Tomography-based Overlay Network Monitoring

Yan Chen, David Bindel, Hanhee Song, and Randy H. Katz (UC Berkeley)

***Abstract*—**
Overlay network monitoring enables distributed Internet applications to detect and recover from path outages and periods of degraded performance within seconds. For an overlay network with n end hosts, existing systems either require $O(n^2)$ measurements, and thus lack scalability, or can only estimate the latency but not congestion or failures. Unlike other network tomography systems, we characterize end-to-end losses (this can also be extended to latency) rather than individual link losses. We find a minimal basis set of k linearly independent paths that can fully describe all the $O(n^2)$ paths. We selectively monitor and measure the loss rates of these paths, then apply them to estimate the loss rates of all other paths. By extensively studying synthetic and real topologies, we find that for reasonably large n (e.g., 100), k is only in the range of $O(n \log n)$. This is explained by the moderate hierarchical nature of Internet routing.

Our scheme only assumes the knowledge of underlying IP topology, and any link can become lossy or return to normal. In addition, our technique is tolerant to topology measurement inaccuracies, and is adaptive to topology changes such as addition/removal of end hosts and routing changes. Both extensive simulation and experiments on the Internet show that we achieve high path loss rate estimation accuracy with only $O(n \log n)$ measurements. For instance, the average absolute error of loss rate estimation for the Internet experiments is only 0.0027.

Keywords: system design, network measurements, experimentation with real networks, numerical linear algebra.

I. Introduction

With the rapid growth of the Internet, a new class of large-scale globally distributed network services and applications have emerged, such as overlay routing and location systems, application-level multicast, and peer-to-peer file sharing. As these systems have flexibility in choosing their communication paths and targets, they can benefit significantly from dynamic network distance prediction (e.g., latency and loss rate).

Existing network distance estimation systems can be grouped into two categories: *static estimation* [1], [2] and *dynamic monitoring* [3], [4], [5], [6]. Previous works on static estimation, such as Global Network Positioning (GNP) [1], achieve a high level of accuracy, but also incur high overhead for continuously updating the estimates.

On the other hand, dynamic monitoring can detect path outages and periods of degraded performance within seconds. It can both facilitate distributed system management (such as a virtual private network (VPN) or a content distribution network), and help build adaptive overlay applications, like streaming media and multiplayer gaming. For instance, it can enable overlay routing to bypass congestion or failures for live media streaming.

However, existing schemes either require pair-wise measurements for all end hosts, and thus lack scalability [6], or can only estimate latency, but not congestion or failures [3], [4], [5]. The basic idea for the latter category is to cluster end hosts to achieve scalability, based on their network proximity or latency similarity under normal conditions. However that does not imply that the end hosts in the same cluster will have similar losses, especially when the losses happen in the last mile.

In this paper, we design a scalable overlay network congestion/failure monitoring system with high accuracy. Consider an overlay network of n end hosts, we define a path to be a routing path between a pair of end hosts, and a link to be an IP link between routers. A path is a concatenation of links. There are $O(n^2)$ paths among the n end hosts, and we wish to select a minimal subset of paths to monitor so that the loss rates/latency of all other paths can be inferred. The loss rates are used to estimate the congestion/failures on the overlay paths. Besides being accurate, efficient, and scalable, our network monitoring system must also be incrementally deployable.

To this end, we propose a tomography-based overlay network monitoring system in which we selectively monitor k ($k \ll O(n^2)$) paths from which we can infer all path loss rates. We choose the monitored paths to form a *basis* for the full set of paths: that is, any end-to-end path can be written as a unique linear combination of the monitored paths. The end-to-end path loss rates can be computed even when the paths contain *unidentifiable links* for which loss rates cannot be computed. We also provide an intuitive picture of this characterization process in terms of *virtual links*.

Our key observation is that k grows relatively slowly as a function of n. The dimension k is bounded by the number of links in the subgraph induced by the routing paths. In an Internet-like topology with the power-law degree distribution, there are $O(N)$ links, where N is the total number of end hosts in the network. This is because a small number of nodes have high degree and the links between them are heavily used [7]. Consequently, if $n = O(N)$, then $k < O(n)$. However, even when $n \ll N$, the moderately hierarchical structure of the network causes many routing paths to overlap [8], so that the number of links in the routing path subgraph grows much slower than $O(n^2)$. Our extensive study of both synthetic and real Internet topologies suggests that for a randomly selected subset of n end hosts, k grows like $O(n \log n)$ when n is sufficiently large (say 100).

Furthermore, our technique is tolerant to topology measurement inaccuracies, and is adaptive to topology changes. Efficient algorithms are designed to add/remove end hosts and to handle routing changes.

Besides simulating our system with various synthetic and real topologies, we implement our system on the Planet Lab testbed [9]. All results show we achieve high accuracy when estimating path loss rates with $O(n \log n)$ measurements. For example, the average absolute error of loss rate estimation for the Internet experiments is only 0.0027.

The rest of the paper is organized as follows. We survey related work in Sec. II, describe our model and basic theory in Sec. III and present algorithms in Sec. IV. The methodology and results of simulation are described in Sec. V, and those of real Internet experiments are presented in Sec. VI. Finally, we discuss the generalization of our framework in Sec. VII and conclude in Sec. VIII.

II. Related work

Network tomography has been extensively studied ([10] provides a good survey). Most existing systems assume that limited measurement information is given (often in a multicast tree-like structure), and they try to infer the characteristics of the links [11], [12], [13], [14] or shared congestion [15] in the middle of the network.

In many cases, these inferences are limited due to limited measurement and the irregularity of Internet topologies. In contrast, we do not care about the characteristics of *individual* links. Furthermore, we do not have any restriction on the paths to measure. Our goal is to selectively measure a small subset of paths so that we can infer the loss rates of all other paths.

As the closest work to ours, Shavitt *et al.* also use algebraic tools to compute the distances that are not explicitly measured [16]. However, they do not have a path selection problem, and they do not study how to adapt to topology changes. Recently, Ozmutlu *et al.* select a minimal subset of paths to cover all links for monitoring, assuming link-by-link latency is available via end-to-end measurement [17]. Their approach has the following three limitations. 1) Traceroute cannot give accurate link-by-link latency. Many routers in the Internet hide their identities. Besides, traceroute uses the ICMP protocol for measurement, and routers often treat ICMP packet differently from TCP/UDP packets. Therefore, latency data is not representative. 2) It is not applicable for loss rate, because it is difficult to estimate link-by-link loss rates from end-to-end measurements. Loss rate is often more important for applications than latency. 3) It assumes static routing paths and does not consider topology changes.

Many of the previous findings can be leveraged to refine loss rate prediction. For example, [14] finds that the end-to-end losses are dominated by a small number of lossy links. Thus, the path space to be monitored can be reduced to those paths that include lossy links. Consequently, the basis set and the amount of measurement will be reduced. Another similar assumption used in NWS [18] is that losses are primarily due to failures over long hops.

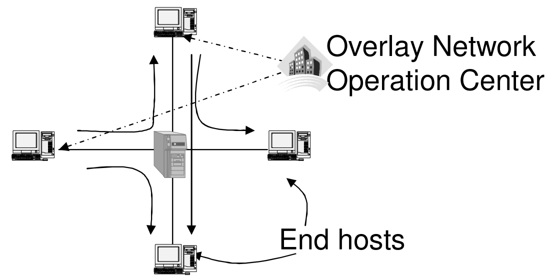


Fig. 1. Architecture of a tomography-based overlay network monitoring system

III. The Model

In this section, we develop the model for tomography-based overlay monitoring.

Given n end hosts to be monitored, we assume that they belong to an overlay network (such as a virtual private network), or that they cooperate to share the monitoring services. Thus, we can measure the routing topology and loss rate of any path. The end hosts are under the control of a central authority (e.g., an overlay network operation center (ONOC)) to measure the topology and loss rates of paths, though in the future we plan to investigate techniques to distribute the work of the central authority.

For simplicity, we mostly assume symmetric routing and undirectional links in the paper. But our techniques work without changes for asymmetric routing, as we did for the Internet experiments. Fig. 1 shows an example where there are four end hosts on the overlay network. There are six paths and four links. The end hosts measure the topology and report to the ONOC, which selects four paths and instruments two of the end hosts to measure the loss rates of the four paths. The end hosts periodically report the loss rates measured to the ONOC. Then the ONOC infers the loss rates of every link, and consequently the loss rates of the other two paths. Applications can query the ONOC for the loss rate of any path, or they can set up triggers to receive alerts when the loss rates of paths of interest exceed a certain threshold. Our technique is applicable to any additive metrics, such as latency.

The path loss rates can be measured by either passive observation of normal traffic to estimate packet drop rate [14] or active measurement. The measurements of selected paths do not have to be taken at exactly the same time because Zhang *et al.* report that the loss rate remains operationally stable in the time scale of an hour [19]. The network topology can be measured via traceroute or other advanced tools [20], [21]. We discuss topology changes in Sec. IV-D.

A. Theory and Notations

Suppose an overlay network spans s IP links. We represent a path by a column vector $v \in \mathbb{R}^s$, where the jth entry v_j is one if link j is part of the path, and zero otherwise. Suppose link j drops packets with probability l_j; then the probability p of packet loss on the path represented by v

Symbols	Meanings
M	total number of nodes
N	number of end hosts
n	number of end hosts on the overlay
$r = O(n^2)$	number of end-to-end paths
s	# of IP links that the overlay spans on
t	number of identifiable links
$G \in R^{r\times s}$	original path matrix
$\bar{G} \in R^{k\times s}$	reduced path matrix
$k \le s$	rank of G
l_i	loss rate on ith link
p_i	loss rate on ith measurement path
x_i	$\log(1-l_i)$
b_i	$\log(1-p_i)$
v	vector in R^s (represents path)
p	loss rate along a path
$\mathcal{N}(G)$	null space of G
$\mathcal{R}(G^T)$	row(path) space of G (== range(G^T))

TABLE I
TABLE OF NOTATIONS

is given by

$$1 - p = \prod_{j \text{ s.t. } v_j=1} (1 - l_j)\,. \tag{1}$$

By taking logarithms on both sides of (1), we have

$$\log(1-p) = \sum_{j=1}^{s} v_j \log(1-l_j) \tag{2}$$

If we define a column vector $x \in \mathbb{R}^s$ with elements $x_j := \log(1-l_j)$, and write v^T for the row vector which is the transpose of v, we can rewrite (2) in the following dot product form:

$$\log(1-p) = \sum_{j=1}^{s} v_j x_j = v^T x \tag{3}$$

Considering all $r = O(n^2)$ paths in the overlay network, there are r linear equations of the form (3). Putting them together, we form a rectangular matrix $G \in \mathbb{R}^{r\times s}$ to represent these paths. Each row of G represents a path in the network: $G_{ij} = 1$ when path i contains link j, and $G_{ij} = 0$ otherwise. Let p_i be the probability of packet loss during transmission on the ith path, and let $b \in \mathbb{R}^r$ be a column vector with elements $b_i := \log(1-p_i)$. Then we write the system of equations relating the link losses to path losses as

$$Gx = b \tag{4}$$

In general, the measurement matrix G may be rank deficient: i.e., k = rank(G) and $k < s$. If G is rank deficient, we will be unable to determine the loss rate of some links from (4). We call these links *unidentifiable* as in [13].

We illustrate how rank deficiency can occur in Fig. 2. There are three end hosts (A, B, and C) on the overlay, three links (1, 2 and 3) and three paths between the end

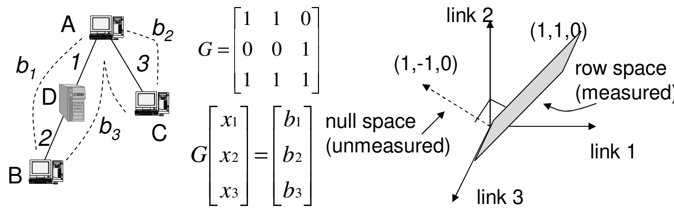


Fig. 2. Sample overlay network.

hosts. Because links 1 and 2 always appear together, their individual loss rates cannot be computed from the measurements. For example, suppose that $x_1 + x_2 = b_1 = -0.06$ and $x_3 = b_2 = -0.01$. We know that $x_1 = -0.03 + \alpha$ and $x_2 = -0.03 - \alpha$ for some α, but the value of α cannot be determined from the end-to-end measurements. The set of vectors $\alpha\begin{bmatrix}1 & -1 & 0\end{bmatrix}^T$ which are not defined by (4) can be added to x without affecting b. This set of vectors is the *null space* of G.

To separate the identifiable and unidentifiable components of x, we write x as $x = x_G + x_N$, where $x_G \in \mathcal{R}(G^T)$ is in the *row space* of G and $x_N \in \mathcal{N}(G)$ is in the orthogonal *null space* of G (i.e. $Gx_N = 0$). The vector x_G contains all the information we can know from (4) and the path measurements. For instance, we can determine $x_1 + x_2$ in Fig. 2, but not $x_1 - x_2$. Intuitively, links 1 and 2 together form a single *virtual link* with an identifiable loss rate $x_1 + x_2$. All end-to-end paths can be written in terms of such *virtual links*, as we describe in more details in Sec. III-C. So x_G involves all the links, while x_N only involves unidentifiable links. The decomposition of x for the sample overlay network is shown below.

$$x_G = \frac{(x_1 + x_2)}{2}\begin{bmatrix}1\\1\\0\end{bmatrix} + x_3\begin{bmatrix}0\\0\\1\end{bmatrix} = \begin{bmatrix}b_1/2\\b_1/2\\b_2\end{bmatrix} \tag{5}$$

$$x_N = \frac{(x_1 - x_2)}{2}\begin{bmatrix}1\\-1\\0\end{bmatrix} \tag{6}$$

Because x_G lies in the k-dimensional space $\mathcal{R}(G^T)$, only k independent equations of the r equations in (4) are needed to uniquely identify x_G. By measuring k independent paths, we can compute x_G. Since $b = Gx = Gx_G + Gx_N = Gx_G$, we can compute all elements of b from x_G, and thus obtain the loss rate of all other paths. For example, in Fig. 2, we only need to measure b_1 and b_2 to compute x_G, from which we can calculate b_3. Detailed algorithms are described in Sec. IV.

B. Dimension Analysis of Path Space ($\mathcal{R}(G^T)$)

In this section, we will examine asymptotically how big k is in terms of n.

Theorem 1: Given a power-law degree network topology of M nodes, the frequency f_d of an outdegree d is proportional to the outdegree to the power of the outdegree exponent constant c (i.e., $f_d \propto d^c$). With $d \ge 1$

and $c < -2$ (as found in [7]), the number of end hosts $N > M/2$.

See Appendix A for the proof. It also follows the intuition that the number of end hosts should be more than the number of routers in the Internet.

Meanwhile, Faloutsos *et al.* prove that such a topology has only $O(M)$ links (Lemma 2 in [7]). Combining the two facts, given N end hosts, there are at most $O(N)$ links in the topology. Thus, if the majority of the end hosts are on the overlay network ($n = O(N)$), the dimension of $\mathcal{R}(G^T)$ is $O(n)$.

What about if only a small portion of the end hosts are on the overlay? Tangmunarunkit *et al.* found that the power-law degree Internet topology has moderate hierarchy due to the heavy-tailed degree distribution [8]. Because G is an r by s matrix, k is bounded by the number of links s. If it is strict hierarchy like a tree, $s = O(n)$, thus $k = O(n)$. But if there is no hierarchy at all (e.g., clique), $k = O(n^2)$ because all the $O(n^2)$ paths are linearly independent. Moderate hierarchy should fall in between. We found that for reasonably large n (e.g, 100), $k = O(n \log n)$.

We run linear regression tests of our hypothesis on both synthetic and real topologies. We experiment with three types of BRITE router-level topologies: Barbarasi-Albert, Waxman and hierarchical models, as well as a real router level topology with 284,805 nodes [20]. For hierarchical topologies, BRITE first generates an autonomous system (AS) level topology with a Barabasi-Albert model or a Waxman model. Then for each AS, BRITE generates the router-level topologies with another Barabasi-Albert model or Waxman model. So there are four types of possible topologies, and they all have the same trend for regression. We just show the results of AS-level Barabasi-Albert model and router-level Waxman model as the representative.

We randomly select end hosts, which has the least degree, to form an overlay network. We fit linear regression of k on $O(n)$, $O(n \log n)$, $O(n^{1.25})$, $O(n^{1.5})$, and $O(n^{1.75})$. As shown in Fig. 3, results for each type of topology are averaged over three runs with different topologies for synthetic ones and with different random sets of end hosts for the real one. We find that for Barabasi-Albert, Waxman and real topologies, $O(n)$ regression has the least residual errors, while for a hierarchical model, $O(n \log n)$ fits best. Conservatively speaking, we have $k = O(n \log n)$.

C. Intuition through Virtual Links

In Sec. III-A, we explain in algebraic terms how to compute all end-to-end path loss rates from only k path measurements. Our actual computations are based completely on this algebraic picture; however, these formulas may seem unintuitive. We now describe a more intuitive picture using the notion of *virtual links*. The key idea is that although the loss rates of some individual links are incomputable (unidentifiable links), each of them is covered by some path segment whose loss rate is computable,

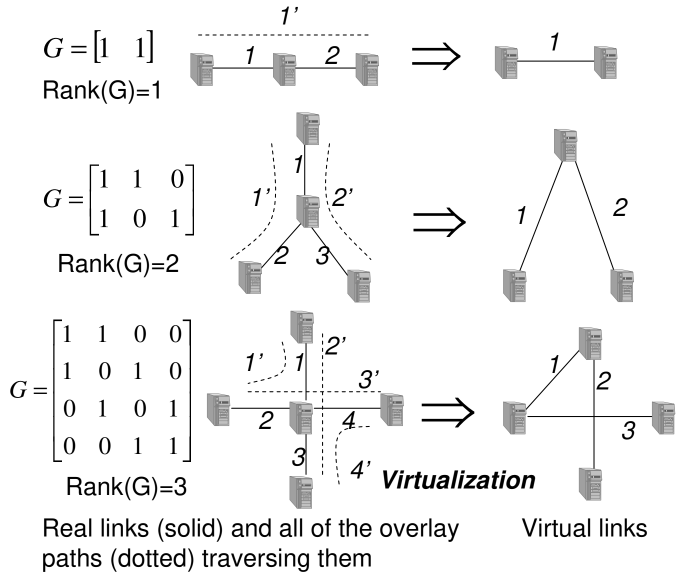


Fig. 4. Sample parts of IP network and overlay paths. The B matrix maps physical links to virtual links.

and the loss rates of these path segments are sufficient to compute the path loss rates in which we are interested.

We choose a minimal set of such path segments that can fully describe all end-to-end paths, and refer to them as *virtual links*. If a link is identifiable, the link itself is a virtual link.

Fig. 4 illustrates some examples. In the top figure, the virtual link is a concatenation of two sequential physical links as we discussed before. In the middle figure, there are three links, but only two paths traverse these links. Thus, rank(G) = 2 and none of the links are identifiable. In the bottom figure, there are four links, and totally four paths traversing them. But the four paths are linearly dependent, so rank(G) = 3, and none of the link loss rate are computable. We can use any three out of the four paths as virtual links, and the other one can be linearly represented by the virtual links. For example, path $4'$ can be described as virtual links 2+3-1.

Since the dimension of $\mathcal{R}(G^T)$ is k, the minimum number of virtual links which can fully describe $\mathcal{R}(G^T)$ is also k. x_G is a linear combination of the vectors representing the virtual links. Since virtual links are identifiable, x_G is also computable. From x_G, we can compute the loss rates of all end-to-end paths as we can do with virtual links.

IV. Algorithms

In this section, we describe implementation techniques.

A. Selecting Measurement Paths

To characterize all $O(n^2)$ end-to-end paths, we monitor k linearly independent end-to-end paths and form a reduced system

$$\bar{G} x_G = \bar{b} \tag{7}$$

where $\bar{G} \in \mathbb{R}^{k \times s}$ and $\bar{b} \in \mathbb{R}^k$ consist of k rows of G and b, respectively. Linearly independent sets of rows and columns in rank-deficient problems are usually computed

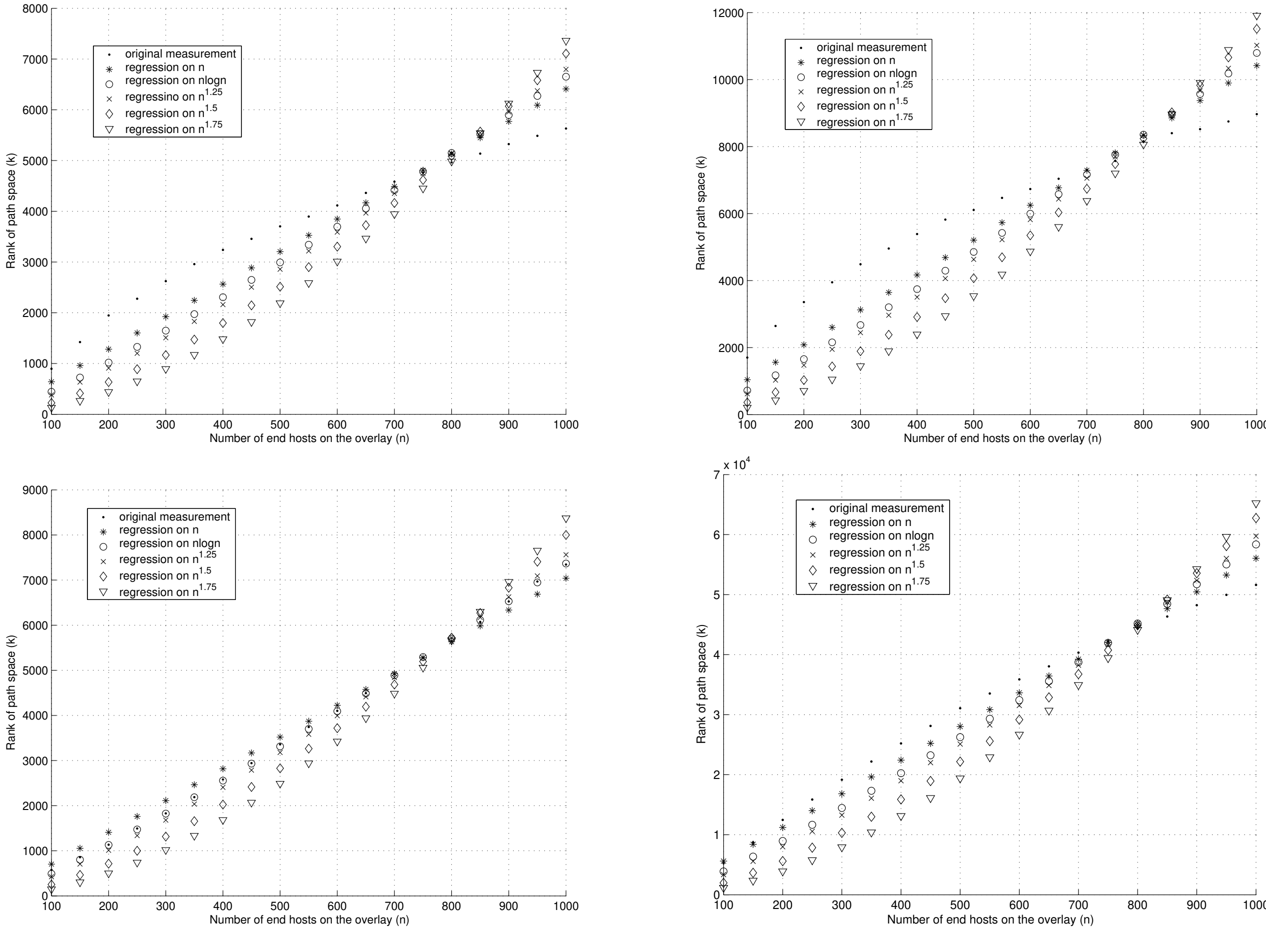


Fig. 3. Regression of k in various functions of n under different router-level topologies. Top: Barabasi-Albert model of 20K nodes (left), and Waxman model of 10K nodes (right). Bottom: hierarchical model of 20K nodes (left) and a real topology of 284K routers (right).

using *rank-revealing decompositions* [22]. For a dense r by s matrix with rank k, common rank-revealing decompositions include Gaussian elimination with complete pivoting (as used in [16]), QR with column pivoting, and the singular value decomposition (SVD). The former two cost $O(rks)$, and the SVD costs $O(rs^2)$. Our G matrix is very sparse; that is, there are only a few nonzeros per row. Rank-revealing decompositions for many sparse problems can be computed much more quickly than in the dense case. However, the exact cost depends strongly on the structure of the problem, and efficient computation rank-revealing decompositions of sparse matrices is an open area of research [23], [24].

We select rows using Algorithm 1, which is a variant of the QR procedure [22, p.223]. The procedure incrementally builds a decomposition

$$\bar{G}^T = QR \tag{8}$$

where $Q \in \mathbb{R}^{s \times k}$ is a matrix with orthonormal columns and $R \in \mathbb{R}^{k \times k}$ is upper triangular. We do not store Q explicitly; instead, we write Q as $R^{-1}\bar{G}^T$. The idea is the same as the classical Gram-Schmidt algorithm: as each row is inspected, we subtract off any components in the space spanned by the previous rows, so that the remainder is orthogonal to all previous rows. If the remainder is zero, then the row was linearly dependent upon the previous rows; otherwise, we extend the factorization.

procedure SelectPath(G)
for *every row v in G* **do**
  $\hat{R}_{12} := R^{-T}\bar{G}v^T = Q^T v^T$
  $\hat{R}_{22} := \|v\|^2 - \|\hat{R}_{12}\|^2$
  if $\hat{R}_{22} \neq 0$ **then**
    Mark v as a measurement path
    $\bar{G} := \begin{bmatrix} \bar{G} \\ v \end{bmatrix}$
    $R := \begin{bmatrix} R & \hat{R}_{12} \\ 0 & \hat{R}_{22} \end{bmatrix}$
  end
end

Algorithm 1: Path (row) selection algorithm

In practice, we use a variant of Algorithm 1 which uses optimized routines from the LAPACK library [25] and inspects several rows at a time. The time complexity of processing each vector is dominated by the solution of a triangular system to compute $\hat{R}_{12}$, which costs $O(k^2)$. The total cost of the algorithm is $O(rk^2)$ and the constant in the bound is modest: on a Pentium 4 running at 1.5 GHz, our code takes just over ten minutes to process a problem with $n = 350$ ($r = 61075$) and $k = 2958$. The memory cost is roughly $k^2/2$ single-precision floating point numbers for storing the R factor.

When k exceeds 10000 the $O(k^2)$ memory requirement becomes too onerous. We note that dense factorization methods may still be feasible if the number of overlay end-hosts is small or if we relax our original problem statement.

B. Path Loss Calculations

The QR decomposition which we use to select measurement paths is also used to compute a solution to the underdetermined system (7). To choose a unique solution x_G to $\bar{G}x_G = \bar{b}$, we impose the additional constraint that $x_G = \bar{G}^T y$. We can then compute

$$\begin{aligned} y &:= R^{-1}R^{-T}\bar{b} \\ x_G &:= \bar{G}^T y. \end{aligned}$$

This is a standard method for finding the minimum norm solution to an underdetermined system (see [22], [26]). The dominant cost in the computation is the solution of two triangular linear systems for y, which costs $O(k^2)$. Once we have computed x_G, we can compute $b := Gx_G$, and from there infer the loss rates of the remaining paths.

C. Topology Measurement Error Tolerance

Our technique is tolerant to network topology measurement errors because our goal is to estimate the end-to-end path loss rate instead of any interior link loss rate. For example, poor alias resolution of routers may present one physical link as several links. At worst, our failure to recognize the links as the same will result in a few more path measurements because the rank of G is higher. But we can still get accurate path loss rate estimation as verified by Internet experiments in Sec. VI.

D. Topology Changes

During normal operation, new links may appear or disappear, routing paths between end hosts may change, and hosts may enter or exit the overlay network. These changes may cause rows or columns to be added to or removed from G, or entries in G may change.

1) Path Additions and Deletions: The basic building blocks for topology updates are path additions and deletions. We have already handled path additions in Algorithm 1; adding a path v during an update is no different than adding a path v during the initial scan of G. In both cases, we decide whether to add v to $\bar{G}$ and update R.

To delete a path that is not in $\bar{G}$ is trivial; we just remove it from G. But to remove a path from $\bar{G}$ is more complicated. We need to update R; this can be done in $O(k^2)$ time by standard algorithms (see e.g. Algorithm 3.4 in [27, p.338]). In general, we may then need to replace the deleted path with another measurement path. Finding a replacement path, or deciding that no such path is needed, can be done by re-scanning the rows of G as in Algorithm 1; however, this would take time $O(rk^2)$.

We now describe Algorithm 2 to delete a path more efficiently. Suppose $\bar{G} \in \mathbb{R}^{k \times s}$ is the measurement path matrix before path deletion, and $\bar{G}' \in \mathbb{R}^{(k-1) \times s}$ is the measurement path matrix after deleting the ith row of $\bar{G}$. By deleting a path from $\bar{G}$, we increase the dimension of the null space from $s - k$ to $s - k + 1$. To find a vector y which is in the null space of $\bar{G}'$ but not of $\bar{G}$, we solve the linear system

$$\bar{G}y = e_i \tag{9}$$

where e_i is the vector of all zeros except for a one in entry i. This system is similar to the linear system we solved to find x_G, and one solution is $y = \bar{G}^T R^{-1} R^{-T} e_j$.

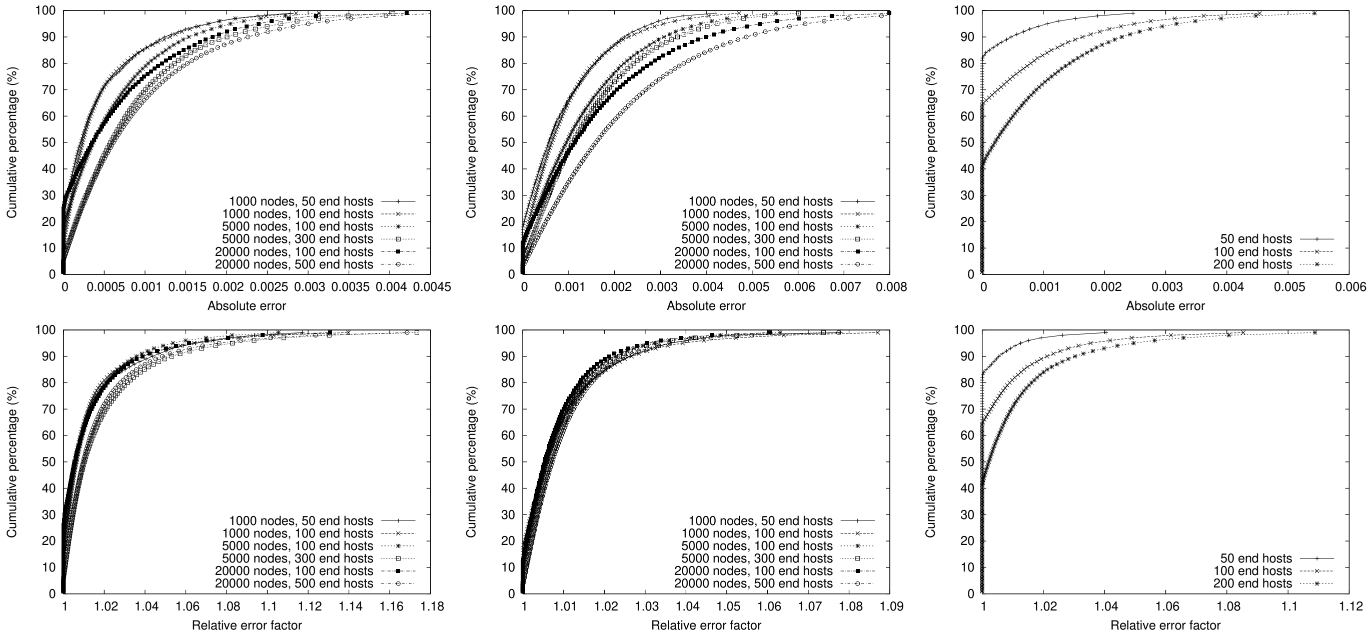
Once we have computed y, we compute $r := G'y$, where G' is the updated G matrix. Because we chose y to make $\bar{G}'y = 0$, all the elements of r corresponding to selected rows are zero. Paths such that $r_j \neq 0$ are guaranteed to be independent of $\bar{G}'$, since if row j of G could be written as $w^T \bar{G}'$ for some w, then r_j would be $w^T \bar{G}' y = 0$. If all elements of r are zero, then y is a null vector for all of G'; in this case, the dimension k' of the row space of G' is $k-1$, and we do not need to replace the deleted measurement path. Otherwise, we can find any j such that $r_j \neq 0$ and add the jth path to $\bar{G}'$ to replace the deleted path.

procedure DeletePath(v, G, $\bar{G}$, R)
if *deleted path v is measured* **then**
    $j :=$ index of v in $\bar{G}$
    $y := \bar{G}R^{-1}R^{-T}e_j$
    Remove v from G and $\bar{G}$
    Update R (Algorithm 3.4 in [27, p.338])
    $r := Gy$
    if $\exists$ *i such that* $r_i \neq 0$ **then**
        Add the ith path from G to $\bar{G}$ (Algorithm 1, steps 2-7)
    end
end
else Remove v from G

Algorithm 2: Path deletion algorithm

When deleting a path used in $\bar{G}$, the factor R can be updated in $O(k^2)$ time. To find a replacement row, we need to compute a sparse matrix-vector product involving G, which takes $O(n^2 \times \text{(average path length)})$ operations. Since most routing paths are short, the dominant cost will typically be the update of R. Thus, the running time for Algorithm 2 is $O(k^2)$.

2) End hosts join/leave the overlay: To add an end host, we use Algorithm 1 to scan all the new paths from that host, for a cost of $O(nk^2)$. To delete an end host, we use Algorithm 2 to delete all the paths involving that host.



BRITE Barabasi-Albert topology | BRITE Hierarchical topology | Real topology of 284K routers

Fig. 5. Cumulative distribution of absolute (top) and relative (bottom) errors under Gilbert loss model for various topologies.

Each path that was used in measurement can be removed in $O(k^2)$ time; the total cost of end host deletion is then at most $O(nk^2)$.

3) Routing changes: In the network, routing changes or link failures can affect multiple paths in G. Most Internet paths remain stable over days [28]. So we can incrementally measure the topology to detect the changes. Each end host measures the paths to all other end hosts daily, and for each end host, such measurement load can be evenly distributed throughout the day.

For each link, we keep a list of the paths that traverse it. If any path is reported as changed for certain link(s), we will examine all other paths that go through those link(s) because it is highly likely that those paths can change their routes as well. We use Algorithms 1 and 2 to incrementally incorporate each path change.

E. Robustness and Real-time Response

There are some scenarios such that the overlay monitoring system can fail to provide real-time loss rate estimation for some paths. This can happen when a routing change is just detected, or the measurement node(s) crash, or some node(s) just join or leave the overlay network. Before we incrementally set up new measurement path(s) and collect results, for a short period, there are some paths for which we can not compute loss rates. However, we can still return bounds on the computed loss rate (see Sec. VII). For example, we can check whether all the links on the incomputable path are covered by $\bar{G}$, and if so, yield an upper bound (though possibly a pessimistic one) quickly. Furthermore, such bounds may be already sufficient for some applications.

V. EVALUATION

In this section, we present the simulation results of tomography-based network monitoring. We consider the following dimensions for simulation.

- Topology type: three types of synthetic topologies from BRITE (see Sec. V-A) and a real router-level topology from [20].
- Topology size: the number of nodes ranges from 1000 to 20000. Note that the node count includes both internal nodes (i.e., routers) and end hosts.
- Fraction of end hosts on the overlay network: we define end hosts to be the nodes with the least degree. Then we randomly choose from 10% to 50% of end hosts to be on the overlay network. This gives us pessimistic results because other distributions of end hosts will probably have more sharing of the routing paths among them. We prune the graphs to remove the nodes and links that are not referenced by any path on the overlay network.
- Link loss rate distribution: 90% of the links are classified as “good” and the rest as “bad”. We use two different models for assigning loss rate to links as in [14]. In the first model ($LLRD_1$), the loss rate for good links is selected uniformly at random in the 0-1% range and that for bad links is chosen in the 5-10% range. In the second model ($LLRD_2$), the loss rate ranges for good and bad links are 0-1% and 1-100% respectively. Given space limitations, most results are under model $LLRD_1$ except for Sec. V-B.
- Loss model: After assigning each link a loss rate, we use either a Bernoulli or a Gilbert model to simulate the loss processes at each link. For a Bernoulli model, each packet traversing a link is dropped at in-

# of nodes	# of end hosts		# of paths(r)	# of links		rank (k)	MPR (k/r)	lossy paths (Bernoulli)			lossy paths (Gilbert)		
	total	OL(n)		original	AP			real	coverage	FP	real	coverage	FP
1000	506	50	1225	1997	443	275	22%	437	99.6%	1.3%	437	100.0%	0.2%
		100	4950		791	543	11%	2073	99.0%	2.0%	1688	99.9%	0.2%
5000	2489	100	4950	9997	1615	929	19%	2271	99.1%	2.0%	2277	99.7%	0.1%
		300	44850		3797	2541	6%	19952	98.6%	4.1%	20009	99.6%	0.3%
20000	10003	100	4950	39997	2613	1318	27%	2738	98.4%	3.4%	2446	99.5%	0.6%
		500	124750		11245	6755	5%	67810	97.8%	5.5%	64733	99.5%	0.4%

# of nodes	# of end hosts		# of paths(r)	# of links		rank (k)	MPR (k/r)	lossy paths (Bernoulli)			lossy paths (Gilbert)		
	total	OL(n)		original	AP			real	coverage	FP	real	coverage	FP
1000	335	50	1225	2000	787	486	40%	704	99.0%	1.1%	579	99.6%	0.4%
		100	4950		1238	909	18%	2544	98.5%	4.6%	2539	99.7%	0.5%
5000	1680	100	4950	10000	2996	1771	36%	3067	97.5%	3.9%	3024	99.5%	0.4%
		300	44850		6263	4563	10%	29135	96.8%	7.1%	28782	99.1%	1.1%
20000	6750	100	4950	40000	5438	2606	53%	3735	98.4%	2.3%	3607	99.6%	0.4%
		500	124750		20621	13769	11%	93049	96.1%	5.7%	92821	99.1%	1.5%

# of nodes	# of end hosts		# of paths(r)	# of links		rank (k)	MPR (k/r)	lossy paths (Bernoulli)			lossy paths (Gilbert)		
	total	OL(n)		original	AP			real	coverage	FP	real	coverage	FP
1000	312	50	1225	2017	441	216	18%	1034	98.8%	2.0%	960	99.6%	0.5%
		100	4950		796	481	10%	4207	98.4%	1.6%	3979	99.6%	0.3%
5000	1608	100	4950	10047	1300	526	11%	4688	99.1%	0.6%	4633	99.8%	0.2%
		300	44850		3076	1787	4%	42331	99.2%	0.8%	42281	99.8%	0.1%
20000	6624	100	4950	40077	2034	613	12%	4847	99.8%	0.2%	4830	100.0%	0.1%
		500	124750		7460	3595	3%	122108	99.5%	0.3%	121935	99.9%	0.1%

TABLE II

SIMULATION RESULTS FOR THREE TYPES OF BRITE ROUTER TOPOLOGIES: BARABASI-ALBERT (TOP), WAXMAN (MIDDLE) AND HIERARCHICAL MODEL (BOTTOM). OL GIVES THE NUMBER OF END HOSTS ON THE OVERLAY NETWORK. AP SHOWS THE NUMBER OF LINKS AFTER PRUNING, WHERE PRUNING REMOVES THE NODES AND LINKS THAT ARE NOT ON THE OVERLAY PATHS. MPR IS THE MONITORED PATH RATIO BETWEEN OUR SCHEME AND THE PAIR-WISE SCHEME. FP IS THE FALSE POSITIVE RATE.

dependently fixed probability as the loss rate of the link. For a Gilbert model, the link fluctuates between a good state (no packet dropped) and a bad state (all packets dropped). According to Paxon's observed measurement of Internet [29], the probability of remaining in bad state is set to be 35% as in [14]. Thus, the Gilbert model is more likely to generate bursty losses than the Bernoulli model. The other state transition probabilities are selected so that the average loss rates matches the loss rate assigned to the link.

We repeat our experiments five times for each simulation configuration unless denoted otherwise, where each repetition has a new topology and new loss rate assignments. The path loss rate is simulated based on the transmission of 10000 packets. Using the loss rates of selected paths as input, we compute x_G, then the loss rates of all other paths.

The metric is path loss rate estimation accuracy. To compare the inferred loss rate $\hat{p}$ with real loss rate p, we analyze both absolute error and relative error. The absolute error is $|p - \hat{p}|$. We adopt the relative error factor $F_\varepsilon(p, \hat{p})$ defined in [13] as follows:

$$F_\varepsilon(p, \hat{p}) = \max\left\{\frac{p(\varepsilon)}{\hat{p}(\varepsilon)}, \frac{\hat{p}(\varepsilon)}{p(\varepsilon)}\right\} \tag{10}$$

where $p(\varepsilon) = \max(\varepsilon, p)$ and $\hat{p}(\varepsilon) = \max(\varepsilon, \hat{p})$. Thus, p and $\hat{p}$ are treated as no less than ε, and then the error factor is the maximum ratio, upwards or downwards, by which they differ. We use the default value $\varepsilon = 0.001$ as in [13]. If the estimation is perfect, the error factor is one.

Furthermore, we classify a path to be lossy if its loss rate exceeds 5%, which is the threshold between "tolerable loss" and "serious loss" as defined in [19]. We report the true number of lossy paths, the percentage of real lossy paths identified (coverage) and the false positive rate, all averaged over five runs of experiment for each configuration.

A. Different Topologies

We test three types of router-level topologies from BRITE [30]: Barabasi-Albert, Waxman and hierarchical models. Four types of hierarchical models described in III-B are tested. They all have similar results, thus we only show the results of AS-level Barabasi-Albert and router-level Waxman topology as the representative.

We achieve high loss rate estimation accuracy in terms of small absolute and relative errors. Results for the Bernoulli and the Gilbert models are similar. Since the

# of end hosts on overlay (n)	# of paths(r)	# of links after pruning	rank (k)	MPR (k/r)	lossy paths (Bernoulli)			lossy paths (Gilbert)		
					real	coverage	FP	real	coverage	FP
50	1225	2098	1017	83%	891	99.7%	0.9%	912	100.0%	0.2%
100	4950	5413	3193	65%	3570	98.7%	1.9%	3651	99.6%	0.3%
200	19900	12218	8306	42%	14152	97.9%	3.1%	14493	99.6%	0.4%

TABLE III

SIMULATION RESULTS FOR A REAL ROUTER TOPOLOGY. MPR IS THE MONITORED PATH RATIO BETWEEN OUR SCHEME AND THE PAIR-WISE SCHEME. FP IS THE FALSE POSITIVE RATE.

Gilbert loss model is more realistic, we plot the cumulative distribution functions (CDFs) of absolute and relative errors with the Gilbert model in Fig. 5. For all the configurations, the absolute errors are less than 0.008 and the relative errors are less than 1.18. Waxman topologies have similar results, and we omit them in the interest of space.

The lossy path inference results are shown in Table II. Notice that k is much smaller than the number of IP links that the overlay network spans, which means that there are many IP links whose loss rates are unidentifiable. For all configurations, the lossy path coverage is more than 96% and the false positive ratio is less than 8%. Many of the false positives and false negatives are caused by small estimation errors for paths with loss rates near the 5% threshold.

We also test our algorithms in the 284,805-node real router-level topology from [20]. There are 65,801 end host routers and 860,683 links. We get the same trend of results as illustrated in Fig. 5 and Table III. The CDFs include all the path estimation, such as the monitored paths of which we know the real loss rates. Given the same number of end hosts, the ranks in the real topology are higher than those of the synthetic ones. But as we find in Sec. III-B, the growth of k is still in the range of $O(n)$.

B. Different Link Loss Rate Distribution

We have also run all the simulations above with model $LLRD_2$. The loss rate estimation is a bit less accurate than it is under $LLRD_1$, but we still find over 95% of the lossy paths with a false positive rate under 10%. Given space limitations, we only show the lossy path inference with the Barabasi-Albert topology model and the Gilbert loss model in Table IV.

C. Topology Changes

We study two common scenarios in overlay networks: end hosts joining and leaving as well as routing changes. Again, the Bernoulli and the Gilbert models have similar results, thus we only show those of the Gilbert model.

1) End hosts join/leave: For the real router topology, we start with an overlay network of 40 random end hosts. Then we randomly add an end host to join the overlay, and repeat the process until the size of the overlay reaches 45 and 50. Averaged over three runs, the results in Table V show that there is no obvious accuracy degradation caused by accumulated numerical errors.

The results for removing end hosts are similar, and we omit them in the interest of space.

# of nodes	# of end hosts		lossy paths (Gilbert)		
	total	on overlay	real	coverage	FP
1000	506	50	495	99.8%	1.1%
		100	1989	99.8%	3.0%
5000	2489	100	2367	99.6%	3.5%
		300	21696	99.2%	1.4%
20000	10003	100	2686	98.8%	1.1%
		500	67817	99.0%	4.6%

TABLE IV

SIMULATION RESULTS WITH MODEL $LLRD_2$. USE THE SAME BARABASI-ALBERT TOPOLOGIES AS IN TABLE II. REFER TO TABLE II FOR STATISTICS LIKE RANK. FP IS THE FALSE POSITIVE RATE.

# of end hosts	# of paths	rank	lossy paths		
			real	coverage	FP
40	780	616	470	99.9%	0.2%
+5 (45)	+210 (990)	+221 (837)	+153 (623)	100.0%	0.1%
+5 (50)	+235 (1225)	+160 (997)	+172 (795)	99.8%	0.2%

TABLE V

SIMULATION RESULTS FOR ADDING END HOSTS ON A REAL ROUTER TOPOLOGY. FP IS THE FALSE POSITIVE RATE. DENOTED AS +ADDED_VALUE (TOTAL_VALUE).

2) Routing changes: We form an overlay network with 50 random end hosts on the real router topology. Then we simulate topology changes by randomly choosing a link that is on some path of the overlay and removing of such link will not cause disconnectivity for any pair of overlay end hosts. Then we assume that link is broken, and have the affected path(s) re-route. Algorithms in Sec. IV-D are applied to incrementally incorporate each path change. Averaged over three runs, results in Table VI show that we still have good accuracy in path loss rate estimation.

VI. INTERNET EXPERIMENTS

We implemented our system on the Planet Lab [9] testbed. In this section, we present the experimental results from the implementation.

A. Methodology

We choose 51 Planet Lab hosts, each from a different organization as shown in Table VII. All the international

# of paths affected	40.7
# of monitored paths affected	36.3
# of unique nodes affected	41.7
# of real lossy paths (before/after)	761.0/784.0
coverage (before/after)	99.8%/99.8%
false positive rate (before/after)	0.2%/0.1%

TABLE VI
SIMULATION RESULTS FOR REMOVING A LINK FROM A REAL ROUTER TOPOLOGY.

Planet Lab hosts are universities.

Areas and Domains			# of hosts
US (40)	.edu		33
	.org		3
	.net		2
	.gov		1
	.us		1
International (11)	Europe (6)	France	1
		Sweden	1
		Denmark	1
		Germany	1
		UK	2
	Asia (2)	Taiwan	1
		Hong Kong	1
	Canada		2
	Australia		1

TABLE VII
DISTRIBUTION OF PLANET LAB HOSTS FOR EXPERIMENTS.

First, we measure the topology among these sites by simultaneously running "traceroute" to find the paths from each host to all others. Each host saves its destination IP addresses for sending measurement packets later. Then we measure the loss rates between every pair of hosts. Our measurement consists of 300 trials, each of which lasts 300 msec. During a trial, each host sends a 40-byte UDP packet [1] to every other host. Usually the hosts will finish sending before the 300 msec trial is finished. For each path, the receiver counts the number of packets received out of 300 to calculate the loss rate.

To prevent any host from receiving too many packets simultaneously, each host sends packets to other hosts in a different random order. Furthermore, any single host uses a different permutation in each trial so that each destination has equal opportunity to be sent later in each trial, because when sending packets in a batch, the packets sent later are more likely to be dropped. Such random permutations are pre-generated by each host. To ensure that all hosts in the network take measurements at the same time, we set up sender and receiver daemons, then use a well-connected server to broadcast a "START" command.

Will the probing traffic itself cause losses? We did sensitivity analysis on sending frequency as shown in Fig. 6.

[1] 20-byte IP header + 8-byte UDP header + 12-byte data on sequence number and sending time.

All experiments were executed between 1am-3am PDT June 24, 2003, when most networks are free. The traffic rate from or to each host is $(51-1) \times sending\_freq \times 40$ bytes/sec. The number of lossy paths does not change much when the sending rate varies, except when the sending rate is over 12.8Mbps, since many servers can not sustain that sending rate. We choose a 300 msec sending interval to collect loss rate statistics quickly but with moderate bandwidth consumption.

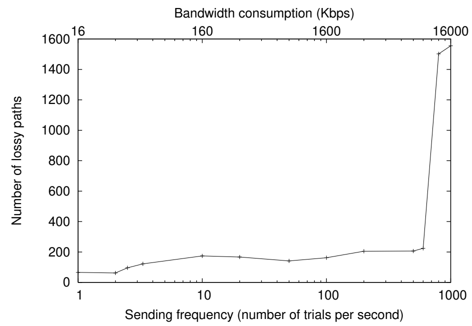


Fig. 6. Sensitivity test of sending frequency

Note that the experiments above use $O(n^2)$ measurements so that we can compare the real loss rates with our inferred loss rates. In fact, our technique only requires $O(n \log n)$ measurements. Thus, given good load balancing, each host only needs to send to $O(\log n)$ hosts. Even for an overlay network of 400 random end hosts on the real topology of 284K nodes that we used before, k = 18668. If we reduce the measurement frequency to one trial per second, the traffic consumption for each host is $18668/400\times$ 40 bytes/sec = 14.9Kbps, which is typically less than 5% of the bandwidth of today's "broadband" Internet links. We can use adaptive measurement techniques in [6] to further reduce the overheads.

B. Results

From June 24 to June 27, 2003, we ran the experiments 100 times, mostly during peak hours 9am - 6pm PDT. Each experiment generates 51 $\times$ 50$\times$ 300 = 765K UDP packets, totaling 76.5M packets for all experiments. We run the loss rate measurements three to four times every hour, and run the pair-wise traceroute every two hours. Across the 100 runs, the average number of selected monitoring paths ($\bar{G}$) is 871.9, about one third of total number of end-to-end paths, 2550. Table VIII shows the loss rate distribution on all the paths of the 100 runs. About 96% of the paths are non-lossy. Among the lossy paths, most of the loss rates are less than 0.5. Though we try to choose stable nodes for experiments, about 25% of the lossy paths have 100% losses and are likely caused by node failures or other reachability problems as discussed in Sec. VI-B.2.

1) Accuracy: When identifying the lossy paths (loss rates > 0.05), the average coverage is 95.6% and the average false positive rate is 2.75% Fig. 7 shows the CDFs for the coverage and the false positive rate. Notice that 40 runs have 100% coverage and 90 runs have coverage over

loss rate	[0, 0.05)	lossy path [0.05, 1.0] (4.1%)				
		[0.05, 0.1)	[0.1, 0.3)	[0.3, 0.5)	[0.5, 1.0)	1.0
%	95.9%	15.2%	31.0%	23.9%	4.3%	25.6%

TABLE VIII
LOSS RATE DISTRIBUTION: LOSSY VS. NON-LOSSY AND THE SUB-PERCENTAGE OF LOSSY PATHS.

85%. 58 runs have no false positives and 90 runs have false positive rates less than 10%.

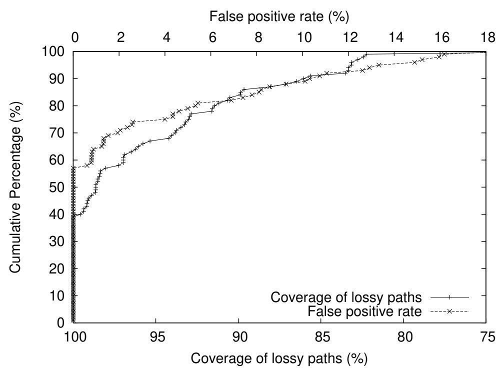


Fig. 7. Cumulative percentage of the coverage and the false positive rates for lossy path inference in the 100 experiments.

As in the simulations, many of the false positives and false negatives are caused by the 5% threshold boundary effect. The average absolute error across the 100 runs is only 0.0027 for all paths, and 0.0057 for lossy paths. We pick the run with the worst accuracy in coverage (69.2%), and plot the CDFs of absolute and relative errors in Fig. 8. Since we only use 300 packets to measure the loss rate, the loss rate precision granularity is 0.0033, so we use $\varepsilon = 0.005$ for relative error calculation.

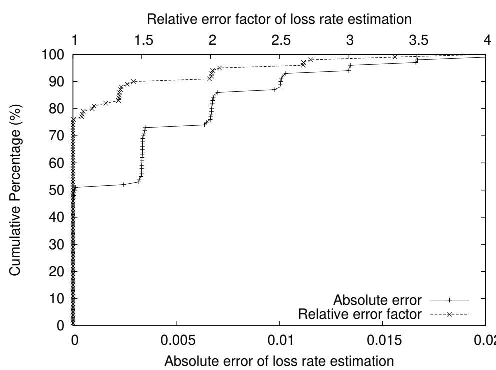


Fig. 8. Cumulative percentage of the absolute and relative errors for the experiment with the worst accuracy in coverage.

Even for the worst case, 95% of absolute error in loss rate estimation is less than 0.014, and 95% of relative error factor is less than 2.1. To further view the overall statistics, we pick 95 percentile of absolute and relative errors in each run, and plot the CDFs on those metrics. The results are shown in Fig. 9. Notice that 90 runs have the 95 percentile of absolute error less than 0.0135, and 90 runs have the 95 percentile of relative error factor less than 2.0.

In short, we achieve high accuracy for loss rate estimation with both Internet experiments and simulations.

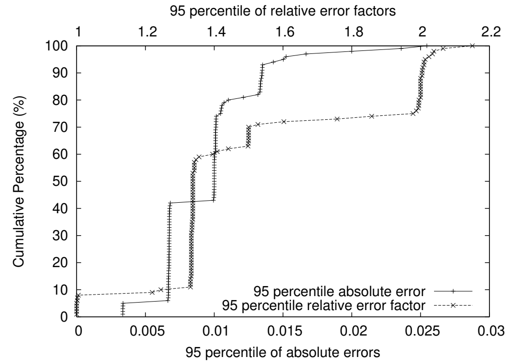


Fig. 9. Cumulative percentage of the 95 percentile of absolute and relative errors for the 100 experiments.

2) Topology error tolerance: The limitation of traceroute, which we use to measure the topology among the end hosts, led to many topology measurement inaccuracies. As found in [31], many of the routers on the paths among Planet Lab nodes have aliases. We did not use sophisticated techniques to resolve these aliases. Thus, the topology we have is far from accurate. But we are still be able to get good results as above.

Some nodes were down, or were unreachable from certain nodes. For instance, planetlab1.ipls.internet2.planet-lab.org, planetlab2.sttl.internet2.planet-lab.org and planet2.berkeley.intel-research.net often can not reach uw1.accretive-dsl.nodes.planet-lab.org, while other nodes can. Meanwhile, some routers are hidden and we only get partial routing paths. Averaging over 14 sets of traceroutes, 245 out of $51 \times 50 = 2550$ paths have no or incomplete routing information.

For completely untraceable paths, we add a direct link between the source and the destination. In our system, these paths will become selected paths for monitoring ($\bar{G}$). For traceroutes that only gives partial paths, we add links from where the normal route becomes unavailable (self loops or displaying "* * *"), to where the traceroute resumes or the destination if such anomalies persist to the end. The accurate loss rate estimation results show that the technique is robust against topology measurement errors.

VII. DISCUSSION

In this section, we generalize our framework to infer the path loss rate bound when we have only restricted measurements.

We note that, in addition to the equations (4), the unknown x_j must satisfy the inequalities $x_j \geq 0$. While we do not make use of them in our current work, these inequalities can be used in conjunction with (4) to bound failure probabilities, both from below and from above. For example, the loss probability l_j is bounded above by the loss probability of the least lossy path that includes link j. More generally, we have the following theorem:

Theorem 2: Let $v \in \mathbb{R}^s$ represent a network path with loss probability p, and let $w = G^T c$ for some $c \in \mathbb{R}^r$ (i.e. $w \in \mathcal{R}(G^T)$). Then

1) If $v \leq w$ elementwise, then $\log(1-p) \geq c^T b$
2) If $v \geq w$ elementwise, then $\log(1-p) \leq c^T b$

See Appendix B for the proof.

In principle, we can compute good upper and lower bounds on path loss rates by solving two linear programming problems:

1) Maximize $c_u^T b$ subject to $G^T c_u \geq v$,
2) Minimize $c_l^T b$ subject to $G^T c_l \leq v$.

Then $1 - \exp(c_l^T b) \leq p \leq 1 - \exp(c_u^T b)$. When $v \in \mathcal{R}(G^T)$ we have $v = G^T c_u = G^T c_l$, and the bound is tight. While this approach seems to offer bounds on path loss probabilities that are possibly optimal given the measured data, we have not yet applied the technique in practice.

VIII. Conclusions

In this paper, we present a tomography-based overlay network monitoring system. For an overlay of n end hosts, the space of $O(n^2)$ paths can be characterized by a basis of $O(n \log n)$ paths. We selectively monitor these basis paths, then use the measurements to infer the loss rates of all other paths. Both simulation and real implementation on the Internet show that our techniques achieve accurate loss rate estimation.

For more efficient monitored path selection, we plan to investigate the use of iterative methods [32], [23] such as CGNE or GMRES both to select rows and to compute loss rate vectors. In our preliminary experiments, the path matrix G has been well-conditioned, which suggests that iterative methods may converge quickly. Besides, we are applying the inequality bounds in Sec. VII for diagnostics, to detect which links or path segments may fail when end-to-end congestion occurs. We will also study random reordering techniques to achieve load balancing of path monitoring across the overlay nodes. Furthermore, we are building applications, such as streaming media, on top of such monitoring systems.

References

[1] T. S. E. Ng and H. Zhang, "Predicting Internet network distance with coordinates-based approaches," in *Proc.of Infocom*, 2002.

[2] S. Ratnasamy et al., "Topologically-aware overlay construction and server selection," in *Proc. of IEEE Infocom*, 2002.

[3] P. Francis et al., "IDMaps: A global Internet host distance estimation service," *IEEE/ACM Trans. on Networking*, Oct. 2001.

[4] W. Theilmann and K. Rothermel, "Dynamic distance maps of Internet," in *Proceedings of IEEE Infocom*, 2000.

[5] Y. Chen, K. Lim, C. Overton, and R. H. Katz, "On the stability of network distance estimation," in *ACM SIGMETRICS Performance Evaluation Review (PER)*, Sep. 2002.

[6] D. G. Andersen, H. Balakrishnan, M. F. Kaashoek, and R. Morris, "Resilient overlay networks," in *Proc. of ACM SOSP*, 2001.

[7] M. Faloutsos, P. Faloutsos, and C. Faloutsos, "On power-law relationship of the Internet topology," in *ACM SIGCOMM*, 1999.

[8] H. Tangmunarunkit et al., "Network topology generators: Degree-based vs structural," in *ACM SIGCOMM*, 2002.

[9] Planet Lab, "http://www.planet-lab.org/," .

[10] Mark Coates, Alfred Hero, Robert Nowak, and Bin Yu, "Internet Tomography," *IEEE Signal Processing Magazine*, vol. 19, no. 3, pp. 47–65, 2002.

[11] A. Adams et al., "The use of end-to-end multicast measurements for characterizing internal network behavior," in *IEEE Communications*, May, 2000.

[12] M. Adler et al., "Tree layout for internal network characterizations in multicast networks," in *3rd International Workshop on Networked Group Communication (NGC)*, 2001.

[13] T. Bu, N. Duffield, F. Presti, and D. Towsley, "Network tomography on general topologies," in *ACM SIGMETRICS*, 2002.

[14] V. Padmanabhan, L. Qiu, and H. Wang, "Server-based inference of Internet performance," in *IEEE INFOCOM*, 2003.

[15] D. Rubenstein, J. F. Kurose, and D. F. Towsley, "Detecting shared congestion of flows via end-to-end measurement," *IEEE/ACM Transactions on Networking*, vol. 10, no. 3, 2002.

[16] Y. Shavitt, X. Sun, A. Wool, and B. Yener, "Computing the unmeasured: An algebraic approach to Internet mapping," in *IEEE INFOCOM*, 2001.

[17] H. C. Ozmutlu, N. Gautam, and R. Barton, "Managing end-to-end network performance via optimized monitoring strategies," *Journal of Network and System Management, Special Issue on Management of Converged Networks*, vol. 10, no. 1, 2002.

[18] R. Wolski, N. Spring, and J. Hayes, "The network weather service: A distributed resource performance forecasting service for metacomputing," *Journal of Future Generation Computing Systems*, vol. 15, no. 5–6, pp. 757–768, 1999.

[19] Y. Zhang et al., "On the constancy of Internet path properties," in *Proc. of SIGCOMM Internet Measurement Workshop 2001*.

[20] R. Govindan and H. Tangmunarunkit, "Heuristics for Internet map discovery," in *IEEE INFOCOM*, 2000.

[21] M. Coates, R. Castro, and R. Nowak, "Maximum likelihood identification of network topology from edge-based unicast measurements," in *ACM SIGMETRICS*, 2002.

[22] G.H. Golub and C.F. Van Loan, *Matrix Computations*, The Johns Hopkins University Press, 1989.

[23] C. Meyer and D. Pierce, "Steps toward an iterative rank-revealing method," Tech. Rep. ISSTECH-95-013, Boeing Information and Support Services, 1995.

[24] D. Pierce and J. Lewis, "Sparse multifrontal rank revealing QR factorization," *SIAM Journal on Matrix Analysis and Applications*, vol. 18, no. 1, January 1997.

[25] E. Anderson et al., *LAPACK Users' Guide*, Society for Industrial and Applied Mathematics, Philadelphia, PA, third edition, 1999.

[26] J.W. Demmel, *Applied Numerical Linear Algebra*, SIAM, 1997.

[27] G. W. Stewart, *Matrix Algorithms: Basic Decompositions*, Society for Industrial and Applied Mathematics, 1998.

[28] V. Paxon, "End-to-end routing behavior in the Internet," *IEEE/ACM Transactions on Networking*, vol. 5, no. 5, 1997.

[29] V. Paxon, "End-to-end Internet packet dynamics," in *ACM SIGCOMM*, 1997.

[30] A. Medina, I. Matta, and J. Byers, "On the origin of power laws in Internet topologies," in *ACM Computer Communication Review*, Apr. 2000.

[31] N. Spring, D. Wetherall, and T. Anderson, "Scriptroute: A facility for distributed internet measurement," in *USITS*, 2003.

[32] R. Barrett et al., *Templates for the Solution of Linear Systems: Building Blocks for Iterative Methods, 2nd Edition*, SIAM, Philadelphia, PA, 1994.

Appendix A
Proof for Theorem 1

Proof:

Given that the power-law distribution topology has outdegree exponent: the frequency f_d of an outdegree d is proportional to the outdegree to the power of a constant, i.e., $f_d = Nd^c$, where N is the proportion constant. Assume that end hosts have degree 1, then the number of end hosts is N.

If $c < -1$, then

$$\begin{aligned} M &= N \sum_{d=1}^{M-1} d^c && (11) \\ &\leq N \left(1 + \int_1^{M-1} x^c \, dx\right) && (12) \\ &\leq N \left(1 + \int_1^{\infty} x^c \, dx\right) && (13) \\ &= N \left(1 - \frac{1}{1+c}\right) && (14) \\ &= N \frac{c}{1+c} && (15) \end{aligned}$$

Therefore, the fraction $\frac{N}{M}$ is at least $\frac{1+c}{c} = 1 + \frac{1}{c}$. If $c \leq -2$ then $\frac{N}{M} \geq \frac{1}{2}$.

■

Appendix B
Proof for Theorem 2

Proof: In the first case, $v \leq w$ so that $v - w \leq 0$ elementwise. Since $x \leq 0$ elementwise, $(v-w)^T x \geq 0$, or $v^T x \geq w^T x$. We know $\log(1-p) = v^T x$ from (3), and $w^T x = c^T G x = c^T b$. By substitution, we have $\log(1-p) \geq c^T b$. The second case is nearly identical. ■