Netscope: Practical Network Loss Tomography

Denisa Ghita, Hung Nguyen*, Maciej Kurant, Katerina Argyraki, and Patrick Thiran
EPFL, Switzerland

***Abstract*—We present Netscope, a tomographic technique that infers the loss rates of network links from unicast end-to-end measurements. Netscope uses a novel combination of first- and second-order moments of end-to-end measurements to identify and characterize the links that cannot be (accurately) characterized through existing practical tomographic techniques. Using both analytical and experimental tools, we show that Netscope enables scalable, accurate link-loss inference: in a simulation scenario involving 4000 links, 20% of them lossy, Netscope correctly identifies 94% of the lossy links with a false positive rate of 16%—a significant improvement over the existing alternatives. Netscope is robust in the sense that it requires no parameter tuning, moreover its advantage over the alternatives widens when the number of lossy links increases. We also validate Netscope's performance on an "Internet tomographer" that we deployed on an overlay of 400 PlanetLab nodes.**

I. Introduction

Inferring the characteristics of individual network links from end-to-end path measurements can be a powerful tool for network troubleshooting: Network operators can use it to identify faulty or congested links without having to monitor every single link in their network. In a different context, an overlay of Internet end-hosts can use such inference techniques to identify where their packets are getting lost or delayed, which enables them to evaluate the performance of their providers, even draw a map of Internet congestion status.

Network tomography formulates the problem of inferring link characteristics from end-to-end path measurements as a system of linear equations of the form

$$\bar{\mathbf{Y}} = \mathbf{R} \cdot \bar{\mathbf{X}} \tag{1}$$

where $\bar{\mathbf{Y}}$ is the vector of available measurements, $\bar{\mathbf{X}}$ is the vector of unknowns to be estimated (e.g., the link delays or the logarithm of the link loss rates), and $\mathbf{R}$ is the *routing matrix*, which specifies the links included in each path [17]. The amount of information we can get from Eq. 1 depends on the properties of the routing matrix: If $\mathbf{R}$ is full column rank, we can solve Eq. 1 and obtain $\bar{\mathbf{X}}$. If $\mathbf{R}$ is rank deficient, then there are many different $\bar{\mathbf{X}}$'s that satisfy Eq. 1—which means that we need additional information to identify the real $\bar{\mathbf{X}}$.

Unfortunately, routing matrices are always [1] rank-deficient [5], [12], which means that we need additional information to solve Eq. 1. Researchers have suggested various ways of getting this additional information: One approach is to assume strong temporal correlation between the measurement probes, achievable in a multicast environment [1], [4] or with back-to-back probes that emulate multicast [3], [7], [8], [16]. Another approach is to turn Eq. 1 into an optimization problem: of all the possible $\bar{\mathbf{X}}$'s that satisfy Eq. 1, pick the one that meets certain practical constraint, for example, includes the least number of congested links [9], [14], [15]. A third approach (valid for link loss inference) is to first compute the *variances* of link loss rates (which are easier to compute than the loss rates themselves), then use this information to identify links with negligible loss rate, thereby reducing the number of unknowns in Eq. 1 [13].

In this paper, we present Netscope, a link-loss inference technique that uses a new approach. Our focus is to be practical: we want to build an actual "Internet tomographer," i.e., a system that runs on an overlay of Internet hosts and infers the loss rates of the links between them. This leads us away from the (real or emulated) multicast approach. We do use elements from both the second and third approaches outlined above, but combine them with a new algorithm in a way that achieves significantly higher accuracy.

The gist of our approach is the following: Consider a network of n links. Suppose we identify a subset of these links that are *unlikely* to be lossy. We choose k of these links and approximate their loss rates with zero, reducing the number of unknowns in Eq. 1 by k. If k is sufficiently large, Eq. 1 becomes solvable, and we can use it to obtain the loss rates of the remaining $n-k$ links. On the other hand, if k is too large, then we end up approximating the loss rates of too many links with 0, and our inference becomes inaccurate (which, as we will see, is precisely the flaw of the technique proposed in [13]). We design an efficient algorithm that identifies the minimum possible k (and optimal set of k links) such that Eq. 1 becomes solvable; as we will show, in practice, our algorithm leads to accurate link-loss inference.

We evaluate Netscope using a combination of analysis and simulation, the latter over topologies collected through PlanetLab. In a simulation scenario involving 4000 links, 20% of them lossy, Netscope correctly identifies 94% of the lossy links (the best alternative achieves 85%) with a false positive rate of 16% (the best alternative achieves 28%)—a significant improvement over the latest link-loss inference techniques [13], [15]. Moreover, our algorithm is at least twice more accurate in terms of identifying the actual link loss rates than the alternatives. Netscope is robust in the sense that it requires no parameter tuning, and its advantage over the alternatives widens when the number of lossy links increases. We also validate Netscope's performance using PlanetLab experiments: we have built a "tomographer" that runs on PlanetLab nodes and infers the loss rates of the links located between them; we use some of the measured paths for inference and others for validation, and show that the results

* Currently at the University of Adelaide, Australia.

[1] Except from the trivial case where all end-hosts are directly interconnected, i.e., there are no routers in the network.

are consistent.

The rest of the paper is organized as follows: we first establish our terminology and notation and state our assumptions (§II); then we describe Netscope (§III) and evaluate its performance (§IV), discuss related work (§V), and conclude (§VI).

II. Setup

A. Terminology

We consider a network that consists of end-hosts and routers, connected over one-way communication links. An end-host can act as a source, a destination, or both; each source can send traffic to multiple destinations. We model this network as a directed graph $\mathcal{G} = (\mathcal{N}, \mathcal{E})$, where the set of directed edges $\mathcal{E}$ represents the one-way communication links, while the set of nodes $\mathcal{N}$ consists of the set of routers $\mathcal{V}$, and the set of end-hosts $\mathcal{H}$. The set of end-hosts and the set of routers cannot overlap $\mathcal{H} \cap \mathcal{V} = \emptyset$. We define a "path" P as a sequence of links starting from a source and ending at a destination; we denote the set of all active paths in our network by $\mathcal{P}$. Our network graph includes only links that participate in at least one active path.

Before applying Netscope on a network $\mathcal{G}$ with active paths $\mathcal{P}$, we perform the following transformation on $\mathcal{G}$: we identify all sets of links that participate in exactly the same paths; since the characteristics of such links cannot be distinguished from one another (from the point of view of an end-to-end measurement tool), we "merge" each such set into one "virtual link."

Given a network $\mathcal{G} = (\mathcal{N}, \mathcal{E})$ and a set of active paths $\mathcal{P}$, we compute the routing matrix $\mathbf{R}$ of dimensions $|\mathcal{P}| \times |\mathcal{E}|$ as follows: $\mathbf{R}_{i,j} = 1$, if path P_i traverses link e_j and $\mathbf{R}_{i,j} = 0$ otherwise. Hence, the i-th row of the routing matrix $\mathbf{R}$ corresponds to path P_i. Analogously, the j-th column of $\mathbf{R}$ corresponds to link e_j. In the rest of the paper, we use the term "link" to refer both to a link $e \in \mathcal{E}$ and the corresponding column of $\mathbf{R}$; hence, when we say that "a set of links are linearly independent," we mean that the columns of $\mathbf{R}$ which correspond to these links are linearly independent. Finally, we should note that all rows and all columns of $\mathbf{R}$ are nonzero—i.e., we consider only active paths.

The rank of the routing matrix $\mathbf{R}$ is the number of linearly independent columns of $\mathbf{R}$ (equivalently, the number of linearly independent links in $\mathcal{E}$). We say that $\mathbf{R}$ is "full column rank" when all its columns (i.e., all links in $\mathcal{E}$) are linearly independent, i.e., $Rank(\mathbf{R}) = |\mathcal{E}|$. We say that $\mathbf{R}$ is "rank deficient" when $Rank(\mathbf{R}) < |\mathcal{E}|$.

B. Assumptions

We make four assumptions. The first three are common in network tomography—they are necessary in order to establish linear relationships between link and path characteristics. The last assumption is inherited from [13].

1. Routing Stability: Netscope works in rounds; within a round, each source sends probes to multiple destinations. We assume that the routing matrix $\mathbf{R}$ does not change throughout a round. In practice, when we detect that a path has changed, we discard it from our measurements.

$\mathcal{G} = (\mathcal{N}, \mathcal{E})$	A network with a set of nodes $\mathcal{N}$ and unidirectional links $\mathcal{E}$
$\mathcal{H}, \mathcal{V}$	The sets of end-hosts, and routers
$v \in \mathcal{V}$	A router
$e \in \mathcal{E}$	A network link
$\mathcal{P}$	The set of active paths
$P \in \mathcal{P}$	An active network path
$\mathbf{R}$	The routing matrix
$\mathbf{R}_{i,j}$	The element of $\mathbf{R}$ in the i-th row and j-th column, i.e., corresponding to path $P_i \in \mathcal{P}$ and edge $e_j \in \mathcal{E}$
$\mathbf{R}^{\mathcal{E}'}$	The matrix whose columns correspond to a subset of links $\mathcal{E}' \subseteq \mathcal{E}$

TABLE I
Defined Symbols

2. Link-Loss Independence: In each round, we send S unicast probes along each path. Let $\widehat{\phi}_{e_k}$ be the random variable giving the fraction of probes from all paths passing through link e_k that have not been lost at that link in the current round; we define the "transmission rate of link e_k" as $\phi_{e_k} = \mathbb{E}[\widehat{\phi}_{e_k}]$ and its "loss rate" as $1 - \phi_{e_k}$. We assume that the random variables $\widehat{\phi}_{e_k}$ are independent. This assumption has been justified in earlier work [9], [14].

3. Loss Uniformity: For each path P_i, let $\widehat{\phi}_i$ be the random variable giving the fraction of the S probes that arrive correctly at the destination in the current snapshot; we define the "transmission rate of path P_i" as $\phi_i = \mathbb{E}[\widehat{\phi}_i]$ and its "loss rate" as $1 - \phi_i$. Let $\widehat{\phi}_{i,e_k}$ be the fraction of these probes that have traversed link e_k successfully; we assume that $\widehat{\phi}_{i,e_k} = \widehat{\phi}_{e_k}$ almost surely (a.s.) for all paths P_i that traverse e_k, i.e., that the fraction of packets lost at link e_k is the same for all paths P_i traversing the link. This assumption has also been justified for a large enough S [13].

Let $Y_i = \log \widehat{\phi}_i$ and $X_k = \log \widehat{\phi}_{e_k}$. We group these in vectors: $\bar{\mathbf{Y}} = [Y_1\ Y_2 \dots\ Y_{|\mathcal{P}|}]^T$ and $\bar{\mathbf{X}} = [X_1\ X_2 \dots\ X_{|\mathcal{E}|}]^T$, where T denotes transposition. Network tomography relies on the assumptions stated above to establish linear relationships between $\bar{\mathbf{X}}$ and $\bar{\mathbf{Y}}$ and derive Eq. 1.

4. Monotonicity of Link-Loss Variance: We assume that the variance of X_k is a non-decreasing function of the corresponding link loss rate $1 - \phi_{e_k}$, i.e., the more congested a link is, the higher the variance of its loss rate over time. This assumption has been shown to hold based on Internet measurements [18], [19], including recent PlanetLab experiments with more than two million samples of path loss rates [13].

C. Background

Consider a subset of links $\mathcal{B} \subseteq \mathcal{E}$, which contains exactly $|\mathcal{B}| = Rank(\mathbf{R})$ links and all of them are linearly independent. Such a subset is called a "basis" of $\mathcal{E}$. If we partition $\mathcal{E}$ into two sets, $\mathcal{B}$ and $\mathcal{F} = \mathcal{E} \setminus \mathcal{B}$, and somehow obtain the loss rates of the links in $\mathcal{F}$, then we can plug these into Eq. 1 and compute the loss rates of the links in $\mathcal{B}$ by solving

$$\bar{\mathbf{Y}} - \mathbf{R}^{\mathcal{F}} \bar{\mathbf{X}}^{\mathcal{F}} = \mathbf{R}^{\mathcal{B}} \bar{\mathbf{X}}^{\mathcal{B}} \tag{2}$$

where $\bar{\mathbf{Y}}$ is the vector of path loss rates, $\mathbf{R}^{\mathcal{F}}$ is a matrix whose columns correspond to the links in $\mathcal{F}$, $\bar{\mathbf{X}}^{\mathcal{F}}$ is the vector of the loss rates of the links in $\mathcal{F}$, $\mathbf{R}^{\mathcal{B}}$ is a matrix whose columns correspond to the links in $\mathcal{B}$, and $\bar{\mathbf{X}}^{\mathcal{B}}$ is the vector of the (unknown) loss rates of the links in $\mathcal{B}$. Unlike Eq. 1, Eq. 2 can be easily solved since $\mathbf{R}^{\mathcal{B}}$ is full column rank, i.e., it consists of linearly independent columns. So, $|\mathcal{B}| = Rank(\mathbf{R})$ is the maximum number of links whose loss rates we can obtain directly from solving Eq. 2.

A key point for our work is that there exist multiple bases $\mathcal{B}$ for $\mathcal{E}$, and we can solve Eq. 2 for any of them. I.e., whichever basis $\mathcal{B}$ we choose, if we can somehow obtain the loss rates of the remaining links $\mathcal{E} \setminus \mathcal{B}$, then we can use Eq. 2 to compute the loss rates of the links in $\mathcal{B}$.

III. NETSCOPE

A. Overview

Netscope works in rounds: in each round, it collects a set of unicast end-to-end loss measurements $\bar{\mathbf{Y}}$ from a network $\mathcal{G} = (\mathcal{N}, \mathcal{E})$ and outputs estimates for the loss rate of each link in $\mathcal{E}$. It consists of three steps: (i) It computes an approximate *ordering* of the links in $\mathcal{E}$ by loss rate, as proposed in [13]. (ii) It partitions $\mathcal{E}$ into a basis $\mathcal{B}$ and another set $\mathcal{F} = \mathcal{E} \setminus \mathcal{B}$, such that $\mathcal{B}$ includes the lossiest links. This second step is the main contribution of this paper. (iii) It approximates the loss rates of the links in $\mathcal{F}$ with 0, plugs them in Eq. 2 and computes the loss rates of the links in $\mathcal{B}$. We call the links in $\mathcal{F}$ "frozen" links, because we need to determine ("freeze") their loss rates rather than compute them directly from end-to-end measurements.

The intuition is simple: Suppose we are told which are the lossy links. If we can find a basis $\mathcal{B}$ that includes all of them, we can obtain their loss rates by solving Eq. 2. If such a basis does not exist, we can try at least to find the basis $\mathcal{B}$ that includes the lossiest links; if the lossy links that are left *outside* $\mathcal{B}$ are few and not very lossy, then we can approximate their loss rates with 0 and obtain the loss rates of the links in $\mathcal{B}$ by solving Eq. 2. This is precisely what Netscope does, with the difference that it cannot know a-priori exactly which are the lossy links; it can only obtain an approximate ordering of the links by loss rate, as provided by [13].

We now describe each step in more detail.

B. Ordering Links by Loss Rate

We have said that, during each round, we collect a vector of path-loss measurements $\bar{\mathbf{Y}}$. It has been shown that, by processing the $\bar{\mathbf{Y}}$'s collected over multiple rounds, we can compute the variance of the loss rates of all the links in $\mathcal{E}$ [13]. According to our 4th assumption (§II-B), the less lossy a link, the lowest its loss variance; hence, if we obtain an ordering of the links by loss variance, we also have an (approximate) ordering of the links by loss rate. So, in each round, we process the end-to-end measurements collected over all previous rounds, identify the loss variances of all the links, and obtain an ordering of the links, $\mathcal{O}_{\mathcal{E}} = \langle e_1, e_2, ...e_{|\mathcal{E}|} \rangle$, by decreasing loss variance—i.e., e_i has higher variance than $e_{j>i}$. If our 4th assumption holds, then this is also an approximate ordering of the links by loss rate.

Algorithm 1 Netscope Algorithm ($\mathbf{R}$, $\bar{\mathbf{Y}}$)

1: Compute an approximate ordering, $\mathcal{O}_{\mathcal{E}}$, of the links in $\mathcal{E}$ by decreasing loss rate
2: Partition $\mathcal{E}$ into $\mathcal{B}$ and $\mathcal{F}$ such that $|\mathcal{B} \cap \hat{\mathcal{O}}|$ is maximized, where $\hat{\mathcal{O}}$ is any prefix of $\mathcal{O}_{\mathcal{E}}$
3: Set $\bar{\mathbf{X}}^{\mathcal{F}}$ to 0
4: Compute $\bar{\mathbf{X}}^{\mathcal{B}}$ by minimizing $\|\bar{\mathbf{Y}} - \mathbf{R}^{\mathcal{B}}\bar{\mathbf{X}}^{\mathcal{B}}\| + w\|\bar{\mathbf{X}}^{\mathcal{B}}\|$
5: **return** $\bar{\mathbf{X}}^{\mathcal{B}}$

C. Choosing The Optimal Basis

Once we have obtained $\mathcal{O}_{\mathcal{E}}$, we partition $\mathcal{E}$ into a basis set $\mathcal{B}$ and another set $\mathcal{F}$, such that $\mathcal{B}$ meets the following constraint: given *any prefix* $\hat{\mathcal{O}} = \langle e_1, e_2, ...e_{m \leq |\mathcal{E}|} \rangle$ of the ordering $\mathcal{O}_{\mathcal{E}}$, then $\mathcal{B}$ maximizes $|\mathcal{B} \cap \hat{\mathcal{O}}|$; we call $\mathcal{B}$ the "optimal" basis. Practically speaking, this constraint helps us build a basis that includes the highest ranked (hence, lossiest) links.

To compute $\mathcal{B}$, we need to (i) order the columns of the routing matrix $\mathbf{R}$ according to the ordering $\mathcal{O}_{\mathcal{E}}$ and (ii) apply to $\mathbf{R}$ Gaussian-Jordan elimination [10]. The latter computes a basis of the given matrix by iterating over the columns of the matrix: if the current column is linearly independent of the columns already selected for the basis, it is also selected for the basis; otherwise, it is evicted. As long as we iterate starting from the lossiest link, we construct a basis that meets the above constraint—i.e., from a practical point of view, includes the lossiest links.

D. Inferring Loss Rates

Once we have chosen the optimal basis $\mathcal{B}$, we approximate the loss rates of all the remaining links with 0 and plug these values into Eq. 2. In theory, given that $\mathbf{R}^{\mathcal{B}}$ is full rank, we can directly compute the loss rates of the links in $\mathcal{B}$. Nevertheless, we have approximated the links in $\mathcal{F}$ by 0, which may introduce noise in our measurements. As well, it turns out that $\mathbf{R}^{\mathcal{B}}$, although full rank, tends to be ill-conditioned—which means that, if we just solve Eq. 2, we get inaccurate results.

Instead, we use "L1-norm minimization with non-negativity constraints" [15], i.e., minimize $\|\bar{\mathbf{Y}} - \mathbf{R}^{\mathcal{B}}\bar{\mathbf{X}}^{\mathcal{B}}\| + w\|\bar{\mathbf{X}}^{\mathcal{B}}\|$, where w is a configurable parameter, under the constraint that $\bar{\mathbf{X}}^{\mathcal{B}}$ has non-negative elements. This optimization chooses an $\bar{\mathbf{X}}^{\mathcal{B}}$ that may not exactly satisfy Eq. 2, but minimizes the corresponding error (hence the first term of the objective function) and favors solutions that involve fewer lossy links (hence the second term). We use the default value $w = 0.01$ proposed in [15]—it also worked well for the scenarios we tried (§IV-B).

In the beginning of the paper, we said that routing matrices are rank-deficient, which means that we need additional information (on top of end-to-end measurements) to infer the characteristics of individual links; each different tomographic technique essentially discovers different sources for obtaining this additional information. Of course, every source of

	5%	10%	15%	20%	25%
random	0.072	0.326	0.764	1.502	2.53
edge	0.042	0.308	0.696	1.522	2.75

TABLE II
Percentage of the expected number of linearly dependent links within a set of chosen links. The links are chosen either at random — "random", or links located closer to the end-hosts are preferred — "edge". The number of chosen links varies from 5% to 25% of all network links.

additional information is also a potential source of error. In our case, we use two such sources: One is the approximation of the loss rates of the frozen links with 0. The other one is the norm minimization, which allows us to solve the system of Eq. 2. Each of these "tricks" introduces a certain amount of error in our inference; in the next two sections we quantify this error using a combination of analytical and experimental tools.

IV. Evaluation

A. Scalability Analysis

The main source of error in our inference is approximating the loss rates of the frozen links (the links outside the optimal basis) with 0: If all the lossy links *happen* to form a linearly independent set, then Netscope complements this set with enough non-lossy links to form a basis for $\mathcal{E}$ and (correctly) sets the loss rates of the remaining links to 0. However, if there happen to exist lossy links that are linear combinations of other lossy links, then Netscope inevitably "freezes" these links (leaves them outside the optimal basis) and (incorrectly) sets their loss rates to 0. So, Netscope works best when all the lossy links form a linearly independent set; its performance decreases as the number of linearly dependent lossy links (lossy links that are linear combinations of other lossy links) increases.

The first relevant question, then, is: given a certain network, what is the *maximum* number of linearly dependent lossy links in it? Since these links are the main cause of error in our inference, it is worth looking at how many they can be and how their number changes with network size. We already know half the answer: the maximum number of linearly dependent lossy links is equal to the total number of linearly dependent links in a network, which is always equal to $|\mathcal{E}| - Rank(\mathbf{R})$. To understand how this quantity changes with network size, we studied the scalability properties of $Rank(\mathbf{R})$ and formally derived a lower bound on it (the proof can be found in [12], §3.4):

$$Rank(\mathbf{R}) \geq |\mathcal{E}| - \alpha|\mathcal{V}| \quad (3)$$

where $|\mathcal{E}|$ is the number of links and $|\mathcal{V}|$ the number of routers in the network, and α depends on how paths meet and split at each router (a closed form is provided in [12]). In all the topologies we collected through PlanetLab, α was around 1.2. This bound tells us that there can be no more than α (a couple, if our PlanetLab topologies are representative) of linearly dependent links per router; these are the links that, if lossy, introduce error in our inference.

On the other hand, it is unlikely that *all* these "problematic" links will happen to be lossy. So, the next question is: given a certain network, what is the *expected* number of linearly dependent lossy links in it? The answer depends not only on the particular topology, but also on the type of failures observed in the network in question, hence, we cannot provide a general answer.

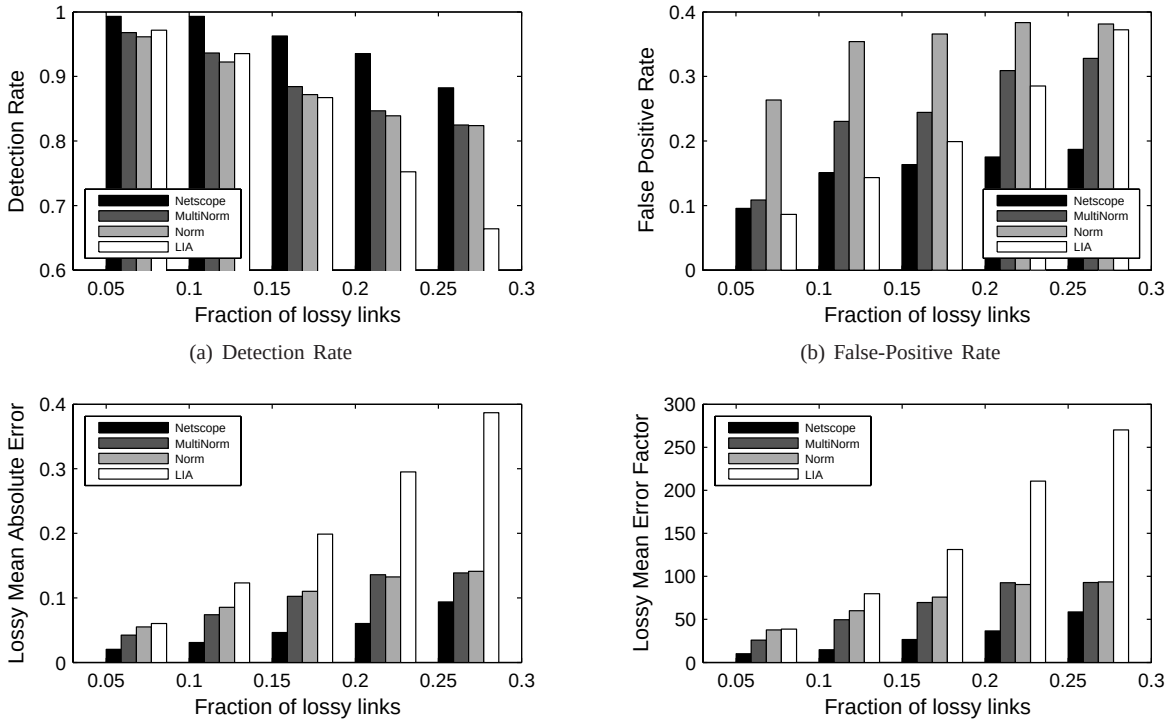
Instead, we ask the following, simpler question: given a particular topology and a particular algorithm for choosing a subset of links in this topology, what is the expected number of linearly dependent links in the chosen subset? We answered this question for our PlanetLab topologies and for two algorithms for choosing links: (i) "random," where all links have the same probability of being chosen, and (ii) "edge," which favors links located closer to the end-hosts (and was inspired by the fact that congestion in the Internet happens typically at the edge of the network). In other words, we computed the fraction of linearly dependent lossy links, given our PlanetLab topologies assuming that congestion occurs (i) at random locations and (ii) toward the edge of the network.

Table II shows the results for a representative 4000-link topology: if we choose a subset of 400 links from this topology, on average 1.2 of these links are linearly dependent on the others, both when all links have the same probability of being chosen, and when links located closer to the end-hosts are favored . In other words, if 10% of the links are lossy, then only 0.3% of these lossy links are linearly dependent, hence introduce error in our inference. Moreover, even when the number of lossy links increases to 25% of the network links, less than 3% of the lossy links are linearly dependent.

B. Simulation Results

Simulator: We first tested Netscope's performance through simulation. To this end, we designed a packet-level, event-driven simulator that works as follows: The network is represented as a graph, where vertices represent nodes and edges represent links. An event corresponds to a packet reaching a node; in response, we determine the outgoing link of the packet and flip a coin to determine whether the packet will be successfully transmitted; if yes, a new event is scheduled that corresponds to the packet reaching the next node. Like Netscope, the simulation works in rounds. In the beginning of each round, each link is assigned a loss rate, which determines the success rate of the coin associated with the link.

On the positive side, our simulator captures the fact that the actual loss rates of paths are, in practice, different from the *measured* loss rates of paths. We capture this, because we measure the loss rate of each path as the fraction of packets successfully received along that path, which is what a real measurement tool like our PlanetLab tomographer (§IV-C) does—and which is precisely why we use a packet-level simulator. On the negative side, we determine which packets get lost through independent Bernoulli processes, which means that we miss the potential inter-dependencies between successive probes. To our defense, this is the standard



(a) Detection Rate

(b) False-Positive Rate

(c) Mean Absolute Error for Lossy Links

(d) Error Factor for Lossy Links

Fig. 1. Performance as a function of the number of lossy links, when lossy links are randomly chosen.

way to evaluate tomographic techniques, and it is expected to be accurate as long as probe inter-arrival times are large enough that the fates of two successive probes are more or less independent from one another.

Setup: We use real Internet topologies, collected in the following manner: we got hold of as many PlanetLab nodes as we could (400 was the maximum) and ran traceroute between them to identify the set of routers on each path; we discarded all paths with incomplete traceroute results. Even though we repeated the above process several times to collect different topologies, they all look similar in terms of in- and out-degree of the nodes and, hence, yield similar results. Thus, we show results that correspond to one topology of 4000 links (the largest we were able to collect) and note that these are consistent with the results we got from all topologies.

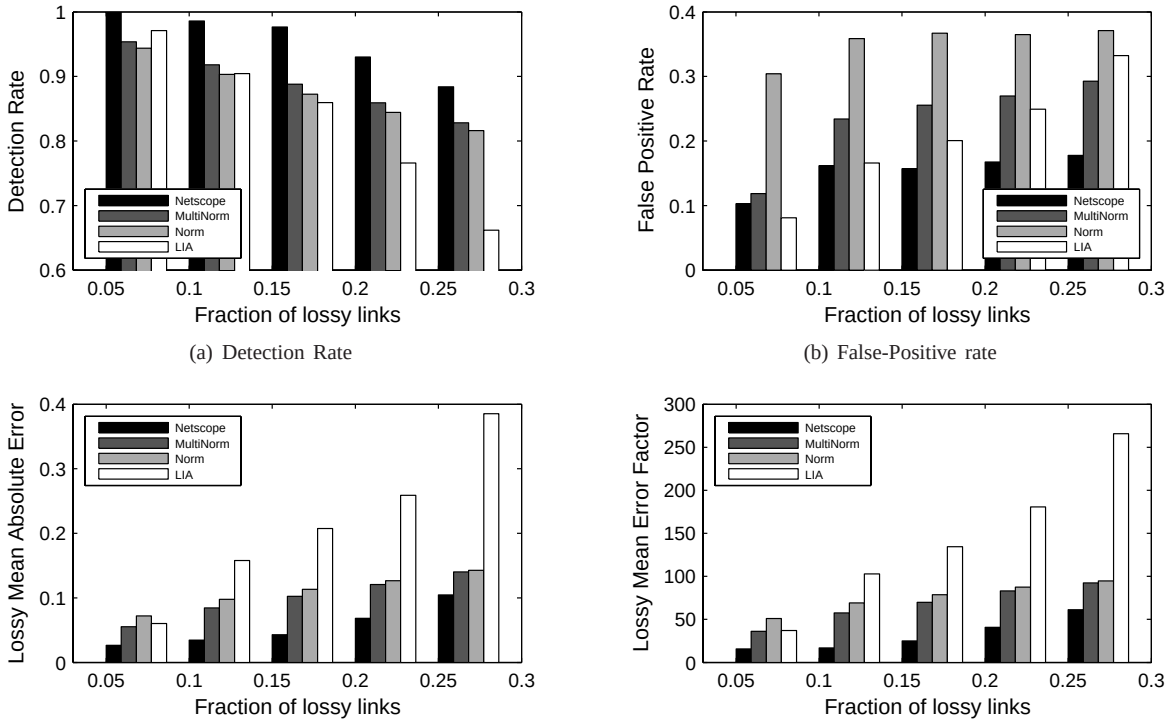
To assign loss rates to links, we use the same loss model with [13] (also similar to the models used in [14], [15]), which assigns loss rates between 0 and 0.002 to non-lossy links and between 0.05 and 1 to lossy links. We chose this over alternative milder models (which assign lower loss rates to the lossy links), because we found that all tomographic techniques we tried did worse with it—hence, we deemed it a better test.

For each test, we choose a topology and the total number of lossy links. Moreover, we need to choose which *particular* links will be lossy: we choose either randomly or based on a "weight" parameter, which specifies how likely a link is to be lossy as a function of its distance from the edge of the network. The latter approach allows us to simulate scenarios where congestion happens mostly close to the end-hosts, which is often the case in the current Internet.

Alternative Solutions: We compare Netscope to three link-loss inference techniques: "Norm" (L1-norm minimization with non-negativity constraints) [15], "MultiNorm," a modified version of Norm described below, and "LIA" [13].

Norm is essentially Netscope's third step without the other two: it simply takes Eq. 1 and solves it using the norm-minimization approach outlined in §III-D. Norm applies the "L1-norm minimization with non-negativity constraints" on all the links in the network, while Netscope applies the minimization only on the links in the optimal basis. A direct comparison between Norm and Netscope would be in some sense unfair, because, unlike Netscope, Norm does not use information from previous rounds; as a result, Norm more often misclassifies a non-lossy link as lossy, hence, has a higher false-positive rate. To make a fair comparison, we introduce "MultiNorm," a modified version of Norm, which (just like Netscope and LIA) uses information from previous rounds: instead of applying L1-norm minimization on *all* the links in the network, it applies it only on links that were lossy in more than T% of the previous rounds. Essentially, MultiNorm tries to enforce a certain amount of "stability" across different rounds, i.e., if L1-norm minimization happens to misclassify a good link as lossy, MultiNorm corrects the mistake, as long as it is infrequent. As expected, MultiNorm's performance depends on the threshold T. We found that $T = 75\%$ gave good results in all our simulation scenarios, hence we show results obtained with this threshold.

LIA shares the same first and third steps (§III) with



(a) Detection Rate

(b) False-Positive rate

(c) Mean Absolute Error for Lossy Links

(d) Mean Error factor for Lossy Links

Fig. 2. Performance as a function of the number of lossy links, when lossy links are located closer to the end-hosts.

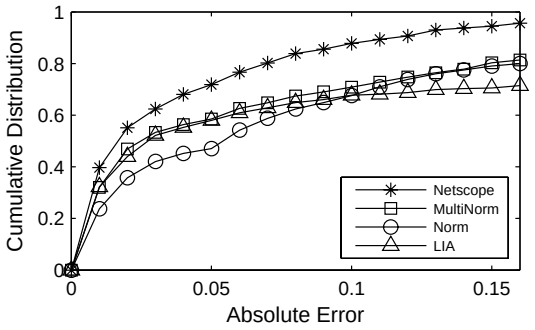
Netscope. In the second step, it partitions $\mathcal{E}$ into two sets, $\mathcal{B}^*$ and $\mathcal{F}^*$, as follows: it goes over the links in $\mathcal{E}$, starting from the lowest-ranked link (according to the ordering obtained in the first step), and greedily removes links until the remaining links form a linearly independent set, or a pre-configured minimum number of links is reached; at that point, all the remaining links are assigned to $\mathcal{B}^*$, while all the removed links are assigned to $\mathcal{F}^*$. Note that this is different from Netscope's second step: Netscope removes links *optimally*, so as to identify a *basis for* $\mathcal{E}$, i.e., a set of $Rank(\mathbf{R})$ linearly independent links. In contrast, LIA removes links *greedily*, until it identifies *any* set of linearly independent links; in many cases, it never identifies such a set and has to stop at an arbitrary point, when the next link to be removed has a loss rate variance above a certain threshold. Therefore, LIA is sensitive to the threshold that specifies when a link has a high loss rate variance such that its loss rate cannot be approximated by 0. As a result, the set $\mathcal{B}^*$ ends up being significantly smaller than $Rank(\mathbf{R})$—which means that LIA freezes significantly more links than necessary.

Netscope is essentially a combination of Norm and LIA plus our optimal-basis selection algorithm (§III-C). Nevertheless, Netscope is robust in the sense that it requires no parameter tuning, unlike MultiNorm which uses a threshold to reduce the number of non-lossy links misclassified as lossy, or LIA, which needs a threshold to determine when a link has a high loss rate variance and cannot be approximated by 0. Hence, comparing Netscope's performance to the one achieved by each of these two techniques alone is essential in quantifying the value of our contribution.

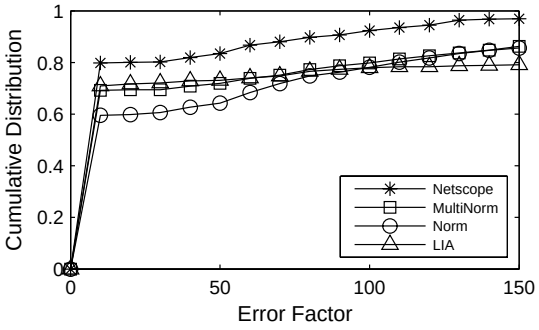
Metrics: We use four metrics: The *detection rate* specifies the fraction of the lossy links that were correctly identified as lossy. The *false positive rate* specifies the fraction of the links identified as lossy that were actually not lossy. For a given link, the *absolute error* is the absolute difference between the link's actual and inferred loss rate; e.g., absolute error 0.1 means that a link has $X\%$ loss and we incorrectly inferred it has $X \pm 10\%$ loss (so, 0.1 can be a significant error). For a given link, the *error factor* is the actual loss rate of the link divided by its inferred loss rate, or the other way around, such that the outcome is always larger than 1 [3]. Note that, in our graphs, we show the mean absolute error (and error factor) *of the lossy links only*; otherwise, given that the number of lossy links is relatively small (5% to 25%) the errors in inferring the loss rates of the lossy links can be diluted.

Random Congestion: We look at the performance of the four techniques for different numbers of lossy links. Fig. 1 shows the results when the lossy links are randomly selected. We make the following observations:

When there are few lossy links, all techniques perform well, with Netscope having a small advantage by all metrics; as the number of lossy links increases, the gap between Netscope's performance and that of the other techniques widens, especially regarding the false-positive rate and the absolute error and error factor. For instance, when 20% of the links are lossy, Netscope detects 94% of the lossy links (MultiNorm, the best alternative, detects 85%) with a false-positive rate of 16% (31% for MultiNorm) and a mean absolute



(a) Cumulative Distribution of the Absolute Errors for the Lossy Links



(b) Cumulative Distribution of the Error Factors for the Lossy Links

Fig. 3. Cumulative distribution function of the absolute errors (the difference between the actual and the inferred loss rates) and of the error factors for the lossy links, when 15% of the network links are lossy.

error of 7.5% for the lossy links (14% for MultiNorm). Our interpretation is the following: Of all the $\bar{\mathbf{X}}$'s that satisfy Eq. 1, Norm chooses one that minimizes the error between the measurements and the link loss rates, and has the fewest lossy links. The larger the number of lossy links in the network, the larger the number of different $\bar{\mathbf{X}}$'s that satisfy Eq. 1, hence, the less likely it is for Norm to pick the right solution without any additional information. MultiNorm achieves better performance than Norm by using information from previous rounds, however, it still suffers from the same flaw as Norm. In contrast, Netscope uses the additional information derived from the variance-based ordering of the links to discard unlikely solutions.

LIA performs well when there are only few lossy links in the network, but its performance degrades as the number of lossy links increases. Among all four techniques, LIA does the worst in terms of identifying the actual link loss rates. This is because of the effect we described above: because of its greedy nature, LIA ends up freezing significantly more links than necessary; as the number of lossy links increases, this unnecessary freezing has a worse impact on performance, because frozen links are more likely to be lossy. These results show that it is not enough to just use the additional information provided by the variance-based ordering, it is also important to use it the right way.

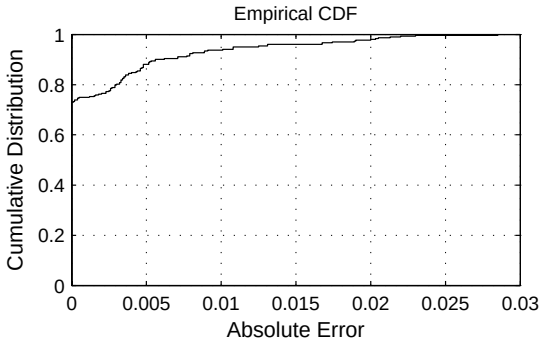


Fig. 4. Cumulative distribution of the validation error for the PlanetLab experiments.

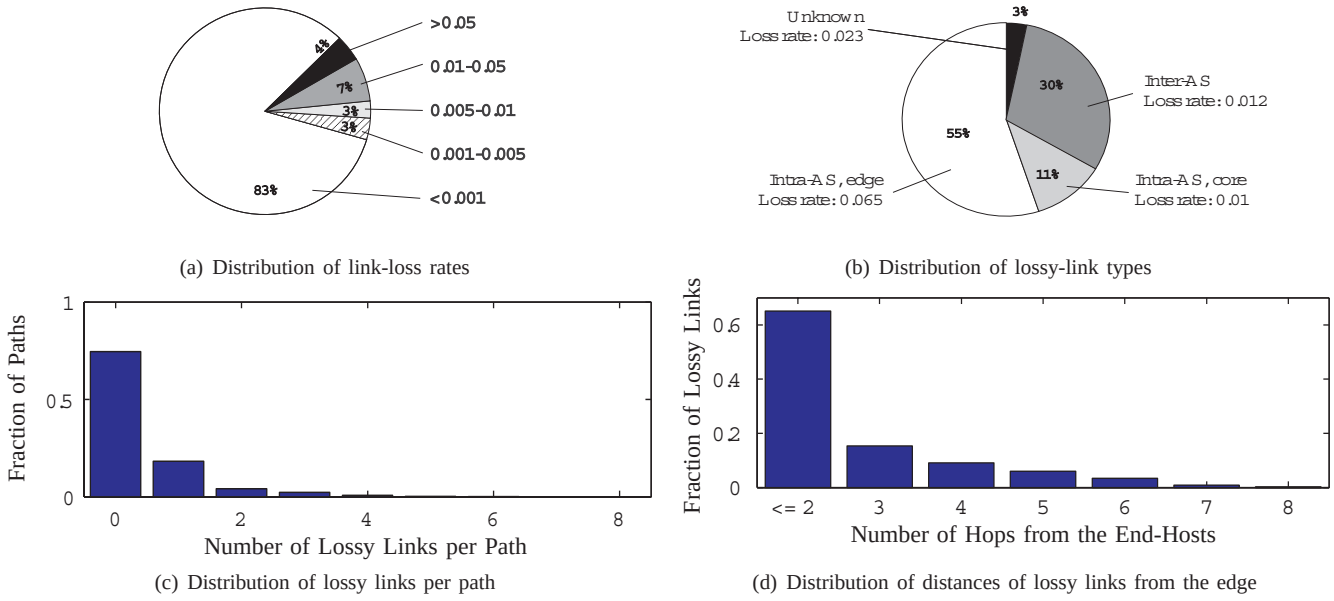
Congestion at the Edge: Next, we look at the performance of the four techniques when the lossy links are located closer to the end-hosts. Fig. 2 shows the performance as the fraction of lossy links in the network increases. We observe that the gap between Netscope and Norm widens when the lossy links are mostly at the network edge—when 10% of the links are lossy, Norm has more than twice Netscope's false-positive rate, three times its mean absolute error, and five times its mean error factor. This may be due to the fact that Norm favors solutions that involve fewer lossy links; such solutions tend to involve links that participate in many paths—hence, are not at the edge of the network. MultiNorm performs better than Norm, but its mean absolute error is still two times higher than the one achieved by Netscope.

Fig. 3 shows the cumulative distribution function of the absolute errors (the difference between the actual and the inferred loss rates) and the error factors for the lossy links, for the particular case where 15% of the links in the network are lossy. With Netscope, 80% of the lossy links have an absolute error of less than 0.06, which means that, if a link has X% loss, we infer that it has a loss in the range $X \pm 6$%, while the best alternative, MultiNorm, infers that it has a loss in the range $X \pm 16$%, so it is 10% worse in identifying the actual link loss rates. Similarly, for 80% of the lossy links Netscope has an error factor of less than 10, while MultiNorm, the best alternative, achieves an error factor of less than 150.

C. Experimental Validation

We built a tomographer that runs on PlanetLab nodes and uses Netscope to infer the loss rates of the links between them. Of course, there is no way to directly measure the accuracy of our tomographer. Instead, we use the indirect validation method proposed in [14]: the paths probed in each round are divided into *inference* paths and *validation* paths, such that each link is represented in both sets; the inference paths are used to infer the loss rates of all the links; these inferred link loss rates are then used to compute the path loss rates for the validation paths. For a given validation path, we use the term "validation error" to refer to the difference between the computed (from inferred link loss rates) and measured path loss rate.

Fig. 4 shows the validation error for different paths. We



(a) Distribution of link-loss rates

(b) Distribution of lossy-link types

(c) Distribution of lossy links per path

(d) Distribution of distances of lossy links from the edge

Fig. 5. Loss statistics. The numbers on the y-axes and the pie percentages are averages over 800 rounds.

observe that 70% of the paths have 0 validation error, while 94% have a validation error below 1%.

D. Loss Characteristics of Internet Links

We close with a summary of the statistics we collected over 7 days using our PlanetLab tomographer; the numbers concern a total of about 4000 links: On average, about 83% of links, in each round, had negligible loss rate, while only 4% had a loss rate above 0.05 (Fig. 5(a)). More than half of the lossy links in each round were edge links; these were also the links with the highest loss rate (Fig. 5(b)). About 18% of paths had one lossy link, while less than 5% had 2 or more lossy links (Fig. 5(c)). Finally, about 62% of lossy links were one or two hops away from the edge, while less than 5% were more than 5 hops away (Fig. 5(d)). We state these without further discussion, as an in-depth analysis of Internet loss characteristics is outside the scope of this paper.

V. Related Work

We looked at the problem of inferring link loss rates from end-to-end path measurements using the linear system specified by Eq. 1. Researchers have already proposed different approaches to this problem: The first proposals relied on multicast probes: they showed that the assumptions we can make on the temporal correlation between such probes allow us to solve Eq. 1 [1], [3], [4]. Later proposals "emulated" multicast probes with "clusters" of back-to-back unicast probes; these methods removed the need for multicast deployment at the cost of somewhat lower accuracy and higher administrative costs [7], [8], [16]. Other work showed how to identify not the loss rates of links, but which are the lossy links [9] or the loss rates of *sequences* of consecutive links—in particular, those sequences whose loss rates can be readily computed by solving Eq. 1 [20].

Our work relies on two more recent proposals, which identify the loss rates of links without using multicast or back-to-back probes. The first one turns Eq. 1 into an optimization problem, in particular it chooses the $\bar{\mathbf{X}}$ that minimizes $\|\bar{\mathbf{Y}}-\mathbf{R}\bar{\mathbf{X}}\|+w\|\bar{\mathbf{X}}\|$. This was first proposed in [14] and later evaluated against (and shown to be better than) similar techniques that use different objective functions [15]. The second proposal first infers the *variance* of the loss rates of links (rather than the loss rates themselves), then uses this information to identify links that are unlikely to be lossy and approximates their loss rates with 0 [13].

We use elements from both of these techniques: Netscope's first step consists of computing the variances of the loss rates of links, exactly as proposed in [13]; on the other hand, that proposal feeds this information into a heuristic, whereas we feed it into an optimal algorithm, which results in significantly higher performance (§IV-B). Netscope's third step involves solving Eq. 2 using precisely the L1-norm minimization with non-negativity constraints proposed in [15]; however, we use this technique *after* having reduced the routing matrix to a full-rank matrix (using the information provided from the first step), which results in higher performance (§IV-B).

We should also mention the Bayesian Inference technique from [14], which iterates over different possible solutions using Markov Chain Monte Carlo sampling, until one that sufficiently matches the observed end-to-end measurements is found. We are trying to use this technique, but we have not managed yet to implement it such that the simulation completes in a reasonable amount of time, given the numbers of links that we consider. However, this technique and Netscope are not mutually exclusive—if computational complexity is not an issue, we expect that combining the two techniques will yield better results than either one alone.

Our tomographer can benefit from techniques that smartly

choose which paths to measure: It has been shown that, in practice, it is possible to approximately characterize all paths by measuring only a small subset of heavily used paths [6]. NetQuest uses Bayesian learning to choose paths to measure so as to obtain the maximum amount of information about the network [15]. VScope uses a "staged" approach: it only measures a subset of paths at any given round so as to meet resource constraints; whenever a congested path is identified, it collects more measurements in order to diagnose that particular path [21].

There also exist non-tomographic techniques for estimating link loss rates, which exploit the responses generated by routers to especially crafted probes [2], [11]. Compared to tomography, these techniques have the advantage of fewer fundamental assumptions and the disadvantage of needing to probe individual routers (as opposed to probing end-to-end paths only); moreover, they require that routers do indeed generate the expected responses to the corresponding probes, e.g., in the case of Tulip, ICMP messages with continuous IP identifiers. We consider tomographic and router-based techniques complementary rather than competing: the former scale better, in the sense that they require probing paths rather than routers; the latter can provide more accurate results regarding links of particular interest at the cost of more probes and time.

VI. Conclusion

We presented Netscope, a tomographic technique that infers the loss rates of links from end-to-end measurements. Netscope gains initial information about the network by computing the variances of the link loss rates, as proposed in [13]. Its novelty lies in the way it uses this information—to identify and characterize the minimum set of links whose loss rates cannot be accurately inferred through standard practical tomography. We showed that combining Netscope with norm minimization [15] significantly improves the latter's performance: In a simulation scenario involving a real Internet topology consisting of 4000 links, 20% of them congested, Netscope achieved a detection rate of 94% and a false positive rate of 16%, whereas norm minimization alone achieved 84% and 38%, respectively. Moreover, Netscope is robust in the sense that it requires no parameter tuning, and its advantage over the alternatives widens when the number of lossy links increases.

We have used Netscope to build a "tomographer" that runs on PlanetLab nodes and infers the loss rates of the links between them; we are currently extending it to cover as large as possible a fraction of the Internet. Our ultimate goal is to turn it into an online tool, constantly analyzing the status of Internet links.

Acknowledgments

This work is financially supported by grant ManCom 2110 of the Hasler Foundation, Bern, Switzerland, and by an Ambizone grant from Swiss National Science Foundation. We would like to thank the anonymous reviewers for their constructive feedback (especially the reviewer whose comments inspired the MultiNorm approach). We are particularly grateful to Adrian Suter, for his help in writing the Netscope code and his fruitful input on the Netscope algorithm.

References

[1] A. Adams, T. Bu, T. Friedman, J. Horowitz, D. Towstey, R. Caceres, N. Duffield, F. L. Presti, S. B. Moon, and V. Paxson. The Use of End-to-end Multicast Measurements for Characterizing Internal Network Behavior. *IEEE Communications Magazine*, May 2000.

[2] K. Anagnostakis, M. Greenwald, and R. Ryger. Cing: Measuring Network Internal Delays Using Only Existing Infrastructure. In *IEEE INFOCOM*, 2003.

[3] T. Bu, N. Duffield, F. L. Presti, and D. Towsley. Network Tomography on General Topologies. In *ACM SIGMETRICS*, 2002.

[4] R. Caceres, N. G. Duffield, J. Horowitz, and D. Towsley. Multicast-based Inference of Network-Internal Loss Characteristics. *IEEE Transactions on Information Theory*, 45:2462–2480, 1999.

[5] Y. Chandramouli and A. Neidhardt. Analysis of network congestion inference techniques. *SIGMETRICS Perform. Eval. Rev.*, 33(3):3–9, 2005.

[6] D. Chua, E. Kolaczyk, and M. A. Crovella. Efficient Monitoring of End-to-End Network Properties. In *IEEE INFOCOM*, 2005.

[7] M. Coates and R. Nowak. Network Loss Inference Using Unicast End-to-End Measurement. In *ITC Seminar on IP Traffic, Measurements and Modelling*, 2000.

[8] N. Duffield, F. L. Presti, V. Paxson, and D. Towsley. Inferring Link Loss Using Striped Unicast Probes. In *IEEE INFOCOM*, 2001.

[9] N. G. Duffield. Network Tomography of Binary Network Performance Characteristics. *IEEE Transactions on Information Theory*, 52(12):5373–5388, Dec. 2006.

[10] G. H. Golub and C. F. V. Loan. *Matrix Computations*. The Johns Hopkins University Press, 1996.

[11] R. Mahajan, N. Spring, D. Wetherall, and T. Anderson. User-level Internet Path Diagnosis. In *ACM SOSP*, 2003.

[12] H. Nguyen, D. Ghita, M. Kurant, K. Argyraki, and P. Thiran. Fundamental Properties of Routing Matrices and Their Implications for Network Tomography. Technical report, EPFL, February 2009. Available (for Infocom 2010 review only) at http://icwww.epfl.ch/~argyraki/RoutingMatrixRank.pdf.

[13] H. X. Nguyen and P. Thiran. Network Loss Inference with Second Order Statistics of End-to-End Flows. In *IEEE IMC*, 2007.

[14] V. N. Padmanabhan, L. Qiu, and H. J. Wang. Server-based Inference of Internet Performance. In *IEEE INFOCOM*, 2003.

[15] H. H. Song, L. Qiu, and Y. Zhang. NetQuest: A Flexible Framework for Large-Scale Network Measurement. In *ACM SIGMETRICS*, 2006.

[16] Y. Tsang, M. Coates, and R. Nowak. Passive Network Tomography Using the EM Algorithms. In *IEEE ICASSP*, 2001.

[17] Y. Vardi. Network Tomography: Estimating Source-Destination Traffic Intensities. *Journal of the American Statistical Association*, 91:365–377, 1996.

[18] V.Paxson. End-to-End Routing Behaviour in the Internet. In *ACM SIGCOMM*, 1996.

[19] Y. Zhang, N. Duffield, V. Paxson, and S. Shenker. On the Constancy of Internet Path Properties. In *ACM SIGCOMM Internet Measurement Workshop*, 2001.

[20] Y. Zhao, Y. Chen, and D. Bindel. Toward Unbiased End-to-End Network Diagnosis. In *ACM SIGCOMM*, 2006.

[21] Y. Zhao, Z. Zhu, Y. Chen, D. Pei, and J. Wang. Towards Efficient Large-Scale VPN Monitoring and Diagnosis under Operational Constraints. In *IEEE INFOCOM*, 2009.