodes in D in sequence to construct the routing tree topology incrementally, as described in Algorithm 2

Algorithm 2: Sequential Topology Inference Algorithm

Input: Source Node, Destination Nodes,$D = \{1,2, \ldots\ldots N\}$, $\Delta > 0$
1.$V_0 = \{s\}, E_0 = \emptyset, T_0 = (V_0, E_0)$.
2. For $j = 1$ *to* N: $T_j = add\_node(T_{j-1}, s, j, \Delta)$
Output: Tree Tree $\hat{T} = T_N$.

When we compare the RNJ algorithm and the sequential topology inference algorithms By assuming all probings have the same sample size and time interval between two consecutive probes. Under multicast probing, the RNJ algorithm is more efficient (for building the whole tree); while under unicast packet pair probing, the sequential topology inference algorithm is more efficient, in terms of the probing traffic and probing time. In both cases, the sequential topology inference algorithm $O(log_l N)$ is more computationally efficient than the RNJ algorithm $O(N^2 logN)$ where N Destination nodes l −ary Tree with Depth $O(log_l N)$.

VI.CONCLUSION:

In this paper, we have proposed different types of tree topologies such as unicast, multicast topologies and traceroute mechanisms in which probing overhead have occurred during the inferring of network topology. To overcome this problem we have proposed fast and scalable algorithms for network routing tree topology inference using a framework based on a RNJ algorithm that is a grouping type algorithm that recovers the tree topology by recursively joining the neighbors based on additive metrics, which is considered from the existing algorithms to handle these problems. In particular a sequential topology inference algorithm is proposed from this framework to address the probing scalability problem and handles dynamic node joining and leaving efficiently. These frameworks provide powerful tools for large-scale network inference in communication networks.

ACKNOWLEDGMENT

We are greatly delighted to place my most profound appreciation to Er.K.Satyanarayana Chancellor of K.L.University, Assistant Prof V. SRUJANA Guide, Dr.K.Raja Sekhara Rao Principal, Prof S.Venkateswarlu Head of the department, and Dr.Subramanyam in charge for M.Tech under their guidance and encouragement and kindness in giving us the opportunity to carry out the paper. Their pleasure nature, directions, concerns towards us and their readiness to share ideas enthused us and rejuvenated our efforts towards our goal. We also thank the anonymous references of this paper for their valuable comments.

REFERENCES

[1] Jian Ni, Haying Xie, Sekhar Tatikonda, Yang Richard Yang,"Efficient and Dynamic Routing Topology Inference from End-to-End Measurements" IEEE/ACM transactions on networking, vol. 18, no. 1, February 2010
[2] D. Antonova, A. Krishnamurthy, Z. Ma, and R. Sundaram, "Managing a portfolio of overlay paths," in *Proc. NOSSDAV*, Kinsale, Ireland, Jun. 2004, pp. 30–35.
[3] P. Buneman, "The recovery of trees from measures of dissimilarity," in *Mathematics in the Archaeological and Historical Sciences*. Edinburgh,
Scotland: Edinburgh Univ. Press, 1971, pp. 387–395.
[4] J. Hartigan, *Clustering Algorithms*. New York: Wiley, 1975.
[5] N. G. Duffield, J. Horowitz, F. L. Presti, and D. Towsley, "Multicast topology inference from End-to-End measurements," *Adv. Perform. Anal.*, vol. 3, pp. 207–226, 2000.
[6] O. Gascuel and M. Steel, "Neighbor-joining revealed," *Mol. Biol. Evol.*, vol. 23, no. 11, pp. 1997–2000, 2006.
[7] R. Castro, M. Coates, G. Liang, R. Nowak, and B. Yu, "Network tomography: Recent developments," *Statist. Sci.*, vol. 19, no. 3, pp. 499–517, 2004.
[8] BOLOT, J.-C. End-to-end packet delay and loss behavior in the Internet. In *Proceedings of SIGCOMM '93* (San Francisco, CA, Sept. 1993), ACM,pp. 289–298.
[9] M. Coates, R. Castro, R. Nowak, M. Gadhiok, R. King,and Y. Tsang, "Maximum likelihood network topology

identification from edge-based unicast measurements," in *SIGMETRICS'02: Proceedings of the 2002 ACM SIGMETRICS international conference on Measurement and modeling of computer systems*. New York, NY, USA: ACM, 2002, pp.11–20.

[10] J.-C. Bolot, "End-to-end packet delay and loss behavior in the Internet," in *Proc. ACMSIGCOMM*, San Francisco, CA, Sep. 1993, pp. 189–199.

[11] A. Bestavros, J. Byers, and K. Harfoush, "Inference and labeling of metric-induced network topologies," in *Proc. IEEE INFOCOM*, New York, Jun. 2002, pp. 628–637.