Queueing Network Topology Inference Using Passive and Active Measurements

Akash Kumar, *Student Member, IEEE*, Yudi Huang, and Ting He, *Senior Member, IEEE*

***Abstract*—We revisit a classic problem of inferring the routing tree for a given source in a packet-switched network from end-to-end measurements, with two critical differences from existing solutions: (i) instead of exclusively relying on active measurements obtained by probing, we strive to maximally utilize passive measurements obtained from data packets; (ii) instead of inferring a logical topology that omits degree-2 nodes, we want to recover the physical topology containing all the nodes. Our main idea is to utilize the detailed queueing dynamics inside the network to estimate a certain parameter (residual capacity) of each queue, and then use the estimated parameters as fingerprints to detect the queues shared across paths and thus infer the topology. To this end, we develop a Laplace-transform-based estimator to estimate the parameters of a tandem of queues from end-to-end delays, and efficient algorithms to infer the topology by identifying the parameters associated with the same queue. To improve the accuracy, we further develop a hybrid algorithm that uses the information from active measurements to identify (generalized) siblings and the information from passive measurements to detect shared queues on the paths from the source to each pair of identified siblings. Our inferred topology is guaranteed to converge to the ground-truth topology as the number of measurements increases, up to a permutation of the queues traversed by the same set of paths. Our evaluations in both queueing-theoretic and packet-level simulations show that the proposed solutions, particularly the hybrid algorithm, significantly improve the accuracy over the state of the art.**

***Index Terms*—Network topology inference, passive and active measurement, queueing network, neighbor joining.**

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 topology information is typically available to the network administrator (who can obtain such information through local monitoring agents), obtaining this information for multi-domain networks such as the Internet is much more challenging. In these cases, it is desirable to have alternative methods that are implementable by hosts. These methods can be broadly classified into the protocol-based approach and the measurement-based approach. The former approach utilizes specific control-plane protocols such as Internet Control Message Protocol (ICMP) to discover internal nodes (e.g., routers) by triggering responses from them through specially-designed probes (such as `traceroute`), but its success hinges on the cooperation of internal nodes. The latter approach, which is a branch of *network tomography* [2], tries to infer the network topology from end-to-end measurements, which complements the protocol-based approach when internal cooperation is unavailable (e.g., in a network of anonymous routers [3], [4]).

Since the introduction in the late 1990s [5], a number of measurement-based topology inference algorithms have been developed, which used multicast measurements [6], [7], [8], their unicast-based approximations [9], [10], or network coding [11] to infer the routing tree rooted at each probing source, and then merged the trees for multiple sources to form a more comprehensive topology [12], [13]. The foundation of these works is a probing scheme that generates *specifically correlated measurements* across paths, so that the correlation (caused by shared links) 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 (routing) topology.

Despite the extensive studies, existing topology inference algorithms have the critical limitations that (i) they rely on *active probes* to generate the specifically correlated measurements 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 address these limitations in the context of *single-source topology inference*, by developing topology inference algorithms that are designed to utilize *passive measurements* with arbitrary (or no) correlation across paths, and recover the *physical (network-layer) topology* with degree-2 nodes. This is achieved by a fundamentally different approach that utilizes the detailed queueing dynamics inside the network to fingerprint each link (modeled as a queue) and then uses these fingerprints to identify the links shared between paths. As algorithms for multi-source topology inference usually require inferred single-source topologies as input [12], [13], our solution also lays the foundation for improving multi-source topology inference.

A. Related Work

Network topology inference: Our work belongs to a branch of network tomography aiming at inferring routing topologies from end-to-end measurements [2]. The technique originated from the observation that correlated losses observed at multicast receivers can be used to infer the multicast tree [5], and was then extended to utilize a variety of multicast measurements, including losses [6], [14], delays [7], [14] and a combination of both [8]. As multicast is not widely supported, solutions based on unicast were proposed [9], [10], [15]. These solutions, however, were based on specially-designed probing schemes such as stripes of back-to-back unicast packets [9],

Received 15 August 2024; revised 28 February 2025 and 4 June 2025; accepted 17 June 2025; approved by IEEE TRANSACTIONS ON NETWORKING Editor S. Magnusson. This work was supported by the National Science Foundation under Award CNS-2106294. Part of this work was presented at IFIP Networking [DOI: 10.23919/IFIPNetworking52078.2021.9472774]. *(Corresponding author: Ting He.)*

Akash Kumar and Ting He are with the Department of Computer Science and Engineering, The Pennsylvania State University, University Park, PA 16802 USA (e-mail: ajk6173@psu.edu; tinghe@psu.edu).

Yudi Huang was with The Pennsylvania State University, University Park, PA 16802 USA. He is now with Nvidia, Santa Clara, PA 95050 USA (e-mail: hyd1123camel@gmail.com).

This article has supplementary downloadable material available at https://doi.org/10.1109/TON.2025.3582205, provided by the authors.

Digital Object Identifier 10.1109/TON.2025.3582205

[15] or "sandwiches" of small and large packets [10], 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., [12], [13] 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., [11]). These solutions require the internal nodes to perform network coding, thus not applicable in current packet-switched networks.

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

Queueing parameter inference: Our approach is based on the inference of queueing parameters from end-to-end measurements. To this end, a variety of parameters have been tackled in the context of communication networks. For example, it was shown in [16] that the difference between the delays measured when the buffer is full/empty can be used to estimate the buffer size at the bottleneck link. In [17], packet arrival times and flow identifiers were used to detect bottleneck links shared between flows and estimate their bandwidths. In [18], periodic probes were used to detect the "dominant congested link" on a path and estimate the maximum queueing 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 queueing systems, the inference of queueing 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 [19]. However, when the system becomes more complex (e.g., a tandem of queues), inversion techniques became unstable [19], and solutions fell back to standard algorithms based on maximum likelihood estimation [19], [20]. We refer to [21] for a more comprehensive bibliography in this space. To our knowledge, *all* the existing works assumed the queueing network topology to be known.

B. Summary of Contributions

We aim at inferring the topology of a queueing network modeling the connections from a given source to a given set of destinations from the measurements of end-to-end delays. Our contributions are:

1) We propose a novel approach for topology inference that can utilize passive measurements and recover the physical queueing network topology by exploiting the detailed queueing dynamics inside the network.
2) We develop a Laplace-transform-based estimator that can estimate certain parameters (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) Using the estimated parameters as queue fingerprints, we develop computationally efficient algorithms to identify the queues shared by different paths, and then construct a tree topology accordingly. The constructed topology is guaranteed to be identical to the ground-truth topology up to a permutation of the queues traversed by the same set of paths, if the parameter estimation is sufficiently accurate.
4) To improve the accuracy for larger networks where many queues may have similar parameters, we develop a hybrid algorithm that uses both information from passive measurements and information from active measurements to infer the topology, which can correctly infer the ground-truth topology (up to a permutation of the queues traversed by the same set of paths) in some cases where the algorithms using only one type of information cannot.
5) Our evaluations based on real Internet topology show that: (i) the proposed estimator outperforms existing estimators in efficiency and accuracy, (ii) the proposed algorithm based on only passive measurements significantly outperforms the state-of-the-art algorithm based on active probing due to its ability of inferring degree-2 nodes, and (iii) the proposed algorithm that uses both passive and active measurements further improves the accuracy.

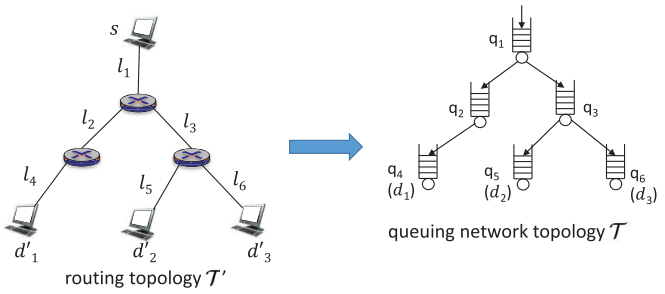


Fig. 1. Queueing network model.

Roadmap. Section II formulates our problem, Section III addresses parameter estimation for a tandem of queues, Section IV addresses topology inference based on only the estimated parameters, Section V presents a way to jointly utilize the estimated parameters and information from active measurements, Section VI evaluates the proposed algorithms against benchmarks, and Section VII concludes the paper. **Proofs and additional evaluations are provided in the appendix in Supplementary Material.**

II. Problem Formulation

A. Network Model

Given the routing tree $\mathcal{T}'$ connecting a given source s' to a given set of destinations $\{d'_1, \ldots, d'_N\}$, we model this topology by a queueing network as shown in Fig. 1, where each queue q_l models the outgoing interface at the beginning of a link $l \in \mathcal{T}'$. This model is motivated by the fact that queueing in packet-switched networks typically occurs at the outgoing interfaces [22]. It is easy to see that the resulting queueing network also has the topology of a rooted tree, denoted by $\mathcal{T}$, where each vertex represents a queue that corresponds to a link in the original routing topology. One can easily obtain the original topology from $\mathcal{T}$. Let $d_i \in \mathcal{T}$ $(i \in [N])$[1] denote the leaf modeling the access link for destination d'_i, and p_i denote the path from the root of $\mathcal{T}$ to d_i.

We model each q_l as an $M/M/1$ queue, whose sojourn time Y_l models the delay imposed by link l on a packet traversing l. Specifically, let λ_l denote the unknown load on link l and

[1]Given a positive integer N, $[N]$ denotes $\{1, \ldots, N\}$.

μ_l denote the unknown capacity of this link, both measured in packets per unit time. We assume that $\mu_l > \lambda_l$, which guarantees queue stability. Then it is well-known [23] that the sojourn time Y_l of q_l in the steady state is exponentially distributed with a parameter $\delta_l := \mu_l - \lambda_l$ that represents the *residual capacity*, i.e., the PDF of Y_l at $t_l > 0$ is $\delta_l e^{-\delta_l t_l}$. Moreover, we assume that the sojourn times of a given packet at different queues are independent of each other, and hence the end-to-end delay on path p_i follows the *hypoexponential distribution* with parameters $(\delta_l)_{l \in p_i}$, where "$l \in p_i$" means for each queue q_l on path p_i.

Remark 1: Our assumptions are automatically satisfied if $\mathcal{T}$ is a Jackson network with $M/M/1$ queues, which is a commonly-used model in queueing theory. We refer to [19].2] for a detailed discussion on the realism of this model. We 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 independent cross-traffic, and thus a measurement packet will incur independent waiting times and hence largely independent sojourn times at different queues (assuming the sojourn times to be dominated by the waiting times). While M/M/1 is a simplification of the actual queueing behavior at each link, prior studies have shown that this model allows one to approximately estimate the actual residual capacity [24]. Note that this queueing network model is only used for its tractability, and we will evaluate the solutions derived from this model in more realistic scenarios (see Section VI).

B. Observation Model

As typical in network topology inference, we assume the measurements to be collected through the cooperation between the source and the destinations. However, in contrast to the previous works that only rely on active measurements from controlled probing, we utilize both active and passive measurements. Specifically, we assume that by passively monitoring the transmission of data packets, we can obtain a sequence of end-to-end delays on each path p_j, denoted by $\boldsymbol{x}_j^{(0)} := (x_{j,h}^{(0)})_{h=1}^{n_j^{(0)}}$, where $n_j^{(0)}$ is the number of passive measurements on p_j and $x_{j,h}^{(0)}$ is the h-th passive measurement on path p_j. In addition, the source may also collect active measurements. We adopt the probing scheme in [9], where the source sends pairs of back-to-back probes to each pair of destinations to mimic multicast to these destinations. Let $\boldsymbol{x}_j^{(i)} := (x_{j,h}^{(i)})_{h=1}^{n_{ij}}$ ($i \in [N] \setminus \{j\}$) denote the sequence of end-to-end delays on path p_j measured by (mimicked) multicast on p_j and p_i, where n_{ij} is the number of multicast probes on p_j and p_i. As common in the literature, we assume the measurements to be temporally i.i.d. The assumption of temporal independence can be ensured by having sufficient spacing between consecutive measurements (i.e., between probe pairs or monitored data packets) as in [10]. The assumption of identical distribution can be approximated by performing measurements within a small time window and/or during off-peak hours so that there is minimal fluctuation in background traffic. In our experiments, we find that topology inference can converge on as few as 50 seconds of measurements. As in [9], we assume that probes in the same pair incur the same delay at shared links (if any). We do not make any assumption about the cross-path correlation of the delays from passive measurements (i.e., they may be arbitrarily correlated or uncorrelated). Note that this observation model is only used for deriving our solutions, and we will evaluate their actual performance in more realistic scenarios in Section VI.

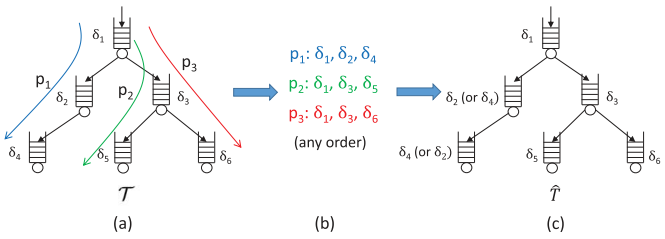


Fig. 2. Motivating example: (a) true topology, (b) estimated queue parameters for each path, (c) topology inferred from the estimated parameters.

C. Objective

Given the passive measurements from each path $(\boldsymbol{x}_j^{(0)})_{j \in [N]}$ and (optionally) the active measurements from each pair of paths $\left((\boldsymbol{x}_j^{(i)}, \boldsymbol{x}_i^{(j)})\right)_{i,j \in [N]}$, we want to infer the queueing network topology $\mathcal{T}$ to the maximum accuracy.

D. Motivating Example

It is well-known that through carefully-designed probing schemes such as mimicked multicast [9], we can detect the existence of links shared between paths and thus infer the topology. However, we will illustrate by a simple example that the topology may still be inferrable even if probing is not allowed. As illustrated in Fig. 2a, $\mathcal{T}$ models the connection from a source to three destinations via paths p_1, p_2, and p_3. Under the network model in Section II-A, the end-to-end delay on p_j ($j = 1, \ldots, 3$) follows a hypoexponential distribution with the parameters listed in Fig. 2b, which can be accurately estimated after collecting sufficiently many passive measurements as shown later (in Section III). Under the assumption that different queues have different parameters, we can infer that there is a queue shared by all three paths with parameter δ_1 and another queue shared by paths p_2 and p_3 with parameter δ_3. Due to the tree structure, we can then infer the topology as in Fig. 2c, where the only uncertainty is in the order of the queues with parameters δ_2 and δ_4 on p_1. Since this order does not affect the end-to-end delay distribution on any path, Fig. 2c is the most accurate estimate of $\mathcal{T}$ that can be obtained from end-to-end measurements.

Following the idea in this example, we will develop our solution in two steps: (1) estimating the δ-parameters (i.e., residual capacities) for a tandem of queues from the end-to-end delays, and then (2) inferring the queueing network topology by identifying the queues shared between paths from the estimated δ-parameters.

III. Parameter Estimation for Tandem Queues

We start by studying how to estimate the parameters $\boldsymbol{\delta}_j := (\delta_{j,k})_{k=1}^{K_j}$ for a tandem of queues on a given path p_j based on the delay measurements $\boldsymbol{x}_j := (x_{j,h})_{h=1}^{n_j}$ from p_j, where K_j is the number of queues on p_j (i.e., the hop count in the routing topology), and n_j is the number of measurements.[2] For

[2]These can include passive measurements alone (i.e., $n_j = n_j^{(0)}$) or both passive and active measurements on p_j (i.e., $n_j = n_j^{(0)} + \sum_{i \in [N] \setminus \{j\}} n_{ij}$).

simplicity, we will omit the path index and simply denote the delay measurements on the considered path by $\boldsymbol{x} := (x_h)_{h=1}^n$ and the queue parameters by $\boldsymbol{\delta} := (\delta_k)_{k=1}^K$, where K is the number of queues on the path, and n is the number of delay measurements. We assume that the hop count of each source-to-destination path is known, as even if all the routers are anonymous, we can still measure the hop count by tools like `traceroute` (i.e., the minimum TTL for a packet to reach the destination).[3] As the order of queues does not affect the end-to-end delay distribution and hence cannot be identified by the delay measurements from a single path, we assume $\delta_1 \le \cdots \le \delta_K$ when comparing the estimated and the true parameters.

A. Maximum Likelihood Estimation (MLE)

The existing approach as proposed in [19] is to apply MLE. MLE is considered the state-of-the-art estimator for phase-type distributions [25], 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:

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

and the MLE aims at finding the value of $\boldsymbol{\delta}$ that maximizes the log-likelihood $\sum_{h=1}^n \log g(x_h;\, \boldsymbol{\delta})$. If solved exactly, the MLE has a desirable property that it is asymptotically efficient under regularity conditions,[4] i.e., as the number of measurements increases, it converges to the true parameter at a rate approximating the *Cramér-Rao bound* [26].

However, the log-likelihood function is non-concave, making it challenging to compute the MLE. To address this challenge, various algorithms have been proposed to compute approximations [25]. In particular, the *Expectation-Maximization (EM)* algorithm is guaranteed to converge to a local maximum and was adopted to solve this problem in [19]. 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 [20] 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 $\mathbf{E}[e^{-sX}]$ 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 [27]. While the transform has been used to estimate PDF/CDF from data [28], [29], to our knowledge, we are the first to apply it to parameter estimation.

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

[3]Using `traceroute` to obtain the hop count implies that the inferred topology is at the network layer. However, our solution applies to any queueing network satisfying the model in Section II-A for which the number of queues on each measurement path is known.

[4]The conditions are: (i) the log-likelihood function is twice differentiable, and (ii) the Fisher Information Matrix is non-zero, both satisfied in our case.

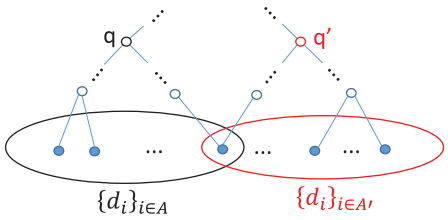


Fig. 3. A queue q on the paths to destinations $\{d_i\}_{i\in A}$ and a queue q' on the paths to destinations $\{d_i\}_{i\in A'}$ cannot coexist in a tree if $A \cap A' \neq \emptyset$, $A \not\subseteq A'$, and $A' \not\subseteq A$.

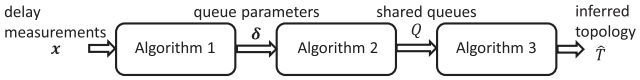


Fig. 4. Overall solution: topology inference using passive measurements.

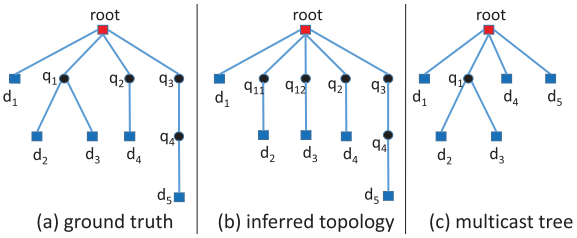


Fig. 5. Example: (b) has edit distance 1 to (a); (c) has edit distance 3 to (a).

the Laplace transform of an exponential random variable with parameter δ_i is $\delta_i/(\delta_i + s)$ $(s > -\delta_i)$, the Laplace transform of the end-to-end delay is

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

Our idea is to estimate the Laplace transform at a predetermined set of values for s, and find the parameter $\boldsymbol{\delta}$ that achieves the best fit.

Estimator: By definition, the empirical Laplace transform

$$\hat{L}(s;\, \boldsymbol{x}) := \frac{1}{n} \sum_{h=1}^{n} e^{-sx_h} \tag{3}$$

gives an unbiased estimate of $L(s;\, \boldsymbol{\delta})$. Given the empirical Laplace transform, we propose to estimate $\boldsymbol{\delta}$ by fitting the empirical Laplace transform at a given set of points in S (a design parameter):

$$\min \quad \sum_{s\in S} |L(s;\, \boldsymbol{\delta}) - \hat{L}(s;\, \boldsymbol{x})| \tag{4a}$$

$$\text{s.t. } 0 < \delta_1 \le \cdots \le \delta_K. \tag{4b}$$

The above uses absolute error, but other error measures such as squared error can also be used with similar accuracy.

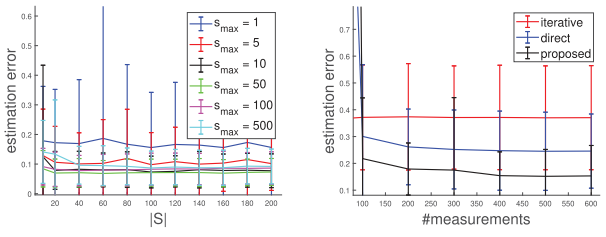
We note that there are other ways to use the Laplace transform for estimating $\boldsymbol{\delta}$. For example, [28] proposed to rewrite the Laplace transform (2) as

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

where the coefficients are related to $\boldsymbol{\delta}$ as

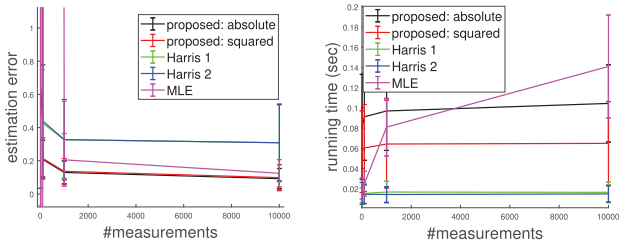
$$a_j := \sum_{A\subseteq\{1,\ldots,K\}, |A|=j} \prod_{i\in\{1,\ldots,K\}\setminus A} \delta_i, \;\; j = 0, \ldots, K-1. \tag{6}$$

Then they estimated the coefficients $a_0, \ldots, a_{K-1}$ from the empirical moments or the empirical Laplace transform, and



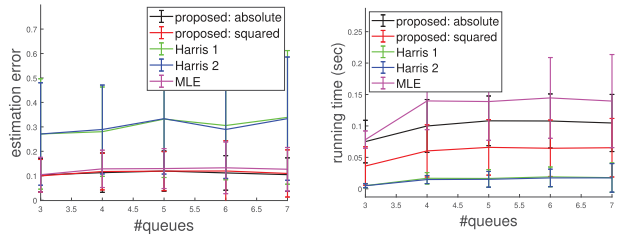
(a) design of S (b) initialization method

Fig. 6. Evaluation of design choices (y-axis trimmed for better visibility).



(a) estimation error (b) running time

Fig. 7. Parameter estimation: vary #measurements (#queues =4).



(a) estimation error (b) running time

Fig. 8. Parameter estimation: vary #queues (#measurements $= 10,000$).

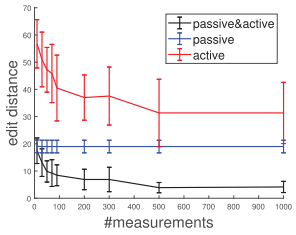


Fig. 9. Topology inference: vary sample size ($N = 20$, $K = 4$, #active measurements = #passive measurements).

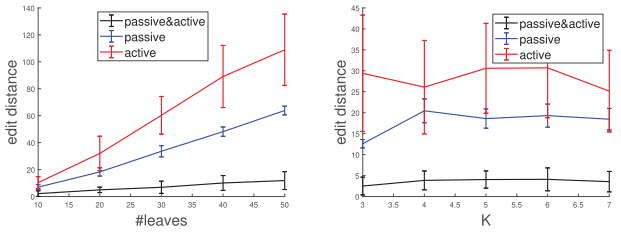
solved the system of equations (6) for $\boldsymbol{\delta}$. However, we find that the proposed estimator achieves much better accuracy (see Fig. 7- 8 and Fig. 18- 19 in the Supplementary Material).

Performance analysis: The proposed estimator is *consistent* under sufficiently large $|S|$ (see proof in Appendix A.1 in the Supplementary Material).

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

Besides the condition in Theorem 1, the values in S also affect the accuracy of the proposed estimator at finite samples sizes. Intuitively, S should contain a diverse range of values to provide a good description of the Laplace transform. It has been suggested in [28] that these values should be evenly distributed. We also find the range and the density of the points in S to matter (see Fig. 6a).

Algorithm: Although one can directly apply a generic optimization method to solve (4), we find the accuracy to be



(a) vary N ($K = 4$) (b) vary K ($N = 20$)

Fig. 10. Topology inference: vary tree size (#active/passive measurements $= 1,000$).

Algorithm 1 Laplace-Based Tandem Queue Inference

input : end-to-end delays $\boldsymbol{x} := (x_h)_{h=1}^n$, number of queues K, input parameters for Laplace transform S, #iterations for initialization J

output: estimated queue parameters $\hat{\boldsymbol{\delta}}$

1 **for** $s \in S$ **do**

2 $\quad \hat{L}(s;\boldsymbol{x}) \leftarrow \frac{1}{n}\sum_{h=1}^n e^{-sx_h}$;

3 let $\boldsymbol{\delta}^{(0)}$ be an arbitrary initial guess;

4 **repeat**

5 $\quad$ **for** $i = 1$ *to* K **do**

6 $\quad\quad \delta_i^{(0)} \leftarrow \text{argmin}_{\delta_i} \sum_{s\in S} |L(s;\boldsymbol{\delta}) - \hat{L}(s;\boldsymbol{x})|$, where $\delta_j = \delta_j^{(0)}$ for all $j \neq i$;

7 **until** J *times*;

8 $\hat{\boldsymbol{\delta}} \leftarrow \text{argmin}_{\boldsymbol{\delta}} \sum_{s\in S} |L(s;\boldsymbol{\delta}) - \hat{L}(s;\boldsymbol{x})|$ with initial value $\boldsymbol{\delta} = \boldsymbol{\delta}^{(0)}$;

sensitive to initialization. Therefore, we propose a two-step estimation algorithm as shown in Algorithm 1. Lines 1– 1 are used to find a good initial value $\boldsymbol{\delta}^{(0)}$ by iteratively optimizing one δ_i at a time, while fixing the other δ_j for $j \neq i$. Then line 1 further optimizes the estimate by a joint optimization starting from $\boldsymbol{\delta}^{(0)}$. We find that optimization based on such an initialization method outperforms optimization from an arbitrary initial value (see Fig. 6b).

IV. Topology Inference Using Passive Measurements

Given the inferred δ-parameters for each source-to-destination path in $\{p_i\}_{i\in[N]}$, we are ready to infer the queueing network topology $\mathcal{T}$. The key observation is that if each queue has a distinct δ-parameter (which is likely 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 and hence infer the topology. Specifically, given the inferred parameters $\boldsymbol{\delta}_i := (\delta_{i,k})_{k=1}^{K_i}$ for path p_i ($i \in [N]$) where $\delta_{i,k}$ is the k-th smallest parameter on p_i, we can view p_i as a tandem of K_i queues, referred to as *input queues* and denoted by $\boldsymbol{q}_i := (q_{i,k})_{k=1}^{K_i}$, where $q_{i,k}$ is the input queue with the k-th smallest parameter on p_i. Topology inference aims at merging these tandems of input queues at the shared queues into a tree-shaped network. We will solve this problem in two steps: (1) inferring which input queues represent the same queue in the underlying network, and (2) inferring the corresponding queueing network topology. Let $K := \max_{i\in[N]} K_i$ denote the maximum number of queues per path.

A. Inferring Shared Queues

We define a set of input queues, each on a different path, as a *shared queue*, if they correspond to the same queue in the underlying queueing network $\mathcal{T}$. Algorithm 2 shows our algorithm to identify the shared queues based on similarities of the estimated δ-parameters.

1) Algorithm: Specifically, define

$$D_{\{q_{i_1,j_1},\ldots,q_{i_k,j_k}\}} := \max\{\delta_{i_1,j_1},\ldots,\delta_{i_k,j_k}\} - \min\{\delta_{i_1,j_1},\ldots,\delta_{i_k,j_k}\} \tag{7}$$

as the error in associating the input queues $\{q_{i_1,j_1},\ldots,q_{i_k,j_k}\}$ to the same queue in $\mathcal{T}$. If the maximum estimation error for any $\delta_{i,j}$ is $\Delta/2$, then a set of input queues $\{q_{i_1,j_1},\ldots,q_{i_k,j_k}\}$ may represent the same queue in $\mathcal{T}$ only if $D_{\{q_{i_1,j_1},\ldots,q_{i_k,j_k}\}} \le \Delta$. Therefore, we refer to a set $\tilde{q}$ of input queues (each on a different path) satisfying $D_{\tilde{q}} \le \Delta$ as a *candidate set*, where Δ is a design parameter that controls the tradeoff between detecting truly shared queues and not detecting non-shared queues as shared queues.

However, the feasibility of different candidate sets (even if they are disjoint) cannot be determined independently due to the constraint of tree topology. To see this, we define the *category* of a set of input queues $\tilde{q} := \{q_{i_1,j_1},\ldots,q_{i_k,j_k}\}$ as the set of indices of the paths traversing the queues in $\tilde{q}$, denoted by

$$c(\tilde{q}) := \{i_1,\ldots,i_k\}. \tag{8}$$

We will also refer to the indices of the paths traversing a queue q in $\mathcal{T}$ as the category of q. Clearly, if all the input queues in $\tilde{q}$ represent the same queue q in $\mathcal{T}$, then $\tilde{q}$ and q have the same category. For example, the inference results in Fig. 2b suggest that there is a queue with parameter δ_1 that has category $\{1,2,3\}$. The problem is that if we associate a set of input queues of category $A \subseteq [N]$ with the same queue q in $\mathcal{T}$, then q must reside on the subtree containing "destinations" $\{d_i\}_{i\in A}$ (recall that d_i is the queue modeling the last link towards destination d_i'), and thus there cannot be another queue q' of category A' for any A' satisfying

$$A' \cap A \neq \emptyset,\ A' \not\subseteq A,\ \text{and}\ A \not\subseteq A', \tag{9}$$

as illustrated in Fig. 3. We say that two candidate sets $\tilde{q}$ and $\tilde{q}'$ *conflict with each other* if $c(\tilde{q})$ and $c(\tilde{q}')$ satisfy (9).

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 (7). However, a straightforward implementation of this idea will incur an exponential complexity as there can be $O((K+1)^N)$ 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 $\tilde{Q}$ the candidate sets that will be searched in the next iteration. For each $\tilde{q} \in \tilde{Q}$, let $F_{\tilde{q}} \in \{0,1\}$ indicate whether the candidate set $\tilde{q}$ is feasible, i.e., not conflicting with any $\tilde{q}' \in Q$. Starting by selecting all the singletons $\{q_{i,j}\}$ into Q as $D_{\{q_{i,j}\}} = 0$ (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) $\tilde{Q}$ 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).

Algorithm 2 Inference of Shared Queues

input : estimated parameters $\boldsymbol{\delta}_1,\cdots,\boldsymbol{\delta}_N$ for all the paths ($\boldsymbol{\delta}_i = (\delta_{i,j})_{j=1}^{K_i}$); threshold Δ
output: collection Q of sets of queues, where each $\tilde{q} \in Q$ is a set of input queues inferred to be associated with the same queue in $\mathcal{T}$

1 $Q \leftarrow \emptyset$; $\tilde{Q} \leftarrow \emptyset$;
2 **for** $i = 1,\ldots,N$ **do**
3 $\quad Q \leftarrow Q \cup \{\{q_{i,1}\},\ldots,\{q_{i,K_i}\}\}$;
4 **for** $\{q_{i_1,j_1}\},\{q_{i_2,j_2}\} \in Q$ *with* $i_1 \neq i_2$ **do**
5 $\quad$ **if** $|\delta_{i_1,j_1} - \delta_{i_2,j_2}| \le \Delta$ **then**
6 $\quad\quad \tilde{Q} \leftarrow \tilde{Q} \cup \{\{q_{i_1,j_1},q_{i_2,j_2}\}\}$;
7 $\quad\quad D_{\{q_{i_1,j_1},q_{i_2,j_2}\}} \leftarrow |\delta_{i_1,j_1} - \delta_{i_2,j_2}|$;
8 $\quad\quad F_{\{q_{i_1,j_1},q_{i_2,j_2}\}} \leftarrow 1$;
9 **while** $\exists \tilde{q} \in \tilde{Q}$ *with* $F_{\tilde{q}} = 1$ **do**
10 $\quad \tilde{q}^* \leftarrow \operatorname{argmin}_{\tilde{q}\in\tilde{Q},F_{\tilde{q}}=1} D_{\tilde{q}}$;
11 $\quad Q \leftarrow Q \cup \{\tilde{q}^*\}$;
12 $\quad Q \leftarrow Q \setminus \{\tilde{q} \in Q : \tilde{q} \subset \tilde{q}^*\}$;
13 $\quad \tilde{Q} \leftarrow \tilde{Q} \setminus \{\tilde{q} \in \tilde{Q} : \tilde{q} \cap \tilde{q}^* \neq \emptyset\}$;
14 $\quad$ **for** $\tilde{q}' \in \tilde{Q}$ **do**
15 $\quad\quad F_{\tilde{q}'} \leftarrow 1$;
16 $\quad\quad$ **for** $\tilde{q} \in Q$ **do**
17 $\quad\quad\quad$ **if** $\tilde{q}'$ *conflicts with* $\tilde{q}$ **then**
18 $\quad\quad\quad\quad F_{\tilde{q}'} \leftarrow 0$;
19 $\quad\quad\quad\quad$ break;
20 $\quad$ **for** $\tilde{q} \in Q$ *such that* $c(\tilde{q}) \cap c(\tilde{q}^*) = \emptyset$ **do**
21 $\quad\quad d \leftarrow \max_{q_{i,j}\in\tilde{q}\cup\tilde{q}^*} \delta_{i,j} - \min_{q_{i,j}\in\tilde{q}\cup\tilde{q}^*} \delta_{i,j}$;
22 $\quad\quad$ **if** $d \le \Delta$ **then**
23 $\quad\quad\quad \tilde{Q} \leftarrow \tilde{Q} \cup \{\tilde{q} \cup \tilde{q}^*\}$;
24 $\quad\quad\quad D_{\tilde{q}\cup\tilde{q}^*} \leftarrow d$;
25 $\quad\quad\quad F_{\tilde{q}\cup\tilde{q}^*} \leftarrow 1$;
26 $\quad\quad\quad$ **for** $\tilde{q}' \in Q$ **do**
27 $\quad\quad\quad\quad$ **if** $\tilde{q} \cup \tilde{q}^*$ *conflicts with* $\tilde{q}'$ **then**
28 $\quad\quad\quad\quad\quad F_{\tilde{q}\cup\tilde{q}^*} \leftarrow 0$;
29 $\quad\quad\quad\quad\quad$ break;

2) Complexity: 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, $\tilde{Q}$ contains the union of each pair of sets in Q, and hence $|\tilde{Q}| = O(K^2N^2)$. This implies a space complexity of $O(K^2N^3)$, dominated by the space for storing $\tilde{Q}$, as the cardinality of each set in $\tilde{Q}$ is at most N. For time complexity, the initialization (lines 2– 2) takes $O(K^2N^2)$, each while loop (lines 2– 2) takes $O(K^3N^4)$, dominated by the update of $(F_{\tilde{q}})_{\tilde{q}\in\tilde{Q}}$ in lines 2– 2 (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 Δ.

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

parameter, and $\tilde{q}_e$ the true set of all the input queues associated with e. Algorithm 2 will correctly identify all the shared queues under the following condition (see proof in Appendix A.2 in the Supplementary Material).

Theorem 2: Let $\Delta^* := \min_{e,e' \in \mathcal{T}, e \neq e'} |\delta^*_e - \delta^*_{e'}|$. Algorithm 2 will output $Q = \{\tilde{q}_e\}_{e \in \mathcal{T}}$ if every input parameter $\delta_{i,j}$ associated with some $e \in \mathcal{T}$ satisfies $|\delta_{i,j} - \delta^*_e| \leq \frac{1}{2}\Delta < \frac{1}{4}\Delta^*$.

B. Constructing Queueing Network Topology

The result of Algorithm 2 helps to infer the queueing network $\mathcal{T}$ by revealing the set of queues and their positions in the tree. Specifically, if Algorithm 2 infers that a set of input queues $\{q_{i_1,j_1}, \ldots, q_{i_k,j_k}\}$ correspond to the same queue q in $\mathcal{T}$, then q must reside on the tree branch covering leaves $\{d_{i_1}, \ldots, d_{i_k}\}$. We now use this idea to construct the tree.

Algorithm 3 Topology Construction

input : estimated parameters $\boldsymbol{\delta}_1, \cdots, \boldsymbol{\delta}_N$, inferred shared queues Q
output: inferred topology $\widehat{\mathcal{T}}$

1 $\widehat{\mathcal{T}} \leftarrow \emptyset$; $R \leftarrow \emptyset$;
2 **for** $\tilde{q} \in Q$ *in increasing order of* $|\tilde{q}|$ **do**
3 create a vertex $v_{\tilde{q}}$ in $\widehat{\mathcal{T}}$ with parameter $\delta_{v_{\tilde{q}}} = \text{mean}(\{\delta_{i,j}\}_{q_{i,j} \in \tilde{q}})$;
4 **for** $v_{\tilde{q}'} \in R$ **do**
5 **if** $c(\tilde{q}') \cap c(\tilde{q}) \neq \emptyset$ **then**
6 create an edge $(v_{\tilde{q}}, v_{\tilde{q}'})$ in $\widehat{\mathcal{T}}$;
7 $R \leftarrow R \setminus \{v_{\tilde{q}'}\}$;
8 $R \leftarrow R \cup \{v_{\tilde{q}}\}$;
9 **if** $|R| > 1$ **then**
10 merge all the vertices in R;

1) Algorithm: Algorithm 3 constructs the tree by going through the shared queues inferred by Algorithm 2 in the increasing order of cardinality[5] (line 2), breaking ties arbitrarily, and constructing a vertex (i.e., a queue) 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 subtree. After constructing a new vertex $v_{\tilde{q}}$, the algorithm will connect it with each vertex $v_{\tilde{q}'} \in R$ that is in the same subtree as $v_{\tilde{q}}$ (indicated by $c(\tilde{q}') \cap c(\tilde{q}) \neq \emptyset$) and update R (lines 3– 3). If Q does not contain any shared queue of category $[N]$, then the constructed topology will be a forest, in which case we merge the roots to form a tree (line 10).

2) Complexity: Recall from Section IV-A2 that $|Q| = O(KN)$. 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 $O(KN^2)$, dominated by the space for storing Q, and a time complexity of $O(KN^3)$, dominated by line 3 (as there are $O(KN)$ loops in line 3, $O(N)$ loops in line 3, and $|c(\tilde{q})| \leq N$ for all $\tilde{q} \in Q$).

3) Correctness: We say that v is a *branching point* in $\mathcal{T}$ if it is a vertex with at least two children. We say that two vertices v_1, v_2 in $\mathcal{T}$ are *on the same branch* if they are between the same pair of adjacent branching points (excluding the branching point closer to the root, including the branching point closer to the leaves), i.e., traversed by the same set of root-to-leaf paths. Based on these notions, we will show that the output of Algorithm 3 is correct if its input Q is correct (see proof in Appendix A.3 in the Supplementary Material).

Theorem 3: If $Q = \{\tilde{q}_e\}_{e \in \mathcal{T}}$, where $\tilde{q}_e$ is the set of all the input queues associated with queue $e \in \mathcal{T}$, then the topology $\widehat{\mathcal{T}}$ constructed by Algorithm 3 will be identical to $\mathcal{T}$, except that vertices on the same branch may be permuted.

Remark 2: The order of vertices (each representing a queue) on the same branch of $\mathcal{T}$ does not affect the end-to-end delay distribution on any path and is hence not identifiable.

C. Summary of Solution

Fig. 4 illustrates the overall solution, which (i) first uses end-to-end delay measurements $\boldsymbol{x} := (\boldsymbol{x}_i)_{i \in [N]}$ to infer the queue parameters $\boldsymbol{\delta} := (\boldsymbol{\delta}_i)_{i \in [N]}$ via Algorithm 1, (ii) then uses these parameters as fingerprints to infer the shared queues Q via Algorithm 2, and (iii) finally constructs an inferred topology $\widehat{\mathcal{T}}$ based on the shared queues via Algorithm 3. Theorems 1, 2, and 3 together guarantee that the inferred topology will converge to the ground truth topology (up to a permutation of queues on the same branch) as the number of measurements per path increases.

V. Topology Inference Using Passive and Active Measurements

While the algorithms in Section IV can correctly estimate the queueing network topology in theory as the number of measurements goes to infinity, it has a practical limitation that due to the limited accuracy of parameter estimation at finite sample sizes, it can be difficult to distinguish the parameters associated with one queue from those associated with a different queue, particularly in large networks when many queues have similar residual capacities. This issue cannot be addressed by simply increasing the number of (passive) measurements, as this will take a longer time during which the underlying routing topology could change. Our idea of addressing this problem is to augment the information from passive measurements with information from limited active measurements.

A. Information From Active Measurements

We will leverage a classical way of extracting topology information from active measurements. Classical topology inference relies on additive metrics that can be estimated from end-to-end measurements to measure the "lengths" of paths and shared portions between paths, hence inferring the topology. There are three commonly-used additive metrics: loss-based metric, utilization-based metric, and delay-based metric [9], where the *delay-based metric* is most consistent with our network model. Specifically, the delay-based metric w_l for each link l (modeled as a queue q_l) is defined as the variance of its delay Y_l (i.e., sojourn time of q_l), i.e., $w_l := \text{var}(Y_l)$. Under the M/M/1 queueing model in Section II-A, we can express w_l as $1/\delta_l^2$, where δ_l is the residual capacity of q_l. According to the delay-based metric, the shared length between paths p_i and p_j is defined as

$$\rho^*_{ij} := \text{cov}(X_i, X_j), \tag{10}$$

where (X_i, X_j) denotes the pair of delays on p_i and p_j from a (mimicked) multicast on these paths. Under the assumption of independent delays across links in Section II-A, it is easy

[5]The cardinality is well-defined because each shared queue is by definitiona set of input queues.

to see that $\rho^*_{ij} = \sum_{l\in p_i\cap p_j} w_l = \sum_{l\in p_i\cap p_j} 1/\delta_l^2$. Although the true value of ρ^*_{ij} is unknown, we can estimate it via the empirical covariance based on the delay pairs $((x^{(j)}_{i,h}, x^{(i)}_{j,h}))_{h=1}^{n_{ij}}$ measured by (mimicked) multicast probes:

$$\rho_{ij} := \frac{1}{n_{ij}-1}\sum_{h=1}^{n_{ij}}(x^{(j)}_{i,h} - \overline{x}^{(j)}_i)(x^{(i)}_{j,h} - \overline{x}^{(i)}_j), \tag{11}$$

where $\overline{x}^{(i)}_j := \frac{1}{n_{ij}}\sum_{h=1}^{n_{ij}} x^{(i)}_{j,h}$ is the sample mean, and the factor $1/(n_{ij}-1)$ (instead of $1/n_{ij}$) is used to correct the bias in the estimation [9]. By the definition of covariance, it is easy to see that as $n_{ij}\to\infty$, ρ_{ij} will converge to ρ^*_{ij}, i.e., the estimator in (11) is consistent.

B. Idea for Combining Active and Passive Measurements

Recall that $\tilde{q}_e$ (referred to as a *shared queue*) denotes the set of input queues corresponding to $e\in\mathcal{T}$, i.e., the estimated parameters $\{\delta_{j,k}\}_{q_{j,k}\in\tilde{q}_e}$ are all associated with the same queue e in the underlying network, and $c(\tilde{q}_e)$ (referred to as the *category* of $\tilde{q}_e$) denotes the set of indices of the paths traversing e. We have shown in Theorem 3 that the set Q of all the shared queues uniquely determines the queueing network topology $\mathcal{T}$, up to a permutation of queues on the same branch. Thus, it suffices to infer the set of shared queues Q correctly.

Although Theorem 2 shows that Algorithm 2 can correctly identify the shared queues if all the queues have sufficiently distinct parameters that can be estimated with sufficient accuracy, the challenge in applying this solution is that (i) the parameter estimation errors may not be upper-bounded by $\Delta^*/4$ (where Δ^* is the minimum difference between the parameters of different queues in $\mathcal{T}$), and (ii) the maximum estimation error (which is needed for setting the threshold Δ) is unknown. Nevertheless, we still expect the following to hold under reasonable estimation accuracy:

1) $\delta_{j,k}$'s will be similar for all $q_{j,k}\in\tilde{q}_e$;
2) $\sum_{\tilde{q}_e:i,j\in c(\tilde{q}_e)} 1/\delta_e^2$ will be close to the estimated shared path length ρ_{ij}, where $\delta_e := \left(\sum_{q_{j,k}\in\tilde{q}_e}\delta_{j,k}\right)/|\tilde{q}_e|$.

The above observations motivate the following idea: *first* using the passive (and possibly active) measurements from each path to estimate the queue parameters $\boldsymbol{\delta} := (\boldsymbol{\delta}_j)_{j\in[N]}$ and the active measurements from each pair of paths to estimate the shared path lengths $\boldsymbol{\rho} := (\rho_{ij})_{i,j\in[N]}$, and *then* using both $\boldsymbol{\delta}$ and $\boldsymbol{\rho}$ to infer the shared queues Q. Intuitively, the inference of shared queues should try to minimize (i) the deviation between parameters associated with the same queue as well as (ii) the deviation between the shared path length estimated from probes (i.e., ρ_{ij}) and the shared path length predicted by the estimated queue parameters (i.e., $\sum_{\tilde{q}_e:i,j\in c(\tilde{q}_e)} 1/\delta_e^2$), subject to feasibility constraints. From Section IV-A, we know that a candidate solution for Q is feasible if and only if

1) $\nexists j\in[N]$ and $k,k'\in[K_j]$ $(k\neq k')$ such that $q_{j,k}, q_{j,k'}\in\tilde{q}$ for some $\tilde{q}\in Q$, and
2) $\nexists\tilde{q},\tilde{q}'\in Q$ such that $c(\tilde{q})\cap c(\tilde{q}')\neq\emptyset$, $c(\tilde{q})\not\subseteq c(\tilde{q}')$, and $c(\tilde{q}')\not\subseteq c(\tilde{q})$.

The first constraint is because different input queues traversed by the same path are already known to represent different queues in the underlying network (due to the assumption of tree-based routing in Section II-A), and the second constraint is because the coexistence of two queues traversed by partially-overlapping sets of paths will lead to a contradiction of the assumption that the underlying topology $\mathcal{T}$ is a tree, as illustrated in Fig. 3.

Remark 3: Directly solving the above optimization is highly nontrivial due to the discrete and huge solution space. Specifically, the number of possible ways for the paths $(p_j)_{j\in[N]}$ to share queues is given by

$$\sum_{A\subseteq[N]:A\neq\emptyset}\prod_{j\in A} K_j, \tag{12}$$

as each nonempty subset of paths $\{p_j\}_{j\in A}$ may share a queue, and this shared queue can be any of the K_j queues on each p_j. For $K := \max_{j\in[N]} K_j$ and $K' := \min_{j\in[N]} K_j$, we have

$$(K')^N \leq \quad (12) \leq (2K)^N. \tag{13}$$

Since each possible shared queue may or may not appear in the inferred set of shared queues Q, the number of possible solutions for Q is between $2^{(K')^N}$ and $2^{(2K)^N}$, which is super-exponential in the number of measurement paths N. This observation motivates us to seek more efficient algorithms.

C. Algorithm Design

Algorithm 4 Fingerprint-Aware Neighbor Joining

input : Number of destinations N, estimated parameters for each path $(\boldsymbol{\delta}_j)_{j\in[N]}$ and the sample sizes for such estimation $(n_j)_{j\in[N]}$, estimated shared path lengths $(\rho_{ij})_{i,j\in[N]}$ for each pair of paths and the sample sizes for such estimation $(n_{ij})_{i,j\in[N]}$

output: The set of shared queues Q

$D\leftarrow[N]$; ▷ vertices under consideration
$b\leftarrow N+1$; ▷ next branching point
$Q\leftarrow\emptyset$; ▷ initial shared queues
foreach $j\in[N]$ **do**
  foreach $k\in[K_j]$ **do**
    $\tilde{q}_{j,k}\leftarrow\{q_{j,k}\}$;
    $Q\leftarrow Q\cup\{\tilde{q}_{j,k}\}$;
while $|D|>1$ **do**
  $(i,j) = \operatorname{argmax}_{i',j'\in D:i'\neq j'}\rho_{i'j'}$;
  sort elements of $[K_i]\times[K_j]$ into $\left((k_l,k'_l)\right)_{l=1}^{K_iK_j}$ in increasing order of $|\delta_{i,k_l}-\delta_{j,k'_l}|$;
  $K_b\leftarrow$ $\operatorname{argmin}_J\left|\sum_{(k_l,k'_l)\in Q_b(J)}\frac{(n_i+n_j)^2}{(n_i\delta_{i,k_l}+n_j\delta_{j,k'_l})^2} - \rho_{ij}\right|$, where $Q_b(J)$ contains the first J pairs of (k_l,k'_l)'s that form a matching;
  $D\leftarrow D\cup\{b\}\setminus\{i,j\}$;
  foreach $v\in D\setminus\{b\}$ **do**
    $\rho_{vb}\leftarrow\frac{1}{n_{vi}+n_{vj}}(n_{vi}\rho_{vi}+n_{vj}\rho_{vj})$;
    $n_{vb}\leftarrow n_{vi}+n_{vj}$;
  for *the l-th element* $(k_l,k'_l)\in Q_b(K_b)$ **do**
    $\delta_{b,l}\leftarrow\frac{1}{n_i+n_j}(n_i\delta_{i,k_l}+n_j\delta_{j,k'_l})$;
    $\tilde{q}_{b,l}\leftarrow\tilde{q}_{i,k_l}\cup\tilde{q}_{j,k'_l}$;
    $Q\leftarrow Q\cup\{\tilde{q}_{b,l}\}\setminus\{\tilde{q}_{i,k_l},\tilde{q}_{j,k'_l}\}$;
  $n_b\leftarrow n_i+n_j$;
  $b\leftarrow b+1$;

To infer the shared queues efficiently, we leverage the approach of *neighbor-joining*, which is a bottom-up approach

widely used in inferring trees from distance measurements [9], [30]. While the original neighbor-joining algorithm for network topology inference [9] only used the distance information, we jointly use the distances in $\boldsymbol{\rho}$ and the fingerprints in $\boldsymbol{\delta}$ to improve the accuracy. The proposed algorithm, called *Fingerprint-aware Neighbor Joining (FNJ)*, is shown in Algorithm 4. For ease of presentation, we use p_v to denote the path from the root to vertex v in the tree (p_v is one of the measurement paths $\{p_j\}_{j\in[N]}$ if v is a leaf), and K_v to denote the number of queues (i.e., vertices) on p_v. We use $\tilde{q}_{v,k}$ to denote the k-th shared queue on p_v according to an arbitrary order. We use ρ_{vb} to denote the estimated shared path length between p_v and p_b according to the delay-based metric in Section V-A, and $\delta_{v,k}$ to denote the estimated parameter of the shared queue $\tilde{q}_{v,k}$. We further use n_v to denote the number of measurements used for parameter estimation for p_v, and n_{vb} to denote the number of measurements used for estimating the shared path length ρ_{vb}.

FNJ follows the basic procedure of neighbor joining, i.e., it uses a set D to track the vertices the algorithm is trying to connect (initially the set of leaves), joins the two vertices in D with the longest shared path from the root, and then uses the nearest common ancestor of the two vertices to replace them in D, which leads to a recursive algorithm that constructs a tree from the leaves to the root. However, it differs from the original neighbor-joining algorithm [9] in the use of estimated queue parameters as fingerprints to place all the queues onto the tree. To this end, it initially treats each δ-parameter as representing a distinct queue (lines 4– 4), and then iteratively detects the parameters associated with the same queue by considering two paths at a time (lines 4– 4). Each iteration considers the pair of vertices (i, j) in D with the longest shared path (line 4), and identifies the shared queues between p_i and p_j using both the fingerprints and the shared path length (lines 4– 4). It then updates the state variables in preparation for the next iteration, including replacing vertices $\{i, j\}$ in D by a newly constructed vertex b representing their nearest common ancestor (line 4), updating the shared path lengths between the new vertex and every other vertex in D (lines 4– 4), and estimating the parameter of each newly identified shared queue (line 4) while recording these shared queues (lines 4– 4). The weighted average in lines 4 and 4 is designed to better mitigate estimation error during aggregation, by weighing each estimate by the number of samples used to obtain it. The output of FNJ can then be used as input for Algorithm 3 to construct the inferred topology.

Remark 4: In lines 4– 4, FNJ identifies the shared queues between p_i and p_j by (i) prioritizing candidate pairings between the queues on these paths based on the similarity of their fingerprints (line 4), and (ii) finding a feasible subset of K_b pairings that lead to a delay-based metric best approximating the estimated shared path length between p_i and p_j (line 4). Here, "feasibility" means that no queue on p_i or p_j is paired with more than one queue on the other path, i.e., the selected pairings form a matching.

D. Performance Analysis

Complexity: The memory consumption by Algorithm 4 is mainly for storing the set of shared queues Q, the estimated shared path lengths $\boldsymbol{\rho}$, the estimated parameters $\boldsymbol{\delta}$, and the sorted result of $[K_i] \times [K_j]$ in line 4. The sizes of these variables are bounded by $O(NK)$ for Q (recall that $K := \max_{j\in[N]} K_j$ is the maximum number of queues per path), $O(N^2)$ for $\boldsymbol{\rho}$, $O(NK)$ for $\boldsymbol{\delta}$, and $O(K^2)$ for $[K_i]\times[K_j]$. Thus, the space complexity of Algorithm 4 is $O(N^2 + NK + K^2)$. The running time of Algorithm 4 is dominated by the loop in lines 4– 4. The loop will be repeated $N - 1$ times as each loop will reduce $|D|$ by one. In each loop, line 4 takes $O(N^2)$ time as there are $O(N^2)$ pairs of vertices in D, line 4 takes $O(K^2 \log K)$ time to sort K^2 elements, and the other steps are subsumed by these steps in terms of time. Thus, the time complexity of Algorithm 4 is $O(N^3 + NK^2 \log K)$.

Correctness: We define the following notations for the purpose of analysis. We will refer to vertices i and j in a tree as *generalized siblings* if there is no branching point between them and their nearest common ancestor, e.g., q_1 and d_4 in Fig. 5a are generalized siblings but d_2 and d_4 are not. We will use $\tilde{D}$ to denote the set of vertices in the ground truth topology $\mathcal{T}$ that are either leaves or branching points, and Q_{ij} $(i, j \in \tilde{D})$ to denote the set of index pairs for the queues shared between p_i and p_j (recall that p_v denotes the root-to-v path), i.e., $\tilde{q}_{i,k}$ and $\tilde{q}_{j,k'}$ represent the same queue if and only if $(k, k') \in Q_{ij}$. We will use ρ^*_{ij} to denote the true value of the estimated shared path length ρ_{ij} between paths p_i and p_j, and $\delta^*_{i,k}$ to denote the true value of the estimated parameter $\delta_{i,k}$ of the k-th queue on path p_i. With these notations, we will establish a set of sufficient conditions under which FNJ correctly infers the set of shared queues, and thus correctly infers the queueing network topology with the help of Algorithm 3 (see proof in Appendix A.4 in the Supplementary Material).

Theorem 4: Let $\widetilde{\Delta}$ be the minimum length (measured by the delay-based metric) of any branch in $\mathcal{T}$, and Δ^*_{ij} be the minimum difference between the parameters of any two queues in $p_i \cup p_j$ $(\forall i, j \in \tilde{D})$. Then FNJ (Algorithm 4) correctly infers all the shared queues if the following conditions hold:

1) $|\rho_{ij} - \rho^*_{ij}| < \widetilde{\Delta}/2$, $\forall i, j \in \tilde{D}$;
2) $|\delta_{i,k} - \delta^*_{i,k}| < \Delta^*_{ij}/4$ and $|\delta_{j,k'} - \delta^*_{j,k'}| < \Delta^*_{ij}/4$, $\forall i, j \in \tilde{D}$ that are generalized siblings and $\forall k \in [K_i], k' \in [K_j]$;
3) $\exists \delta_0 > 0$ such that $\forall i, j \in \tilde{D}$ that are generalized siblings and $\forall k \in [K_i]$, $k' \in [K_j]$,

$$\delta^*_{i,k}, \delta^*_{j,k'} \in \left[\delta_0 + \frac{\Delta^*_{ij}}{4}, \left((2|Q_{ij}| + 1) f\left(\delta_0, \frac{\Delta^*_{ij}}{4}\right) + \widetilde{\Delta}\right)^{-1/2}\right], \tag{14}$$

where $f(x, y) := 1/x^2 - 1/(x + y)^2$.

Although the conditions in Theorem 4 depend on some internal variables of FNJ (e.g., ρ_{ij} and $\delta_{i,k}$ for a branching point i), we can simplify them into conditions only depending on the input parameters (see proof in Appendix A.5 in the Supplementary Material).

Corollary 1: Let $\widetilde{\Delta}$ and $f(x, y)$ be defined as in Theorem 4, and $\Delta^\dagger := \min\{\Delta^*_{ij} : i, j \in \tilde{D}, (i, j) \text{ is a pair of generalized siblings}\}$. Then FNJ correctly infers all the shared queues if the following holds:

1) $|\rho_{ij} - \rho^*_{ij}| < \widetilde{\Delta}/2$, $\forall i, j \in [N]$;
2) $|\delta_{i,k} - \delta^*_{i,k}| < \Delta^\dagger/4$, $\forall i \in [N], k \in [K_i]$;
3) $\exists \delta_0 > 0$ such that $\forall i \in [N], k \in [K_i]$,

$$\delta^*_{i,k} \in \left[\delta_0 + \frac{\Delta^\dagger}{4}, \left((2K - 1) f\left(\delta_0, \frac{\Delta^\dagger}{4}\right) + \widetilde{\Delta}\right)^{-1/2}\right]. \tag{15}$$

TABLE I
STATISTICS IN TOPOLOGY GENERATION FROM AS6461

height K	root s'	#covered nodes	#candidate destinations
3	127	92	81
4	21	151	125
5	61	175	141
6	21	180	139
7	61	182	142

Remark 5: Theorem 4 and Corollary 1 essentially state that FNJ will correctly infer the shared queues if the given shared path lengths and queue fingerprints are sufficiently accurate, and the true parameters are not too large or too small. In comparison, RNJ [9] requires $|\rho_{ij} - \rho^*_{ij}| < \widetilde{\Delta}/4$ ($\forall i, j \in [N]$) to have guaranteed correctness[6],and Algorithm 2 requires $|\delta_{i,k} - \delta^*_{i,k}| < \Delta^*/4$ ($\forall i \in [N], k \in [K_i]$). Comparing these conditions with the conditions in Corollary 1 shows that by leveraging both shared path lengths and queue fingerprints, *FNJ can correctly infer the topology based on lower accuracy of each of these input parameters compared to algorithms that only use one type of information*. Note that $\Delta^* \leq \Delta^\dagger$, as Δ^* is between any two queues in $\mathcal{T}$ and $\Delta^\dagger$ is only between the queues on two paths (to a pair of generalized siblings). Although it appears that FNJ requires more conditions than RNJ or Algorithm 2, we note that these are only sufficient conditions, and will resort to empirical evaluations to compare their actual accuracy.

VI. PERFORMANCE EVALUATION

We evaluate the proposed algorithms against benchmarks via both queueing-theoretic simulations under our network model and packet-level simulations in NS3 [31] that stress-test our model, based on real Internet topology.

A. Simulation Setup

1) Topology Generation: We generate the ground-truth tree topology based on an *Autonomous System (AS)* topology AS6461 from [32], which represents IP-level connections between the routers in Abovenet with 182 nodes and 294 links. Similar results have been obtained under other topologies (see Appendix B.1 in the Supplementary Material). Given a maximum path length (measured in #nodes) of K, we start from a source node s' selected to cover the maximum number of nodes within path length K, and perform a breadth first search to obtain a tree of height K. We then randomly pick N of the leaves of this tree as destinations to form a tree with N leaves, which is used as the ground truth topology $\mathcal{T}$ in one Monte Carlo run. The statistics for each tested value of K is given in Table I. We note that although the constructed topology is technically a routing topology, the corresponding queueing network topology will have the same structure as illustrated in Fig. 1 if s' is treated as the gateway router for the actual source. We can thus interpret the constructed $\mathcal{T}$ as a queueing network topology.

2) Parameter Setting: We set all the link capacities to 1 Gbps according to [33]. Assuming a link utilization of 20–80%, we randomly generate a residual capacity δ_l from $[0.2, 0.8]$ Gbps for each node $q_l \in \mathcal{T}$ (representing its incoming link in the routing topology). For queueing-theoretic simulations, we convert the unit to packets/sec using a packet size of $1,500$ bytes. For NS3 simulations, we randomly draw the sizes of background packets from 80, 606, and $1,500$ bytes with probabilities 0.4, 0.2, and 0.4 according to [34], and set the data packet size to $1,500$ bytes and the probing packet size to 50 bytes (these include the 30-byte header added by NS3 to every packet). In NS3, we collect both passive and active measurements at an interval of 5 ms, which corresponds to a data rate of 2.4 Mbps per path, and a probing rate of 1.52 Mbps per path for $N = 20$ (i.e., a probing load of no more than 3% of the link capacity). By default, we set $K = 4$, $N = 20$, and both #passive measurements per path and #active measurements per path pair to $1,000$, which will take 5 seconds to collect.[7] We will vary these parameters to test their impacts. All the results are averaged over 20 Monte Carlo runs per combination of (K, N).

3) Benchmarks: For estimating the δ-parameters on each path from end-to-end delays, we compare the proposed estimation algorithm (Algorithm 1), based on either absolute or squared error, with the MLE (solved by the Nelder-Mead simplex method [35]) and the two methods proposed by Harris et al. [28] ('Harris 1', 'Harris 2') discussed in Section III-B.

For topology inference, we use the *Rooted Neighbor Joining* algorithm (RNJ) [9] as the benchmark, which is a state-of-the-art topology inference algorithm guaranteed to correctly reconstruct the *multicast tree* from sufficiently many active measurements. The multicast tree, which is a logical topology obtained by merging nodes on the same branch (e.g., Fig. 5c), is a common target of existing topology inference algorithms (see Section I-A). None of the existing solutions can perform topology inference by only using passive measurements or jointly using both active and passive measurements.

A challenge in applying FNJ is that the relationship between the δ-parameters (residual capacities) and the ρ-parameters (delay covariances) often deviates from the ideal relationship given by the model of M/M/1 queue, as the queueing dynamics are not exactly M/M/1 and the parameters are estimated from different measurements. To correct such deviation, we normalize the estimated values of $(\rho_{ij})_{i,j\in[N]}$ with respect to the estimated values of $(\boldsymbol{\delta}_j)_{j\in[N]}$ as follows: (1) replace all the negative ρ-values by 0; (2) normalize all the ρ-values to $[0, 1]$; (3) scale all the ρ-values by the maximum possible delay covariance according to the estimated δ-parameters, calculated by summing $1/\delta_e^2$ for the $K-1$ smallest δ_e values (because in a tree topology of height K, the maximum number of queues shared between any two paths is at most $K-1$).

4) Metrics: We evaluate the accuracy of parameter estimation by the normalized absolute error, defined as $\|\hat{\boldsymbol{\delta}} - \boldsymbol{\delta}^*\|_1/\|\boldsymbol{\delta}^*\|_1$, where $\hat{\boldsymbol{\delta}}$ is the estimate and $\boldsymbol{\delta}^*$ the ground truth.

We evaluate the accuracy of topology inference by a version of graph edit distance [36] that allows merging/splitting nodes. Graph edit distance is a typical performance metric for topology inference algorithms [37]. In our case, a common error is duplicating the same node due to the failure in recognizing that some queues traversed by different paths are the same shared queue, and another common error is incorrectly merging nodes due to mistakenly identifying different queues as a shared

[6]Note that being correct for RNJ only means to correctly infer the multicasttree that ignores the degree-2 vertices.

[7]This refers to the simulated time, which differs from the running time of the simulation. Due to its single-threaded execution, the NS3 simulation often takes much longer than the time period it simulates, as it needs to simulate all the discrete events (packet transmissions/receptions) during this period.

queue. Both types of errors can be captured by the above graph edit distance. As illustrated in Fig. 5, the inferred topology (Fig. 5b) has an edit distance of 1 to the ground truth (Fig. 5a), as merging q_{11} and q_{12} will make it identical to the ground truth. The multicast tree (Fig. 5c) has an edit distance of 3 to the ground truth, requiring d_4 to be split once and d_5 to be split twice.

For conciseness, we only present the results from queueing-theoretic simulations below and defer the results from NS3 simulations to Appendix B in the Supplementary Material, as the observations are similar.

B. Results on Parameter Estimation

We start by evaluating the parameter (i.e., residual capacity) estimation for a single path.

Evaluation of design choices: We first evaluate the design of the points S for fitting the empirical Laplace transform. As [28] suggested the values in S to be evenly distributed, we set S to contain $|S|$ points evenly distributed in $[0, s_{\max}]$, and evaluate the parameter estimation error under various combinations of $|S|$ and $s_{\max}$ as shown in Fig. 6a. The results show that $s_{\max}$ should be large enough so that the Laplace transform is sufficiently described within $[0, s_{\max}]$, but not too large to avoid fitting the noise when the Laplace transform is nearly zero. Meanwhile, $|S|$ should be large enough to provide enough points to fit. Setting $|S|$ too large, however, will unnecessarily increase the running time (as the complexity of evaluating the objective function (4a) is proportional to $|S|$) without further decreasing the estimation error. Based on the results of Fig. 6a, we chose $s_{\max} = 10$ and $|S| = 50$, i.e., $S = \{1, 1.2, 1.4, \ldots, 10\}$. Next, we evaluate the proposed two-step method of solving (4) as in lines 1– 1 of Algorithm 1 ($J = 100$) against only performing the iterative single-variate optimization in lines 1– 1 ('iterative') or directly optimizing (4) from an arbitrary initial value ('direct'). The results, given in Fig. 6b, show that the proposed method significantly improves the estimation accuracy.

Comparison with benchmarks: We compare the proposed estimator ('proposed: absolute/squared') against benchmarks in a variety of settings as in Fig. 7- 8 in terms of both estimation error and running time. We see that: (i) the estimators based on the Laplace transform are much more efficient than MLE as the number of measurements or queues increases; (ii) directly optimizing the parameter of interest (i.e., $\boldsymbol{\delta}$) to fit the Laplace transform as in the proposed solution is much more accurate than first estimating the Laplace transform and then inferring the corresponding parameter as in 'Harris 1' and 'Harris 2'; (iii) which error measure to use in the proposed estimator does not affect its accuracy, although the squared error is faster to optimize than the absolute error. We also see that the proposed estimator converges quickly with #measurements (Fig. 7a), and that it largely maintains the same accuracy as the path length increases (Fig. 8a).

C. Results on Topology Inference

Having validated the proposed estimator, we now evaluate the accuracy of using its outputs in topology inference. We compare the accuracy of the FNJ algorithm proposed in Section V that jointly uses passive and active measurements ('passive&active'), the solution proposed in Section IV that only uses passive measurements ('passive'), and the RNJ algorithm [9] that only uses active measurements ('active'). To have a conservative estimation of the advantage of combining passive and active measurements, we set the design parameters of the other algorithms to their ideal values, even though in practice these ideal values are unknown.[8]

TABLE II
IMPACT OF PROBING RATE (50 BYTES/PROBE, $N = 20$, 1000 ACTIVE MEASUREMENTS PER PATH PAIR)

rate (probes/s)	time (s)	load increase	delay increase (ms)
50	20	0.76%	0.002
100	10	1.52%	0.005
200	5	3.04%	0.011
400	2.5	6.08%	0.026

We first compare the algorithms as the number of passive/active measurements increases, as shown in Fig. 9. The results show that (i) the FNJ algorithm that combines passive and active measurements achieves much better accuracy than both the solution using only passive measurements and the solution using only active measurements; (ii) all the algorithms converge fast (within $1,000$ measurements). We also observe that the accuracy of the passive-only solution does not notably improve after a few passive measurements, even if theoretically its error should converge to zero as the conditions of Theorems 1 and 2 are satisfied here. This is because of the deviation between the actual queueing behavior and the model of independent M/M/1 queues adopted in our analysis, as well as the suboptimality in numerically solving the parameter estimation problem in (4). Meanwhile, as FNJ only uses the estimated parameters to detect the queues shared between two sub-paths (from the root to a pair of generalized siblings) instead of all the root-to-leaf paths, it can achieve better accuracy under the same parameter estimation error.

We then compare the scalability of the algorithms as the width (controlled by N) or height (controlled by K) of the ground truth topology increases, as shown in Fig. 10. We see that (i) the FNJ algorithm ('passive&active') can maintain good accuracy as the tree grows wider (Fig. 10a) or taller (Fig. 10b), while (ii) the passive-only or active-only solution deteriorates quickly with the increase of width (Fig. 10a), and (iii) the width of the tree has a greater impact on the accuracy of topology inference than the height. Qualitatively similar observations have been obtained through packet-level simulations in NS3; see Appendix B in the Supplementary Material.

While augmenting passive measurements with active probing can improve the accuracy of topology inference, the probing will also impact the network. Table II examines such impact in terms of the time to collect enough active measurements as well as the maximum increase in link load and link delay as the probing rate varies. The results show that the amount of active measurements required by our FNJ algorithm can be collected within a reasonably short time with negligible performance impact to data traffic. Meanwhile, our evaluation of the computation time shows that generating the inferred topology from given measurements takes negligible time compared to the time to collect the measurements (see Appendix B.2 in the Supplementary Material).

[8]For 'passive', we set the threshold Δ in Algorithm 2 to $\Delta^*/2$ (Δ^*: minimum difference between the parameters of different queues); for 'active', we set the parameter of RNJ (also denoted as 'Δ' in [9]) to the minimum delay-based metric of a link in the multicast tree.

Although we have assumed a static network state to focus on the one-shot topology inference problem, our proposed algorithms can be combined with change detection techniques to handle dynamic network states to some extent, as detailed in Appendix B.3 in the Supplementary Material. However, the design of algorithms tailored for detecting network state changes from end-to-end measurements remains an open problem that is left to future work.

VII. Conclusion

We revisited a classic problem of inferring a tree topology from end-to-end measurements sent by a single source, through a fundamentally different approach that utilizes the detailed queueing dynamics inside the network. Compared to the existing solutions based on multicast or its approximations, our approach has the advantages that it can utilize passive measurements to reduce probing overhead and recover the physical topology with degree-2 nodes. Our solution consists of (i) a novel estimator that infers the residual capacities of a tandem of queues from end-to-end delays, (ii) an algorithm that uses the estimated residual capacities as fingerprints to identify the queues shared between paths, and (iii) an algorithm that combines the estimated residual capacities and the shared path lengths estimated from active measurements to improve the accuracy in identifying the shared queues. Our solutions were theoretically proved to be asymptotically accurate, and empirically validated to beat the state-of-the-art solution in data-driven simulations based on real Internet topologies. Meanwhile, the proposed algorithms are designed for a fixed network state (including routing topology and queue parameters), and need to be combined with other algorithms, e.g., change detection algorithms that can identify suitable time windows for performing inference and adaptation algorithms that can update the inferred topology while maximally reusing previous results, to keep up with network dynamics, the detailed investigation of which is left to future work.

References

[1] Y. Lin, T. He, and G. Pang, "Queuing network topology inference using passive measurements," in *Proc. IFIP Netw. Conf. (IFIP Netw.)*, Jun. 2021, pp. 1–9.

[2] T. He, L. Ma, A. Swami, and D. Towsley, *Network Tomography: Identifiability, Measurement Design, and Network State Inference*. Cambridge, U.K.: Cambridge Univ. Press, 2021.

[3] B. Yao, R. Viswanathan, F. Chang, and D. Waddington, "Topology inference in the presence of anonymous routers," in *Proc. 22nd Annu. Joint Conf. IEEE Comput. Commun. Societies (IEEE INFOCOM)*, vol. 1, Aug. 2003, pp. 353–363.

[4] M. H. Gunes and K. Sarac, "Resolving anonymous routers in Internet topology measurement studies," in *Proc. 27th Conf. Comput. Commun. (IEEE INFOCOM)*, Apr. 2008, pp. 1076–1084.

[5] R. Caceres, N. G. Duffield, J. Horowitz, F. Lo Presti, and D. Towsley, "Loss-based inference of multicast network topology," in *Proc. 38th IEEE Conf. Decis. Control*, vol. 3, Jul. 1999, pp. 3065–3070.

[6] 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.

[7] 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.

[8] N. G. Duffield, J. Horowitz, and F. Lo Prestis, "Adaptive multicast topology inference," in *Proc. Conf. Comput. Communications. 20th Annu. Joint Conf. IEEE Comput. Commun. Soc. (IEEE INFOCOM)*, vol. 3, Sep. 2001, pp. 1636–1645.

[9] 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.

[10] M. Coates, R. Castro, R. Nowak, M. Gadhiok, R. King, and Y. Tsang, "Maximum likelihood network topology identification from edge-based unicast measurements," in *Proc. ACM SIGMETRICS Int. Conf. Meas. Model. Comput. Syst.*, Jun. 2002, pp. 11–20.

[11] H. Yao, S. Jaggi, and M. Chen, "Passive network tomography for erroneous networks: A network coding approach," *IEEE Trans. Inf. Theory*, vol. 58, no. 9, pp. 5922–5940, Sep. 2012.

[12] P. Sattari, M. Kurant, A. Anandkumar, A. Markopoulou, and M. G. Rabbat, "Active learning of multiple source multiple destination topologies," *IEEE Trans. Signal Process.*, vol. 62, no. 8, pp. 1926–1937, Apr. 2014.

[13] M. Rabbat, M. Coates, and R. Nowak, "Multiple source Internet tomography," *IEEE J. Sel. Areas Commun.*, vol. 24, no. 12, pp. 2221–2234, Dec. 2006.

[14] N. Duffield, J. Horowitz, F. L. Presti, and D. Towsley, "Multicast topology inference from end-to-end measurements," *Adv. Perform. Anal.*, vol. 3, pp. 207–226, Jan. 2013.

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

[16] M. Hirabaru, "Impact of bottleneck queue size on TCP protocols and its measurement," *IEICE Trans. Inf. Syst.*, vol. 89, no. 1, pp. 132–138, Jan. 2006.

[17] D. Katabi and C. Blake, "Inferring congestion sharing and path characteristics from packet interarrival times," Mass. Inst. Technol., Cambridge, MA, USA, Tech. Rep. MIT-LCS-TR-828, 2001.

[18] W. Wei, B. Wang, D. Towsley, and J. Kurose, "Model-based identification of dominant congested links," in *Proc. ACM SIGCOMM Conf. Internet Meas. (IMC)*, 2003, pp. 115–128.

[19] F. Baccelli, B. Kauffmann, and D. Veitch, "Inverse problems in queueing theory and Internet probing," *Queueing Syst.*, vol. 63, nos. 1–4, pp. 59–107, Dec. 2009.

[20] F. Pin, D. Veitch, and B. Kauffmann, "Statistical estimation of delays in a multicast tree using accelerated EM," *Queueing Syst.*, vol. 66, no. 4, pp. 369–412, Dec. 2010.

[21] A. Asanjarani, Y. Nazarathy, and P. K. Pollett, "Parameter and state estimation in queues and related stochastic models: A bibliography," 2017, *arXiv:1701.08338*.

[22] J. Kurose and K. Ross, *Computer Networking: A Top-Down Approach*, 7th ed., Hoboken, NJ, USA: Pearson, 2017.

[23] J. F. Shortle, J. M. Thompson, D. Gross, and C. M. Harris, *Fundamentals of Queuing Theory*. Hoboken, NJ, USA, 5th ed., 2018.

[24] Y. Huang, Y. Lin, and T. He, "Optimized cross-path attacks via adversarial reconnaissance," *ACM SIGMETRICS Perform. Eval. Rev.*, vol. 52, no. 1, pp. 51–52, Jun. 2024.

[25] L. Esparza, "Maximum likelihood estimation of phase-type distributions," Ph.D. dissertation, Dept. Inform. Math. Model., Tech. Univ. Denmark, Lyngby, Denmark, 2011.

[26] H. L. Van Trees, *Detection, Estimation, and Modulation Theory*. Hoboken, NJ, USA: Wiley, 2004.

[27] J. Abate and W. Whitt, "Numerical inversion of Laplace transforms of probability distributions," *ORSA J. Comput.*, vol. 7, no. 1, pp. 36–43, Feb. 1995.

[28] C. M. Harris and W. G. Marchal, "Distribution estimation using Laplace transforms," *INFORMS J. Comput.*, vol. 10, no. 4, pp. 448–458, Nov. 1998.

[29] A. V. den Boer and M. Mandjes, "Convergence rates of laplace-transform based estimators," *Bernoulli*, vol. 23, no. 4A, pp. 2533–2557, Nov. 2017.

[30] N. Saitou and M. Nei, "The neighbor-joining method: A new method for reconstructing phylogenetic trees," *Mol. Biol. Evol.*, vol. 4, no. 4, pp. 406–425, 1987.

[31] T. R. Henderson, M. Lacage, G. F. Riley, C. Dowell, and J. Kopena, "Network simulations with the NS-3 simulator," *SIGCOMM Demonstration*, vol. 14, no. 14, p. 527, 2008.

[32] N. Spring, R. Mahajan, and D. Wetherall, "Measuring ISP topologies with rocketfuel," in *Proc. Conf. Appl., Technol., Architectures, Protocols Comput. Commun.*, Aug. 2002, pp. 133–145.

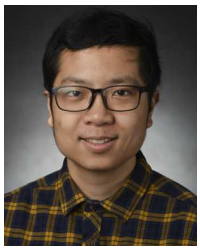
[33] S. Gay, P. Schaus, and S. Vissicchio, "REPETITA: Repeatable experiments for performance evaluation of traffic-engineering algorithms," 2017, *arXiv:1710.08665*.

[34] A. Svigelj, M. Mohorcic, G. Kandus, A. Kos, M. Pustisek, and J. Bester, "Routing in ISL networks considering empirical IP traffic," *IEEE J. Sel. Areas Commun.*, vol. 22, no. 2, pp. 261–272, Feb. 2004.

[35] J. C. Lagarias, J. A. Reeds, M. H. Wright, and P. E. Wright, "Convergence properties of the nelder-mead simplex method in low dimensions," *SIAM J. Optim.*, vol. 9, no. 1, pp. 112–147, Jan. 1998.

[36] P. Bille, "A survey on tree edit distance and related problems," *Theor. Comput. Sci.*, vol. 337, nos. 1–3, pp. 217–239, Jun. 2005.
[37] Y. Lin, T. He, S. Wang, K. Chan, and S. Pasteris, "Looking glass of NFV: Inferring the structure and state of NFV network from external observations," *IEEE/ACM Trans. Netw.*, vol. 28, no. 4, pp. 1477–1490, Aug. 2020.
[38] O. A. Grigg, V. T. Farewell, and D. J. Spiegelhalter, "Use of risk-adjusted CUSUM and RSPRTcharts for monitoring in medical contexts," *Stat. Methods Med. Res.*, vol. 12, no. 2, pp. 147–170, Apr. 2003.
[39] T. Benson, A. Akella, and D. A. Maltz, "Network traffic characteristics of data centers in the wild," in *Proc. 10th ACM SIGCOMM Conf. Internet Meas.*, Nov. 2010, pp. 267–280.

Akash Kumar (Student Member, IEEE) received the B.S. and M.S. degrees in computer science from West Chester University. He is currently pursuing the Ph.D. degree in computer science and engineering with The Pennsylvania State University, advised by Dr. Ting He. His research interests include computer networking, network topology inference, network management, and machine learning.



Yudi Huang received the B.Eng. and M.Eng. degrees in communication and information engineering from the University of Electronic Science and Technology of China and the Ph.D. degree in computer science and engineering from The Pennsylvania State University. He is currently a Senior Software Engineer with Nvidia. His interests include computer networking, wireless communication, and machine learning.

Ting He (Senior Member, IEEE) received the Ph.D. degree in ECE from Cornell University. She is currently an Associate Professor with the School of EECS, The Pennsylvania State University, University Park, PA, USA. Her research interests include computer networking, performance evaluation, and machine learning. She received multiple paper awards from IEEE Communications Society, ICDCS, SIGMETRICS, ICASSP, IMC, and SmartGridComm. She served as an Associate Editor for IEEE TRANSACTIONS ON COMMUNICATIONS and IEEE/ACM TRANSACTIONS ON NETWORKING, the General Co-Chair of IEEE RTCSA, the TPC Co-Chair of ACM MobiHoc and IEEE ICCCN, and the Area TPC Chair of IEEE INFOCOM.