A Distance-based Agglomerative Clustering Algorithm for Multicast Network Tomography

Grigorios Kakkavas
School of Electrical & Comp. Engin.
National Technical University of Athens
Iroon Polytechniou 9, Zografou,
Athens, 15780, Greece
Email: gkakkavas@netmode.ntua.gr

Vasileios Karyotis
Department of Informatics
Ionian University
Tsirigoti Sq. 7,
Corfu, 49132, Greece
Email: karyotis@ionio.gr

Symeon Papavassiliou
School of Electrical & Comp. Engin.
National Technical University of Athens
Iroon Polytechniou 9, Zografou,
Athens, 15780, Greece
Email: papavass@mail.ntua.gr

***Abstract*—In this paper, we address the network tomography problems of inferring the multicast routing tree topology and estimating core link performance characteristics (i.e., loss rate, jitter) based on end-to-end measurements from a source node to a set of destination nodes. We extend the agglomerative hierarchical clustering algorithm that works in a bottom-up manner and iteratively joins siblings (i.e., nodes with the same parent) by incorporating the concepts of reciprocal nearest neighbors and nearest neighbors chains. We employ two alternative ways for calculating the required distance matrix of terminal nodes. One based on additive tree metrics and another utilizing several normalized dissimilarity measures on the binary sequences of received/lost probes maintained at each node. Finally, we evaluate the performance of the proposed algorithm in terms of estimation accuracy and correctness of the inferred logical routing tree over real network topologies constructed in an open testbed of the Fed4FIREPlus federation.**

***Index Terms*—Network tomography, agglomerative clustering, end-to-end measurements, topology inference, link metrics.**

I. Introduction

The proliferation of network architectures and technologies calls for efficient and accurate monitoring tools, which will allow for the realization of advanced network management. Among the various monitoring frameworks, *network tomography* has emerged as one of the lean approaches for efficient network monitoring, since it mitigates the need for special purpose cooperation and participation from the various network elements, and reduces the associated traffic overhead footprint on the total network load. Network tomography refers to estimating network topology and performance parameters based on traffic measurements at a limited subset of accessible network elements by the authority requiring them [1].

Network tomography can be classified into various categories depending on the type of acquired data and network parameters of interest [2]. In this paper, we focus on the inference of the unknown topology and the estimation of link-level performance parameters, namely loss rate and jitter (square root of delay variance), based on end-to-end path-level measurements conducted on the network edge and obtained without the cooperation/participation of the internal nodes. The key contributions of our work are summarized as follows:

- We introduce the use of *nearest neighbors (NN) chains* and *reciprocal nearest neighbors (RNNs)* to the agglomerative hierarchical clustering method, improving in that way the computational complexity of the bottom-up clustering algorithm that is widely employed for topology inference and link-level parameter estimation.
- In addition to the typical distances based on loss and delay-related tree additive metrics, we explore an alternative approach that constructs binary sequences at the nodes from the reception/loss of the sent probes and utilizes several well-known normalized dissimilarity measures of binary sequences to express the distance between each pair of nodes.
- We implement a network tomography utility that realizes all discussed methods for both topology inference and link-level parameter estimation. We publish[1] the source code under a permissive free software license, accompanied with detailed documentation.
- We evaluate the performance of the proposed LCA-RNN algorithm (Algorithm 1) over bare-metal hardware provided by an open large-scale testbed (Fed4FIREPlus [3]), testing all possible parametrizations when alternative options are available (e.g., multiple reduction update formulas, dissimilarity measures, etc.).

The remainder of this paper is organized as follows. In Section II, we briefly overview some related work. We introduce the network model and the considered inference problem in Section III, along with two alternative ways of constructing the required distance matrix. In Section IV, we analyze the extended NN chain based bottom-up hierarchical clustering algorithm. In Section V, we describe the experimental setup for the performance evaluation of the proposed algorithm, whereas the results of several physical routing trees are presented in Section VI. Finally, Section VII concludes the paper.

II. Related Work

In [4], the authors introduce three types of algorithms that utilize end-to-end loss measurements in order to infer the

This work was supported by the European Commission, partially in the framework of the H2020-ICT-19-2019 project 5G-HEART (Grant Agreement No. 857034), and in part by H2020 Fed4FIREplus Project (Grant Agreement No. 732638).

[1]https://gitlab.com/gkakkavas/lca-rnn-clustering

underlying multicast logical topology, namely a grouping estimator exploiting the monotonicity of loss rates with increasing path lengths, a maximum-likelihood estimator (MLE), and a Bayesian estimator. The first approach, which belongs to the wider class of hierarchical clustering that is also the focus of our paper, is proven to offer best performance in terms of accuracy and computational cost.

The RNJ algorithm proposed in [5] is a bottom-up grouping algorithm recursively joining neighbors in the routing tree. It can be employed both for the inference of the logical routing tree topology and the estimation of internal link performance parameters. It is based on tree additive metrics, and it is the most relevant to our work, while it can be considered as the typical agglomerative clustering paradigm.

In [6], the authors employ unicast probing and manage to reduce significantly the number of pairwise probes needed to infer the logical routing topology by exploiting a Depth-First-Search (DFS) ordering of the end-hosts. The BHC clustering algorithm presented in [7] identifies the multicast routing topology based on the Hamming distances between all pairs of sequences of received and lost probes maintained at each node, while also incorporating hop-count information. However, such information is not always available in practice, thus limiting the applicability of the algorithm.

III. Network Model and Problem Formulation

Let $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ denote the topology of the network that consists of end hosts, internal switches and routers (node set $\mathcal{V}$), and the communication links that join them (link set $\mathcal{E}$). Assuming that during the measurement period the underlying routing algorithm determines a unique path from a source to each destination that is reachable from it, the physical routing topology from a source node to a set of destination nodes is a directed tree called *physical routing tree*. From the physical routing tree, we can derive the *logical routing tree* which is defined by the branching points, i.e., internal nodes with two or more outgoing links. Internal nodes where no branching of traffic occurs do not appear in the logical tree. A logical link may comprise more than one consecutive physical links. Each node in the logical tree has at least two children, except for the root (which has one) and the leaves (which have none). Thus, the degree of an internal node on the logical routing tree is at least three. If all internal nodes have exactly two children, the tree is called *binary*.

Assume a source node s and the set of destination nodes D that are reachable from s. The logical routing tree from s to nodes in D with node set $V \subseteq \mathcal{V}$ and link set $E \subseteq \mathcal{E}$ is denoted by $T(s, D) = (V, E)$. Every node $k \in V \setminus \{s\}$ has a parent $f(k) \in V$ such that link $e_k \equiv (f(k), k) \in E$. Let $c(k) = \{j \in V : f(j) = k\}$ be the set of children of node $k \in V$. Then, $|c(s)| = 1$, $c(i) = \emptyset, \forall i \in D$ and $|c(i)| \geq 2, \forall i \in V \setminus \{s\} \cup D$. The path $p(i, j)$ denotes the sequence of links that connect node j to i on the logical routing tree. Finally, each link $e \in E$ is associated with a performance parameter θ_e. The network tomography problem under consideration involves using end-to-end measurements taken at the terminal nodes (i.e., nodes in $\{s\} \cup D$) to infer the topology of the logical routing tree and the parameters $\theta_e,\ e \in E$ associated with its links.

The state of each link on the logical routing tree is represented by the set of *link state variables* Z_e, $\forall e \in E$, $Z_e \in \mathcal{Z}$, whose distribution is parameterized by the respective θ_e parameter. Similarly, a set of *outcome variables* is defined for all nodes of the tree. More precisely, for each node $k \in V$, X_k takes value in set $\mathcal{X}$ and denotes the (random) outcome at node k. The outcome at node k is determined by the outcome at its parent node and the state of the link that connects them.

An additive metric on $T(s, D)$ associates each link $e \in E$ with a finite, strictly positive link length $d(e)$, whereas the distance $d(i, j)$ between any pair of nodes, $i, j \in V$, is expressed as the summation of the link lengths along the path $p(i, j)$ that connects them. The topology and the link lengths of a tree $T(s, D)$ are uniquely determined by the distances $\mathcal{D} = \{d(i, j) : i, j \in \{s\} \cup D\}$ between the terminal nodes under any additive metric on the tree [8]. Furthermore, if there is an one-to-one mapping between the link performance parameters θ_e and link lengths $d(e),\ e \in E$, then the former can be recovered from the latter. However, in actual network inference problems, we only have end-to-end (from source s to destination nodes in D) measurements taken at the terminal nodes, based on which the required pairwise distances are estimated. Subsequently, we can use the estimated distances between the terminal nodes to infer the logical routing tree topology and the link performance parameters.

A. Multicast-based Additive Tree Metrics

Under the assumption of independence and stationarity of link states during the measurement period, we construct the following additive tree metrics:

1) *Loss rate*: the negative logarithm of the complement of the packet loss rate is additive. In greater detail, the link state variable Z_e is a Bernoulli random variable that takes value 1 (or 0) with probability θ_e (or $1-\theta_e$) if the probe travels through (or is lost on) link e. Therefore, θ_e expresses the success rate of link e, whereas $1 - \theta_e$ denotes the respective loss rate. Similarly, the outcome variable X_k is a Bernoulli random variable that takes value 1, if the probe reaches node k, and value 0 otherwise. For the source s generating the probes, we assume $X_s \equiv 1$ and $\mathbb{P}(X_s = 1) = 1$. For the rest nodes, i.e., $\forall k \in V \setminus \{s\}$, we have:

$$X_k = X_{f(k)} \cdot Z_{e_k} = \prod_{e \in p(s,k)} Z_e, \quad \text{and}$$

$$\mathbb{P}(X_k = 1) = \prod_{e \in p(s,k)} \mathbb{P}(Z_e = 1) = \prod_{e \in p(s,k)} \theta_e.$$

Consequently, we can construct an additive tree metric with link length $d(e) = -\log(\theta_e),\ \forall e \in E$, where θ_e denotes the complement of the loss rate (i.e., the success rate) of link e.

2) *Delay variance*: the square of the jitter is additive (when jitter is defined as the standard deviation of latency), since the variance of a sum of independent random variables is the sum of their variances. More precisely, the link state variable Z_e

is a random variable that expresses the random queuing delay of link e. The performance parameter associated with link e is defined as the second moment of the respective link state variable, i.e., $\theta_e = \text{Var}(Z_e)$. The outcome variable X_k denotes the cumulative end-to-end queuing delay that is experienced by the probe until reaching node k. Clearly, for the source s it is $X_s \equiv 0$ and $\text{Var}(X_s) = 0$. For the rest nodes, i.e., $\forall k \in V \setminus \{s\}$, we have:

$$X_k = X_{f(k)} + Z_{e_k} = \sum_{e \in p(s,k)} Z_e, \quad \text{and}$$

$$\text{Var}(X_k) = \text{Var}\left(\sum_{e \in p(s,k)} Z_e\right) = \sum_{e \in p(s,k)} \text{Var}(Z_e) = \sum_{e \in p(s,k)} \theta_e.$$

Therefore, we can construct an additive tree metric with link length $d(e) = \theta_e = \text{Var}(Z_e),\ \forall e \in E$, where $\text{Var}(Z_e)$ is the square of jitter of link e.

B. Additive Tree Metric Distance Matrix Estimation

The joint distributions of the outcome variables at the terminal nodes (i.e., nodes $\{s\} \cup D$) enable us to calculate the respective pairwise distances and form the *distance matrix* under the employed additive tree metric, which is necessary for inferring the logical routing topology and the corresponding link performance parameters. However, in practice we do not know the actual distributions. Instead, we have end-to-end measurements between the source and each of the destination nodes, which we must leverage in order to estimate the required pairwise distances. In particular, assuming source s sends n probes to the nodes in the destination set D, only the outcome variables at the terminal nodes can be measured and observed $\left(X_k^{(t)} :\ k \in \{s\} \cup D \text{ and } t = 1, 2, \cdots, n\right)$. Then, depending on the employed additive metric we have:

1) *Loss rate*: the pairwise distances between the terminal nodes can be calculated as:

$$d(s,i) = \log\left(\frac{1}{\mathbb{P}(X_i = 1)}\right) = -\log\left(\mathbb{P}(X_i = 1)\right),\ \forall i \in D,$$

$$\text{and } d(i,j) = \log\left(\frac{\mathbb{P}(X_i = 1) \cdot \mathbb{P}(X_j = 1)}{\left[\mathbb{P}(X_i X_j = 1)\right]^2}\right),\ \forall i,j \in D.$$

For every probe $t = 1, 2, \cdots, n$ the corresponding value of the outcome variable at destination node k, i.e, $X_k^{(t)}$, is equal to 1, if the probe reaches that specific node and equal to 0, if it gets lost somewhere along the way. Given that for a Bernoulli random variable, the maximum likelihood estimator (MLE) of the probability that it takes value 1 is equal to the sample mean, we can construct the following estimators for the pairwise distances between the terminal nodes:

$$\hat{d}(s,i) = \log\left(\frac{1}{\overline{X_i}}\right) = -\log\left(\overline{X_i}\right),\ \forall i \in D,$$

$$\text{and } \hat{d}(i,j) = \log\left(\frac{\overline{X_i} \cdot \overline{X_j}}{\overline{X_i X_j}^2}\right),\ \forall i,j \in D,$$

$$\text{where } \overline{X_i} = \frac{1}{n}\sum_{t=1}^{n} X_i^{(t)} \text{ and } \overline{X_i X_j} = \frac{1}{n}\sum_{t=1}^{n} X_i^{(t)} X_j^{(t)}.$$

2) *Delay variance*: the pairwise distances between the terminal nodes can be calculated as:

$$d(s,i) = \text{Var}(X_i),\ \forall i \in D, \quad \text{and}$$

$$d(i,j) = \text{Var}(X_i) + \text{Var}(X_j) - 2 \cdot \text{Cov}(X_i, X_j).$$

Using the unbiased sample variance (with lost packets ignored), we can construct the following estimators for the pairwise distances between the terminal nodes:

$$\hat{d}(s,i) = \widehat{\text{Var}}(X_i),\ \forall i \in D, \quad \text{and}$$

$$\hat{d}(i,j) = \widehat{\text{Var}}(X_i) + \widehat{\text{Var}}(X_j) - 2\widehat{\text{Cov}}(X_i, X_j),\forall i,j \in D,$$

$$\text{where } \widehat{\text{Var}(X_i)} = \frac{1}{n-1}\sum_{t=1}^{n}\left(X_i^{(t)} - \overline{X_i}\right)^2, \text{ and}$$

$$\widehat{\text{Cov}}(X_i, X_j) = \frac{1}{n-1}\sum_{t=1}^{n}\left(X_i^{(t)} - \overline{X_i}\right)\left(X_j^{(t)} - \overline{X_j}\right).$$

C. Multicast-based Binary Sequences Dissimilarities

An alternative approach for constructing the distance matrix under the aforementioned loss model involves a slight modification of the outcome variables and the use of binary sequences. In particular, assuming source s sends n probes to the nodes in the destination set D, we maintain the sequences $S_k = \{s_k^{(t)},\ 1 \le t \le n\},\ \forall k \in D$, with $s_k^{(t)} = 1$ if the t-th probe has reached node k and $s_k^{(t)} = 0$ otherwise. By convention, we consider that all probes "reach" the source s, therefore all n elements of sequence S_s are equal to 1. Subsequently, we can calculate the pairwise distances between the terminal nodes and form the respective distance matrix by measuring the dissimilarity of the corresponding binary sequences. More specifically, we employ the following normalized (with values in the range $[0,1]$) dissimilarity measures [9], [10]:

- Jaccard-Needham

$$d_{JN} = \frac{C_{10} + C_{01}}{C_{11} + C_{01} + C_{10}}, \tag{1}$$

- Hamming

$$d_H = \frac{C_{01} + C_{10}}{n}, \tag{2}$$

- Dice

$$d_D = \frac{C_{10} + C_{01}}{2C_{11} + C_{01} + C_{10}}, \tag{3}$$

- Rogers-Tanimoto

$$d_{RT} = \frac{2C_{10} + 2C_{01}}{C_{11} + C_{00} + 2C_{10} + 2C_{01}}, \tag{4}$$

where $C_{ij},\ i,j \in \{0,1\}$ is the number of occurrences of matches with i in the first sequence and j in the second sequence at the corresponding positions.

IV. Logical Routing Tree and Link Parameters Inference

In order to reconstruct the binary logical routing tree from the (estimated) pairwise distances of terminal nodes, we will employ an agglomerative hierarchical clustering approach that begins with a set including all the destination nodes in D and each destination node in its own cluster. In each step, two neighboring nodes are joined to one cluster and are replaced in the set by a new node designated as their parent. Clustering continues recursively on this reduced set, until there is only one node left in the set, which will be the child of the root s.

The various agglomerative clustering algorithms are differentiated based on two important components of the previous procedure: the neighbor selection criterion according to which the nodes are joined, and the reduction update formula used for the calculation of the distances in the reduced set (i.e., the distances from the newly generated parent to the rest nodes).

A. Agglomerative Hierarchical Clustering for Additive Tree Metrics

Given the source node s, let $\text{LCA}(i,j),\ \forall i,j \in D$ denote the lowest common ancestor of the nodes i and j in the logical routing tree, which is the node that is located the farthest from the root and has both i and j as descendants (where we define each node to be a descendant of itself). Then, $\ell(i,j) = d(s, \text{LCA}(i,j))$ expresses the depth of the lowest common ancestor of i and j, which can be viewed as the shared path length from s to i and j, and $\ell(i,i) = d(s,i)$ denotes the path length from s to i. Under any additive tree metric, there is an one-to-one mapping between pairwise distances and lowest common ancestor depths of terminal nodes as defined by:

$$\ell(i,j) = \frac{1}{2}\left[d(s,i) + d(s,j) - d(i,j)\right],\ \forall i,j \in D. \tag{5}$$

Therefore, given the set of (estimated) pairwise distances $\mathcal{D} = \{d(i,j) : i,j \in \{s\} \cup D\}$ of terminal nodes, we can construct the (symmetric) matrix of LCA depths $\mathbf{L} = [\ell(i,j)],\ \forall i,j \in D$. In [5], the neighbor selection criterion is the maximization of the LCA depth, that is nodes i and j with the largest $\ell(i,j)$ are considered siblings. However, it is not necessary to find the (computationally expensive) pair of nodes with the deepest lowest common ancestor (i.e, largest LCA depth). Instead, nodes i and j can be considered siblings if they are the nearest neighbor of each other, in which case they are called *mutual or reciprocal nearest neighbors* (RNNs). In the context of the LCA depths matrix, this means that the selected pair i, j should correspond to a maximal off-diagonal entry in rows i, j, but not necessarily in the entire matrix, i.e., nodes i and j are siblings that can be joined and substituted by parent v if and only if $\forall k \neq i,j,\ \ell(i,j) \geq \ell(i,k), \ell(j,k)$. Such pairs of RNNs can be found efficiently based on the construction of nearest neighbors (NN) chains [11], [12].

More precisely, a NN chain consists of an (arbitrarily chosen) initial node i, followed by its nearest neighbor in terms of LCA, meaning node j such that $\ell(i,j) = \max_{k \neq i} \ell(i,k)$ (or equivalently the node that corresponds to the column index of the maximal off-diagonal value of the row i in matrix $\mathbf{L}$), followed by the nearest neighbor of this second node and so on, until it eventually gets complete by terminating at a pair of nodes that are nearest neighbors of each other (reciprocal nearest neighbors). The latter are chosen to be merged as soon as they are found and they are about to be replaced by a newly generated parent node u. Provided that the reduction is convex, i.e., $\forall k \neq u,\ \min\{\ell(i,k), \ell(j,k)\} \leq \ell(u,k) \leq \max\{\ell(i,k), \ell(j,k)\}$, Bruynooghe's reducibility property [11] is preserved and the remaining nodes after the removal of i and j continue to form an incomplete NN chain (meaning that consecutive pairs in the chain remain nearest neighbors) that needs to be extended until the termination condition is met again. Hence, the largest part of the chain can be reused along subsequent iterations and can be extended starting from the node preceding the last found RNNs or from an arbitrary node if those RNNs happened to be the only two elements of the chain. By merging RNNs as they appear in the NN chain, nodes are merged in a different order than that of greedy methods that always join the overall closest pair. However, the generated hierarchy results to be the same.

The NN chain method consists of a sequence of extending operations, some of which lead to termination, whereas the rest lead to the extension of the chain by one additional node. Each such operation requires the computation of a maximal row element of matrix $\mathbf{L}$, which can be done in linear time. Thus, the overall time complexity of constructing and maintaining the complete NN chain is determined by the total number of extending operations that are performed throughout the execution of the algorithm. Assuming n nodes in the destination set D, $n-1$ operations lead to termination (we must find and remove from the chain $n-1$ pairs of RNNs, one in each iteration). Furthermore, since in each iteration only two nodes are removed from the chain and by the end of the algorithm the chain is reduced to a single element, the total number of nodes that ever get added to the chain cannot be more than $2n-2$. Consequently, the total number of extending operations cannot be more than $3n-3$, thus yielding a total running time of $O(n^2)$.

Regarding the reduction update formula, we consider two alternative options. The first is the so-called mid-point reduction that is also employed by the Rooted Neighbor-Joining (RNJ) algorithm presented in [5]. According to this approach $\ell(u,k) = \frac{1}{2}\left[\ell(i,k) + \ell(j,k)\right],\ \forall k \in D : k \neq u$, where u is the newly generated parent of the nodes i and j that are being joined. The second option is the so-called maximal-value reduction expressed by the formula $\ell(u,k) = \max\{\ell(i,k), \ell(j,k)\}$.

Taking into account all of the above, the logical tree topology inference algorithm (henceforth denoted LCA-RNN algorithm) is presented in Algorithm 1. Assuming there are n destination nodes (i.e., $|D| = n$), the proposed algorithm terminates after $n-1$ iterations. In each iteration, the reduction stage requires $O(n)$ operations, thus resulting to a total $O(n^2)$. Furthermore, the RNN pair selections can be achieved by constructing and maintaining a complete NN chain throughout the execution of the algorithm in $O(n^2)$ time as previously explained. Therefore, the overall time complexity of the algorithm has a bound of $O(n^2)$ (for comparison, the RNJ [5] algorithm has a complexity of $O(n^2 logn)$).

Algorithm 1 LCA-RNN Algorithm

```
Input: Source $s$, destinations $D$, distances $d(i,j) : i,j \in \{s\} \cup D$.

1: Initialization
     Set $V = \{s\} \cup D$ and $E = \emptyset$.
2: LCA depths matrix deduction:
     Compute the $|D| \times |D|$ matrix of LCA depths $\mathbf{L} = [\ell(i,j)]$, $\forall i,j \in D$ where:
       $\ell(i,j) = \frac{d(s,i)+d(s,j)-d(i,j)}{2}, \ \forall i,j \in D,$
       and $\ell(i,i) = d(s,i), \ \forall i \in D.$
3: Neighbor selection:
     Find $i', j' \in D$ that are reciprocal nearest neighbors by constructing the complete NN chain on matrix $\mathbf{L}$.
4: Reduction:
     Create node $u$ as the parent of $i'$ and $j'$.
     Reduce (symmetric) matrix $\mathbf{L}$ by deleting row and column $j'$ and replacing row $i'$ with a new row $u$ with values:
       $\ell(u,k) = \frac{\ell(i',k)+\ell(j',k)}{2}$ or
       $\ell(u,k) = \max\{\ell(i',k), \ell(j',k)\}, \ \forall k \in D, \ k \neq u, i', j',$
       and $\ell(u,u) = \ell(i',j').$
5: Tree reconstruction:
     Remove nodes $i'$ and $j'$ from set $D$:
       $D = D \setminus \{i',j'\}.$
     Add parent node $u$ to the set of tree nodes $V$ and to set $D$:
       $V = V \cup \{u\}$ and $D = D \cup \{u\}.$
     Add the two new edges to set $E$:
       $E = E \cup \{(u,i'),(u,j')\}.$
     Connect $i'$ and $j'$ to $u$ with link lengths:
       $d(u,i') = \max\{0, \ \ell(i',i') - \ell(i',j')\},$
       $d(u,j') = \max\{0, \ell(j',j') - \ell(i',j')\}.$
6: Stopping condition:
     If $|D| = 1$ (or equivalently $\mathbf{L} = [\ell]$) then for the node $k \in D$, set $d(s,k) = 0$ and $E = E \cup (s,k)$ (i.e., connect the remaining node with source $s$) else go to Step 3.

Output: Tree $T = (V,E)$ and link lengths $d(e), \forall e \in E$.
```

B. Agglomerative Hierarchical Clustering for Binary Sequences

In the case of the binary sequences based loss model presented in Section III-C, the LCA-RNN algorithm is slightly modified. Given that the binary sequences of the internal nodes can be calculated from the sequences maintained at the terminal nodes, there is no need of employing a reduction update formula at Step 4 and the lengths of the new edges at Step 5 can be calculated directly. More precisely, we assume that a specific probe reaches an internal node, if at least one of the node's children also receives it successfully. Therefore, at each iteration the binary sequence of the newly constructed parent node u can be calculated by applying the element-wise logical-OR operation to the respective sequences of the chosen reciprocal nearest neighbors i' and j' that are considered its children: $S_u = S_{i'} \vee S_{j'}$.

After that, we can employ one of the dissimilarity measures (1) to (4) to calculate the pairwise distances between u and the rest nodes in set $\{s\} \cup D$ (i.e., $d(u,k)$, $\forall k \in \{s\} \cup D$, $k \neq u$) and update the matrix of LCA depths by applying (5). We note that under this model there is no one-to-one mapping between the link performance parameters θ_e and link lengths $d(e)$, $e \in E$. As a consequence, we can only employ the LCA-RNN algorithm to infer the underlying topology and not the link characteristics.

C. Extension to General Trees

Since an RNN pair is chosen as a pair of siblings (i.e., nodes that have the same parent) at every iteration, the proposed algorithm will result in a binary inferred tree. Thus, the application of the LCA-RNN algorithm to an arbitrary topology produces a binary tree that contains "imaginary" edges of zero-length. A straightforward way to extend the inferred topology to general trees is to add a post-processing pruning stage, during which the nodes that are connected with zero-length edges (excluding the edge between the root and its single child that is of zero-length by design) are joined.

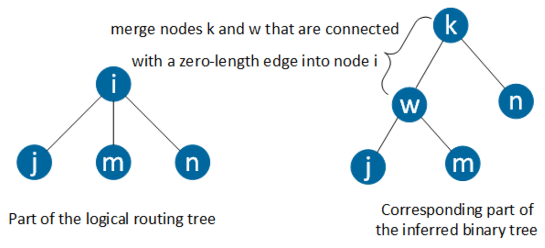

merge nodes k and w that are connected
with a zero-length edge into node i
i
j
m
n
k
w
n
j
m
Part of the logical routing tree
Corresponding part of
the inferred binary tree

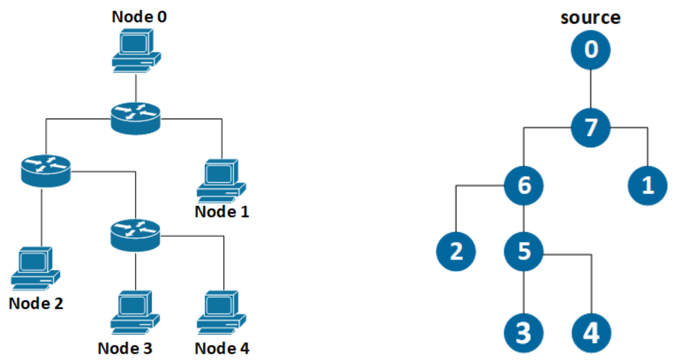
Fig. 1. Binary tree pruning.

Fig. 1 illustrates this process. Nodes k and w of the inferred binary tree are to be merged in order to recreate node i. It should be noted that the lengths of the remaining edges involved are estimated correctly and the only correction needed is the update of the respective end with the new joined node. In practice, because of accumulated errors introduced during the measurement process and the estimation of the involved distances, we employ a threshold $\tau > 0$ and prune all the links e_k for which $\hat{d}(e_k) \leq \tau$ (or equivalently join nodes k and $f(k)$ at the ends of the edge).

V. Experimental Setup

The effectiveness of the proposed approach is demonstrated and quantitatively evaluated by exploiting the capabilities of the Virtual Wall testbed of ILAB.T [13] (part of the

Fed4FIREPlus federation [3]). More precisely, the provided hardware was employed for the construction of several physical routing tree topologies (as described in Section III) with Ubuntu 18.04 LTS 64-bit operating system running directly on the nodes. The nodes acting as routers required specialized routing software to enable networking protocols. To that end, we installed the *Quagga* network routing software suite [14] and leveraged its Open Shortest Path First (OSPFv2) and Protocol Independent Multicast for Source-specific Multicast (PIM-SSM) implementations. Network impairment (i.e., delay and loss) on links between nodes was implemented on software, using the *netem* kernel module that provides network emulation functionality and the *tc* utility that configures the kernel structures required to support traffic control [15].



(a) Real physical routing tree (b) Inferred logical routing tree

Fig. 2. Binary Tree Topology Inference.

TABLE I
BINARY TREE LINK PERFORMANCE PARAMETER ESTIMATION

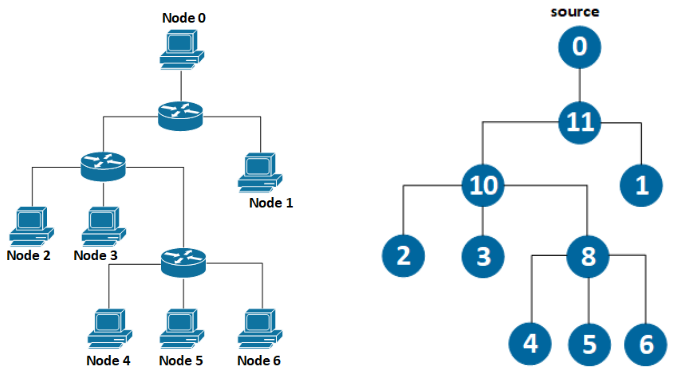
	Loss Rate (%)				Jitter (ms)			
# Pr.		**2k**	**5k**	**10k**		**2k**	**5k**	**10k**
e	**tc**	**Estimation**			**tc**	**Estimation**		
(7,1)	20	19.88	19.67	19.34	200	200.26	203.16	199.27
(7,6)	13	11.87	12.24	12.90	150	152.39	150.72	151.10
(6,2)	10	10.58	10.34	9.63	100	101.27	98.62	101.02
(6,5)	5	5.15	5.21	4.73	50	48.38	49.9	48.88
(5,3)	7	6.63	6.94	7.11	20	24.30	23.67	23.71
(5,4)	15	14.33	15.49	14.48	40	37.79	37.64	37.96

Active end-to-end probing was carried out using the iPerf network traffic tool [16]. In particular, the destination nodes (i.e., leaves of the tree) joined the same multicast group (e.g., 239.1.2.3) by operating the appropriate iPerf servers and the source (i.e., root of the tree) generated UDP traffic to that group via a suitable iPerf client. The target probing bandwidth was set to 1 Mbit/s and the measurement period was selected as 25s, 60s or 120s, thus resulting in 2k, 5k or 10k sent datagrams respectively. For the purpose of our experiments, each datagram was assumed a probe packet. In order to obtain the desired end-to-end measurements, the aforementioned probes were captured at the outgoing interface of the source and at the incoming interfaces of the destination nodes via *tcpdump* and *libpcap* [17]. The description of the contents of the packets at the network interfaces that matched the provided filter expression (in our experiments, udp and port 5001 and dst 239.1.2.3) preceded by the corresponding timestamps expressed in hours, minutes, seconds, and fractions of a second since midnight were saved to text files and were later processed by a Python script that estimated the distances between the terminal nodes as described in Subsection III-B.

It should be noted that whenever the kernel needs to send a packet to an interface, it en-queues it to the queuing discipline that exists between the protocol output and the driver queue of the associated NIC. Since network impairment is achieved by configuring that queuing discipine using tc, and end-to-end measurements depend on tcpdump capturing probe packets at the appropriate NICs, in order to obtain accurate starting measurements for the end-to-end paths between the source and each destination node, we leverage the fact that in the assumed network model the source always has a single child and we never impair that specific link.

VI. Results and Discussion

We have developed in Python 3.7.4 a fully functional command-line network tomography utility that implements the presented algorithm with all alternative options, namely loss rate, delay variance and loss sequence models, mid-point and maximal-value reduction formulas, and the four binary sequences dissimilarity measures. The utility takes as input the tcpdump files of the source and the destination nodes and outputs the inferred logical routing tree topology, along with the link performance parameters, if applicable.



(a) Real physical routing tree (b) Inferred logical routing tree

Fig. 3. General Tree Topology Inference.

TABLE II
GENERAL TREE LINK PERFORMANCE PARAMETER ESTIMATION

	Loss Rate (%)				Jitter (ms)			
# Pr.		**2k**	**5k**	**10k**		**2k**	**5k**	**10k**
e	**tc**	**Estimation**			**tc**	**Estimation**		
(11,10)	10	10.37	9.75	9.89	50	45.3	50.6	48.9
(11,1)	15	14.50	15.17	14.14	220	220.4	223.6	217.8
(10,2)	13	13.38	13.31	13.08	150	150.4	149.9	150.5
(10,3)	20	19.85	19.92	19.15	80	79.4	79.9	80.9
(10,8)	8	7.50	7.87	7.80	110	108.6	110.8	107.8
(8,4)	15	15.99	14.72	14.34	70	66.8	69.6	69.9
(8,5)	11	10.36	11.64	11.55	50	49.6	49.1	51.4
(8,6)	13	11.85	13.32	13.16	60	60.4	60.2	58.6

Fig. 2a illustrates the simple topology used for testing. It is verified that the logical routing tree, which in this case coincides with the physical routing tree, is inferred correctly. Furthermore, Table I presents in fine-grained detail the estimations of the link performance parameters when the mid-point reduction formula is employed. Finally, Fig. 3 and Table II demonstrate similar results for the general tree extension.

Tables III and IV provide an overview of the accuracy of the link characteristics estimation for several physical routing trees, both binary and general. More precisely, they provide the Mean Square Error (MSE) statistic for every conducted experimental scenario, which is defined as:

$$\mathrm{MSE} = \frac{1}{|E|-1} \sum_{e \in E \setminus (s, c(s))} \left(y_e - \hat{y}_e\right)^2 ,$$

where $T(s,D) = (V,E)$ is the logical routing tree, y_e is the loss rate (or the jitter) of link e and the edge between the source and its unique child has been excluded from the calculation, given that it is never impaired by design.

TABLE III
AGGREGATED EXPERIMENTAL RESULTS FOR LOSS RATE (%)

Physical Routing Tree		Mid-point			Maximal Val.		
		2k	5k	10k	2k	5k	10k
Type	# Nds	Mean Square Error					
Bin.	8	0.3751	0.1804	0.1569	0.3472	0.1379	0.1649
Bin.	16	1.3856	0.3419	0.1387	3.2834	0.4841	0.1723
Bin.	24	1.2981	0.2588	0.1571	1.6536	0.3185	0.2439
Bin.	32	1.4352	0.6979	0.1935	2.4624	0.9732	0.3123
Gen.	10	0.4392	0.101	0.287	0.6063	0.1006	0.4088
Gen.	20	0.7687	0.2856	0.1007	0.9362	0.4861	0.1325
Gen.	30	0.5523	0.2659	0.2226	0.7902	0.2999	0.2137
Gen.	40	1.3112	0.4353	0.2622	1.6839	0.7002	0.4794

TABLE IV
AGGREGATED EXPERIMENTAL RESULTS FOR JITTER (MS)

Physical Routing Tree		Mid-point			Maximal Val.		
		2k	5k	10k	2k	5k	10k
Type	# Nds	Mean Square Error					
Bin.	8	5.564	5.24	3.657	6.17	5.499	3.899
Bin.	16	51.2	22.35	11.47	57.038	22.474	16.039
Bin.	24	58.438	22.057	12.198	79.298	22.295	12.079
Bin.	32	122.269	46.111	19.361	213.994	102.907	42.716
Gen.	10	4.421	1.851	2.03	13.22	2.417	2.393
Gen.	20	33.485	22.296	9.696	86.869	34.617	14.872
Gen.	30	46.75	27.684	11.575	164.415	70.508	26.968
Gen.	40	105.291	26.493	18.382	196.975	59.68	46.198

As it can be observed, the use of the mid-point reduction update formula leads consistently to better estimations. Furthermore, as expected, the accuracy of the estimations is improved as the number of the employed probes increases. However, in the case of loss rates, the differences in the produced errors are small enough to avoid prohibiting the use of less probes. Regarding the reported errors, we should note that during the calculations we consider as true values (i.e., y_e) the arguments provided to the *tc* command, which may differ from the actual realized values, especially for jitter that is susceptible to various limitations, such as the clock resolution of the kernel and the employed distribution tables. Finally, it should be pointed out that in the case of general trees, the logical routing tree topology is inferred correctly in all of the conducted experiments after a suitable threshold has been identified, both for the additive tree metrics and the binary sequences based distances. The binary routing trees are also reconstructed correctly.

VII. CONCLUSION

In this paper, we addressed the problem of inferring the multicast routing tree and estimating the internal link performance parameters of loss rate and jitter based on end-to-end measurements from a source to a set of destination nodes. We extended the traditional bottom-up hierarchical clustering algorithm by incorporating the concepts of reciprocal nearest neighbors and nearest neighbors chain. We devised two alternative ways of constructing the required matrix of pairwise distances between terminal nodes. Finally, we evaluated the performance and accuracy of the proposed algorithm with experiments conducted on a Fed4FIREPlus testbed.

REFERENCES

[1] A. Coates, A. O. Hero III, R. Nowak, and B. Yu, "Internet tomography," *IEEE Signal Process. Mag.*, vol. 19, no. 3, pp. 47–65, May 2002.

[2] R. Castro, M. Coates, G. Liang, R. Nowak, and B. Yu, "Network tomography: Recent developments," *Statist. Sci.*, vol. 19, no. 3, pp. 499–517, Aug. 2004.

[3] Federation for FIRE Plus. [Online]. Available: https://www.fed4fire.eu/

[4] N. Duffield, J. Horowitz, F. Lo Presti, and D. Towsley, "Multicast topology inference from measured end-to-end loss," *IEEE Trans. Inform. Theory*, vol. 48, no. 1, pp. 26–45, 2002.

[5] J. Ni and S. Tatikonda, "Network tomography based on additive metrics," *IEEE Trans. Inform. Theory*, vol. 57, no. 12, pp. 7798–7809, Dec. 2011.

[6] B. Eriksson, G. Dasarathy, P. Barford, and R. Nowak, "Efficient network tomography for internet topology discovery," *IEEE/ACM Trans. Networking*, vol. 20, no. 3, pp. 931–943, Jun. 2012.

[7] H. Tian and H. Shen, "Multicast-based inference for topology and network-internal loss performance from end-to-end measurements," *Comput. Commun.*, vol. 29, no. 11, pp. 1936–1947, Jul. 2006.

[8] P. Buneman, "The recovery of trees from measures of dissimilarity," *Mathematics of Archeological & Historical Sciences*. Edinburgh University Press, pp. 387–395, 1971.

[9] B. Zhang and S. N. Srihari, "Binary vector dissimilarity measures for handwriting identification," in *Doc. Recogn. and Retr. X*, T. Kanungo, E. H. B. Smith, J. Hu, and P. B. Kantor, Eds. SPIE, Jan. 2003.

[10] S.-S. Choi, S.-H. Cha, and C. C. Tappert, "A survey of binary similarity and distance measures," *J. Systemics, Cybernetics & Informatics*, vol. 8, no. 1, pp. 43–48, 2010.

[11] F. Murtagh, "Complexities of hierarchic clustering algorithms: State of the art," *Comput Stat Quarterly*, vol. 1, no. 2, pp. 101–113, 1984.

[12] F. Murtagh and P. Contreras, "Algorithms for hierarchical clustering: An overview," *WIREs Data Mining Knowl Discov*, vol. 2, no. 1, pp. 86–97, Dec. 2011.

[13] Virtual wall. [Online]. Available: https://doc.ilabt.imec.be/ilabt/virtualwall/

[14] Quagga routing suite. [Online]. Available: https://www.nongnu.org/quagga/

[15] The Linux Foundation. Network emulation (netem). [Online]. Available: https://wiki.linuxfoundation.org/networking/netem

[16] iperf2: A tool for measuring TCP and UDP network performance. [Online]. Available: https://sourceforge.net/projects/iperf2/

[17] Tcpdump & libpcap. [Online]. Available: https://www.tcpdump.org/