

Received March 3, 2020, accepted March 19, 2020, date of publication March 23, 2020, date of current version April 8, 2020.

Digital Object Identifier 10.1109/ACCESS.2020.2982653

Network Topology Inference Using Higher-Order Statistical Characteristics of End-to-End Measured Delays

GAOLEI FEI¹, JIAN YE¹, SHENG WEN², (Member, IEEE), AND GUANGMIN HU¹

¹School of Information and Communication Engineering, University of Electronic Science and Technology of China, Chengdu 611731, China

²School of Software and Electrical Engineering, Swinburne University of Technology, Hawthorn, VIC 3122, Australia

Corresponding author: Gaolei Fei (fgl@uestc.edu.cn)

This work was supported in part by the National Natural Science Foundation of China under Grant 61301274, and in part by the Sichuan Science and Technology Program under Grant 2019YFG0456.

ABSTRACT Network topology is important information for many network control and management applications. Network tomography infers network topology from end-to-end measured packet delays or losses, which is more feasible than internal cooperation-based methods and attracts many studies. Most of the existing methods for network topology inference usually function under the assumption that the distribution of packet delay or loss follows a given distribution (e.g., Gaussian or Gaussian mixture), and they estimate network topology from the parameters of the given distribution. However, these methods may fail to obtain an accurate estimation because the real distribution of packet delay or loss usually cannot be described by a certain distribution. In this paper, we present a novel network topology inference method based on the unicast end-to-end measured delays. The method abandons the assumption of packet delay distribution and constructs network topology by inferring the higher-order cumulants of internal links from the end-to-end measured delays. The analytical and simulation results show that the proposed method offers over 10% improvement in accuracy compared with that of the state-of-the-art works.

INDEX TERMS Higher-order cumulant, topology inference, end-to-end measurement, network tomography.


I. INTRODUCTION

As size of the Internet has grown dramatically, it has become a giant system with a complex structure. Network routing topology describes the connections among the devices on the Internet and plays a significant role in many network management tasks such as understanding network bottlenecks, localization of failures, learning network internal performance and optimizing network design. As a result, determining how to understand the topological information of a network has become an important area in the field of network measurement research and attracted many recent studies [1].

The existing methods for network topology measurement can be divided into two categories: internal cooperation-based methods and network tomography-based methods. The internal cooperation-based methods construct network topology using the information retrieved from

internal routers such as routing information and ICMP packets [2]–[4]. However, these methods may fail if the internal routers refuse to return the topology information because this may incur security problems and represents deployed prohibition by ISPs. The network tomography-based methods [5], [6] (also called network topology inference methods) collect the path performance parameters (e.g., packet queue delay or packet loss) via end-to-end measuring and infer the network topology from the path performance parameters using statistical methods. In practical use, the network tomography-based methods are more feasible than the internal cooperation-based methods because they are capable of obtaining the topology without the cooperation of internal nodes. Based on the estimated topology, network tomography can also be extended to learn more network internal performance parameters [7]–[9] such as delay [10], loss [11] and bandwidth [12].

The existing network topology inference methods estimate the network topology using the means and variances of the

The associate editor coordinating the review of this manuscript and approving it for publication was Muhammad Imran .

performance parameters of shared paths from a source to a set of destination nodes, which indicate that the packet queue delay or packet loss in the network follows a given distribution (e.g., Gaussian [13] or Gaussian mixture [14]). Then, the network topology can be recovered by computing the first or the second statistical characteristic of the path performance parameters. However, due to the complexity of real network environments, the performance parameters in the network usually cannot be accurately described using a certain distribution, and the statistical information of the path performance parameters is not fully utilized by simply relying on the first or the second statistical characteristic, which makes the estimated topology exhibit significant deviation from the actual topology.

Aiming at the problem described above, this paper proposes a higher-order cumulant-based topology inference method. Our method first utilizes the back-to-back probe packets to probe the network and collects the packet delays of the paths from a source to a set of destinations; then, it infers the network topology by analyzing the measured path delay sequences. We summarize our contributions as follows.

- In a non-internal-cooperative network, the only information that can be used to infer the topology is that regarding end-to-end measured packet delays or losses. Therefore, how to effectively discover the topology information from the end-to-end measurements determines the accuracy of topology inference. In this paper, we develop a method to estimate the cumulants of second order and above (higher-order cumulants) of each shared path from a source node to two destination nodes according to the end-to-end path delays. Higher-order cumulants contain more abundant statistical information compared with the mean or variance widely used in existing methods. Especially in actual networks, the path delay cannot be modeled by a known distribution, and the higher-order cumulant is a more appropriate description method for delay distribution, which can help us to infer a more accurate topology.
- A key issue in obtaining accurate topology from end-to-end measurement is to find an effective metric to represent the lengths of all shared paths from a source node to several sets of destination nodes. Network tomography constructs the tree topology of a network based on the inputting of the shared path length metric. To this end, this paper introduces a new metric for the problem of shared path length representation using the estimated higher-order cumulants of delays. We first combine multiple orders of cumulants using the weighted sum method in order to better use the statistical information of path delays. A binary tree topology inference algorithm is then developed based on the new metric. Finally, we alternatively optimize the weights of cumulants and the topology in order to obtain an optimal binary tree that can best fit the measured path delays.
- In an actual network, the topology from a source node to a set of destination nodes usually cannot be a binary

tree. Existing methods usually obtain the general tree topology by deleting links or restricting the number of nodes in a binary tree using a fixed threshold. However, the complexity of traffic distribution (such as traffic imbalance and nonstationary nature) in the actual network makes a fixed threshold ineffective, and such a threshold may not be suitable for all links and nodes. To solve this problem, this paper first designs a link walking algorithm to cluster the links in the binary tree and then uses a two-state automaton model with Bayes inference to identify the true or false links (the links can be deleted). We estimate the general tree topology by deleting the false links in the above optimal binary tree according to the Bayes inference procedure. Our method estimates the general tree topology without setting a fixed threshold, which is more feasible in many practical applications.

Compared with the existing methods, the proposed approach is capable of obtaining more accurate topology estimation because we use the delay cumulants of second order and above to infer the topology, and the statistical information of the path delays can be more fully utilized in our method. The remainder of the paper is organized as follows. In Section II, we review the related works. In Section III, we introduce related concepts and models. In Section IV, we present the method of higher-order cumulant inference. In Section V and Section VI, we describe the topology inference method. In Section VII, we describe the evaluations of the method under NS2 simulations. We finally conclude the paper in Section VIII.

II. RELATED WORKS

Capturing the routing topology knowledge using network tomography is of great interest to networking research. Ratnasamy and McCanne [15] were considered the pioneers in the field of network topology inference research; they proposed a bottom-up tree construction algorithm by sending multicast probe packets and estimating the loss rates of the shared paths. Based on the loss rate estimates of the shared paths, Duffield *et al.* [16], [17] proposed the binary loss tree (BLT) classification algorithm to estimate binary tree topology. A general tree topology can be obtained by removing the links with loss rates below a threshold. The authors also developed the binary delay variance tree (BDT) classification algorithm [18], which constructs the binary tree topology by utilizing the link delay variances. Recently, Bowden and Veitch [19] followed the multicast-based end-to-end measurement method and developed a topology inference method without the assumption that link losses are mutually independent; they estimated tree topology from the bottom up by finding the lost packets on the shared path based on multicast probing.

Since multicast routing is not widely deployed in today's Internet, the investigation of unicast topology inference is very important for practical applications. Castro *et al.* [20] invented a unicast probing method termed sandwich probe

and proposed the agglomerative likelihood tree (ALT) algorithm to estimate network topology using the delay differences of the probes. However, the ALT could only be applied to estimate the binary tree topology. To overcome the limitation, Castro *et al.* introduced the Monte Carlo Markov Chain algorithm (MCMC) to generate a sequence of candidate trees by birth (inserting a node) and death (deleting a node) steps. The final estimate is the tree with the maximum likelihood (MLT: maximum likelihood tree) among the candidates. Usually, the unicast-based topology inference method requires a high number of probe packets, and hence, Eriksson *et al.* [21], [22] proposed a methodology called depth-first search (DFS), which is capable of efficiently reducing the number of unicast probe packets without loss of estimation accuracy.

One of the key issues with respect to improving the accuracy of topology inference is finding a metric that can effectively describe the lengths of all shared paths from a source node to all pairs of destination nodes. Aiming at this problem, Shih and Hero [14] introduced a finite mixture model to describe the delay distribution of packet delay. The topology is estimated via a hierarchical algorithm that recursively optimizes the likelihood function and recovers the topology level by level. Ni *et al.* [23], [24] proposed a framework to merge various metrics that were obtained from end-to-end packet probing measurements and traceroute type measurements. The method may degenerate into a general network topology inference method if the traceroute-based information is not available in a noncooperative network. In this paper, we propose a new metric based on the higher-order cumulants of delays on the shared paths. The metric is capable of capturing richer statistical information of end-to-end packet probing measurements and is also beneficial to more accurately estimating topology.

III. PROBLEM STATEMENT

A. MODEL AND ASSUMPTION

In this paper, we aim at the problem of how to understand the underlying routing topology from a node to a set of other nodes in a network, where all internal nodes (e.g., routers and three-layer switcher) may refuse to return any information about topology. The information of underlying routing topology of a network is particularly useful for many applications such as failure link diagnosis [25], [26], P2P network optimization [27], and link performance parameter inference [28]. For example, the works presented in [25] and [26] focus on the problem of discovering and localizing the failure links in networks by combining the topology and other path-level performance information under the assumption that the underlying routing topology is known. Hence, understanding the underlying routing topology is a necessary preliminary work if those methods can be applied to actual networks.

This paper follows most of the existing works [13]–[19] that represent the network topology as a directed logic tree $T = (V, E)$. T comprises physical nodes V and links E between them. V is composed of one root node s

(the source node, which is the probe packet sender), a set of internal nodes W , and the remaining leaf nodes $D = \{D_1, D_2, \dots, D_M\}$ (destination node, probe packet receiver), where M is the number of leaf nodes. The source and leaf nodes can be normal computers that are proactively deployed in the network or the virtual nodes that are passively generated by network applications (such as P2P). The path from the source node s to a destination node d_i is denoted by $path(s, d_i)$. Apart from the root node, each node $w \in W \cup D$ has a unique parent $f(w)$. We use e_w to denote the link between node w and the corresponding parent node $f(w)$. For a pair of nodes $\{d_i, d_j\}$, we use $f(d_i, d_j)$ to denote the nearest parent node.

The basic principle of network topology inference is to estimate logic tree topologies based on the path-level performance parameters. The path-level performance parameters are obtained by sending a small number of normal data packets (these normal data packets are called probe packets) from a source node to a set of destination nodes. Based on this approach, the general topology of a network can be recovered by estimating multiple trees [29] and merging them together [30]–[32]. In this paper, we probe the network via back-to-back packets. For a pair of leaf nodes $\{d_i, d_j\}$, the root node sends two small packets each time, and the destinations of the two packets are d_i and d_j , respectively. Since both the packet size and the packet interval are very small, it can be determined that the two packets are sent simultaneously and with the same delay on the shared path. Based on the back-to-back probing, we make the following assumptions:

Assumption 1 (Structure Stability): The topology of the network remains unchanged during the measurement and estimation period.

Assumption 2 (Spatial Independence): The packet delays on different links are statistically independent.

Assumption 3 (Temporal Independence and Stationary): The delays of different probes on the same link are statistically independent and identically distributed.

It is noteworthy that network tomography sends normal data packets to probe the network, and these packets cannot be recognized as potentially threatening packets and prohibited by the firewalls or the IDS system [33]–[36].

B. METRIC FOR PATH LENGTH

One of the key tasks for network topology inference is to obtain the length metrics of all shared paths from the source node to all pairs of destination nodes. As shown in Fig. 1, $path(s, w_1)$ and $path(s, w_2)$ are the shared paths from the source node to pairs of destination nodes $\{d_i, d_j\}$ and $\{d_j, d_k\}$, respectively. From Fig. 1, we can observe that $path(s, w_1)$ contains more links than $path(s, w_2)$. Hence, we deem that the length of $path(s, w_1)$ is greater than that of $path(s, w_2)$. According to this information, we can determine that $f(d_i, d_j)$ is farther away from the source node than $f(d_j, d_k)$, and we can insert each leaf node into a known tree by determining the relative locations of the nearest parents with other nodes. As a result, a simple idea to recover the tree topology is to

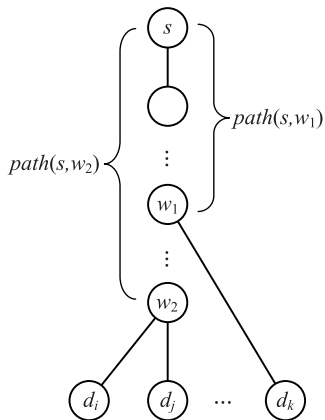


FIGURE 1. Description of the shared path length.

start from a binary tree with two leaf nodes and then insert the leaf nodes into the known binary tree one-by-one.

Unfortunately, obtaining the lengths of shared paths is a tricky task in a noncooperative network. To this end, network tomography collects path-level performance parameters (e.g., path delay, path loss rate) by sending a series of well-designed probe packets and indirectly obtains the length metric of the shared paths. For example, Coates *et al.* [20] designed a probing method called *sandwich*. A sandwich packet is composed of two small packets with a large packet between them. Two small packets may have a larger delay difference if the sandwich packet experiences a longer shared path. Shih *et al.* [14] evaluated the length of the shared path using the delay correlation coefficient of back-to-back packets. Two delay series obtained by sending a series of back-to-back packets may have a larger correlation coefficient if these back-to-back packets experience a longer shared path.

The essence of these methods assumes that the packet delay follows a given distribution, such as Gaussian or Gaussian mixture, and uses the first-order (mean value) or second-order (variance) statistics to describe the length of the shared path. However, an actual network may be affected by various factors such as bursty and nonstationary characteristics of network traffic. The distribution of packet queue delay usually cannot satisfy the given assumption, and it may lead to significant information loss if it relies only on the first- or second-order statistics to estimate the topology. The topology estimated by the existing methods may present obvious deviation from the real topology because the information of path level performance parameters cannot be fully utilized. Therefore, aiming at the problem of how to accurately describe the length of the shared path, this paper computes the higher-order cumulants of the path delays and utilizes them to describe the lengths of the shared paths. The proposed method outperforms the existing methods because the higher-order cumulants contain more statistical information, which enables describing the lengths of the shared paths more accurately.

C. HIGHER-ORDER CUMULANT

The definition of a cumulant for a random variable and the corresponding characteristics are described in [37]–[39]. Here, we review these notations and characteristics. Let X be a random variable (in this paper, X is also used to represent the queue delay of a probe packet), and the definition of the r th-order cumulant of X is given as follows:

$$K^r(X) = \left. \frac{\partial^r G_X(t)}{\partial t^r} \right|_{t=0}, \quad (1)$$

where $G_X(t) = \log(E(e^{tX}))$ is the generation cumulant function of X . In statistics, cumulants can be regarded as an alternative to the moments of a distribution. In general, a cumulant is called a higher-order cumulant if $r > 2$. However, the 2nd-order cumulant is also very useful for topology inference. Hence, in this paper, we refer to a cumulant as a higher-order cumulant if $r \geq 2$. A cumulant has two important characteristics that are critical for inferring the shared path delay cumulants from the end-to-end path delay cumulants:

- First, let X_1, X_2, \dots, X_N be N random variables and let $X = \sum_{i=1}^N X_i$. We have $K^r(X) = \sum_{i=1}^N K^r(X_i)$ if the N random variables are mutually independent.
- Second, $K^r(\alpha X) = \alpha^r K^r(X)$. This characteristic can be derived from the definition of the cumulant.

The cumulant is an important input parameter for many network management and network control operations, such as link failure diagnosis and delay sensitive application optimization. According to the definition of Edgeworth series, the arbitrary form of the distribution function can be fitted by a series with the coefficients of cumulants. However, in practice, we can only compute finite order cumulants. Thus, the order of Edgeworth series is limited, and it is an approximate expression of the distribution function. The Edgeworth series are closer to real distribution functions and contain more statistical information if more order cumulants are used. As a result, this paper estimates network topology by using higher-order cumulants to describe the length of the shared path in tree topology. Compared with the mean and variance that are used in the conventional topology inference method, the higher-order cumulants contain more statistical information and can help to accurately estimate topology.

IV. SHARED PATH LENGTH INFERENCE

To obtain the lengths of all shared paths in a tree topology, we first need to estimate higher-order cumulants from the path-level delays. Given a tree topology, we send back-to-back packets from a source node to two leaf nodes. The two paths experienced by the two packets can be abstracted as a simple binary tree with only two leaf nodes. Fig. 2 presents an example, where Fig. 2(a) displays the original paths from the source node to two leaf nodes d_i and d_j , and Fig. 2(b) shows the corresponding abstract binary tree. The shared path $path(s, f(d_i, d_j))$, the left branch path $path(f(d_i, d_j), d_i)$, and the right branch path $path(f(d_i, d_j), d_j)$ in the original tree are abstracted to three respective links. We use $link_{i,j}^{(1)}$, $link_{i,j}^{(2)}$ and $link_{i,j}^{(3)}$ to represent the three abstract links. Obviously,

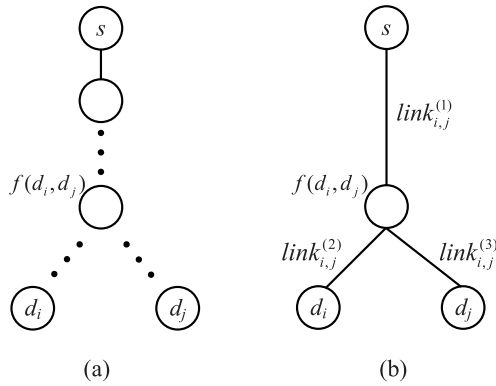


FIGURE 2. Simplified binary tree model. (a) Two paths in true topology. (b) Abstract binary tree.

the delay cumulant in the shared path $path(s, f(d_i, d_j))$ is equivalent to the delay cumulant in link $link_{i,j}^{(1)}$.

For a pair of leaf nodes d_i and d_j , we send a series of back-to-back packets from the source to the two destination nodes. Let Y_i and Y_j represent the packet delays on the paths from the source to two destinations and let $X_{i,j}^{(1)}$, $X_{i,j}^{(2)}$ and $X_{i,j}^{(3)}$ be the respective packet delays on $link_{i,j}^{(1)}$, $link_{i,j}^{(2)}$ and $link_{i,j}^{(3)}$. The two packets experience the same delay on the shared path. Hence, according to the additive packet delay, we have

$$\begin{aligned} Y_i &= X_{i,j}^{(1)} + X_{i,j}^{(2)} \\ Y_j &= X_{i,j}^{(1)} + X_{i,j}^{(3)}. \end{aligned} \quad (2)$$

According to the first property of cumulants, we can obtain the following equations

$$\begin{aligned} K^r(Y_i) &= K^r(X_{i,j}^{(1)}) + K^r(X_{i,j}^{(2)}) \\ K^r(Y_j) &= K^r(X_{i,j}^{(1)}) + K^r(X_{i,j}^{(3)}). \end{aligned} \quad (3)$$

End-to-end measurement is able to obtain the delay of a path from a source node to a destination node, but it is unable to obtain the delay of internal links $link_{i,j}^{(1)}$, $link_{i,j}^{(2)}$ and $link_{i,j}^{(3)}$. Hence, in (3), $K^r(Y_i)$ and $K^r(Y_j)$ are the known variables because they can be computed from the end-to-end path delays according to (1). $K^r(X_{i,j}^{(1)})$, $K^r(X_{i,j}^{(2)})$ and $K^r(X_{i,j}^{(3)})$ are the unknown variables, and they are obtained by solving (3). Unfortunately, for a binary tree shown in Fig. 2(b), the number of paths is smaller than that of links, and hence, the number of unknown variables in (3) is smaller than that of the known variables, and it is infeasible to directly obtain the higher-order cumulants of link delay from (3).

Aiming at the problem described above, we use a key property of back-to-back packets: the two packets of each back-to-back packet have the same delay on the shared path. Hence, we add the path delays of two packets in each back-to-back packet, and we can obtain the following equation:

$$\begin{aligned} Y_{i,j} &= Y_i + Y_j \\ &= 2X_{i,j}^{(1)} + X_{i,j}^{(2)} + X_{i,j}^{(3)}, \end{aligned} \quad (4)$$

where $Y_{i,j}$ indicates the sum of path delays. According to the second property of cumulants, we have

$$K^r(Y_{i,j}) = 2^r K^r(X_{i,j}^{(1)}) + K^r(X_{i,j}^{(2)}) + K^r(X_{i,j}^{(3)}). \quad (5)$$

In (5), $K^r(Y_{i,j})$ can be computed from the sum of path delays $Y_{i,j}$. We combine (3) and (5) and express this into matrix form equations, which is shown as the following:

$$\begin{bmatrix} K^r(Y_i) \\ K^r(Y_j) \\ K^r(Y_{i,j}) \end{bmatrix} = \begin{bmatrix} 1 & 1 & 0 \\ 1 & 0 & 1 \\ 2^r & 1 & 1 \end{bmatrix} \begin{bmatrix} K^r(X_{i,j}^{(1)}) \\ K^r(X_{i,j}^{(2)}) \\ K^r(X_{i,j}^{(3)}) \end{bmatrix}. \quad (6)$$

For (6), let

$$G = \begin{bmatrix} 1 & 1 & 0 \\ 1 & 0 & 1 \\ 2^r & 1 & 1 \end{bmatrix}. \quad (7)$$

It can be easily proven that G is a full rank matrix if $r > 1$. As a result, (6) exhibits a unique solution. We solve (6) to obtain the cumulant of three links of the tree shown in Fig. 2(b). The cumulant of the three links can be expressed as

$$\begin{aligned} K^r(X_{i,j}^{(1)}) &= \frac{K^r(Y_{i,j}) - K^r(Y_i) - K^r(Y_j)}{2^r - 2} \\ K^r(X_{i,j}^{(2)}) &= K^r(Y_i) - \frac{K^r(Y_{i,j}) - K^r(Y_i) - K^r(Y_j)}{2^r - 2} \\ K^r(X_{i,j}^{(3)}) &= K^r(Y_j) - \frac{K^r(Y_{i,j}) - K^r(Y_i) - K^r(Y_j)}{2^r - 2}. \end{aligned} \quad (8)$$

The three equations in (8) are illegal if $r = 1$, which means that the first-order cumulant of link delay cannot be determined by (8). Essentially, the first-order cumulant is the mean value of delays. It is a troublesome problem to estimate the link delay directly because it requires clock synchronization between the source node and the destination nodes, which is difficult to implement in actual networks. Consequently, this paper uses the cumulants of second order and above to describe the lengths of shared paths. For a pair of destination nodes $\{d_i, d_j\}$, these higher-order cumulants on the corresponding shared path can be represented as

$$K_{i,j} = [K^2(X_{i,j}^{(1)}), K^3(X_{i,j}^{(1)}), \dots, K^R(X_{i,j}^{(1)})]^T, \quad (9)$$

where R is the highest order of the cumulant.

In fact, any individual order of a cumulant can be used to describe the lengths of the shared paths, but it may not be sufficient to obtain an accurate topology estimation because a single order of a cumulant usually cannot achieve an appropriate representation of delay distribution. To make full use of the statistical information of the measured path delays, we combine multiple order cumulants and represent the length of the shared path from the source node s to a pair of destinations $\{d_i, d_j\}$ as the following:

$$\rho(d_i, d_j) = \beta_2 \bar{K}^2(X_{i,j}^{(1)}) + \beta_3 \bar{K}^3(X_{i,j}^{(1)}) + \dots + \beta_R \bar{K}^R(X_{i,j}^{(1)}). \quad (10)$$

(10) is the weighted sum form of cumulants, in which β_r is the coefficient for the r -th-order cumulant. β_r satisfies $\beta_r \geq 0$ and $\sum_{r=2}^R \beta_r = 1$. We use a vector $\beta = [\beta_2, \beta_3, \dots, \beta_R]$ to denote the coefficients for cumulants of all orders. In (10), $\bar{K}^r(X_{i,j}^{(1)})$ is the normalized cumulant, and it can be computed from the following equation:

$$\bar{K}^r(X_{i,j}^{(1)}) = \frac{K^r(X_{i,j}^{(1)}) - \min_{1 \leq p, q \leq M, p \neq q} K^r(X_{p,q}^{(1)})}{\max_{1 \leq p, q \leq M, p \neq q} K^r(X_{p,q}^{(1)}) - \min_{1 \leq p, q \leq M, p \neq q} K^r(X_{p,q}^{(1)})}. \quad (11)$$

V. BINARY TREE TOPOLOGY INFERENCE

In this section, we first describe the method about how to infer an optimal binary tree from the higher-order cumulants of the shared paths. Then, in section VI, we will provide the strategy about how to modify the optimization binary tree to obtain the estimation of general tree topology. According to (10), the length of a shared path can be determined if we know the coefficients of the cumulants. In this case, (10) can be used as the input for tree topology inference. This section estimates the optimal binary tree in two steps. First, we assume that all coefficients in (10) are known, and we use a binary tree construction method using the known shared path length. Second, we alternately optimize the binary tree topology and the coefficients of the shared path cumulants, which ensures that the estimated binary tree can be the most suitable topology for the measured end-to-end delays.

Algorithm 1 Binary Tree Construction (BTC) Algorithm

Input: Root node s , the set of leaf nodes $D = \{d_1, d_2, \dots, d_M\}$, and the estimated length of the shared path from the root node to any two leaf nodes $\rho = \{\rho(d_i, d_j) : d_i, d_j \in D, i \neq j\}$;

Output: The estimated binary logic tree topology $\hat{T} = \{\hat{V}, \hat{E}\}$.

Initialize: $\hat{E} = \emptyset, \hat{W} = \emptyset, \hat{D} = D$;

while $|\hat{D}| \geq 2$ **do**

Find two leaf nodes \hat{d}_i and \hat{d}_j s.t. $\{\hat{d}_i, \hat{d}_j\} = \arg \max_{\hat{d}_i, \hat{d}_j \in \hat{D}} \rho(\hat{d}_i, \hat{d}_j)$;

Create a node \hat{d} as the parent of \hat{d}_i and \hat{d}_j ;

$\hat{W} = \hat{W} \cup \hat{d}$;

if $|\hat{D}| > 2$ **then**

$E = E \cup \{\text{link}(d_i, \hat{d}), \text{link}(d_j, \hat{d})\}$;

else

$E = E \cup \{\text{link}(s, \hat{d}), \text{link}(d_i, \hat{d}), \text{link}(d_j, \hat{d})\}$;

end if

$\hat{D} = \hat{D} \setminus \{d_i, d_j\} \cup \{\hat{d}\}$;

end while

$\hat{V} = \{s\} \cup \hat{W} \cup D$;

estimated tree, and let \hat{W} denote the corresponding internal node set. If nodes w_1 and w_2 connect directly, then we use $\text{link}(w_1, w_2)$ to represent the link between them. We assume that the coefficients of cumulants in (10) are known and let $\rho = \{\rho(d_i, d_j) : d_i, d_j \in D, i \neq j\}$ be the set of shared path lengths from the source node to each pair of destination nodes. The algorithm to estimate binary tree topology for a given ρ is described in **Algorithm 1**.

In each loop of the BTC algorithm, we select two nodes with the greatest shared path length. However, not all nodes in \hat{D} exist in the real leaf node set D , which means that for $\forall \{\hat{d}_i, \hat{d}_j\} \in \hat{D}$, the corresponding shared path length may not be able to be directly determined from ρ . Hence, to find two nodes with the greatest shared path length, we summarize three cases with respect to how to compute the shared path length for any pair of leaf nodes $\{\hat{d}_i, \hat{d}_j\}$ in \hat{D} . The details of the three cases are as follows:

Case 1: If $\{\hat{d}_i, \hat{d}_j\} \in D$, then $\rho(\hat{d}_i, \hat{d}_j) \in \rho$, and it can be obtained directly.

Case 2: One of the $\{\hat{d}_i, \hat{d}_j\}$ is included in D (we assume $\hat{d}_i \in D$), but the other is not. In this case, let $L(\hat{d}_j)$ be the set of leaf nodes whose nearest common parent is \hat{d}_j ; $\rho(\hat{d}_i, \hat{d}_j)$ is then computed from the following equation:

$$\rho(\hat{d}_i, \hat{d}_j) = \frac{1}{|L(\hat{d}_j)|} \sum_{\{d_p, d_q : d_p = \hat{d}_i, d_q \in L(\hat{d}_j)\}} \rho(d_p, d_q). \quad (12)$$

Case 3: Both \hat{d}_i and \hat{d}_j are not in D . In this case, we use $L(\hat{d}_i)$ and $L(\hat{d}_j)$ to represent the set of leaf nodes whose nearest parent is \hat{d}_i and \hat{d}_j , respectively, and then, $\rho(\hat{d}_i, \hat{d}_j)$ is computed from the following equation:

$$\rho(\hat{d}_i, \hat{d}_j) = \frac{1}{|L(\hat{d}_i)||L(\hat{d}_j)|} \sum_{\{d_p, d_q : d_p \in L(\hat{d}_i), d_q \in L(\hat{d}_j)\}} \rho(d_p, d_q). \quad (13)$$

Using the BTC algorithm, we start from a binary tree with only two leaf nodes and then insert the leaf nodes one-by-one to construct a binary tree topology.

B. OPTIMIZATION OF THE COEFFICIENTS OF CUMULANTS

The BTC algorithm is capable of estimating an accurate binary tree topology if reasonable coefficients are set to (9). Hence, in this section, we describe the method of estimating the coefficients of (9) and optimize the binary tree topology at the same time.

For a given tree topology \hat{T} and a set of coefficients β , we first define a loss function as the following:

$$\sigma(\hat{T}, \beta) = \sum_{\{w: w \in \hat{W}\}} \sum_{\{d_i, d_j \in U(w)\}} [\bar{\rho}(w) - \rho(d_i, d_j)]^2, \quad (14)$$

where $U(w)$ is the set of destination node pairs. For $\forall \{d_i, d_j\} \in U(w)$, we have $f(d_i, d_j) = w$. The definition of $\bar{\rho}(w)$ is similar to (12) and (13), which represents the average path length from the source node s to internal nodes w ; it also

represents the shared path length from the source node to the set of destination nodes $L(w)$. Then, $\bar{\rho}(w)$ can be computed using the following equation:

$$\bar{\rho}(w) = \frac{1}{|U(w)|} \sum_{\{d_i, d_j\} \in U(w)} \rho(d_i, d_j). \quad (15)$$

If \hat{T} is a correct topology, then for arbitrary $w \in W$ and a pair of leaf nodes $\{d_i, d_j\} \in U(w)$, $path(s, w)$ and $path(s, f(d_i, d_j))$ represent the same path. Hence, in an ideal situation, we have $\bar{\rho}(w) - \rho(d_i, d_j) = 0$. However, due to the measurement error in practice, this may not be able to satisfy $\bar{\rho}(w) - \rho(d_i, d_j) = 0$ even if \hat{T} is a completely correct topology. However, according to the law of large numbers, we have $p(|\bar{\rho}(w) - \rho(d_i, d_j)| \rightarrow 0) = 1$ if the number of probe packets $n \rightarrow \infty$. Consequently, we can obtain the estimations of the binary tree \hat{T} together with the coefficients $\hat{\beta}$ by solving an optimization problem according to the measured path delays. The optimization problem is given as:

$$[\hat{T}, \hat{\beta}] = \arg \min \sigma(T, \beta). \quad (16)$$

However, this problem is very difficult to solve (16) directly because T is a discrete variable. To overcome this problem, we use the alternate optimization algorithm to solve (16). The alternate optimization algorithm estimates T and β by using the following steps:

Step 1 (Initialize the Coefficients and Construct an Initial Binary Tree): For an initial $\beta = \beta_0$, we can obtain an initial binary tree T_0 by calling the BTC algorithm.

Step 2 (Iteratively Update the Coefficients and the Binary Tree): For the $(k + 1)$ th iteration ($k > 0$), we fix the binary tree topology T_k obtained from the k th iteration and minimize $\sigma(T_k, \beta)$ to update β_{k+1} . Herein, minimizing $\sigma(T_k, \beta)$ is a standard quadratic programming problem that can be solved by many sophisticated methods such as the interior point convex method [40] and active set method [41]. Then, according to β_{k+1} , we can construct a new binary tree T_{k+1} by calling the BTC algorithm.

Step 3 (Determine Whether the Iteration Is Complete, and Then Output the Estimations): Repeat the above steps until the convergence conditions are satisfied. Output the estimations $\hat{T} = T_{k+1}$ and $\hat{\beta} = \beta_{k+1}$. The convergence can be determined by giving a fixed iteration number or $\|\beta_{k+1} - \beta_k\|_2^2$ ($\|\cdot\|_2$ is the L_2 norm) is smaller than a given threshold or the binary tree remains unchanged for several iterations.

VI. GENERAL TREE TOPOLOGY INFERENCE

In Section V, we estimated an optimal binary tree topology. However, in an actual network, the topology corresponding to the paths from a source node to a set of destination nodes usually cannot be a binary tree, and hence, we must further optimize the estimated binary tree to obtain a general tree topology, which is consistent with the real topology.

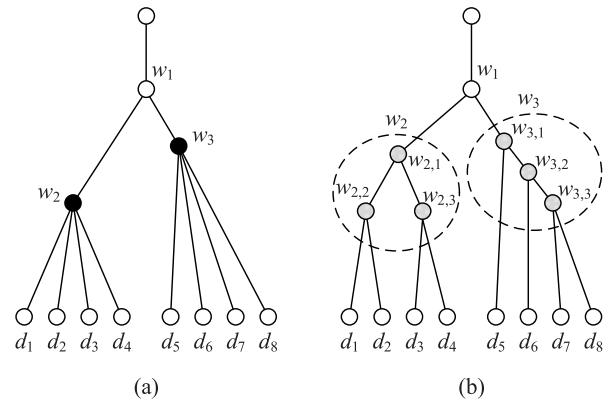


FIGURE 3. Example of true general tree topology and the corresponding estimated binary tree topology. (a) True general tree topology. (b) Estimated binary tree topology.

A. PROBLEM ANALYSIS

For an internal node w ($w \in W$), let $\rho(w)$ be the length of the path from the source node to node w . For the link between w and $f(w)$, we define the length of e_w as $\rho(e_w) = \rho(w) - \rho(f(w))$. Since both the measurement and estimation steps may introduce errors to the estimated tree, an internal node that has more than two child nodes in the real network can be split into multiple internal nodes in the estimated binary tree. Fig. 3 shows an example, where Fig. 3(a) presents the true tree topology and Fig. 3(b) presents the corresponding estimated binary tree topology. Both w_1 and w_2 in the true topology have 4 child nodes, and both of them are split into 3 nodes in the estimated binary tree. We use $\{w_{2,1}, w_{2,2}, w_{2,3}\}$ and $\{w_{3,1}, w_{3,2}, w_{3,3}\}$ to denote the split node sets of w_1 and w_2 , respectively.

To recover the real topology from the estimated binary tree, it is necessary to merge the nodes such as $\{w_{1,1}, w_{1,2}, w_{1,3}\}$ and $\{w_{2,1}, w_{2,2}, w_{2,3}\}$ in Fig. 3(b). Without loss of generality, for an internal node w_i , if w_i is split into N nodes $\{w_{i,1}, w_{i,2}, \dots, w_{i,N}\}$ in the estimated binary tree, we define $\{w_{i,1}, w_{i,2}, \dots, w_{i,N}\}$ as the split nodes of w_i , and the links between $\{w_{i,1}, w_{i,2}, \dots, w_{i,N}\}$ are the split links of w_i . For convenient description, we refer to a split link as a false link because it does not exist in real topology. Conversely, if a link really exists in real topology, then we refer to it as a true link.

To obtain a general tree topology, reference [22] compares the shared path lengths between pairs of leaf nodes during the process of tree topology estimation. If the difference of the shared path lengths between two pairs of leaf nodes is smaller than a given threshold, then the parent nodes of those leaf nodes are merged into one node. By including a penalty factor, reference [13] adds a punishment term into the likelihood function according to the node number of the tree, which obtains the general tree topology by controlling the number of nodes in the network. Reference [24] deems that the false links are usually shorter than the true links and deletes the short links whose lengths are smaller than a given threshold.

TABLE 1. Corresponding link lengths of the binary tree shown in Fig. 3(b).

Index	Link	Length	Index	Link	Length
1	e_{d_1}	5.0	9	e_{w_1}	5.0
2	e_{d_2}	5.0	10	$e_{w_{2,1}}$	4.0
3	e_{d_3}	5.0	11	$e_{w_{2,2}}$	2.0
4	e_{d_4}	5.0	12	$e_{w_{2,3}}$	2.0
5	e_{d_5}	5.0	13	$e_{w_{3,1}}$	2.0
6	e_{d_6}	5.0	14	$e_{w_{3,2}}$	1.0
7	e_{d_7}	5.0	15	$e_{w_{3,3}}$	1.0
8	e_{d_8}	5.0			

In fact, a binary tree has the largest size if given a source and a set of destinations because it has the maximum number of nodes and links. The essence of the abovementioned methods is to restrict the size of the network according to the given threshold. However, in actual networks, different nodes usually carry different amounts of traffic, with the result being that the measurement and estimation errors introduced by different nodes are also quite different, and the correct topology may not be inferred if simply given a threshold to control the size of the network.

Fig. 3 is an intuitive instance of the scenario described above; the internal nodes w_1 and w_2 in Fig. 3(a) are split into two sets of nodes $\{w_{1,1}, w_{1,2}, w_{1,3}\}$ and $\{w_{2,1}, w_{2,2}, w_{2,3}\}$ in the estimated binary tree, as shown in Fig. 3(b). The corresponding link lengths of the estimated binary tree are depicted in Tab. 1. To obtain a general tree that is consistent with the real topology, we should delete links $e_{w_{2,2}}, e_{w_{2,3}}, e_{w_{3,2}}$ and $e_{w_{3,3}}$ from the estimated topology. We can delete links $e_{w_{2,2}}$ and $e_{w_{2,3}}$ by utilizing a threshold that is greater than $\max\{\rho(e_{w_{2,2}}), \rho(e_{w_{2,3}})\}$. Links $e_{w_{3,2}}$ and $e_{w_{3,3}}$ can also be deleted under the given threshold because the lengths of $e_{w_{3,2}}$ and $e_{w_{3,3}}$ are smaller than those of $e_{w_{2,2}}$ and $e_{w_{2,3}}$. Unfortunately, we find that a true link $e_{w_{3,1}}$ is deleted because its length is also smaller than the given threshold, which results in deviation of the estimated topology from the real topology.

To obtain correct tree topology according to the optimal binary tree topology, we use a strategy similar to that described in [24], which estimates a general tree by deleting links from the binary tree. The difference is that we do not set a fixed threshold for all links split from different internal nodes in the real network. Our method is based on the following assumption:

Assumption 4 (Topology Identifiable): For an internal node w in the real topology, if the node is split into multiple nodes $\{w_{i,1}, w_{i,2}, \dots, w_{i,N}\}$, then the links between $\{w_{i,1}, w_{i,2}, \dots, w_{i,N}\}$ exhibit a significant difference from the links between $\{w_{i,1}, w_{i,2}, \dots, w_{i,N}\}$ and other nodes. In this paper, we use link length to represent this significant difference.

The reason for *Assumption 4* is that the false links are introduced by measurement and estimation errors, while the true links are computed from the packet queue delay. The generation mechanisms of the two types of links are quite

different; hence, there is also a significant difference in the link lengths. In fact, any methods that infer the general tree topology by controlling the size of the network using a given threshold are based on *Assumption 4*; otherwise, those methods may also fail even in a network within which the traffic is quite balanced in each node because we have no information that can be used to determine whether a link is true or false.

According to *Assumption 4*, the split nodes can be merged together if we delete the false links between them. Hence, to obtain the true general tree topology, we first design a link walk algorithm to cluster the false links split from the same node in the true topology. The idea of the link walk algorithm is similar to the OPTICS clustering method [42] that produces a cluster ordering instead of explicit partitioning of data points. Our link walk algorithm produces a link ordering. The false links split from the same true node may be ranked together in the link ordering. We then introduce a two-state automaton model with Bayes inference to analyze the link ordering and identify the true or false links.

B. LINK WALK ALGORITHM

For a tree $T = (V, E)$, let $\hat{T} = (\hat{V}, \hat{E})$ be the corresponding estimated binary tree, and let \hat{W} be the corresponding internal nodes of \hat{T} . We use $\rho_{\hat{E}}$ to represent the set of link lengths for all links in \hat{E} . We may not delete a link that is connected to the source node or a destination node, even if the length of the link is smaller than the threshold, and hence set the length of those links as $\max\{\rho(w) : w \in \hat{W}\}$. The purpose of the link walk algorithm is to output a link ordering by traversing the links in the estimated binary tree according to the link lengths and the locations of the links in the binary tree. The links in the output link ordering are sorted according to the order of traveling. A link should be located in the front of the ordering if it is traveled as a priority by the algorithm. Let \vec{E} be the output link ordering and let \bar{E} be a temporary buffer that is used to store the links that may be traveled by the algorithm. For a node $w \in \{s\} \cup \hat{W}$, let $C(w)$ be the set of child nodes of w . Without loss of generality, we use $C(s)$ to denote the child node of s .

The link walk algorithm is described in **Algorithm 2**. From the algorithm, we can observe that the link walk algorithm travels with priority the links that are close in distance and similar in path length; hence, the false links split from a true node are ranked closely with each other in the output link ordering. The output link ordering can be expressed as a graphical form. For the binary tree shown in Fig. 3(b), Fig. 4 shows the corresponding output of the link walk algorithm. From Fig. 4, we can observe that the algorithm visits e_{w_1} first and then visits $e_{w_{3,1}}$ and the corresponding split links $e_{w_{3,2}}$ and $e_{w_{3,3}}$. Next, it jumps to $e_{w_{2,1}}$ and the corresponding split links $e_{w_{2,2}}$ and $e_{w_{2,3}}$. Finally, all of the links connecting to the leaf nodes are visited. The split links of w_1 and w_2 are respectively ranked closely to each other in the output link ordering, and the purpose of the remaining work is then to identify these links by analyzing the output link ordering.

Algorithm 2 Link Walk Algorithm

Input: The length of all links $\rho_{\hat{E}}$ and the estimated binary tree \hat{T} .

Output: The link sequence \vec{E} .

Initialize: $\vec{E} = \{e_{C(s)}\}$, $\tilde{E} = \emptyset$, $\hat{E} = \hat{E} \setminus \{e_{C(s)}\}$;

while $|\tilde{E}| \geq 1$ **do**

 Find a link $e_w \in \tilde{E}$, s.t. $e_w = \arg \min_{e_w \in \tilde{E}} \rho(e_w)$;

$\vec{E} = \vec{E} \cup \{e_w\}$;

for v in $C(w)$ **do**

if $e_v \notin \tilde{E}$ **then**

 Insert e_v into \tilde{E} according to ascending order of link length;

end if

end for

$\tilde{E} = \tilde{E} \setminus \{e_w\}$;

end while

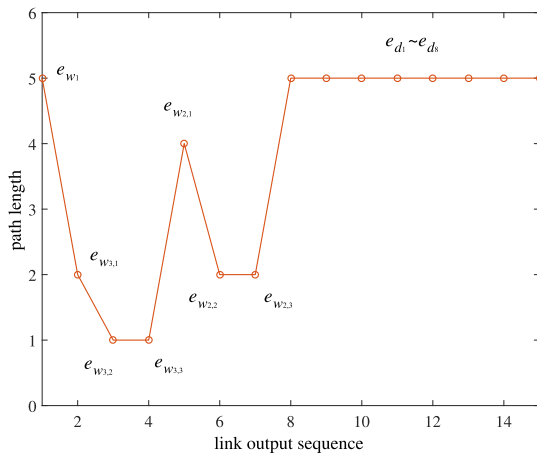


FIGURE 4. An example of output link ordering of the link walk algorithm.

C. FALSE LINK IDENTIFICATION

The link walk algorithm outputs a link ordering in which the links split from the same nodes are ranked together. Hence, if we can identify these links, then we can merge them and obtain an estimated general tree. For convenient description, let $\vec{E} = \{e_1, e_2, \dots, e_{|\hat{E}|}\}$ be the output link ordering, where $|\hat{E}|$ is the number of links in the estimated binary tree, and e_i ($1 \leq i \leq |\hat{E}|$) is the i th link in the sequence \vec{E} . Let $\vec{\rho} = \{\rho_1, \rho_2, \dots, \rho_{|\hat{E}|}\}$ represent the corresponding link length for the links in \vec{E} , in which $\rho_i = \rho(e_i)$ for $1 \leq i \leq |\hat{E}|$.

In this paper, we introduce an automaton-based bursty state detection method to identify the false links in the output link ordering of the link walk algorithm. This bursty state identification method is first used in the problem of identifying the bursty states in a stream (e.g., e-mail or news stream) [43]. The purpose of the method is to find a best state sequence for a known stream. The state sequence indicates whether the elements in the stream are bursty or not. We cluster the false links from the same true node together using the link walk algorithm. Hence, bursty changes exist in the output link

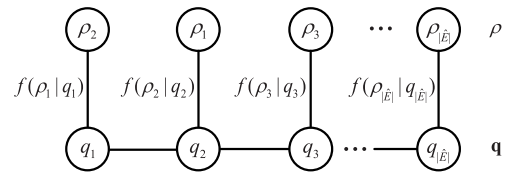


FIGURE 5. The probability model of the state transition process of the two-state automaton for false link identification.

ordering of the link walk algorithm if the algorithm walks from a true link to a false link or from a false link to a true link. The problem described in [43] is quite similar to our problem because we also need to find a state sequence for a given link ordering. The difference is that our state sequence indicates whether a link is true or false.

In this paper, we use a two-state automaton to model the true or false states in the output link ordering. Before describing the model, we first define the distribution of the link length. We use an exponential distribution to model the link length; then, we let $f_0(\rho) = \alpha_0 e^{-\alpha_0 \rho}$ be the density function of the true links and let $f_1(\rho) = \alpha_1 e^{-\alpha_1 \rho}$ be the density function of the false links, where α_0^{-1} and α_1^{-1} are the expected values of the true link and false link, respectively. In actual networks, determining α_0 and α_1 is very difficult, and hence, we follow [43] to obtain the expected value of the exponential distribution by deviating from a base value. In this paper, we use the average value of link length as the base value. According to Assumption 4, the true links and the false links are quite different with respect to length, and hence, we deem that the expected value of the true links is greater than the base value, and the expected value of the false links is smaller than the base value. As a result, we assign $\alpha_0 = (s\bar{\rho})^{-1}$ and $\alpha_1 = (s^{-1}\bar{\rho})^{-1}$, where $s > 1$ is a scaling parameter and $\bar{\rho}$ is the average value of link lengths.

Based on the above link length distribution assumption, we can construct the two-state automaton model for the false link identification problem. Let $q = \{q_1, q_2, \dots, q_{|\hat{E}|}\}$ be the binary state sequence for all links in \vec{E} . e_i ($1 \leq i \leq |\hat{E}|$) is a true link if $q_i = 1$; otherwise, e_i is a false link if $q_i = 0$. The two-state automaton changes state with probability p , and remains in the current state with probability $1 - p$. Changing the state of the automaton means that the link walk algorithm jumps from a true link to a false link (or vice versa), and remaining in the current state means that the algorithm jumps within true links or false links. Fig. 5 shows the probability model of the state transition process of the two-state automaton.

Our purpose is to determine q given the known link length sequence $\vec{\rho}$. The Bayes inference method can be used to conquer the problem, that is,

$$P(q|\vec{\rho}) = \frac{1}{Z} P(q) f(\vec{\rho}|q), \tag{17}$$

where $Z = P(\vec{\rho}) = \sum_{q_i \in \{0,1\}} P(q_i) f(\rho_i|q_i)$ is a constant. To reduce the complexity of the automaton model, we only consider the relationship of the adjacent link in \vec{E} , and hence,

the prior probability $P(q)$ in (17) can be expressed as

$$P(q) = p^k(1-p)^{|\hat{E}|-k}. \quad (18)$$

According to *Assumption 2*, the link lengths are mutually independent, and hence, $f(\vec{\rho}|q)$ in (17) can be expressed as

$$f(\rho|\vec{q}) = \prod_{i=1}^{|\hat{E}|} f(\rho_i|q_i) = \prod_{i=1}^{|\hat{E}|} f_{q_i}(\rho_i). \quad (19)$$

We can then write the detailed form of (17), that is,

$$\begin{aligned} P(q|\vec{\rho}) &= \frac{1}{Z} p^k(1-p)^{|\hat{E}|-k} \prod_{i=1}^{|\hat{E}|} f_{q_i}(\rho_i) \\ &= \frac{1}{Z} \left(\frac{p}{1-p} \right)^k (1-p)^{|\hat{E}|} \prod_{i=1}^{|\hat{E}|} f_{q_i}(\rho_i). \end{aligned} \quad (20)$$

Finding an optimal state sequence q is equivalent to solving the maximum probability of (20). For the sake of convenience, we use the log function on both sides of (20), and then, we have

$$\begin{aligned} \log P(q|\vec{\rho}) &= \log \left(\frac{1}{Z} \right) + k \log \left(\frac{p}{1-p} \right) \\ &\quad + |\hat{E}| \log(1-p) + \sum_{i=1}^{|\hat{E}|} \log f_{q_i}(\rho_i). \end{aligned} \quad (21)$$

Since both the first and the third part of (21) are constants, maximizing (21) is equivalent to minimizing the following cost function:

$$c(q|\vec{\rho}) = k \log \left(\frac{1-p}{p} \right) - \sum_{i=1}^{|\hat{E}|} \log f_{q_i}(\rho_i). \quad (22)$$

For (22), the standard forward dynamic programming algorithm can be used to obtain the optimal q . According to q , we can identify the true links and false links in the estimated binary tree. We delete the false links in the estimated binary tree and merge the two nodes on both sides of each false link, and then, a general tree topology can be obtained. This general tree topology is the final estimation of topology.

VII. EXPERIMENTS AND RESULTS

In this section, we evaluate the performance of our method in two respects. First, we use NS2 [44] simulation to assess the accuracy improvement of the proposed method by implementing a large number of experiments and comparing with two existing classical methods. Second, we conduct real network experiments using PlanetLab [45] to further evaluate the effectiveness of the proposed method.

A. SIMULATION SCENARIO SETUP

To evaluate the performance of our method in practical network environments, we first use NS2 to construct a network experimental environment. In practice, since constructing a real network experiment environment is very difficult, NS2 is regarded as an effective experimental tool for

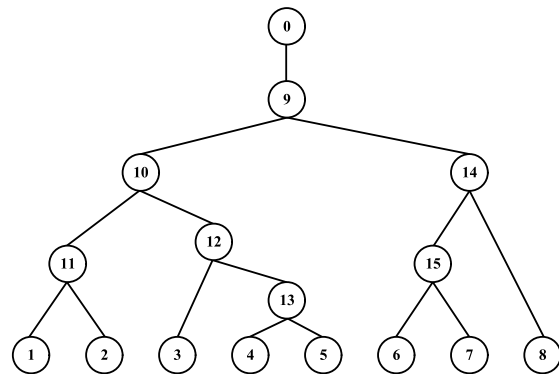


FIGURE 6. Binary tree topology.

network researchers and has been applied to many network studies [46]. Especially for network topology inferences, it is a difficult task to find a real network whose topology is known and for which the corresponding traffic scenarios can be easily set up. Consequently, in this paper, we first conduct NS2 simulations so that we can fully evaluate the performance of the proposed method by implementing a large number of experiments under different traffic scenarios.

We construct two different tree topologies, which are shown in Fig. 6 and Fig. 7, respectively. Fig. 6 shows a small binary tree topology with 8 leaf nodes, and Fig. 7 shows a large general tree topology with 30 leaf nodes. We use Fig. 6 to evaluate the performance of the higher-order cumulant inference method and the corresponding binary tree inference method, and we use Fig. 7 to evaluate the performance of the general tree topology inference method. We set the parameters of the simulation by comprehensively considering a strategy from [14] along with the computing ability of our simulation equipment. The parameter settings of the two simulation topologies are very similar. We assign the bandwidth of all links to 50 Mbps. The propagation delay of the links is set to 2 ms, and each link has a FIFO (first-in-first-out) queue with buffer size of 50 packets. We send back-to-back packets from the source node to the destination nodes. The pair of destinations for each back-to-back packet is randomly selected from the destination nodes. The back-to-back packets are sent by attaching them in constant bit rate UDP streams. The size of the back-to-back packets is set to 40 bytes. During the simulation period, we send approximately 2,000 back-to-back packets to each pair of destination nodes on average.

To simulate the traffic scenario in actual networks, we add two types of background traffic into our simulation topologies. The two types of background traffic are referred to as stationary traffic and bursty traffic. The stationary traffic comprises 150 long durations, which are Pareto distributed On-Off modeled TCP streams, and 50 UDP streams with constant bit rate. Both the burst time and the idle time of the TCP streams are 2 s, and the burst rate is 0.2 Mbps. The rate of each UDP stream is set to 0.2 Mbps. For better simulating the complex network environment, every stationary stream

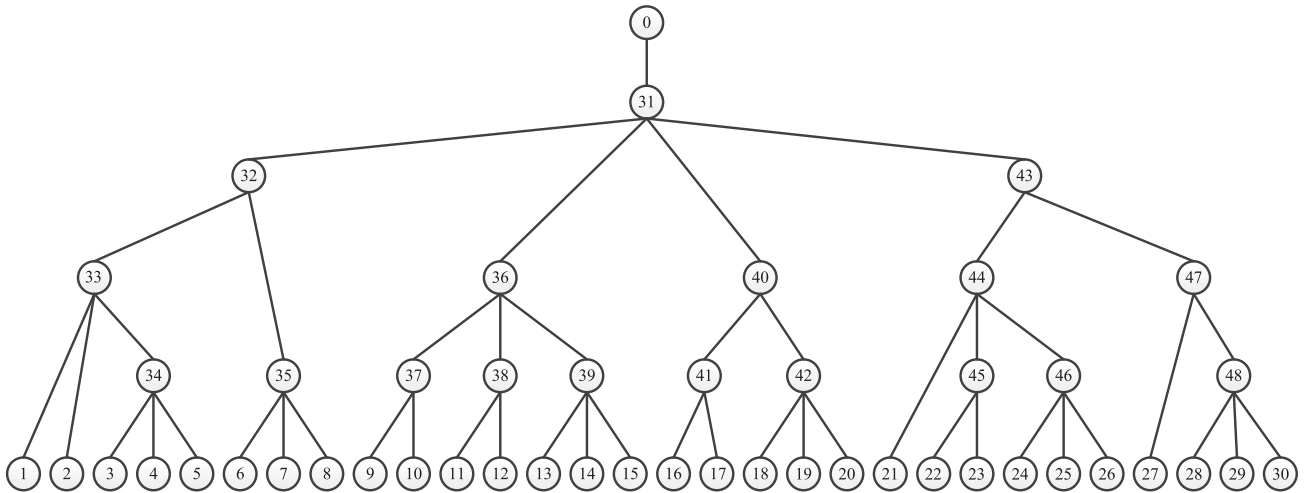


FIGURE 7. General tree topology.

is randomly sent during the first 10% of the simulation time and randomly stopped within the last 10% of the simulation time. The bursty traffic comprises short-term high-speed UDP streams. By adding different bursty traffic, we build three bursty traffic scenarios: slight burst, middle burst and heavy burst. For the slight burst scenario, the number of UDP streams added onto each link is 2, and the corresponding traffic rate is assigned to 2 Mbps. For the middle and heavy burst scenarios, the number of UDP streams is increased to 4, and the corresponding traffic rates are respectively assigned to 3 Mbps and 6 Mbps. Each bursty UDP stream starts at a random time and lasts 2 s. The packet size of all background traffic (both stationary traffic and bursty traffic) is set to 500 bytes. For each traffic burst scenario, we run 1,000 independent simulations, each for 100 s.

B. HIGHER-ORDER CUMULANT INFERENCE RESULT

Obtaining accurate higher-order cumulants is the basis of our approach. Hence, we first evaluate the performance of the higher-order cumulant estimation method. We run simulations under the three traffic burst scenarios and estimate the 2nd- to 4th-order cumulants of the shared path $p(0, 13)$. We select $p(0, 13)$ because it contains the maximum number of links among all shared paths of Fig. 6. We average the true and estimated cumulants from 1,000 simulations and depict the comparison between the real and estimated higher-order cumulants in Fig. 8. From the actual cumulant curves, we observe that the cumulant grows nearly linearly with increasing hop number, which demonstrates that the cumulant exhibits the linear additive property. Hence, it is feasible to use the higher-order cumulants to measure the shared path length.

We also observe that the estimated cumulants slightly deviate from the actual cumulants, but they still approximate the linear additivity property of actual cumulants very well. The reason is because the estimation errors are mainly introduced

by *Assumption 2*, which may not be entirely satisfied in a real network, and this factor impacts all of the estimated cumulants in the same way. To show the estimation errors more intuitively, we plot the averaged relative errors of the estimated cumulants in Fig. 9 and find that the estimated cumulants have low and stable relative errors in the three traffic burst scenarios. As a result, the estimated higher-order cumulants of shared paths can be accurately estimated from the measured path delays, which can also be used to measure the lengths of the shared paths.

C. BINARY TREE TOPOLOGY INFERENCE RESULTS

Herein, we evaluate the performance of the binary tree topology inference in the three traffic burst scenarios using the topology depicted in Fig. 6. We first calculate the 2nd- to 4th-order cumulants of the shared path for an arbitrary pair of destination nodes and then use the proposed binary tree inference method to infer the tree topology. For comparison, we also infer the binary tree using the 2nd- to 4th-order cumulants. To quantify the accuracy of the topology estimation results, we define the tree accuracy as the ratio of the simulation number for which the binary tree is correctly estimated to the total simulation number. We consider the binary tree to be correctly estimated if the estimated tree is exactly the same as the true binary tree.

We calculate tree accuracy from 1,000 simulations under the three respective traffic burst scenarios. A comparison of the estimation accuracy is shown in Fig. 10. From the figure, we observe that the estimation accuracy decreases with the growth of cumulant order when using the individual cumulant order to estimate topology. The reason is because the estimation errors of the cumulants also increase along with the growth of cumulant order. However, we also find that our optimal tree estimation method performs best compared with the methods that use the 2nd-order, 3rd-order, or 4th-order cumulant separately; this result demonstrates that combining

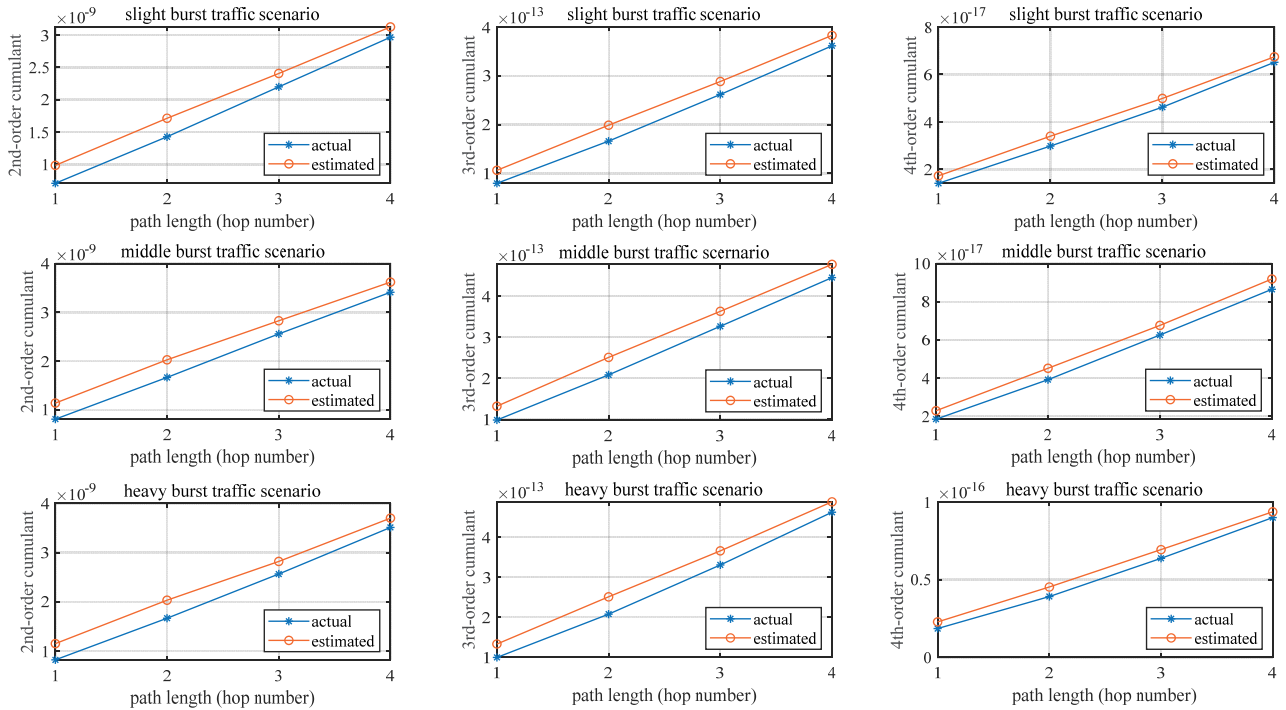


FIGURE 8. Estimated cumulants versus actual cumulants in the three traffic burst scenarios.

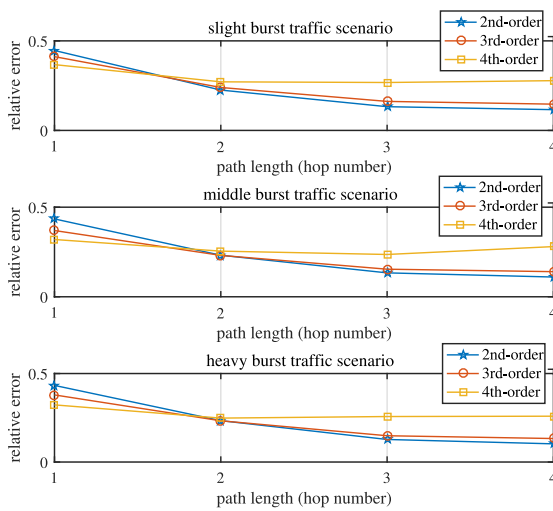


FIGURE 9. Average relative errors of the estimated cumulants in the three traffic burst scenarios.

cumulants of different order can help us to improve the accuracy of topology estimation because the delay distribution can be described more appropriately. Cumulants of different order provide different information for topology estimation. Our method fully integrates this useful information by optimizing the weights of the cumulants and finally obtains the optimized tree that best matches the measured path delay.

Fig. 10 also depicts the topology inference results under different traffic burst scenarios. It is obvious that the proposed method (labeled optimization in Fig. 10) has the highest accuracy for all three traffic burst scenarios. Especially

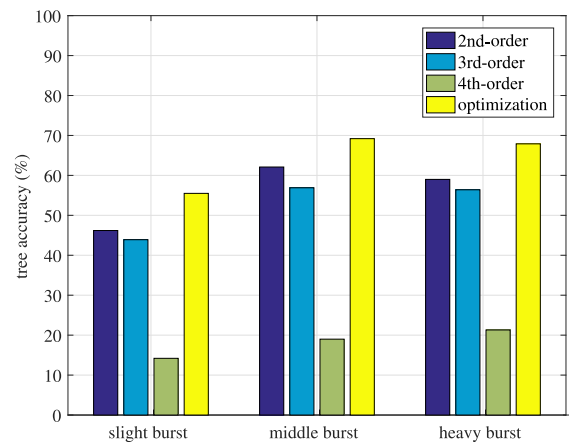


FIGURE 10. Tree accuracy of binary tree inference in the three traffic burst scenarios.

under the middle burst scenario, our method achieves the best performance. The different performances of the proposed method in different traffic burst scenarios result from the following two reasons. First, for the slight burst, the bursty property of background traffic is not obvious, and the packet delays usually contain less higher-order statistical information, which results in the decrease in the accuracy of the estimated higher-order cumulant and the introduction of errors to network topology inference. Second, in the heavy burst scenario, the correlation of two packets in a back-to-back packet may exhibit inconsistent delays when they pass the shared path, which may also introduce errors to the estimation

of shared path delay cumulants and exert negative impacts on network topology inference.

D. GENERAL TREE TOPOLOGY INFERENCE RESULTS

Next, we evaluate the performance of the general tree topology inference method using the topology depicted in Fig. 7. We also implement simulations in the given three traffic burst scenarios, and we run 1,000 simulations for each traffic burst scenario. For large topology, the tree accuracy may not be sufficient to provide an intuitional evaluation of the estimation accuracy because the general tree topology can be correctly estimated only in a few simulations. Hence, we define an F_1 score of the estimated links in the estimated general tree by mapping the internal nodes of the real tree to those of the estimated tree. Let T and \hat{T} be the real tree and the estimated tree, respectively. For an internal node w_1 in T and an internal node w_2 in \hat{T} , we define the similarity of w_1 and w_2 using Jaccard distance:

$$\text{sim}(w_1, w_2) = \frac{|L(w_1) \cap L(w_2)|}{|L(w_1) \cup L(w_2)|}. \quad (23)$$

For $\forall w \in W$ (W is the set of internal nodes of the real tree), we can find an internal node $\hat{w} \in \hat{W}$ (\hat{W} is the set of internal nodes of the estimated tree) which has maximum similarity with w . We regard \hat{w} as an estimation of w , and a link in \hat{T} is a real link if the nodes of both sides of the link are connected in the real tree. As the accuracy evaluation in supervised machine learning, we can define the *precision* as the ratio of true link number and total link number in the estimated tree, and we define *recall* as the ratio of true link number in the estimation tree and total link number in the real tree. In this paper, we use F_1 score to evaluate the accuracy of topology inference. The F_1 score represents the comprehensive assessment of estimation accuracy by merging precision and recall and is computed by the following equation:

$$F_1 = \frac{2 \times \text{precision} \times \text{recall}}{\text{precision} + \text{recall}}. \quad (24)$$

For each simulation, we first estimate a binary tree by using the optimal binary tree estimation method and then obtain link ordering by calling the link walk algorithm. Finally, we identify the false links and delete them from the optimal binary tree by using the burst detection method. Many literature publications estimate the general tree by deleting links whose lengths are smaller than a given threshold, but there is no literature that offers a strategy about how to set the threshold. To demonstrate the performance of our method, we compare the performance of BSD (we use BSD to denote the proposed burst state detection-based method) with two classical methods: RNJ (rooted neighbor-joining algorithm) [24] and OLTD (ordered logical topology discovery algorithm) [22]. Both RNJ and OLTD start from a binary tree and insert leaf nodes one-by-one by comparing the length of the shared path, and both of the algorithms may ignore links if the link length is smaller than a threshold. The difference is that RNJ inserts the leaf node with the highest

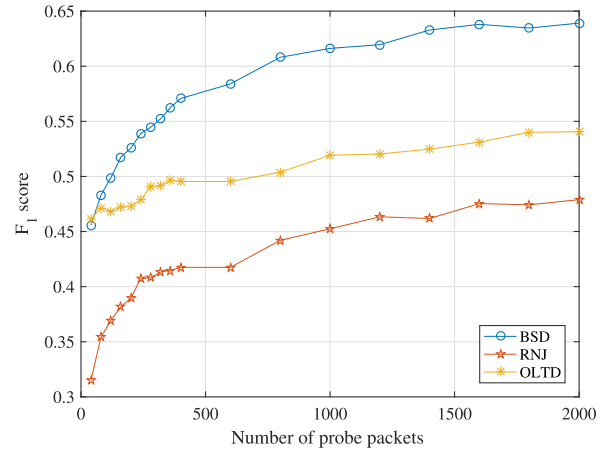


FIGURE 11. F_1 score comparison between the BSD, RNJ and OLTD (in the slight traffic burst scenario).

shared path length each time, whereas OLTD inserts the leaf nodes according to deep first sequence (DFS) ordering. In this paper, we derive the DFS ordering for the OLTD from the estimated binary tree. Note that the reason for choosing RNJ and OLTD as the baseline methods is that the three methods can infer network topology based on unicast probes, which are widely used in today's Internet. In addition, there are seldom unicast-based topology inference studies after RNJ and OLTD.

We then compare the F_1 scores of estimated links in the general topology by using BSD, RNJ and OLTD. We evaluate the performances of the three algorithms under the conditions of different numbers of probe packets and different traffic burst scenarios. For RNJ and OLTD, we estimate general tree topology using different thresholds. We compute the minimum link length ρ_{min} and the maximum link length ρ_{max} from the optimized binary tree and divide the interval between ρ_{min} and ρ_{max} into 8 bins. Each threshold can then be written as follows,

$$\Delta_i = \rho_{min} + \frac{i \cdot (\rho_{max} - \rho_{min})}{8}, \quad i = 1, 2, \dots, 8. \quad (25)$$

Fig. 11 to Fig. 13 plot the variation in F_1 scores of links with the numbers of probe packets. Note that for RNJ and OLTD, we choose the result with the highest accuracy under the conditions of setting different thresholds. In our experiment, we find that both RNJ and OLTD achieve the best accuracy if the threshold is set to Δ_1 .

From Fig. 11 to Fig. 13, we can observe that all of the methods may obtain more accurate topology estimation if more probe packets are used. The reason for this result is that sending more probe packets usually signifies obtaining more accurate path-level delays. This is the basis of inferring accurate topology because the end-to-end measurements represent the only information that can be used. More importantly, Fig. 11 to Fig. 13 show that our method significantly outperforms RNJ and OLTD. Both RNJ and OLTD are capable of returning the correct tree topology if the link length errors

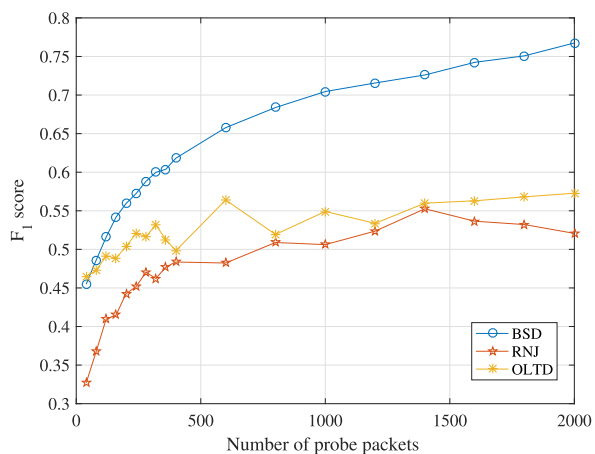


FIGURE 12. F_1 score comparison between the BSD, RNJ and OLTD (in the middle traffic burst scenario).

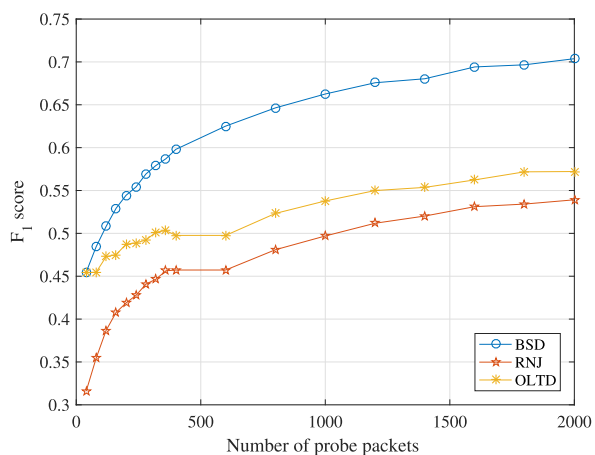


FIGURE 13. F_1 score comparison between the BSD, RNJ and OLTD (in the heavy traffic burst scenario).

are smaller than a quarter of the minimum link length in actual networks, but the link length and the corresponding error can be deeply impacted by many factors such as bursts and imbalance of traffic; hence, choosing an appropriate threshold to obtain accurate estimations is a difficult task. Different from most existing methods, our methods first use the link walk algorithm to cluster the false links that split from the same true link and then introduce a burst state detection method to recognize the false links in binary tree topology by using the differential between the false links and the true links surrounding them. The burst state detection method helps to find a global optimization false link set and results in accurate general tree estimation.

E. REAL NETWORK EXPERIMENT

To verify the effectiveness of the proposed method in a real network, we use PlanetLab [45] to deploy 9 probe nodes in North America (the detailed information of the nodes can be found in Tab. 2). We send probe packets from node0 to the remaining 8 nodes (node1 to node9). To evaluate the accuracy

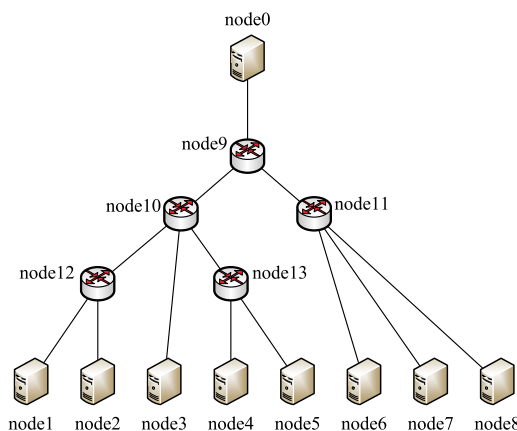


FIGURE 14. Ground-truth topology obtained by implementing traceroute in PlanetLab.

TABLE 2. Node information in the ground-truth topology.

Node name	IP address	Domain name
node0	206.117.37.7	planetlab4.postel.org
node1	128.10.18.53	planetlab2.cs.purdue.edu
node2	129.22.150.29	planetlab-4.eecs.cwru.edu
node3	170.140.119.69	node1.planetlab.mathcs.emory.edu
node4	129.32.84.160	planetlab1.temple.edu
node5	155.225.2.72	planetlab2.citadel.edu
node6	153.90.1.35	pl2.cs.montana.edu
node7	129.97.74.12	plink.cs.uwaterloo.ca
node8	128.223.8.114	planetlab4.cs.uoregon.edu
node9	137.164.27.241	-
node10	137.164.26.201	-
node11	137.164.25.74	-
node12	162.252.70.140	-
node13	162.252.70.158	-

of the estimated topology, we implement traceroute from node0 to other nodes to obtain the ground-truth topology, as shown in Fig. 14. It is noteworthy that traceroute may fail to obtain a topology because the routers usually refuse to cooperate in many other cases. We send back-to-back probe packets by randomly selecting a pair of destination nodes. The interval between two adjacent probes is 50 ms, and the entire measurement period lasted for 120 minutes. For the actual experimental scenario, we collected 100 groups of data from Dec 3-21, 2019, and estimated the topology 100 times using the data obtained during different time periods.

Fig. 15 shows the variation in average link F_1 score of the estimation topology with the probe time. We find that the estimated topology becomes closer to the ground-truth topology as the probe time increases (because more probe packets are sent). We obtain a topology with highest accuracy in the case of probing the network for approximately 1.5 hours. However, only slight improvement can be obtained if more probe packets are sent. Compared with the estimated topology and the ground-truth topology, we find that the links between node9, node10 and node11 may be error estimated

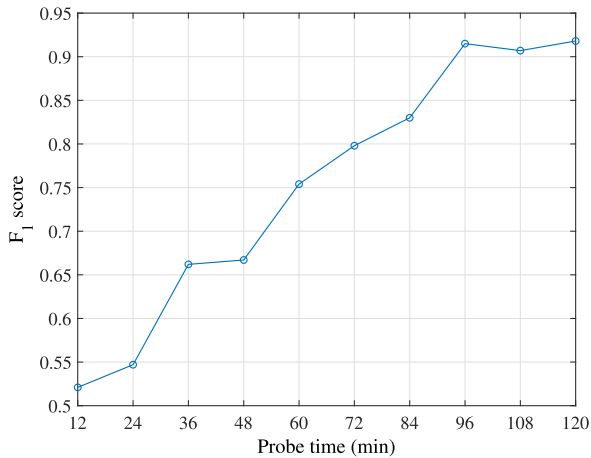


FIGURE 15. Variation in average link F_1 score of the estimation topology with the probe time.

with high probability. We analyzed the IP addresses of these nodes and found that they belong to the same network segment. Hence, to the best of our knowledge, we deem that the delay variations over these links are relatively stable and that the links may not be resolved from the measured path delay. In general, however, the estimated topology is essentially in agreement with the traceroute topology, so the proposed method performs effectively in real networks.

VIII. CONCLUSION

In this paper, we presented a method to infer network topology by using the higher-order cumulants of measured end-to-end delays. First, we estimated the internal link higher-order cumulants of delays from the measured path level delays in order to find a metric that can effectively describe the length of the shared path from a source node to a pair of destinations. Second, we used an alternate optimization-based method combined with a binary tree construction algorithm to estimate an optimization binary tree topology by merging the information of higher-order cumulants. Third, we applied a burst state detection method to recognize and delete the false links in the optimal binary tree to obtain a true general tree topology. We evaluated the performance of our method using NS2 simulation and found that the proposed method offers over 10% improvement in accuracy compared with that of state-of-the-art methods that estimate the tree topology by only relying on the parameters of given loss or delay distribution.

Future works can focus on introducing multiple-state automata to model the link lengths and developing topology measurement tools based on the proposed method.

REFERENCES

- [1] R. Motamedi, R. Rejaie, and W. Willinger, "A survey of techniques for Internet topology discovery," *IEEE Commun. Surveys Tuts.*, vol. 17, no. 2, pp. 1044–1065, 2nd Quart., 2015.
- [2] P. Marchetta, P. Merindol, B. Donnet, A. Pescapé, and J.-J. Pansiot, "Topology discovery at the router level: A new hybrid tool targeting ISP networks," *IEEE J. Sel. Areas Commun.*, vol. 29, no. 9, pp. 1776–1787, Oct. 2011.
- [3] B. Donnet, P. Raoult, T. Friedman, and M. Crovella, "Deployment of an algorithm for large-scale topology discovery," *IEEE J. Sel. Areas Commun.*, vol. 24, no. 12, pp. 2210–2220, Dec. 2006.
- [4] H. Gobjuka and Y. J. Breitbart, "Ethernet topology discovery for networks with incomplete information," *IEEE/ACM Trans. Netw.*, vol. 18, no. 4, pp. 1220–1233, Aug. 2010.
- [5] 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.
- [6] R. Castro, M. Coates, G. Liang, R. Nowak, and B. Yu, "Network tomography: Recent developments," *Stat. Sci.*, vol. 19, no. 3, pp. 499–517, Aug. 2004.
- [7] T. He, A. Gkeliass, L. Ma, K. K. Leung, A. Swami, and D. Towsley, "Robust and efficient monitor placement for network tomography in dynamic networks," *IEEE/ACM Trans. Netw.*, vol. 25, no. 3, pp. 1732–1745, Jun. 2017.
- [8] H. Li, Y. Gao, W. Dong, and C. Chen, "Preferential link tomography in dynamic networks," in *Proc. IEEE 25th Int. Conf. Netw. Protocols (ICNP)*, Oct. 2017, pp. 1–10.
- [9] C. Feng, L. Wang, K. Wu, and J. Wang, "Bound-based network tomography with additive metrics," in *Proc. IEEE Conf. Comput. Commun. (INFOCOM)*, Apr. 2019, pp. 316–324.
- [10] N. Etemadi Rad, Y. Ephraim, and B. L. Mark, "Delay network tomography using a partially observable bivariate Markov chain," *IEEE/ACM Trans. Netw.*, vol. 25, no. 1, pp. 126–138, Feb. 2017.
- [11] Y. Qiao, "Robust loss inference in the presence of noisy measurements," in *Proc. IEEE Conf. Comput. Commun. (INFOCOM)*, Apr. 2018, pp. 738–746.
- [12] F. Thouin, M. Coates, and M. Rabbat, "Large scale probabilistic available bandwidth estimation," *Comput. Netw.*, vol. 55, no. 9, pp. 2065–2078, Jun. 2011.
- [13] R. M. Castro, M. J. Coates, and R. D. Nowak, "Likelihood based hierarchical clustering," *IEEE Trans. Signal Process.*, vol. 52, no. 8, pp. 2308–2321, Aug. 2004.
- [14] M.-F. Shih and A. O. Hero, III, "Hierarchical inference of unicast network topologies based on end-to-end measurements," *IEEE Trans. Signal Process.*, vol. 55, no. 5, pp. 1708–1718, May 2007.
- [15] S. Ratnasamy and S. McCanne, "Inference of multicast routing trees and bottleneck bandwidths using end-to-end measurements," in *Proc. Conf. Comput. Commun. 18th Annu. Joint Conf. IEEE Comput. Commun. Soc.*, vol. 1, Mar. 1999, pp. 353–360.
- [16] N. G. Duffield, J. Horowitz, and F. Lo Prestis, "Adaptive multicast topology inference," in *Proc. Conf. Comput. Commun. 20th Annu. Joint Conf. IEEE Comput. Commun. Soc. (INFOCOM)*, vol. 3, Apr. 2001, pp. 1636–1645.
- [17] N. G. Duffield, J. Horowitz, F. Lo Presti, and D. Towsley, "Multicast topology inference from measured end-to-end loss," *IEEE Trans. Inf. Theory*, vol. 48, no. 1, pp. 26–45, Jan. 2002.
- [18] N. G. Duffield and F. LoPresti, "Network tomography from measured end-to-end delay covariance," *IEEE/ACM Trans. Netw.*, vol. 12, no. 6, pp. 978–992, Dec. 2004.
- [19] R. Bowden and D. Veitch, "Finding the right tree: Topology inference despite spatial dependencies," *IEEE Trans. Inf. Theory*, vol. 64, no. 6, pp. 4594–4609, Jun. 2018.
- [20] M. Coates, R. Castro, R. Nowak, M. Gadhiok, R. King, and Y. Tsang, "Maximum likelihood network topology identification from edge-based unicast measurements," *ACM SIGMETRICS Perform. Eval. Rev.*, vol. 30, no. 1, pp. 11–20, Jun. 2002.
- [21] B. Eriksson, G. Dasarathy, P. Barford, and R. Nowak, "Toward the practical use of network tomography for Internet topology discovery," in *Proc. IEEE INFOCOM*, Mar. 2010, pp. 1–9.
- [22] B. Eriksson, G. Dasarathy, P. Barford, and R. Nowak, "Efficient network tomography for Internet topology discovery," *IEEE/ACM Trans. Netw.*, vol. 20, no. 3, pp. 931–943, Jun. 2012.
- [23] J. Ni, H. Xie, S. Tatikonda, and Y. R. Yang, "Network routing topology inference from end-to-end measurements," in *Proc. IEEE 27th Conf. Comput. Commun. (INFOCOM)*, Apr. 2008, pp. 36–40.
- [24] J. Ni, H. Xie, S. Tatikonda, and Y. R. Yang, "Efficient and dynamic routing topology inference from end-to-end measurements," *IEEE/ACM Trans. Netw.*, vol. 18, no. 1, pp. 123–135, Feb. 2010.
- [25] H. Li, Y. Gao, W. Dong, and C. Chen, "Taming both predictable and unpredictable link failures for network tomography," *IEEE/ACM Trans. Netw.*, vol. 26, no. 3, pp. 1460–1473, Jun. 2018.
- [26] S. M. Srinivasan, T. Truong-Huu, and M. Gurusamy, "Machine learning-based link fault identification and localization in complex networks," *IEEE Internet Things J.*, vol. 6, no. 4, pp. 6556–6566, Aug. 2019.

- [27] J. Shen, Z. Ji, Y. Zhu, and J. Huang, "An analytical method of network service scalability," *IEEE Access*, vol. 6, pp. 5633–5640, 2018.
- [28] M. Rahali, J.-M. Sanner, and G. Rubino, "Unicast inference of additive metrics in general network topologies," in *Proc. IEEE 27th Int. Symp. Model., Anal., Simul. Comput. Telecommun. Syst. (MASCOTS)*, Oct. 2019, pp. 107–115.
- [29] M. Adler, T. Bu, R. Sitaraman, and D. Towsley, "Tree layout for internal network characterizations in multicast networks," in *Proc. 3rd Int. Workshop Netw. Group Commun. (NGC)*, London, U.K., Nov. 2001, pp. 189–204.
- [30] M. Rabbat, R. Nowak, and M. Coates, "Multiple source, multiple destination network tomography," in *Proc. IEEE INFOCOM*, Mar. 2004, pp. 1628–1639.
- [31] M. G. Rabbat, M. J. Coates, and R. D. Nowak, "Multiple-source Internet tomography," *IEEE J. Sel. Areas Commun.*, vol. 24, no. 12, pp. 2221–2234, Dec. 2006.
- [32] Y. Lin, T. He, S. Wang, K. Chan, and S. Pasteris, "Multicast-based weight inference in general network topologies," in *Proc. IEEE Int. Conf. Commun. (ICC)*, May 2019, pp. 1–6.
- [33] X. Chen, C. Li, D. Wang, S. Wen, J. Zhang, S. Nepal, Y. Xiang, and K. Ren, "Android HIV: A study of repackaging malware for evading machine-learning detection," *IEEE Trans. Inf. Forensics Security*, vol. 15, pp. 987–1001, Jul. 2019. [Online]. Available: <https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=8782574>
- [34] J. Jiang, S. Wen, S. Yu, Y. Xiang, and W. Zhou, "Identifying propagation sources in networks: State-of-the-art and comparative studies," *IEEE Commun. Surveys Tuts.*, vol. 19, no. 1, pp. 465–481, 1st Quart., 2017.
- [35] I. Stojmenovic, S. Wen, X. Huang, and H. Luan, "An overview of fog computing and its security issues," *Concurrency Comput., Pract. Exper.*, vol. 28, no. 10, pp. 2991–3005, Jul. 2016.
- [36] S. Liu, G. Lin, Q.-L. Han, S. Wen, J. Zhang, and Y. Xiang, "DeepBalance: Deep-learning and fuzzy oversampling for vulnerability detection," *IEEE Trans. Fuzzy Syst.*, early access, Dec. 9, 2019, doi: [10.1109/TFUZZ.2019.2958558](https://doi.org/10.1109/TFUZZ.2019.2958558).
- [37] J. Kenney and E. Keeping, "Cumulants and the cumulant-generating function," in *Mathematics of Statistics*. Princeton, NJ, USA: Princeton Univ. Press, 1951.
- [38] G. Antichi, A. Di Pietro, D. Ficara, S. Giordano, G. Procissi, and F. Vitucci, "End-to-end inference of link level queuing delay statistics," in *Proc. IEEE Global Telecommun. Conf. (GLOBECOM)*, Nov. 2009, pp. 1–6.
- [39] G. Fei, G. Hu, and X. Jiang, "Unicast-based inference of network link delay statistics," in *Proc. IEEE 13th Int. Conf. Commun. Technol.*, Sep. 2011, pp. 146–150.
- [40] R. H. Byrd, M. E. Hribar, and J. Nocedal, "An interior point algorithm for large-scale nonlinear programming," *SIAM J. Optim.*, vol. 9, no. 4, pp. 877–900, Jan. 1999.
- [41] R. H. Byrd, J. C. Gilbert, and J. Nocedal, "A trust region method based on interior point techniques for nonlinear programming," *Math. Program.*, vol. 89, no. 1, pp. 149–185, Nov. 2000.
- [42] M. Ankerst, M. M. Breunig, H.-P. Kriegel, and J. Sander, "OPTICS: Ordering points to identify the clustering structure," *ACM SIGMOD Rec.*, vol. 28, no. 2, pp. 49–60, Jun. 1999.
- [43] J. Kleinberg, "Bursty and hierarchical structure in streams," in *Proc. 8th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining (KDD)*, 2002, pp. 373–397.
- [44] T. Issariyakul and E. Hossain, *Introduction to Network Simulator 2 (NS2)*. Springer, 2009.
- [45] *Planetlab*. Accessed: Nov. 25, 2019. [Online]. Available: <http://www.planet-lab.org/>
- [46] A. K. Nayak, S. C. Rai, and R. Mall, *Computer Network Simulation Using NS2*. Boca Raton, FL, USA: CRC Press, 2016.



include network topology measurement and social network analysis.



JIAN YE received the B.S. degree from the School of Information and Communication Engineering, University of Electronic Science and Technology of China (UESTC), Sichuan, China, in 2017, where he is currently pursuing the M.S. degree with the School of Information and Communication Engineering. His current research interest is network topology measurement.



SHENG WEN (Member, IEEE) received the Ph.D. degree from Deakin University, Melbourne, in October 2014. He has been a Research Fellow and then as a Lecturer in computer science with the School of Information Technology, Deakin University, since 2015. He is currently working full-time as a Senior Lecturer with the Swinburne University of Technology.



GUANGMIN HU received the B.S. degree from the Department of Computer Science, Nanjing University, China, in 1986, and the M.S. and Ph.D. degrees from the Chengdu University of Technology, Sichuan, China, in 1992 and 2000, respectively. From 2000 to 2003, he was a Postdoctoral Researcher with the School of Communication and Information Engineering, University of Electronic Science and Technology of China (UESTC). From 2002 to 2003, he was a Visiting Scholar with Hong Kong Polytechnic University. He is currently a Professor with the School of Information and Communication Engineering, UESTC. His current research interests include computer networks and signal processing.

...