Queuing Network Topology Inference Using Passive Measurements

Yilei Lin, Ting He, and Guodong Pang
Pennsylvania State University, University Park, PA 16802, USA. Email: \{yjl5282,tzh58,gup3\}@psu.edu

Abstract—In this work, we revisit a classic problem of inferring a tree topology from end-to-end measurements originated by a single source, with two critical differences: (i) instead of relying on measurements with specific correlation across paths that often require active probing, we do not rely on any correlation and can thus utilize passive measurements; (ii) instead of inferring a logical topology that ignores certain nodes, we want to recover the physical topology. Our key idea is to utilize the detailed queuing dynamics inside the network to estimate the number of queues and a certain parameter (residual capacity) of each queue on each measurement path, and then use the estimated parameters as fingerprints to detect shared queues and infer the topology. To this end, we develop a Laplace-transform-based estimator to extract the parameters of a tandem of queues from end-to-end delays, and efficient algorithms to identify the parameters associated with the same queue and infer the topology accordingly. The inferred topology is guaranteed to converge to the ground truth, up to a permutation of queues traversed by the same paths, as the number of measurements increases. Our evaluations validate the proposed solutions against benchmarks and identify potential directions for further improvements.

Index Terms—Network topology inference, passive measurement, queuing network, phase-type distribution.

I. INTRODUCTION

Understanding the internal structure of a network, i.e., the network topology, is critical for a variety of tasks such as routing, content distribution, service placement, load balancing, and overlay construction. While network topology is traditionally maintained by the network administrator (e.g., via the help of local agents running Simple Network Management Protocol or OpenFlow) and discoverable by traceroute, obtaining this information for multi-domain networks such as the Internet is much more challenging due to the lack of internal support.

In these cases, network tomography provides a promising approach that infers the network topology from end-to-end measurements. Since introduction in the 1990s [1], a number of algorithms have been developed based on this idea, which use multicast measurements [2]–[4], their unicasting-based approximations [5], [6], or network coding [7] to infer the routing tree rooted at each probing source, and further stitch the trees for multiple sources to form a general topology [8]. A common foundation of these works is a probing scheme that generates specifically correlated measurements across different paths, so that the correlation (caused by shared links/nodes) can be used to estimate the “lengths” of the shared portions of these paths. These shared path lengths can then be used to reveal the branching/joining points between different paths and thus the network topology.

Despite the extensive studies, existing topology inference algorithms have the limitations that (i) they mostly rely on active probes due to the specific type of correlation required by each algorithm, which increases the network load, and (ii) they can only infer a logical topology that ignores the degree-2 nodes between branching/joining points. In this work, we tackle these limitations by developing a topology inference algorithm that can utilize passive measurements with arbitrary (or no) correlation across paths, and recover the physical topology with possible degree-2 nodes. This is achieved by a fundamentally different approach that utilizes the detailed queuing dynamics inside the network. As a first step, we focus on inferring the tree topology rooted at a single source, and leave the extension to multi-source topologies to future work.

A. Related Work

Network topology inference: Our work belongs to a branch of network tomography aiming at inferring routing topologies using end-to-end measurements. The technique was originated based on the observation that correlated losses observed at multicast receivers can be used to infer the multicast tree [1], and was then extended to utilize a variety of multicast measurements, including losses [2], delays [3] and a combination of both [4]. As multicast is not widely supported, solutions based on unicast were proposed [5], [6]. These solutions, however, were based on specially-designed probing schemes such as stripes of back-to-back unicast packets [5] or “sandwiches” of small and large packets [6], both inducing correlated measurements at different receivers that can reveal performance metrics on the shared portions of end-to-end paths. While the above works aimed at inferring a tree topology rooted at a single source, later works (e.g., [8] and references therein) addressed more general topologies by merging trees rooted at multiple sources. However, these works are still based on multicast or its approximations, which requires active probing. Another line of works relies on network coding (e.g., [9] and references therein). These solutions rely on internal nodes that perform network coding, thus not applicable in current packet-switched networks.

In contrast, we take a fundamentally different approach of fingerprinting queues at internal nodes based on (possibly) uncorrelated end-to-end performance measurements, thus able to leverage passive measurements from existing data packets.

Queuing parameter inference: Our approach is based on the inference of queuing parameters from end-to-end measure-
ments. To this end, a variety of parameters have been tackled in the context of communication networks. For example, it was shown in [10] that the difference between the delays measured when the buffer is full or empty can be used to estimate the buffer size at the bottleneck link. In [11], packet arrival times and flow identifiers were used to detect bottleneck links shared between flows and estimate their bandwidths. In [12], periodic probes were used to detect the “dominant congested link” on a path and estimate the maximum queuing delay at this link. These works only focused on the bottleneck links, and while useful for performance diagnosis, did not provide sufficient information for topology inference.

In the context of generic queuing systems, the inference of queuing parameters has been posed as inverse problems, with several inversion techniques developed to infer input and service time characteristics from delay/loss measurements for a single queue [13]. However, when the system becomes more complex (e.g., a tandem of queues), inversion techniques became unstable [13], and solutions fell back to standard algorithms based on maximum likelihood estimation [13, [14]. We refer to [15] for a comprehensive bibliography in this space. To our knowledge, all the existing works assumed the queuing network topology to be known.

B. Summary of Contributions

We aim at inferring the topology of a queuing network modeling the connections from a given source to a given set of destinations, based on possibly uncorrelated measurements of the end-to-end delays. Our contributions are:

1) We propose a novel approach for topology inference that can utilize passive measurements and potentially recover the physical topology (of the queuing network) by exploiting the detailed queuing dynamics inside the network.

2) We develop a Laplace-transform-based estimator that can estimate the length and the residual capacities of a tandem of queues from end-to-end delays, which outperforms the maximum likelihood estimator (MLE) at finite sample sizes and is asymptotically consistent.

3) Based on the estimated queue parameters (i.e., residual capacities), we develop computationally efficient algorithms to identify the parameters associated with the same queue, and then construct a tree topology accordingly. The constructed topology is guaranteed to be identical to the ground truth, up to a permutation of queues traversed by the same set of measurement paths, if the parameter estimation is sufficiently accurate.

4) Our evaluations based on a real topology show that: (i) the proposed estimator outperforms MLE in both efficiency and accuracy in a wide range of settings, and (ii) for small topologies, the proposed topology inference algorithm can even outperform state-of-the-art solutions based on active probing due to its capability of discovering degree-2 nodes. Meanwhile, we find it difficult to accurately infer large topologies from uncorrelated unicast measurements, and identify potential directions for further improvements.

Roadmap. Section II formulates our problem, Section III addresses parameter estimation for a tandem of queues, Section IV addresses topology inference based on the estimated parameters, Section V evaluates the performance of the proposed algorithms, and Section VI concludes the paper.

II. PROBLEM FORMULATION

A. Network Model

Given the routing tree $T'$ connecting a given source $s$ to a given set of destinations $\{d'_1, \ldots, d'_N\}$, we model this topology by a queuing network as shown in Fig. 1, where each queue $q_i$ models the outgoing interface of a link $l_i \in T'$. This model is motivated by the fact that queuing in packet-switched networks typically occurs at the outgoing interfaces. It is easy to see that the resulting queuing network has the topology of a rooted tree, denoted by $T$, where each vertex represents a queue that corresponds to a link in the original routing topology. One can easily obtain the original topology from $T$. Let $d'_i \in T$ ($i = 1, \ldots, N$) denote the leaf modeling the access link for destination $d'_i$, and $p_i$ denote the path from the root of $T$ to $d'_i$.

We model each $q_i$ as an $M/M/1$ queue, where the sojourn time models the delay imposed by link $l_i$ on a packet traversing $l_i$. Specifically, let $\lambda_i$ denote the unknown load on link $l_i$ and $\mu_i$ denote the unknown capacity of this link, both measured in packets per second. We assume that $\mu_i > \lambda_i$, which guarantees queue stability. Then it is well-known [16] that the sojourn time $T_i$ of $q_i$ in the steady state is exponentially distributed with parameter $\delta_i := \mu_i - \lambda_i$ that represents the residual capacity, i.e., the PDF of $T_i$ is $w(t_i) = \delta_i e^{-\delta_i t_i}$ ($t_i > 0$). Moreover, we assume that the sojourn times of a given packet at different queues are independent of each other in the steady state, and hence the end-to-end delay on path $p_j$ follows the hypoexponential distribution with parameters $(\delta_i)_{q_i \in p_j}$, where “$q_i \in p_j$” means for each queue $q_i$ on path $p_j$.

Remark: Our assumptions are automatically satisfied if $T$ is a Jackson network with $M/M/1$ queues, which is a commonly-used model in queuing theory. We refer to [13, Section 6.2] for a detailed discussion on the realism of this model. We further note that we do not assume the same packet to have independent service times at different queues (these times will be correlated); what we assume is that different queues receive largely disjoint cross-traffic, and thus a measurement packet will incur largely independent waiting times at different queues and hence largely independent sojourn times (assumed to be dominated by the waiting times).
B. Objective

Given i.i.d. measurements \( \mathbf{x}_j := (x_{j,h})_{h=1}^n \) of the end-to-end delay on each path \( p_j (x_{j,h}; \delta \) the \( h \)-th measurement on path \( p_j \)), we want to infer the queuing network topology \( \mathcal{T} \). These measurements can be obtained from active probes, passive monitoring of data packets, or a combination of both. In contrast to the previous works [5], [6], we do not rely on specific correlation of measurements across paths, and thus can utilize passive measurements that can be arbitrarily correlated (possibly uncorrelated) across paths. The temporal independence assumption for measurements on the same path can be justified by ensuring sufficient spacing between measurements as in [6]. For the problem to be well-defined, we assume that the link loads remain fixed during measurement, which can be achieved by collecting the measurements during off-peak hours so that there is minimal fluctuation in traffic.

C. Motivating Example

We illustrate why the problem is solvable by a simple example. As illustrated in Fig. 2 (a), \( \mathcal{T} \) models the connection from a source to three destinations via three paths \( p_1 \), \( p_2 \), and \( p_3 \). The end-to-end delay on \( p_j \) (\( j = 1, \ldots, 3 \)) follows a hypoexponential distribution with the parameters listed in Fig. 2 (b), which can be accurately estimated after collecting sufficiently many measurements. Under the assumption that different queues have different parameters, we can infer that there is a queue shared by all three paths with parameter \( \delta_1 \) and another queue shared by paths \( p_2 \) and \( p_3 \) with parameter \( \delta_3 \). Due to the tree structure, we can then infer the topology as in Fig. 2 (c), where the only uncertainty is in the order of the queues with parameters \( \delta_2 \) and \( \delta_4 \) on \( p_1 \). Since this order does not affect the end-to-end delay distribution on any path, Fig. 2 (c) is the most accurate estimate that can be obtained.

Following the idea in this example, we will develop our solution in two steps: (1) estimating the \( \delta \)-parameters (i.e., residual capacities) for a tandem of queues from their end-to-end delays, and then (2) inferring the queuing network topology by merging the queues with sufficiently similar \( \delta \)-parameters. 

III. PARAMETER ESTIMATION FOR TANDEM QUEUES

We first focus on estimating the \( \delta \)-parameters for a tandem of queues modeling a given path. For simplicity, we will omit the path index and simply denote the delay measurements on this path by \( \mathbf{x} := (x_{h})_{h=1}^n \) and the queue parameters by \( \delta := (\delta_1, \ldots, \delta_K) \), where \( K \) is an upper bound on the number of queues per path (i.e., the maximum hop count between the source and the destinations in the routing topology). As the actual number of queues on a given path is unknown, we use this upper bound to define the number of unknown parameters. As the order of queues does not affect the end-to-end delay distribution and hence cannot be identified from the delay measurements from a single path (later in Section IV we will show how to identify their order up to segments in the tree), we assume \( \delta_1 \leq \cdots \leq \delta_K \) when comparing the estimated and the true parameters. If the actual number of queues is \( K^* \) \((K^* < K)\), then the last \( K-K^* \) queues are virtual queues with no additional delay, i.e., \( \delta_{K^*+1} = \cdots = \delta_K = \infty \). Our estimation algorithm (Algorithm 1) will also estimate \( K^* \).

A. Maximum Likelihood Estimation (MLE)

Assuming that \( K = K^* \), [13] proposed to apply the MLE to this problem. More generally, MLE is considered the state-of-the-art estimator for phase-type distributions [17], which includes the hypoexponential distribution as a special case. Specifically, assuming that \( \delta_i \neq \delta_j \) for any \( i \neq j \), we can express the PDF of the end-to-end delay as:

\[
    g(x; \delta) = \sum_{i=1}^K \delta_i e^{-x\delta_i} \left( \prod_{j=1, j \neq i}^K \frac{\delta_j}{\delta_j - \delta_i} \right) ,
\]

and the MLE aims at finding the value of \( \delta \) that maximizes the log-likelihood \( \sum_{n=1}^n \log g(x_n; \delta) \). If solved exactly, the MLE has a desirable property that it is asymptotically efficient under regularity conditions\(^1\), i.e., as the number of measurements increases, it converges to the true parameter at a rate approximating the Cramér-Rao bound [18].

However, as the log-likelihood function is non-concave, computing the MLE is challenging. To address this challenge, various algorithms have been proposed to compute an approximation to MLE [17]. In particular, the Expectation-Maximization (EM) algorithm is guaranteed to converge to a local maximum and was adopted to solve this problem in [13]. However, we find EM to be extremely slow in our case due to the calculation of numerical integration, which combined with its known slow convergence [14] makes it impractical for our problem.

B. Estimation based on Laplace Transform

Motivated by the need to improve the estimation speed and accuracy, we exploit estimators based on the Laplace transform. Defined as \( E[e^{-sX}] \) for a random variable \( X \), the Laplace transform uniquely determines the distribution of \( X \) (except on a set of Lebesgue measure zero), and can be numerically inverted to compute the CDF/PDF of the distribution [19]. While the transform has been used to estimate PDF/CDF from data [20], [21], to our knowledge, we are the first to apply it to parameter estimation.

The Laplace transform appears promising for our problem due to its property that if \( X \) is a summation of independent random variables \( X_1, \ldots, X_K \), then \( E[e^{-sX}] = \prod_{i=1}^K E[e^{-sX_i}] \), thus providing a simple target function for

\(^1\)These conditions are that (i) the log-likelihood function is twice differentiable, and (ii) the Fisher Information Matrix is non-zero, both satisfied in our case.
fitting. Specifically, since the Laplace transform of an exponential random variable with parameter \( \delta_i \) is \( \delta_i/(\delta_i + s) \) \((s > -\delta_i)\), the Laplace transform of the end-to-end delay is

\[
L(s; \delta) := \prod_{i=1}^{K} \frac{\delta_i}{\delta_i + s}, \quad s > -\min_{i=1,\ldots,K} \delta_i.
\] (2)

Our idea is to estimate the Laplace transform at a predetermined set of values for \( s \), and find the parameter \( \delta \) yielding the best fit.

**Estimator:** By definition, the empirical Laplace transform

\[
\hat{L}(s; \mathbf{x}) := \frac{1}{n} \sum_{h=1}^{n} e^{-s x_h}
\] (3)

gives an unbiased estimate of \( L(s; \delta) \). Given the empirical Laplace transform, we propose to estimate \( \delta \) by solving

\[
\min_{\delta \in S} \sum_{s \in S} |L(s; \delta) - \hat{L}(s; \mathbf{x})|
\] (4a)

s.t. \( 0 < \delta_1 \leq \cdots \leq \delta_K \),

i.e., fitting the empirical Laplace transform at \( s \in S \) by minimizing the absolute error, where \( S \) is a design parameter.

We note that there are other ways to use the Laplace transform for estimating \( \delta \). For example, we can rewrite the Laplace transform (2) as

\[
L(s; \delta) = \frac{a_0}{a_0 + a_1 s + \cdots + a_{K-1} s^{K-1} + s^K},
\] (5)

where the coefficients are related to \( \delta \) by

\[
a_j := \sum_{A \subseteq \{1,\ldots,K\} \setminus \{i\}} \prod_{j \in A} \delta_j, \quad j = 0, \ldots, K-1.
\] (6)

Then we can estimate the coefficients \( a_0, \ldots, a_{K-1} \) from the empirical moments or the empirical Laplace transform as in [20], and solve the system of equations (6) for \( \delta \). We can also use other error measures in (4a) (e.g., squared error). However, we find that the proposed estimator achieves the best performance (see Fig. 7–8).

**Performance:** The proposed estimator is **consistent** under sufficiently large \(|S|\) in the following sense.

**Theorem III.1.** As \( n \to \infty \), (4) has a unique optimal solution that equals the ground truth if \(|S| > K\).

**Proof.** As \( n \to \infty \), \( \hat{L}(s; \mathbf{x}) \) will converge almost surely to \( L(s; \hat{\delta}) \), where \( \hat{\delta} \) denotes the true parameter. Suppose that \( \exists \delta \neq \delta^* \) that is optimal for (4) at convergence. It must satisfy

\[
\sum_{s \in S} |L(s; \delta) - L(s; \delta^*)| \leq 0,
\] (7)

and thus

\[
\prod_{i=1}^{K} \delta_i - \prod_{i=1}^{K} \delta^*_i = 0, \quad \forall s \in S.
\] (8)

The left-hand side of (8) can be converted to an order-\( K \) polynomial of \( s \), which has at most \( K \) distinct solutions. This contradicts with the assumption of \(|S| > K\), thus completing the proof.

At finite sample sizes, we find it easier to find a good solution based on the proposed objective in (4a) than the MLE objective (see Fig. 7–8). Intuitively, this is because our objective function has a larger gradient norm around the optimal value, as illustrated in Fig. 3, and hence optimization algorithms are less likely to stop at suboptimal values. Fig. 3 (b) further justifies our choice of the absolute error in (4a) as opposed to the commonly-used squared error. Similar observations have been obtained under other settings.

**Design of \( S \):** Besides the condition in Theorem III.1, the values in \( S \) also affect the accuracy of the proposed estimator at finite sample sizes. Intuitively, \( S \) should contain a diverse range of values to provide a good description of the Laplace transform. It has been suggested in [20] that these values should be evenly distributed. We also find that increasing the density of points in \( S \) helps (see Fig. 6 (a)), at the cost of increased complexity.

**Algorithm:** Although in theory virtual queues with \( \delta_i = \infty \) can be used to accommodate overestimation of the number of queues, in practice such overestimation tends to increase the error in estimating \( \delta \) as optimization algorithms used to solve (4) will always give finite values. Therefore, we propose to test the fitting error under various numbers of queues up to \( K \) and select the estimate with the minimum fitting error, as shown in Algorithm 1. Lines 6–7 are used to find a good initial value of \( \delta \) by iteratively optimizing one \( \delta_i \) at a time, while fixing the other \( \delta_j \) for \( j \neq i \). We find that optimization based on this initialization outperforms optimization from an arbitrary initial guess (see Fig. 6 (b)).

**IV. QUEUEING NETWORK TOPOLOGY INFERENCE**

Based on the inferred \( \delta \)-parameters from each tandem of queues for which the end-to-end delays can be measured, i.e., from each path \( p_i \) \((i = 1, \ldots, N)\), we are now ready to infer the queuing network topology \( T \). The key observation is that if each queue has a distinct \( \delta \)-parameter, which is likely to be satisfied in production networks due to the unique mix of cross-traffic on each link, then we can use this parameter as a “fingerprint” of the queue to detect the queues shared by different paths.

Specifically, given the inferred parameters \( \hat{\delta}_i := (\hat{\delta}_{ij})_{j=1}^{k_i} \) for path \( p_i \) \((i = 1, \ldots, N)\), we view \( p_i \) as a tandem of \( k_i \) queues, denoted by \( q_i := (q_{ij})_{j=1}^{k_i} \). We will develop efficient algorithms to merge these \( N \) tandems of queues into a tree-shaped queuing network in two steps: (1) inferring which input queues represent the same queue in the underlying queuing network, and (2) inferring the queuing network topology accordingly. We now tackle these two steps separately.
We will also refer to the indices of the paths traversing a queue $q$ in $\mathcal{T}$ as the category of $q$. Clearly, if the queues in $s$ represent the same queue $q$ in $\mathcal{T}$, then $s$ and $q$ have the same category. For example, the inference results in Fig. 2 (b) suggest that there is a queue with parameter $\delta_1$ that has category $\{1, 2, 3\}$. The problem is that if we associate a set of input queues of category $A \subseteq \{1, \ldots, N\}$ with the same queue $q$ in $\mathcal{T}$, then $q$ will reside on the tree branch containing “destinations” $\{d_i\}_{i \in A}$ (recall that $d_i$ is the queue modeling the access link of destination $d_i$), and thus there cannot be another queue $q'$ of category $A'$ for any $A'$ satisfying $A' \cap A \neq \emptyset, A' \not\subseteq A$, and $A \not\subseteq A'$, as illustrated in Fig. 4. We say that two candidate sets $s$ and $s'$ conflict with each other if $c(s)$ and $c(s')$ satisfy (11).

To avoid such conflict, our idea is to iteratively select candidate sets via a greedy procedure, where each iteration selects the candidate set not conflicting with the existing selections that has the minimum error defined as in (9). However, a straightforward implementation of this idea will incur an exponential complexity as there are $O(K^N)$ candidate sets.

Algorithm 2 avoids the exponential complexity by only searching among the candidate sets that may achieve the minimum error. Specifically, let $Q$ denote the currently selected candidate sets and $\hat{Q}$ the candidate sets that will be searched in the next iteration. For each $s \in \hat{Q}$, let $F_s \subseteq \{0, 1\}$ indicate whether the candidate set $s$ is feasible, i.e., not conflicting with any $s' \in Q$. Starting by selecting all the singletons $\{\{d_i\}\}$ into $Q$ as $D_{\{d_i\}} = 0$ (lines 2–3), we see that: (i) for the first iteration, the minimum error among the candidate sets outside $Q$ must be achieved at a set of two queues; (ii) for each of the subsequent iterations, the minimum error among the feasible candidate sets outside $Q$ must be achieved at the union of two sets in $Q$. This allows us to initialize (lines 4–8) and update (lines 13–29) $Q$ and its corresponding properties only for the candidate sets that may be selected in the next iteration. The algorithm continues until all the candidate sets not conflicting with the already-selected sets have been considered (line 9).

2) Complexity: Recall that $K$ is the maximum number of queues per path. By design, $|Q|$ starts at $O(KN)$ and reduces by one in each iteration (as two sets in $Q$ will be replaced by their union). Moreover, $\hat{Q}$ contains the union of each pair of sets in $Q$, and hence $|\hat{Q}| = O(K^2N^2)$. This implies a space complexity of $O(K^2N^3)$, dominated by the space for storing $\hat{Q}$, as the cardinality of each set in $\hat{Q}$ is at most $N$. For time complexity, the initialization (lines 2–8) takes $O(K^2N^2)$, each while loop (lines 10–29) takes $O(K^3N^4)$, dominated by the update of ($F_s$) $s \in \hat{Q}$ in lines 14–19 (as the conflict between two sets can be checked in $O(N)$), and the while loop is repeated $O(KN)$ times (each reducing $|Q|$ by one). The total time complexity is thus $O(K^4N^5)$. Note that this is only the worst-case complexity when all the considered sets are candidate sets; in practice, the complexity will be lower with a smaller $\Delta$.

3) Correctness: We show that the inference by Algorithm 2 will be accurate if the input parameters are sufficiently accurate. For each queue $e \in \mathcal{T}$, let $\delta^*_e$ denote its true parameter.


\begin{algorithm}[H]
\caption{Inference of Associated Queues}
\begin{algorithmic}
\STATE \textbf{input :} inferred parameters $\delta_1, \ldots, \delta_N$ for all the paths $(\delta_i = (\delta_{ij}))_{j=1}^{n_i}$; \textbf{threshold $\Delta$}
\STATE \textbf{output:} Collection $Q$ of sets of queues, where each $s \in Q$ is a set of input queues inferred to be associated with the same queue in $T$
\STATE $Q \leftarrow \emptyset; \hat{Q} \leftarrow \emptyset$
\FOR{$i = 1, \ldots, N$}
\STATE $Q \leftarrow Q \cup \{\{q_{i1}, \ldots, q_{ik}\}\}$
\ENDFOR
\FOR{$\{q_{i1}, j\}, \{q_{i2}, j\} \in Q$ with $i_1 \neq i_2$}
\IF{$D_{i1,j} - D_{i2,j} \leq \Delta$}
\STATE $Q \leftarrow Q \cup \{\{q_{i1}, j\}, \{q_{i2}, j\}\}$
\STATE $D(q_{i1}, j) \leftarrow D_{i1,j}$
\STATE $F(q_{i1}, j) \leftarrow 1$
\ENDIF
\ENDFOR
\WHILE{$\exists s \in Q$ with $F_s = 1$}
\STATE $s^* \leftarrow \arg\min_{s \in Q} D_s$
\STATE $Q \leftarrow Q \setminus \{s^*\}$
\STATE $Q \leftarrow Q \setminus \{s \in Q : s \subset s^*\}$
\STATE $Q \leftarrow Q \setminus \{s \in Q : s \cap s^* = \emptyset\}$
\FOR{$s' \in Q$}
\STATE $F_{s'} \leftarrow 1$
\ENDFOR
\FOR{$s \in Q$}
\IF{$s'$ conflicts with $s$}
\STATE $F_s \leftarrow 0$
\STATE \textbf{break}
\ENDIF
\ENDFOR
\FOR{$s \in Q$ such that $c(s) \cap c(s^*) = \emptyset$}
\STATE $d \leftarrow \max_{q_{ij} \in s \cup s^*} d_{ij} - \min_{q_{ij} \in s \cup s^*} d_{ij}$
\IF{$d \leq \Delta$}
\STATE $Q \leftarrow Q \cup \{s \cup s^*\}$
\STATE $D_{s \cup s^*} \leftarrow d$
\STATE $F_{s \cup s^*} \leftarrow 1$
\ENDIF
\ENDFOR
\ENDWHILE
\ENDFOR
\STATE \textbf{return} $s^*$, the true set of all the input queues associated with $e$.
\end{algorithmic}
\end{algorithm}

**Theorem IV.1.** Let $\Delta^* := \min_{e,c,e' \in T, e \neq e'} |\delta^*_e - \delta^*_{e'}|$. Algorithm 2 will output $Q = \{s_e\} \subset T$ if every input parameter $\delta_{ij}$ is associated with some $e \in T$ and $|\delta_{ij} - \delta^*_{ij}| \leq \frac{1}{2}\Delta < \frac{1}{2}\Delta^*$.

**Proof.** As the estimation error for any $\delta_{ij}$ is no more than $\Delta/2$, we must have $\delta_{ij} \in [\delta^*_e - \Delta/2, \delta^*_e + \Delta/2]$ for each $q_{ij} \in s_e$. Thus, $D_{s_e} \leq \Delta$, making $s_e$ and its subsets candidate sets. Meanwhile, for any $e' \in T$ that $e' \neq e$, we will have $|\delta^*_{e'} - \delta^*_e| \geq \Delta^* > 2\Delta$. Hence, the estimated parameter $\delta_{ij}$ of any input queue associated with $e'$ must satisfy $|\delta_{ij} - \delta^*_{ij}| > 3\Delta/2$, and thus $|\delta_{ij} - \delta^*_{ij}| > \Delta$ for all $q_{ij} \in s_e$. This means that a set of input queues $s$ will be considered a candidate set by Algorithm 2 if and only if $s \subseteq s_e$ for some $e \in T$.

Moreover, we claim that the final output $Q$ of Algorithm 2 must contain $s_e$ for each $e \in T$. We prove this by a top-down induction. First, the set $s_{e_0}$ for $e_0$ at the root of $T$ will not conflict with any other set and thus must be selected into $Q$. Second, if for a given $e \in T$, $s_{e'} \in Q$ for each ancestor $e'$ of $e$, then $s_e$ will not conflict with any $s \in Q$ (associated with some $e'' \in T$), since $c(s) \supseteq c(s_e)$ if $e''$ is an ancestor of $e$, $c(s) \cap c(s_e) = \emptyset$ if $e''$ is on a different branch from $e$, and $c(s) \subseteq c(s_e)$ if $e''$ is at or below $e$ on the same branch, and thus $s_e$ must be selected into $Q$. The proof completes by noting that no $s \subset s_e$ for any $e \in T$ will be in $Q$ due to line 12.

**B. Constructing Queuing Network Topology**

The result of Algorithm 2 helps to infer the queuing network $T$ by revealing the set of queues and the position of each queue. Specifically, if Algorithm 2 infers that a set of input queues $\{q_{i1,j}, \ldots, q_{ik,j}\}$ correspond to the same queue $q$ in $T$, then $q$ must have category $\{i_1, \ldots, i_k\}$, and hence reside on the tree branch containing “destinations” $\{d_{i1}, \ldots, d_{ik}\}$. We now use this idea to construct the tree.

1) **Algorithm:** Algorithm 3 constructs the tree by going through the sets of associated queues inferred by Algorithm 2 in the increasing order of cardinality (line 2), breaking ties arbitrarily, and constructing a vertex to represent each set (line 3). This results in a bottom-up approach that constructs the tree from the leaves to the root. At any time, the set $R$ contains the top-most vertex in each constructed branch. After constructing a new vertex $v_s$, the algorithm will connect it with each vertex $v_{s'} \in R$ that is on the same branch as $v_s$ (indicated by $c(s') \cap c(s) = \emptyset$) and update $R$ (lines 4–8). If $Q$ does not contain any set of category $\{1, \ldots, N\}$, then the constructed topology will be a forest, in which case we merge the roots to form a tree (line 10).

\begin{algorithm}[H]
\caption{Topology Construction}
\begin{algorithmic}
\STATE \textbf{input :} inferred parameters $\delta_1, \ldots, \delta_N$, output $Q$ of Algorithm 2
\STATE \textbf{output:} inferred topology $\hat{T}$
\STATE $\hat{T} \leftarrow \emptyset; R \leftarrow \emptyset$
\FOR{$s \in Q$ in increasing order of $|s|$}
\STATE create a vertex $v_s$ in $\hat{T}$ with parameter $\delta_{v_s} = \text{mean}(\{\delta_{ij} : q_{ij} \in s\})$
\ENDFOR
\FOR{$v_{s'} \in R$}
\IF{$c(s') \cap c(s) \neq \emptyset$}
\STATE create an edge $(v_s, v_{s'})$ in $\hat{T}$
\STATE $R \leftarrow R \setminus \{v_{s'}\}$
\ENDIF
\ENDFOR
\IF{$|R| > 1$}
\STATE merge all the vertices in $R$;
\ENDIF
\ENDFOR
\end{algorithmic}
\end{algorithm}

2) **Complexity:** Recall from Section IV-A2 that $|Q| = O(KN^2)$. Moreover, $|R| = O(N)$, as after constructing the $N$ leaves, each new vertex will replace at least one existing vertex in $R$. This implies a space complexity of $O(KN^3)$, dominated by the space for storing $Q$, and a time complexity of $O(KN^3)$, dominated by line 5 (as there are $O(KN)$ loops in line 2, $O(N)$ loops in line 4, and $c(s) \leq N$ for all $s \in Q$).

3) **Correctness:** We say that $v$ is a branching point in $T$ if it is a vertex with at least two children. We say that two vertices $v_1, v_2$ in $T$ are on the same segment if they are traversed by the same set of root-to-leaf paths. Based on
Algorithm 4: Queuing Network Topology Inference

\textbf{input:} paths \{\(p_i\)\}_{i=1}^n, \#measurements per path \(n\),
maximum \#queues per path \(K\), parameter for
Laplace transform \(S\), threshold \(\Delta\)

\textbf{output:} inferred topology \(\hat{T}\)

1 \textbf{for} \(i = 1, \ldots , N\) \textbf{do}
2 \hspace{1em} measure the delays \(x_i := (x_{i,h})_{h=1}^n\) of \(p_i\);
3 \hspace{1em} \(\delta_i\) \leftarrow \text{parameters for } p_i \text{ inferred by Algorithm 1 or}
MLE based on \(x_i\), \(K\), and \(S\) (if applicable);
4 \hspace{1em} \(Q \leftarrow \text{sets of associated queues inferred by}
\text{Algorithm 2 based on } (\delta_i)_{i=1}^N \text{ and } \Delta;\)
5 \hspace{1em} \(\hat{T} \leftarrow \text{topology constructed by Algorithm 3 based on}
(\delta_i)_{i=1}^N \text{ and } Q;\)

these concepts, we will show that the output of Algorithm 3
is correct if its input \(Q\) is correct in the following sense.

Theorem IV.2. If \(Q = \{s_e\}_{e \in T}\), where \(s_e\) is the set of all
the input queues associated with queue \(e\), then the topology
\(\hat{T}\) constructed by Algorithm 3 will be identical to \(T\), except
that vertices on the same segment may be permuted.

\textbf{Proof.} For each vertex \(e \in T\), let \(\hat{e}\) denote the first branching
point at or below \(e\), and \(v_{se}\) the vertex in \(\hat{T}\) corresponding
to \(e\). As each path traversing a vertex \(e’\) in \(T\) must traverse
its parent \(e\), we have \(c(s_{e’}) \subseteq c(s_e)\), and \(c(s_{e’}) \subset c(s_e)\)
if \(e\) is a branching point. Thus, when Algorithm 3 creates
a vertex \(v_{se}\) based on \(s_e \in Q\), the vertices \(\{v_{se’}\}_{e’ \in E}‘\)
for each child \(e’\) of \(\hat{e}\) must have been created and satisfy
\(c(s_{e’}) \cap c(s_e) \neq \emptyset\), and every \(v_{se’}\) for \(e’\) on a different
branch from \(e\) must have \(c(s_{e’}) \cap c(s_e) = \emptyset\). If \(s_e\) is the first
considered set of category \(c(s_e)\), then \(v_{se}\) will be connected
to \(v_{se’}\), \((\forall e’ \in E)’\); otherwise, \(v_{se}\) will be connected
to the top-most vertex \(v_{se_0}\) with \(c(s_{e_0}) = c(s_e)\) (since \(v_{se_0}\)
will have replaced \(\{v_{se’}\}_{e’ \in E}Δ\), where \(e_0\) and \(e\) must
be on the same segment in \(T\). Vertices on the same segment
will tie in line 2 and thus may be created in any order.
\hspace{1em} \qedsymbol

Remark: The order of vertices (each representing a queue)
on the same segment of \(T\) does not affect the end-to-end delay
distribution on any path and is hence not identifiable.

C. Summary of Solution

Algorithm 4 summarizes our overall solution. Theorems III.1, IV.1, and IV.2 together guarantee that the inferred
topology will converge to the ground truth (up to a permutation
of queues on the same segment) as the number of
measurements per path increases.

V. PERFORMANCE EVALUATION

We evaluate the proposed algorithms against benchmarks
based on real Internet topologies.

A. Simulation Setup

1) Topology Generation: We generate the ground-truth tree
topologies based on a Rocketfuel Autonomous System (AS)
topology [22], which represents IP-level connections between
the routers in AS6461 of Abovenet, containing 182 nodes and
294 links. Given a maximum path length (measured in \#nodes)
of \(K\), we start from a randomly selected high-degree node \(s’\)
(with degree > 6) as the “source” and perform a breadth first
search (BFS) to obtain a tree of height \(K\). We then randomly
pick \(N\) of the leaves of this tree as destinations, forming a
subtree with \(N\) leaves, which is used as the ground truth \(T\)
for topology inference.

The above topology construction guarantees that the num-
ber of nodes (each representing a queue) on each root-to-leaf
path is bounded by \(K\). We randomly assign each node \(q_i\), a
parameter \(\delta_i \in (0, 2)\), while making sure that \(|\delta_i - \delta_j| \geq \Delta^*\)
for \(i \neq j\), where \(\Delta^*\) denotes the minimum (possible)
difference between the parameters of different queues (\(\Delta^* = 0\)
means they can be arbitrarily close).

2) Benchmarks: For estimating the \(\delta\)-parameters from
the end-to-end delays on each path, we compare the proposed
estimator (Algorithm 1) with the MLE (solved by the Nelder-
Mead simplex method [23]), and other Laplace-based estima-
tors discussed in Section III-B, including the two methods
proposed by Harris et al. [20] (‘Harris 1’, ‘Harris 2’), and a
variation of the proposed estimator that replaces the absolute
error (4a) by the squared error (‘squared’).

For topology inference, existing solutions relied on correlated
measurements provided by multicast or its approximations
(see Section I-A), and are thus not applicable to our
measurements. Nevertheless, as these solutions aim at inferring
the multicast tree, which is a logical topology obtained by merging nodes on the same segment
(e.g., Fig. 5 (c)), we will compare with the accuracy of
approximating the ground-truth physical topology by the cor-
responding multicast tree, which can be estimated by Rooted
Neighbor Joining (RNJ) [5] from sufficiently many multicast
measurements. The multicast tree represents the asymptotic
accuracy of existing solutions that employ active probing,
and thus should be treated as a “performance upper bound”.

3) Metrics: We evaluate the accuracy of parameter esti-
mation by the relative error, defined as \(||\hat{\delta} - \delta^*|| / ||\delta^*||\),
where \(\hat{\delta}\) is the estimate and \(\delta^*\) the ground truth. If their
dimensions \(\text{dim}(\hat{\delta})\) and \(\text{dim}(\delta^*)\) are different, we fill \(\hat{\delta}\) with
the maximum value if \(\text{dim}(\hat{\delta}) < \text{dim}(\delta^*)\), and discard
the largest \(\text{dim}(\hat{\delta}) - \text{dim}(\delta^*)\) elements of \(\hat{\delta}\) if \(\text{dim}(\hat{\delta}) > \text{dim}(\delta^*)\).

We evaluate the accuracy of topology inference by a version
Graph edit distance is a typical performance metric for topol-
ygy inference algorithms [25]. In our case, a common error is
duplicating the same node (i.e., queue) or incorrectly merging
different nodes due to errors in the estimated \(\delta\)-parameters,
which is captured by this graph edit distance. As illustrated
in Fig. 5, the inferred topology (Fig. 5 (b)) has an edit distance
of 1 to the ground truth (Fig. 5 (a)), as merging \(q_{11}\)
and \(q_{12}\) will make it identical to the ground truth. The multicast
tree (Fig. 5 (c)) has an edit distance of 3 to the ground
truth, requiring \(d_4\) to be split once and \(d_5\) to be split twice.
C. Results on Topology Inference

Having validated the proposed estimator, we now evaluate the accuracy of using its outputs to infer the topology. All the results are based on 20 Monte Carlo runs.

Impact of threshold $\Delta$ and minimum gap $\Delta^*$: Given the estimated $\delta$-parameters, the remaining design parameter is the threshold $\Delta$ for detecting the parameters associated with the same queue. According to Theorem IV.1, we want $\Delta < \Delta^*/2$, where $\Delta^*$ denotes the minimum gap between the parameters of different queues. Our experiments with various $\Delta$ values (plots omitted due to space limitation) also indicated that a small $\Delta$ (e.g., $\Delta = 0.01$) works well in general.

Intuitively, our approach will work better if the $\delta$-parameters for different queues are more separated, i.e., when the minimum gap $\Delta^*$ is larger. However, as shown in Fig. 9, our algorithm is actually not sensitive to $\Delta^*$. Moreover, for small topologies as evaluated here, the finite-sample accuracy of our algorithm can even beat the asymptotic accuracy of existing multicast-based algorithms (e.g., RNJ [5]). This is because instead of only using low-order statistics (e.g., delay variances and covariances [5]), we examine the entire (empirical) distribution of measurements, which allows us to discover the physical topology that may contain more internal nodes than the (logical) multicast tree.

Impact of tree size: We vary the size of the ground truth topology by separately varying the width of the tree (indicated by the number of destinations $N$) and the height of the tree (indicated by the maximum path length $K$), as shown in Fig. 10. As expected, growing either of these parameters will increase the inference error, as the ground truth topology becomes more complex. Compared with solutions designed to infer the multicast tree, we see that our algorithm achieves...
We revisited a classic problem of inferring a tree topology from end-to-end measurements originated by a single source, through a fundamentally different approach that utilizes the detailed queuing dynamics inside the network. Compared to the classical approaches based on multicast or its approximations, our approach has the advantages that it can utilize passive measurements and potentially recover the physical topology with more hidden nodes (degree-2 nodes). Our solution includes a novel estimator that infers the residual capacities of a tandem of queues from end-to-end delays, and efficient algorithms that use the inferred residual capacities to identify the queues shared between paths and infer the queuing network topology. Our solution was guaranteed to be asymptotically consistent, and even outperformed existing solutions based on active probing for small topologies. Meanwhile, we also identified potential directions for further improvements.

**VI. CONCLUSION**

We revisited a classic problem of inferring a tree topology from end-to-end measurements originated by a single source, through a fundamentally different approach that utilizes the detailed queuing dynamics inside the network. Compared to the classical approaches based on multicast or its approximations, our approach has the advantages that it can utilize passive measurements and potentially recover the physical topology with more hidden nodes (degree-2 nodes). Our solution includes a novel estimator that infers the residual capacities of a tandem of queues from end-to-end delays, and efficient algorithms that use the inferred residual capacities to identify the queues shared between paths and infer the queuing network topology. Our solution was guaranteed to be asymptotically consistent, and even outperformed existing solutions based on active probing for small topologies. Meanwhile, we also identified potential directions for further improvements.

**REFERENCES**


