Algebra-Based Scalable Overlay Network Monitoring: Algorithms, Evaluation, and Applications

Yan Chen, *Member, IEEE*, David Bindel, Han Hee Song, and Randy H. Katz, *Fellow, IEEE*

***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. Our earlier extended abstract [Y. Chen, D. Bindel, and R. H. Katz, "Tomography-based overlay network monitoring," *Proceedings of the ACM SIGCOMM Internet Measurement Conference* (IMC), 2003] briefly proposes an algebraic approach that selectively monitors k linearly independent paths that can fully describe all the $O(n^2)$ paths. The loss rates and latency of these k paths can be used to estimate the loss rates and latency of all other paths. Our scheme only assumes knowledge of the underlying IP topology, with links dynamically varying between lossy and normal. In this paper, we improve, implement, and extensively evaluate such a monitoring system. We further make the following contributions: i) scalability analysis indicating that for reasonably large n (e.g., 100), the growth of k is bounded as $O(n \log n)$, ii) efficient adaptation algorithms for topology changes, such as the addition or removal of end hosts and routing changes, iii) measurement load balancing schemes, iv) topology measurement error handling, and v) design and implementation of an adaptive streaming media system as a representative application. Both simulation and Internet experiments demonstrate we obtain highly accurate path loss rate estimation while adapting to topology changes within seconds and handling topology errors.**

***Index Terms*—Dynamics, load balancing, network measurement and monitoring, numerical linear algebra, overlay, scalability.**

I. Introduction

THE rigidity of the Internet architecture makes it extremely difficult to deploy innovative disruptive technologies in the core. This has led to extensive research into overlay and peer-to-peer systems, such as overlay routing and location, application-level multicast, and peer-to-peer file sharing. These systems flexibly choose their communication paths and targets, and thus can benefit from estimation of end-to-end network distances (e.g., latency and loss rate).

Manuscript received April 21, 2005; revised June 6, 2006; approved by IEEE/ACM Transactions on Networking Editor R. Caceres.

Y. Chen is with the Northwestern University, Technical Institute, Evanston, IL 60208-3118 USA (e-mail: ychen@cs.northwestern.edu).

D. Bindel is with the Department of Mathematics, Courant Institute of Mathematical Sciences, New York University, New York, NY 10012 USA.

H. H. Song is with the University of Texas at Austin, Austin, TX 78712-1188 USA.

R. H. Katz is with the University of California at Berkeley, Berkeley, CA 94720-1776 USA.

Digital Object Identifier 10.1109/TNET.2007.896251

Accurate loss rate monitoring systems can detect path outages and periods of degraded performance within seconds. They facilitate management of distributed systems such as virtual private networks (VPMs) and content distribution networks; and they are useful for building adaptive overlay applications, like streaming media and multiplayer gaming.

Thus it is desirable to have a scalable overlay loss rate monitoring system that is accurate and incrementally deployable. However, existing network distance estimation systems are insufficient for this end. These existing systems can be categorized as *general metric* systems [2] and *latency-only* systems [3]–[6]. Systems in the former category can measure any metric, but require $O(n^2)$ measurements where n is the number of end hosts, and thus lack scalability. On the other hand, the latency estimation systems are scalable but cannot provide accurate congestion/failure detection (see Section II).

We formulate the problem as follows: 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 and latencies of all other paths can be inferred.

In this paper, we propose a tomography-based overlay monitoring system (*TOM*) in which we selectively monitor a *basis set* of k paths. Any end-to-end path can be written as a unique linear combination of paths in the basis set. Consequently, by monitoring loss rates for the paths in the basis set, we infer loss rates for all end-to-end paths. This can also be extended to other additive metrics, such as latency. The end-to-end path loss rates can be computed even when the paths contain *unidentifiable links* for which loss rates cannot be computed.

There are many open questions as follows.

- How scalable is the system? In other words, how will k grow as a function of n?
- In an overlay network, end hosts frequently join/leave the overlay and routing changes occur from time to time. How can the system adapt to these efficiently?
- How should the system distribute the measurement load among end hosts to improve scalability?
- How can the system maintain accuracy when there are topology measurement errors?
- How does TOM perform under various topologies and loss conditions, and in the real Internet?
- How can real applications benefit from TOM?

To address these issues, in this paper, we make the following contributions.

- We show that for reasonably large n (say 100), $k = O(n \log n)$ through linear regression tests on various synthetic and real topologies. We also provide some explanation based on the Internet topology and the AS hierarchy.
- We design incremental algorithms for path addition and deletion which only cost $O(k^2)$ time, instead of the $O(n^2k^2)$ time cost to reinitialize the system.
- We propose randomized schemes for measurement load balancing.
- We design effective schemes to handle topology measurement errors.
- We evaluate TOM through extensive simulations and further validate our results through Internet experiments.
- We build an adaptive overlay streaming media system on top of TOM and achieve skip-free live media playback when switching to overlay routing to bypass faulty or slow links in the face of congestion/failures.

In both simulations and PlanetLab experiments, we estimate path loss rates with high accuracy using $O(n \log n)$ measurements. For the PlanetLab experiments, the average absolute error of loss rate estimation is only 0.0027, and the average error factor is 1.1, even though about 10% of the paths have incomplete or nonexistent routing information. The average setup (monitoring path selection) time is 0.75 s, and the online update of the loss rates for all 2550 paths takes only 0.16 s. In addition, we adapt to topology changes within seconds without sacrificing accuracy. The measurement load balancing reduces the load variation and the maximum versus mean load ratio significantly, by up to a factor of 7.3.

We additionally show how streaming media systems can benefit from TOM. The media application's total adaptation time is less than three seconds on average. This includes the time from when the congestion/failure occurs, to when TOM detects it and sends alternative overlay path to the client, until the client sets up overlay connection to the server and concatenates new streams with the old ones in the buffer. Then, with our buffering techniques, which retransmit lost packets during path switch, we achieve skip-free media playback on the PlanetLab experiments.

The rest of the paper is organized as follows. We survey related work in Section II, describe our model and basic static algorithms in Section III, and evaluate scalability in Section IV. We extend the algorithms to adapt to topology changes in Section V and to handle overload and topology measurement errors in Section VI. The methodology and results of our simulations are described in Section VIII, and those of our Internet experiments are presented in Section IX. Finally, we conclude in Section X.

II. Related Work

There are many existing scalable end-to-end latency estimation schemes, which can be broadly classified into clustering-based [5], [6] and coordinate-based systems [3], [4]. Clustering-based systems cluster end hosts based on their network proximity or latency similarity under normal conditions, then choose the centroid of each cluster as the monitor. But a monitor and other members of the same cluster often take different routes to remote hosts. So the monitor cannot detect congestion for its members. Similarly, the coordinates assigned to each end host in the coordinate-based approaches cannot embed any congestion/failure information.

Network tomography has been well studied ([7] provides a good survey). Most tomography systems assume limited measurements are available (often in a multicast treelike structure), and try to infer link characteristics [8], [9] or shared congestion [10] in the middle of the network. However, the problem is underconstrained: There exist *unidentifiable links* [8] with properties that cannot be uniquely determined. In contrast, we are not concerned about the characteristics of *individual* links, and we do not restrict the paths we measure.

Shavitt *et al.* also use algebraic tools to compute distances that are not explicitly measured [11]. Given certain "Tracer" stations deployed and some direct measurements among the Tracers, they search for path or path segments whose loss rates can be inferred from these measurements. Thus their focus is not on Tracer/path selection.

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 [12]. But the link-by-link latency obtained from traceroute is often inaccurate. And their approach is not applicable for loss rate because it is difficult to estimate link-by-link loss rates from end-to-end measurement. A similar approach was taken for selecting paths to measure overlay network [13]. The minimal set cover selected can only gives *bounds* for metrics like latency, and there is no guarantee as to how far the bounds are from the real values.

In contrast to the pure overlay inference approach, some recent network tomography work [14], [15] try to measure the *link* level (instead of the *path* level as in this paper) performance based on response packets sent by interior routers. Unfortunately, interior routers may be unwilling to respond, or may respond in an insufficiently informative manner, e.g., ICMP rate limiting. Thus there is no accuracy guarantee on the loss rate inference.

Furthermore, very few of the existing network tomography work examines topology change, topology measurement errors, or measurement load balancing problems.

III. Model and Basic Algorithms

A. Algebraic Model

Suppose there are n end hosts that belong to a single or confederated overlay network(s). They cooperate to share an overlay monitoring service and are instrumented by a central authority [e.g., an overlay network operation center (ONOC)] to measure the routing topology and path loss rates as needed.[1] For simplicity, we usually assume symmetric routing and undirected links in this paper. However, our techniques work without change for asymmetric routing, as evidenced in the PlanetLab experiments. Fig. 1 shows a sample overlay network with four links and four end hosts; six possible paths connect the end hosts. 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 those paths. The end hosts periodically report the measured loss rates 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

[1]As part of the future work, we will investigate techniques to distribute the work of the central authority.

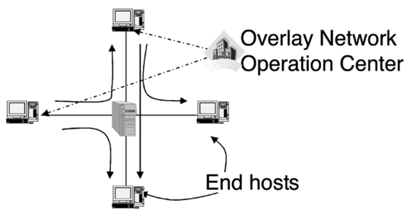


Fig. 1. Architecture of a TOM system.

TABLE I
TABLE OF NOTATIONS

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
$G \in \{0,1\}^{r\times s}$	original path matrix
$\bar{G} \in \{0,1\}^{k\times s}$	reduced path matrix
$k \leq s$	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 $\{0,1\}^s$ (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))

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 [16].

We now introduce an algebraic model that applies to any network topology. Please refer to Table I for notations. Suppose an overlay network spans s IP links. We represent a path by a column vector $v \in \{0,1\}^s$, 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 loss rate p of a path represented by v is given by

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

Equation (1) assumes that packet loss is independent among links. Caceres *et al.* argue that the diversity of traffic and links makes large and long-lasting spatial link loss dependence unlikely in a real network such as the Internet [17]. Furthermore, the introduction of random early detection (RED) [18] policies in routers will help break such dependence. In addition to [17], formula (1) has also been proven useful in many other link/path loss inference works [9], [8], [19], [13]. Our Internet experiments also show that the link loss dependence has little effect on the accuracy of (1).

We take logarithms on both sides of (1). Then by defining a column vector $x \in \mathbb{R}^s$ with elements $x_j = \log(1 - l_j)$, and writing v^T for the transpose of the column vector v, we can rewrite (1) as follows:

$$\log(1-p) = \sum_{j=1}^{s} v_j \log(1 - l_j) = \sum_{j=1}^{s} v_j x_j = v^T x. \tag{2}$$

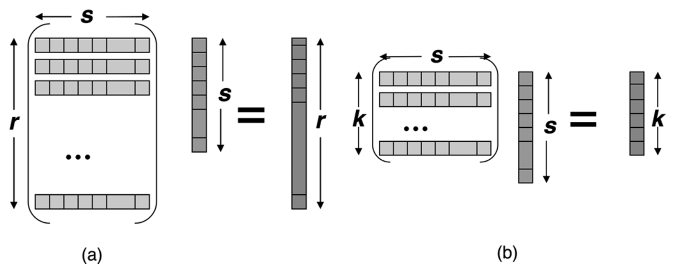


Fig. 2. Matrix size representations. (a) $Gx = b$. (b) $\bar{G}x_G = \bar{b}$.

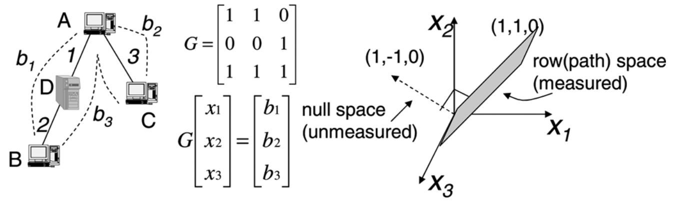

A
b2
b1
1
3
D
C
b3
2
B
$G=\begin{bmatrix}1&1&0\\0&0&1\\1&1&1\end{bmatrix}$
$G\begin{bmatrix}x_1\\x_2\\x_3\end{bmatrix}=\begin{bmatrix}b_1\\b_2\\b_3\end{bmatrix}$
X2
(1,1,0)
(1,-1,0)
row(path) space (measured)
null space (unmeasured)
X1
X3

Fig. 3. Sample overlay network.

There are $r = O(n^2)$ paths in the overlay network, and thus there are r linear equations of the form (2). Putting them together, we form a rectangular matrix $G \in \{0,1\}^{r\times s}$. 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 end-to-end loss rate of 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 r equations in form (2) as

$$Gx = b. \tag{3}$$

Normally, the number of paths r is much larger than the number of links s [see Fig. 2(a)]. This suggests that we could select s paths to monitor, use those measurements to compute the link loss rate variables x, and infer the loss rates of the other paths from (3).

However, in general, G is rank deficient: i.e., $k = \text{rank}(G)$ and $k < s$. If G is rank deficient, we will be unable to determine the loss rate of some links from (3). These links are also called *unidentifiable* in network tomography literature [8].

Fig. 3 illustrates how rank deficiency can occur. There are three end hosts (A, B, and C) on the overlay, three links (1, 2, and 3) and three paths between the end hosts. We cannot uniquely solve x_1 and x_2 because links 1 and 2 always appear together. We know their sum but not their difference.

Fig. 3 illustrates the geometry of the linear system, with each variable x_i as a dimension. The vectors $\{\alpha[1\ -1\ 0]^T\}$ comprise $\mathcal{N}(G)$, the *null space* of G. No information about the loss rates for these vectors is given by (3). Meanwhile, there is an orthogonal *row(path) space* of G, $\mathcal{R}(G^T)$, which for this example is a plane $\{\alpha[1\ 1\ 0]^T + \beta[0\ 0\ 1]^T\}$. Unlike the null space, the loss rate of any vector on the row space can be uniquely determined by (3).

To separate the identifiable and unidentifiable components of x, we decompose x into $x = x_G + x_N$, where $x_G \in \mathcal{R}(G^T)$ is its projection on the row space and and $x_N \in \mathcal{N}(G)$ is its projection on the null space (i.e., $Gx_N = 0$). The decomposition of $[x_1\ x_2\ x_3]^T$ for the sample overlay is shown below.

$$x_G = \frac{(x_1 + x_2)}{2}\begin{bmatrix}1\\1\\0\end{bmatrix} + x_3\begin{bmatrix}0\\0\\1\end{bmatrix} = \begin{bmatrix}b_1/2\\b_1/2\\b_2\end{bmatrix} \tag{4}$$

$$x_N = \frac{(x_1 - x_2)}{2} \begin{bmatrix} 1 \\ -1 \\ 0 \end{bmatrix}. \tag{5}$$

Thus the vector x_G can be uniquely identified and contains all the information we can know from (3) and the path measurements. The intuition of our scheme is illustrated through *virtual links* in [1].

Because x_G lies in the k-dimensional space $\mathcal{R}(G^T)$, only k independent equations of the r equations in (3) are needed to uniquely identify x_G. We measure these k paths to 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. Next, we present more detailed algorithms.

B. Basic Static Algorithms

The basic algorithms involve two steps. First, we select a basis set of k paths to monitor. Such selection only needs to be done once at setup. Then, based on continuous monitoring of the selected paths, we calculate and update the loss rates of all other paths.

1) Measurement Paths Selection: To select k linearly independent paths from G, we use standard rank-revealing decomposition techniques [20], and obtain a reduced system, as follows:

$$\bar{G}x_G = \bar{b} \tag{6}$$

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. The equation is illustrated in Fig. 2(b) (compared with $Gx = b$).

As shown below, our algorithm is a variant of the QR decomposition with column pivoting [20, p. 223]. It incrementally builds a decomposition $\bar{G}^T = QR$, 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.

Algorithm 1: Path (row) selection algorithm

```
procedure SelectPath(G)
1 for every row(path) $v$ in $G$ do
2 $\hat{R}_{12} = R^{-T}\bar{G}v^T = Q^T v^T$
3 $\hat{R}_{22} = \|v\|^2 - \|\hat{R}_{12}\|^2$
4 if $\hat{R}_{22} \neq 0$ then
5     Select $v$ as a measurement path
6     Update $R = \begin{bmatrix} R & \hat{R}_{12} \\ 0 & \hat{R}_{22} \end{bmatrix}$ and $\bar{G} = \begin{bmatrix} \bar{G} \\ v \end{bmatrix}$
    end
  end
```

In general, the G matrix is very sparse; that is, there are only a few nonzeros per row. We leverage this property for speed-up. We further use optimized routines from the LAPACK library [21] to implement Algorithm 1 so that it inspects several rows at a time. The complexity of Algorithm 1 is $O(rk^2)$, and the constant in the bound is modest. The memory cost is roughly $k^2/2$ single-precision floating point numbers for storing the R factor. Notice that the path selection only needs to be executed once for initial setup.

2) Path Loss Rate Calculations: To compute the path loss rates, we must find a solution to the underdetermined linear system $\bar{G}x_G = \bar{b}$. The vector $\bar{b}$ comes from measurements of the paths. Zhang *et al.* report that path loss rates remain operationally stable in the time scale of an hour [22], so these measurements need not be taken simultaneously.

Given measured values for $\bar{b}$, we compute a solution x_G using the QR decomposition we constructed during measurement path selection [20], [23]. We choose the unique solution x_G with minimum possible norm by imposing the constraint $x_G = \bar{G}^T y$, where $y = R^{-1}R^{-T}\bar{b}$. Once we have x_G, we can compute $b = Gx_G$ and, from there, infer the loss rates of the unmeasured paths. The complexity for this step is only $O(k^2)$. Thus we can update loss rate estimates online, as verified in Sections VIII-D and IX-B.

IV. Scalability Analysis

An overlay monitoring system is scalable only when the size of the basis set k grows relatively slowly as a function of n. Given that the Internet has moderate hierarchical structure [24], [25] we proved that the number of end hosts is no less than half of the total number of nodes in the Internet. Furthermore, we proved that when all the end hosts are on the overlay network, $k = O(n)$ [1].

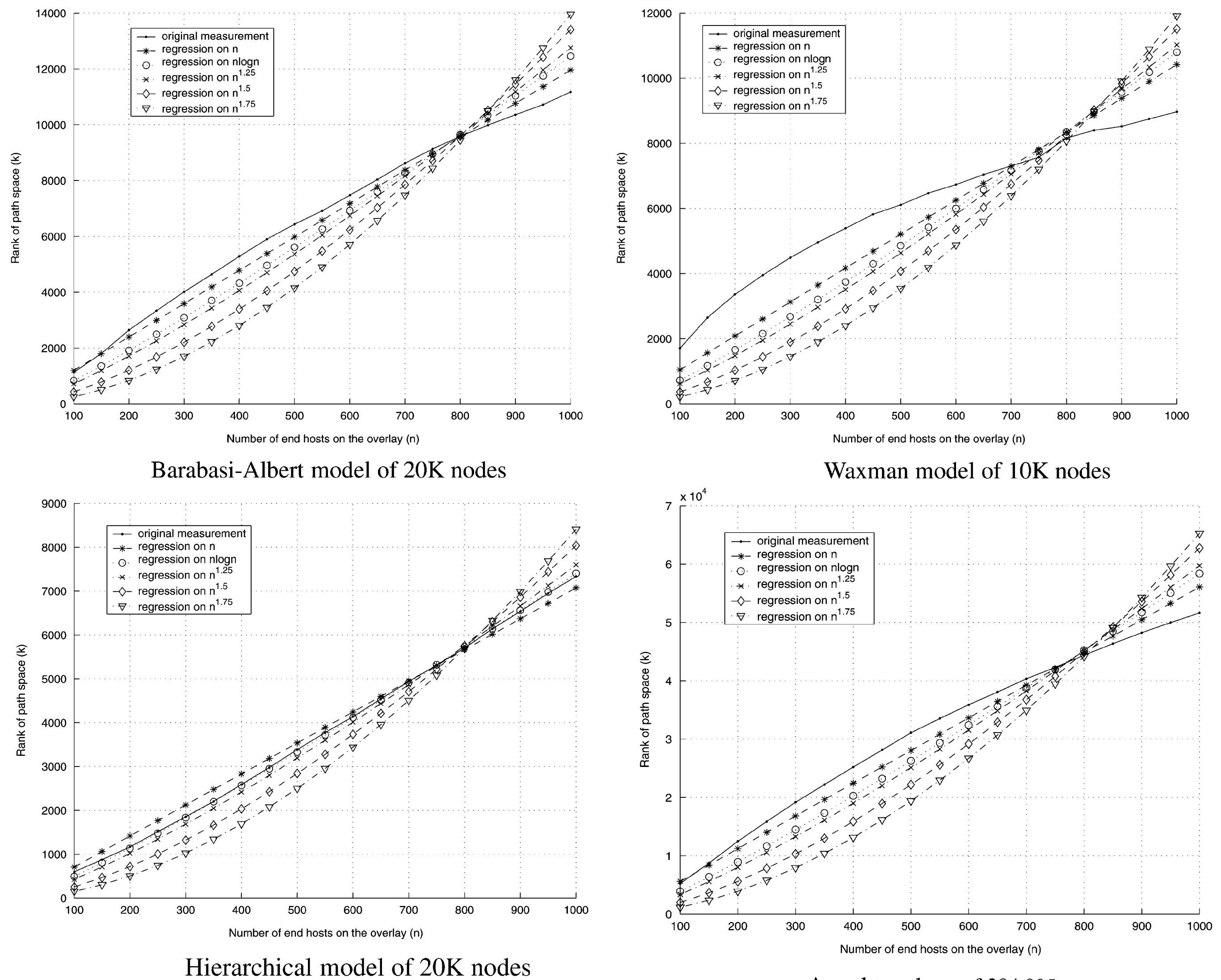
But what about if only a small fraction of the end hosts are on the overlay? Because G is an r by s matrix, k is bounded by the number of links s. If the Internet topology is a strict hierarchy like a tree, $s = O(n)$, thus $k = O(n)$. But if there is no hierarchy at all (e.g., a clique), $k = O(n^2)$ because all the $O(n^2)$ paths are linearly independent. Tangmunarunkit *et al.* found that the power-law degree Internet topology has moderate hierarchy [24]. It is our conjecture that $k = O(n \log n)$.

In this section, we first show through linear regression on both synthetic and real topologies that k is indeed bounded by $O(n \log n)$ for reasonably large n (e.g., 100). Then we explain it based on the power-law degree distribution of the Internet topology and the AS (autonomous system) hierarchy.

A. Empirical Bound on Rank K

We experiment with three types of BRITE [26] router-level topologies—Barabasi–Albert, Waxman, and hierarchical models—as well as with a real router topology with 284 805 nodes [27]. For hierarchical topologies, BRITE first generates an 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. We show one of them as an example because they all have similar trends (see [16] for complete results).

We randomly select end hosts that have the least degree (i.e., leaf nodes) to form an overlay network. We test by 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. 4, 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



Barabasi-Albert model of 20K nodes

Waxman model of 10K nodes

Hierarchical model of 20K nodes
(AS-level: Barabasi-Albert and router level: Waxman)

A real topology of 284,805 routers

Fig. 4. Regression of k in various functions of n under different router-level topologies.

topologies, $O(n)$ regression has the least residual errors—actually, k even grows slower than $O(n)$. The hierarchical models have higher k, and most of them have $O(n \log n)$ as the best fit.

Note that all functions seem to intersect around $n = 800$. This is because *inherently the growth is subquadratic, and roughly linear for the interested range of problem sizes*. Suppose that we have an exactly linear function x (we use a continuous setting for ease of calculation) and we wish to fit a function of the form $c \times x^a$ to this exactly linear function x over the interval [0,1]. The least squares procedure gives a coefficient of the form $c_{ls} = \frac{2+a}{2a+1}$. The data x and the function $c_{ls} \times x^a$ intersect at the following point:

$$x_{\text{intercept}} = \left(\frac{2+a}{2a+1}\right)^{\frac{1}{a-1}}. \tag{7}$$

For a between 1.1 and 2.0, $x_{\text{intercept}}$ varies between 0.728 and 0.800. That is, the fitted function intersects the data about 3/4 of the way across the domain for a wide range of exponents (including the exponents 1.25, 1.5, and 1.75).

Thus conservatively speaking, we have $k = O(n \log n)$.

B. Explanation From Internet Topology

Note that such a trend still holds when the end hosts are sparsely distributed in the Internet, e.g., when each end host is in a different access network. One extreme case is the "star" topology—each end host is connected to the same center router via its own access network. In such a topology, there are only n links. Thus $k = O(n)$. Only topologies with very dense connectivity, like a full clique, have $k = O(n^2)$. Those topologies have little link sharing among the end-to-end paths.

The key observation is that when n is sufficiently large, such dense connectivity is very unlikely to exist in the Internet because of the power-law degree distribution. Tangmunarunkit *et al.* found that link usage, as measured by the set of node pairs (source–destination pairs) whose traffic traverses the link, also follows a power-law distribution, i.e., there is a very small number of links that are on the shortest paths of the majority of node pairs. So there is significant amount of link sharing among the paths, especially for backbone links, customer links, and peering links.

Such link sharing can easily lead to rank deficiency of the path matrix for overlay networks. As an example, consider an overlay within a single AS. The AS with the largest number of links (exclusive of customer and peering links) in [28] has 5300 links. Even considering the coverage factor (55.6% as in Table II of [28]), there are at most 9600 links. Since there are $n(n-1)$ paths among n nodes, link sharing must occur before $n = 100$; in fact, substantial link sharing is likely to occur for even smaller n.

Now consider an overlay network that spans two ASs connected by y customer/peering links, with $n/2$ nodes in one AS and $n/2$ nodes in the other. The $n^2/2$ cross-AS paths can be modeled as a linear combination of $2y \times n + 2y$ virtual links—bidirectional links from each end host to its y peering link routers, and y bidirectional peering links. Thus given that y is normally much less than n and can be viewed as a constant, only $O(n)$ paths need to be measured for the $O(n^2)$ cross-AS paths.

Now consider an overlay on multiple ASs. According to [29], there are only 20 ASs (*tier-1* providers) which form the dense core of the Internet. These ASs are connected almost as a clique, while the rest of the ASs have far less dense peering connectivity. So when the size of an overlay is reasonably big (e.g., $n > 100$), the number of customer and peering links that cross-AS paths traverse tends to grow much slower than $O(n^2)$. For example, a joining end host may only add one customer link to the overlay topology and share the peering links that have been used by other end hosts. Meanwhile, only a few nodes are needed in a single AS before link sharing occurs in paths within an AS.

We believe this heavy sharing accounts for our empirical observation that $k = O(n)$ in a real router-level topology, and k grows at worst like $O(n \log n)$ in several generated topologies. Note that the real 284 805-router topology represents a fairly complete transit portion of the Internet [27]. In our analysis, we conservatively assume that there is only one end host connecting to each edge router to reduce the possible path sharing, but we still find $k = O(n)$ when $n > 100$.

V. Dynamic Algorithms for 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. In this section, we design efficient algorithms to incrementally adapt to these changes.

A. 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., in [30, Algorithm 3.4, 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 rescanning the rows of G as in Algorithm 1; however, this would take time $O(rk^2)$.

Algorithm 2: Path deletion algorithm

```
    procedure DeletePath($v, G, \bar{G}, R$)
1   if deleted path $v$ is measured then
2       $j$ = index of $v$ in $\bar{G}$
3       $y = \bar{G}^T R^{-1} R^{-T} e_j$
4       Remove $v$ from $G$ and $\bar{G}$
5       Update $R$ (Algorithm 3.4 in [30, p. 338])
6       $r = Gy$
7       if $\exists i$ such that $r_i \neq 0$ then
8           Add the $i$th path from $G$ to $\bar{G}$ (Algorithm 1, steps
            2–6)
        end
    end
9   else Remove $v$ from $G$
```

We now describe Algorithm 2 to delete a path v more efficiently. Suppose v corresponds to the ith row in $\bar{G}$ and the jth row in G, we define $\bar{G}' \in \mathbb{R}^{(k-1)\times s}$ as the measurement path matrix after deleting the ith row, and $G' \in \mathbb{R}^{(r-1)\times s}$ as the path matrix after removing the jth row. By deleting v from $\bar{G}$, we reduce the dimension of $\bar{G}$ from k to $k-1$. Intuitively, our algorithm works in the following two steps.

1) Find a vector y that only describes the direction removed by deleting the ith row of $\bar{G}$.
2) Test if the path space of G' is orthogonal to that direction, i.e., find whether there is any path $p \in G'$ that has a nonzero component on that direction. If not, no replacement path is needed. Otherwise, replace v with any of such path p, and update the QR decomposition.

Next, we describe how each step is implemented. To find y which is in the path space of $\bar{G}$ but not of $\bar{G}'$, we solve the linear system $\bar{G}y = e_i$, 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 $y = \bar{G}^T R^{-1} R^{-T} e_i$.

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.

Take the overlay network in Fig. 3. For example, suppose $\bar{G}$ is composed of the paths AB and BC, i.e.,

$$\bar{G} = \begin{bmatrix} 1 & 1 & 0 \\ 1 & 1 & 1 \end{bmatrix}.$$

Then we delete path BC, $\bar{G}' = [1 \quad 1 \quad 0]^T$ and

$$G' = \begin{bmatrix} 1 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}.$$

Applying Algorithm 2, we have $y = [0 \quad 0 \quad 1]^T$ and $r = [0 \quad 1]^T$. Thus the second path in G', AC, should be added to $\bar{G}'$. If we visualize such path deletion in reference to the geometry of the linear system, the path space of G' remains as a plane in Fig. 3, but $\bar{G}'$ only has one dimension of the path space left, so we need to add AC to $\bar{G}'$.

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. Therefore, the complexity of Algorithm 2 is $O(k^2)$.

B. End Hosts Join/Leave the Overlay

To add an end host h, we use Algorithm 1 to scan all the new paths from h, for a cost of $O(nk^2)$. However, it is inefficient to delete an end host h by directly using Algorithm 2 to delete all affected paths. If Algorithm 2 is used to delete a path that starts/ends at h, often another path that starts/ends at h is chosen as a replacement—and soon deleted in turn. To avoid this behavior, we remove all these paths from G first, then use the updated G in Algorithm 2 to select replacements as needed during the removal of paths that start/end at h. Each path in $\bar{G}$ can be removed in $O(k^2)$ time; the worst-case total cost of end host deletion is then $O(nk^2)$.

C. Routing Changes

In the network, routing changes or link failures can affect multiple paths in G. Previous studies have shown that end-to-end Internet paths generally tend to be stable for significant lengths of time, e.g., for at least a day [31], [32]. So we can incrementally measure the topology to detect 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. In addition to the periodic route measurement, if any path is found to have large loss rate changes, we will check its route instantly.

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.

Unlike $O(n^2)$ approaches (e.g., RON), we need some extra traceroute measurement. However, the key point is that the end-to-end routing remains much more stable than its loss rate, thus requires far less frequent measurement. So the savings on loss rate probing dwarf the traceroute overhead.

VI. Load Balancing and Topology Error Handling

To further improve the scalability and accuracy, we need to have good load balancing and handle topology measurement errors, as discussed in this section.

A. Measurement Load Balancing

The current design tends to have a few nodes measure most of the paths in the $\bar{G}$ which will overload these nodes and their access links, and further affect the measurement accuracy. To evenly distribute the measurements among the end hosts, we randomly reorder the paths in G before scanning them for selection in Algorithm 1. Since each path has equal probability of being selected for monitoring, the measurement load on each end host is similar. Note any basis set generated from Algorithm 1 is sufficient to describe all paths G. Thus the load balancing has no effect on the loss rate estimation accuracy.

B. Handling Topology Measurement Errors

As our goal is to estimate the end-to-end path loss rate instead of any interior link loss rate, we can tolerate certain topology measurement inaccuracies, such as incomplete routing information and poor router alias resolution.

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. For paths with incomplete routing information, we add links from where the normal route becomes unavailable (e.g., self loops or displaying "$* \; * \; *$" in traceroute), to where the normal route resumes or to the destination if such anomalies persist until the end. For instance, if the measured route is $(src, ip_1, ``* * *", ip_2, \text{dest})$, the path is composed of three links: $(src\ ip_1), (ip_1, ip_2)$, and (ip_2, dest). By treating the untraceable path (segment) as a normal link, the resulting topology is equivalent to the one with complete routing information for calculating the path loss rates.

For topologies with router aliases presenting one physical link as several links, we have little need to resolve these aliases. At worst, our failure to recognize the links as the same will result in a few more path measurements because the rank of G will be higher. For these links, their corresponding entries in x_G will be assigned similar values because they are actually a single link. Thus the path loss rate estimation accuracy is not affected, as verified by Internet experiments in Section IX. In addition, our system is robust to measurement node failures and node changes by providing bounds on the estimated loss rates.

VII. Sample Application: Streaming Media

In this section, we describe how real applications benefit from real-time path congestion/failure information provided by TOM. The sample application is streaming media delivery, which typically requires sustained network performance in terms of throughput, packet loss, and even latency for interactive applications. In contrast, the Internet provides unpredictable and time-varying service. Existing techniques to address the transport requirement mismatch fall in two categories: source-coding for compression [33] and error-resilience [34], and end-point adaptation, e.g., adjust the quality of the video based on the network throughput [33]. Both categories treat the underlying IP network as a best-effort black box.

Recently, studies have found that overlay routing can effectively improve the performance (e.g., latency, bandwidth, etc.) of IP routing [2], [16]. In this section, we design, implement and evaluate an adaptive live streaming media system that leverages TOM for real-time path congestion/failure information, and an overlay network for adaptive packet relaying and buffering within the delivery infrastructure. Specifically,

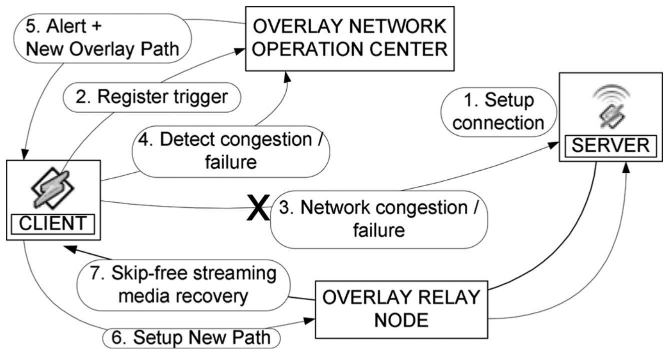

5. Alert + New Overlay Path
OVERLAY NETWORK OPERATION CENTER
2. Register trigger
1. Setup connection
4. Detect congestion / failure
SERVER
CLIENT
3. Network congestion / failure
7. Skip-free streaming media recovery
OVERLAY RELAY NODE
6. Setup New Path

Fig. 5. Event-driven diagram of adaptive overlay media streaming.

streaming clients in our system employ overlay routing to bypass faulty or slow links and re-establish new connection to streaming servers.

A. Architecture and Implementation

The system is composed of streaming media clients, server, and a set of overlay nodes (e.g., content distribution network). All these nodes are the end hosts monitored by TOM, and controlled by an ONOC as in Fig. 1. Normally a client connects to a server directly for streaming media content. The client also registers the path and sets up a trigger for path performance warnings at ONOC. When the path incurs congestion/failure, ONOC detects it, then searches for an alternative overlay path to bypass the faulty link(s), and sends the overlay path to the client if such path exists. The client tears down the current connection, sets up a new connection via overlay node(s), and attempts to concatenate the new streams with the old one for skip-free effect. For live streaming media or when the server is broadcasting the media to multiple clients, the reconnected client may lose part of the data. We apply a simple buffering technique to enable retransmission of lost packets during path switching (see [16] for details). The event driven diagram is shown in Fig. 5.

We add a buffering layer at the server and an overlay layer at the client to work with legacy client and server softwares. The architecture is shown in Fig. 6. Our implementation is built on top of Winamp [35] client and SHOUTcast [36] media server software. Media transport for SHOUTcast is carried using TCP. Nevertheless, our adaptive overlay routing and buffering techniques are applicable to other transport mechanisms such as RTP/UDP.

B. Evaluation

We deploy our system on the PlanetLab [37] testbed. We place ONOC at Stanford University, the Winamp client at U. C. Berkeley and the SHOUTcast server on 49 different locations as used for evaluation of TOM (see Section IX, excluding the Berkeley and Stanford hosts). The client is an Intel PIII/500 MHz Windows XP machine with 256 MB RAM on 100 Mb/s switched Ethernet. All other hosts are PlanetLab nodes, 1.0 GHz–1.8 GHz Linux machines with 512 MB-883 MB RAM.

Network congestion is introduced by using a Packeteer PacketShaper [38]. The normal streaming bitrate is about 300 Kb/s, which is less than the normal available bandwidth between client and server. During streaming, we use PacketShaper to set

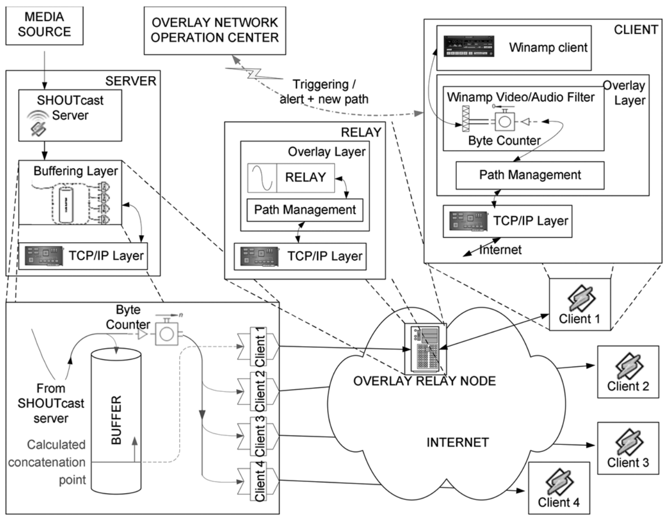


Fig. 6. Architecture of TOM-based adaptive overlay media streaming.

the bandwidth between the SHOUTcast server and the client as 76 Kb/s to emulate the congestion.

For the 51-node overlay network, TOM measures the selected paths (on average 872 paths, see Section IX) every 300 ms by sending out ten UDP packets along each path, then calculates the end-to-end loss rate using an exponential-weighted moving average (EWMA) for better stability. If the estimated loss rate exceeds a certain threshold (e.g., 10%), we assume that congestion occurs.

We evaluate two metrics: 1) the adaptation time, defined as the period from when congestion/failure occurs to when the client gets new streams from the overlay path and successfully concatenates those with the old in the buffer and 2) effectiveness of skip-free live media playback.

The adaptation time can be broken down into three parts as follows. We present the average running time with the SHOUTcast server on 49 different locations.

1) Congestion detection time, the time from introducing the bottleneck link via PacketShaper to when ONOC collects measurements, computes all the loss rates and detects the congestion, is 1.5 s on average.
2) ONOC searches for an overlay path from the server to the client that is relayed by a single overlay node and has the end-to-end loss rate less than 2% along with the smallest end-to-end delay. Then, it sends such path to the client. The total time for this step until the overlay path information is received by the client, is 0.66 s on average.
3) The client tears down the old connection, sets up a new connection to the server via overlay node, retrieves and concatenates the new media data. The average time for this step is 0.73 s.

The total adaptation time on average is less than three seconds. More than 95% of the experiments have adaptation time within five seconds. Since the maximum streaming bit rate for the DSL/cable modem is 450 Kb/s [39], a buffer of 300 KB suffices for skip-free media playback.

For the second metric, we compare to the streaming media system without monitoring-based adaptation. When congestion occurs, its Winamp client gradually runs out of buffer, and stalls.

In contrast, for all experiments, our client adapts to the congestion transparently, and enjoys a skip-free continuous playback without quality degradation.

VIII. Evaluation

In this section, we present our evaluation metrics, simulation methodology and simulation results.

A. Metrics

The metrics include path loss rate estimation accuracy, variation of measurement loads among the end hosts, and speed of setup, update, and topology change adaptation.

To compare the inferred loss rate $\hat{p}$ with real loss rate p, we analyze both absolute error and error factor. The absolute error is $|p - \hat{p}|$. We adopt the error factor $F_\varepsilon(p, \hat{p})$ defined in [8] as follows:

$$F_\varepsilon(p, \hat{p}) = \max\left\{\frac{p(\varepsilon)}{\hat{p}(\varepsilon)}, \frac{\hat{p}(\varepsilon)}{p(\varepsilon)}\right\} \quad (8)$$

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 [8]. 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 [22]. 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.

There are two types of measurement load: 1) sending probes and 2) receiving probes and computing loss rates. The load reflects the CPU and uplink/downlink bandwidth consumption. For each end host h, its measurement load is linearly proportional to, and thus denoted by the number of monitored paths with h as sender/receiver. Then, we compute its variation across end hosts in terms of the *coefficient of variation* (CV) and the *maximum versus mean ratio* (MMR), for sending load and receiving load separately. The CV of a distribution x, defined as below, is a standard metric for measuring inequality of x, while the MMR checks if there is any single node whose load is significantly higher than the average load.

$$CV(x) = \frac{\text{standard deviation}(x)}{\text{mean}(x)}. \quad (9)$$

The simulations only consider undirected links, so for each monitored path, we randomly select one end host as sender and the other as receiver. This is applied to all simulations with or without load balancing.

B. Simulation Methodology

We consider the following dimensions for simulation.

- Topology type: Three types of synthetic topologies from BRITE (see Section VIII-C) and a real router-level topology from [27]. All the hierarchical models have similar results, we use Barabasi–Albert at the AS level and Waxman at the router level as the representative.
- Topology size: The number of nodes ranges from 1000 to 20 000.[2] 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 [9]. 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 $LLRD_1$ except for Section VIII-D.
- 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 independently 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 [40], the probability of remaining in bad state is set to be 35% as in [9]. 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 10 000 packets. Using the loss rates of selected paths as input, we compute x_G, then the loss rates of all other paths.

C. Accurate, Efficient Inference for Various Topologies

For all topologies in Section VIII-B, we achieve high-loss-rate estimation accuracy. Results for the Bernoulli and the Gilbert models are similar. Since the Gilbert loss model is more realistic, we plot the cumulative distribution functions (CDFs) of absolute errors and error factors with the Gilbert model in Fig. 7. For all the configurations, the absolute errors are less than 0.008 ,and the error factors 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. In Fig. 7, we notice that for the same network topology, the smaller an overlay network is, the more accurate overall inference it provides. This is because

[2]20 000 is the largest topology we can simulate on a 1.5 GHz Pentium 4 machine with 512 M memory.

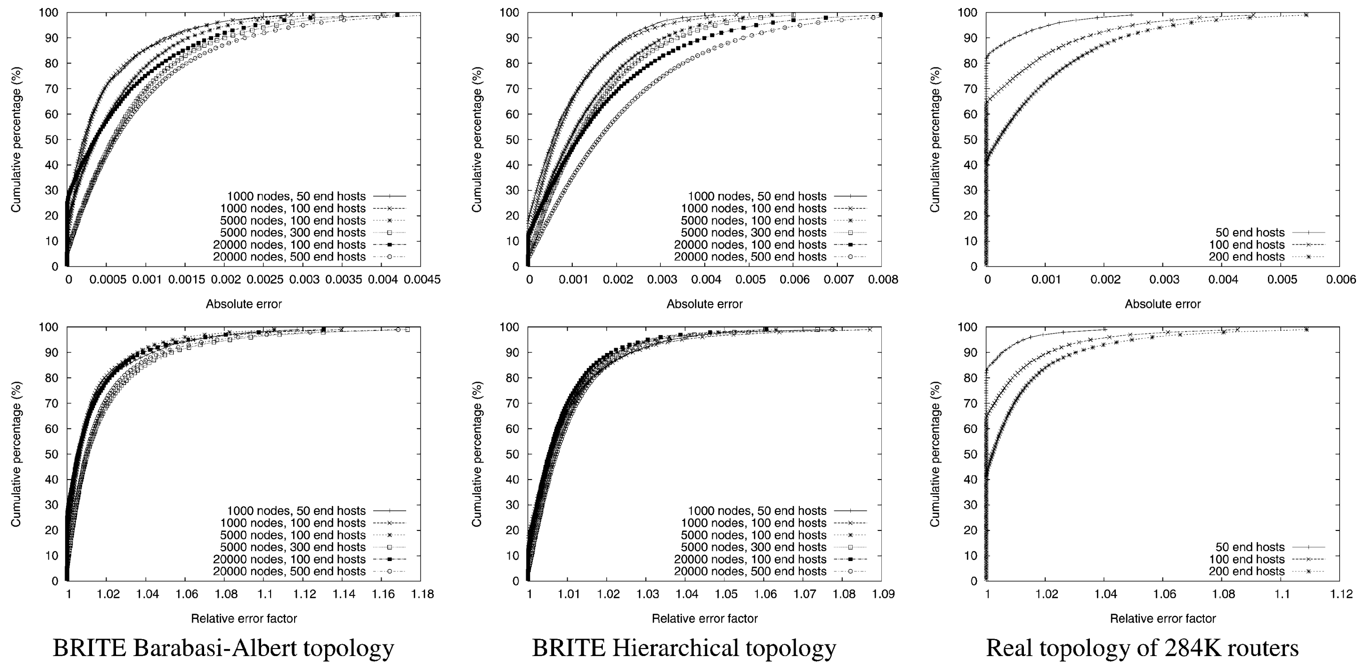


Fig. 7. Cumulative distribution of absolute errors (top) and error factors (bottom) under Gilbert loss model for various topologies.

TABLE II
RESULTS FOR TWO BRITE ROUTER TOPOLOGIES: BARABASI–ALBERT (TOP), HIERARCHICAL MODEL (MIDDLE), AND A REAL ROUTER TOPOLOGY OF 284 805 NODES (BOTTOM). OL: THE NUMBER OF END HOSTS ON THE OVERLAY. AP: THE NUMBER OF LINKS AFTER PRUNING THE NODES AND LINKS THAT ARE NOT ON THE OVERLAY PATHS. MPR: MONITORED PATH RATIO. FP: FALSE POSITIVE RATE

# of	# of end hosts		# of	# of links		rank	MPR	lossy paths (Bernoulli)			lossy paths (Gilbert)		
nodes	total	OL(n)	paths(r)	original	AP	(k)	(k/r)	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	# of end hosts		# of	# of links		rank	MPR	lossy paths (Bernoulli)			lossy paths (Gilbert)		
nodes	total	OL(n)	paths(r)	original	AP	(k)	(k/r)	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%

# of end hosts	# of	# of links	rank	MPR	lossy paths (Bernoulli)			lossy paths (Gilbert)		
on overlay (n)	paths(r)	after pruning	(k)	(k/r)	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%

that for a smaller overlay network, the measurement path ratio (MPR) (i.e., the percentage of paths we monitor out of all possible paths) is larger because the path sharing in the overlay is less. The MPR values are listed in the "MPR" columns in Table II. For those monitored paths, there is no estimation errors and thus a smaller overlay network has larger overall accuracy (for all the paths on the overlay).

Although different topologies have similar asymptotic regression trend for k as $O(n \log n)$, they have different constants. For an overlay network with given number of end hosts, the more IP links it spans on, the bigger k is. We found that Waxman topologies have the largest k among all synthetic topologies. 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 [27]. There are 65 801 end host routers and 860 683 links. We get the same trend of results as illustrated in Fig. 7 and Table II. The CDFs include all the path estimates, including the monitored paths for 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 Section IV, the growth of k is still bounded by $O(n)$.

D. Similar Results for 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

TABLE III
MEASUREMENT LOAD DISTRIBUTION. OL IS OVERLAY. "LB" AND "NLB" MEAN WITH AND WITHOUT LOAD BALANCING

# of nodes	OL size (n)	Barabasi-Albert model								hierarchical model							
		CV				MMR				CV				MMR			
		sender		receiver		sender		receiver		sender		receiver		sender		receiver	
		LB	NLB	LB	NLB	LB	NLB	LB	NLB	LB	NLB	LB	NLB	LB	NLB	LB	NLB
1000	50	0.62	1.10	0.56	0.94	2.41	5.91	3.07	4.09	0.52	0.96	0.53	0.87	2.28	4.80	2.51	4.29
	100	0.61	1.42	0.64	1.34	3.21	11.33	3.61	10.67	0.51	1.38	0.47	1.39	2.74	10.06	2.32	10.27
5000	100	0.44	0.89	0.47	0.97	2.25	6.11	2.36	6.50	0.49	1.18	0.53	1.39	2.60	9.18	2.97	10.16
	300	0.52	1.59	0.51	1.51	2.97	18.70	2.74	17.25	0.47	1.72	0.48	1.76	3.47	23.93	4.13	25.76
20000	100	0.36	0.55	0.40	0.59	1.93	3.20	2.29	3.69	0.48	1.17	0.43	1.09	3.04	8.86	2.56	7.09
	500	0.52	1.36	0.53	1.35	2.64	19.21	3.01	16.82	0.46	1.85	0.46	1.89	5.01	25.85	5.56	27.67

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. OL MEANS OVERLAY NETWORK

# of nodes	end hosts		lossy paths (Gilbert)			speed (second)	
	total	OL	real	coverage	FP	setup	update
1000	506	50	495	99.8%	1.1%	0.13	0.08
		100	1989	99.8%	3.0%	0.91	0.17
5000	2489	100	2367	99.6%	3.5%	1.98	0.22
		300	21696	99.2%	1.4%	79.0	1.89
20000	10003	100	2686	98.8%	1.1%	3.00	0.25
		500	67817	99.0%	4.6%	1250	4.33

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.

E. Fast Setup and Inference

The running time for $LLRD_1$ and $LLRD_2$ are similar, as in Table IV. All speed results in this paper are based on a 1.5 GHz Pentium 4 machine with 512 M memory. Note that it takes about 20 min to set up (select the measurement paths) for an overlay of 500 end hosts, but only several seconds for an overlay of size 100. The update (loss rate calculation) time is small for all cases, only 4.3 s for 124 750 paths. Thus, it is feasible to update online.

F. Effective Measurement Load Balancing

We examine the measurement load distribution for both synthetic and real topologies, and the results are shown in Table III. Given the space constraints, we only show the results for Barabasi–Albert and hierarchical model. Our load balancing scheme reduces CV and MMR substantially for all cases, and especially for MMR. For instance, a 500-node overlay on a 20 000-node network of the Barabasi–Albert model has its MMR reduced by 7.3 times.

We further plot the histogram of measurement load distribution by putting the load values of each node into ten equally spaced bins and counting the number of nodes in each bin as the y-axis. The x-axis denotes the center of each bin, as illustrated in Fig. 8. With load balancing, the histogram roughly follow the normal distribution. In contrast, the histogram without load balancing is close to an exponential distribution. Note that the y-axis in this plot is logarithmic: an empty bar means that the bin contains one member, and 0.1 means the bin is empty.

G. Efficient Incremental Update for Topology Changes

We study two common scenarios in P2P and overlay networks: end hosts joining and leaving as well as routing changes.

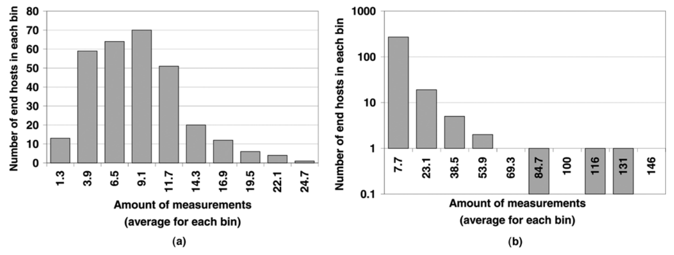


Fig. 8. Histogram of the measurement load distribution (as sender) for an overlay of 300 end hosts on a 5000-node Barabasi–Albert topology. (a) With load balancing and (b) without load balancing.

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)"

# 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%

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 average running time for adding a path is 125 ms, and for adding a node, 1.18 s. Notice that we add a block of paths together to speedup adding node (Section III-B).

Similarly, for removing end hosts, we start with an overlay network of 60 random end hosts, then randomly select an end host to delete from the overlay, and repeat the process until the size of the overlay is reduced to 55 and 50. Again, the accumulated numerical error is negligible as shown in Table VI. As shown in Section V, deleting a path in $\bar{G}$ is much more complicated than adding a path. With the same machine, the average time for deleting a path is 445 ms, and for deleting a node, 16.9 s. We note that the current implementation is not optimized: we can speed up node deletion by processing several paths simultaneously, and we can speed up path addition and deletion with iterative methods such as CGNE or GMRES [41]. Since the time to add/delete a path is $O(k^2)$, and to add/delete a node is $O(nk^2)$, we expect our updating scheme to be substantially faster than the $O(n^2k^2)$ cost of reinitialization for larger n.

TABLE VI
SIMULATION RESULTS FOR DELETING END HOSTS ON A REAL ROUTER TOPOLOGY. FP IS THE FALSE POSITIVE RATE. DENOTED AS "−REDUCED_VALUE (TOTAL_VALUE)"

# of end hosts	# of paths	rank	lossy paths		
			real	coverage	FP
60	1770	1397.0	1180.3	99.9%	0.2%
-5 (55)	-285 (1485)	-245.3 (1151.7)	-210.0 (970.3)	99.8%	0.2%
-10 (50)	-260 (1225)	-156.7 (995.0)	-150.6 (819.7)	99.9%	0.1%

TABLE VII
SIMULATION RESULTS FOR REMOVING A LINK FROM A REAL ROUTER TOPOLOGY

# 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%
average running time	17.3 seconds

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 a link will not cause disconnection for any pair of overlay end hosts. Then, we assume that the link is broken, and reroute the affected path(s). Algorithms in Section V incrementally incorporate each path change. Averaged over three runs, results in Table VII show that we adapt quickly, and still have accurate path loss rate estimation.

We also simulate the topology changes by adding a random link on some path(s) of the overlay. The results are similar as above, so we omit them here for brevity.

IX. INTERNET EXPERIMENTS

A. Methodology

We implemented our system on the PlanetLab [37] testbed, and deployed it on 51 PlanetLab hosts from four continents, each from a different organization. There are 11 International hosts (all are universities) and altogether 18 non-edu hosts.

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 ms. During a trial, each host sends a 40-B UDP packet[3] to every other host. Usually, the hosts will finish sending before the 300 ms trial is finished. For each path, the receiver counts the number of packets received out of 300 to calculate the loss rate. Thus, the loss rate is measured over an interval of $0.3 \times 300 = 90$ s. We believe that such granularity can filter some highly transient congestions, and still be able to capture the relatively stable congestions so that we can select some alternative path to bypass it.

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. This is because when sending packets in a batch, the packets sent later are more likely to be dropped. Such random permutations are pregenerated 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.

[3]20-byte IP header + 8-B UDP header + 12-B data on sequence number and sending time.

Will the probing traffic itself cause losses? We performed sensitivity analysis on sending frequency as shown in Fig. 9. 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 $(51-1) \times \mathrm{sending\_freq} \times 40$ B/s. The number of lossy paths does not change much when the sending rate varies, except when the sending rate is over 12.8 Mb/s, since many servers can not sustain that sending rate. We choose a 300 ms sending interval to balance quick loss rate statistics collection with moderate bandwidth usage.

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. In fact, we achieve similar CV and MMR for measurement load distribution as in the simulation. Even for an overlay network of 400 end hosts on the 284 K-node real topology used before, $k = 18\,668$. If we reduce the measurement frequency to one trial per second, the traffic consumption for each host is $18668/400 \times 40$ B/s $= 14.9$ Kb/s, which is typically less than 5% of the bandwidth of today's "broadband" Internet links. We can use adaptive measurement techniques in [2] to further reduce the overheads.

B. Results

From June 24 to June 27, 2003, we ran the experiments 100 times, mostly during peak hours 9 a.m. to 6 p.m. PDT. Each experiment generates $51 \times 50 \times 300 = 765$ K UDP packets, totaling 76.5 M 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 nonlossy. 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 Section IX-B-2.

1) Accuracy and Speed: 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. 10 shows the CDFs for the coverage and the false-positive rate. Notice that 40 runs have 100% coverage, and 90 runs have coverage over 85%, while 58 runs have no false positives, and 90 runs have false positive rates less than 10%.

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 0.0058 for lossy paths. We pick the run with the worst accuracy in coverage (69.2%) and plot the CDFs of absolute errors and error factors in Fig. 11. Since we only use 300 packets

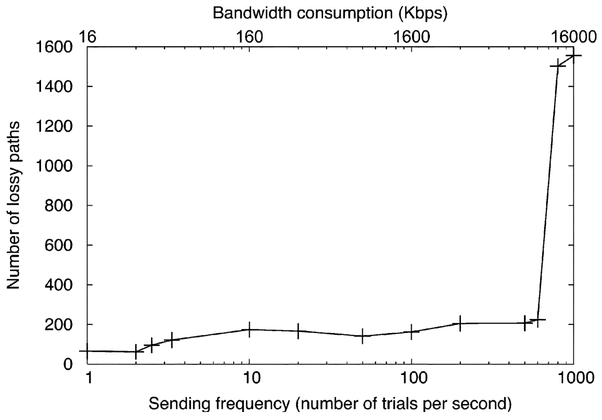


Fig. 9. Sensitivity test of sending frequency.

TABLE VIII
LOSS RATE DISTRIBUTION: LOSSY VERSUS NONLOSSY AND THE SUBPERCENTAGE OF LOSSY PATHS

[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%

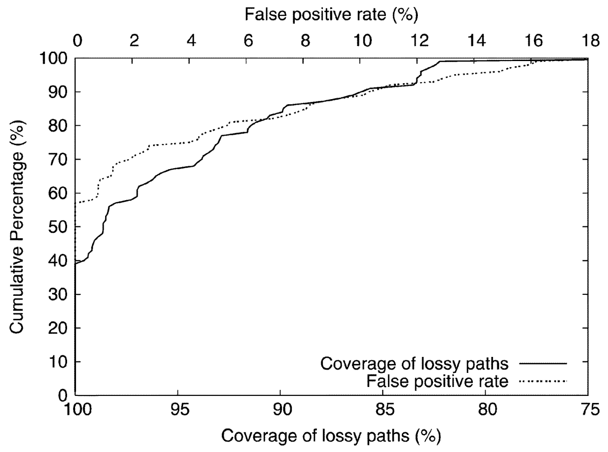


Fig. 10. Cumulative percentage of the coverage and the false positive rates for lossy path inference in the 100 experiments.

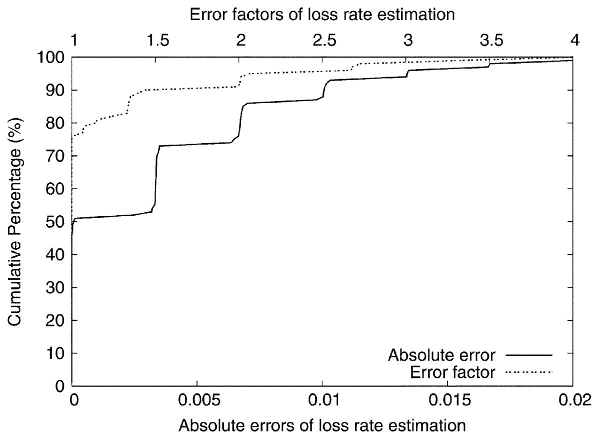


Fig. 11. Cumulative percentage of the absolute errors and error factors for the experiment with the worst accuracy in coverage.

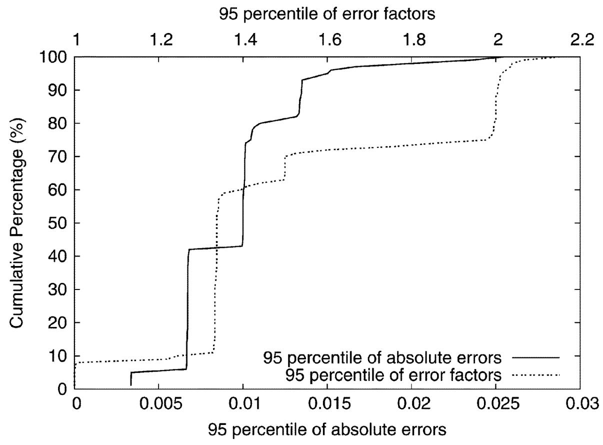


Fig. 12. Cumulative percentage of the 95 percentile of absolute errors and error factors for the 100 experiments.

to measure the loss rate, the loss rate precision granularity is 0.0033, so we use $\varepsilon = 0.005$ for error factor calculation. The average error factor is only 1.1 for all paths.

Even for the worst case, 95% of absolute errors in loss rate estimation are less than 0.014, and 95% of error factors are less than 2.1. To further view the overall statistics, we pick 95 percentile of absolute errors and error factors in each run and plot the CDFs on those metrics. The results are shown in Fig. 12. Notice that 90 runs have the 95 percentile of absolute errors less than 0.0133, and 90 runs have the 95 percentile of error factors less than 2.0.

The average running time for selecting monitoring paths based on topology measurement is 0.75 s, and for loss rate calculation of all 2550 paths is 0.16 s.

2) Topology Error Handling: The limitation of traceroute, which we use to measure the topology among the end hosts, led to many topology measurement inaccuracies. As found in [42], many routers on the paths among PlanetLab nodes have aliases. We did not use sophisticated techniques to resolve these aliases. Thus, the topology we have is far from accurate. Furthermore, in the PlanetLab experiments, some nodes were down or were unreachable from certain nodes. 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. The accurate loss rate estimation results show that our topology error handling is successful.

X. CONCLUSION AND FUTURE WORK

In this paper, we design, implement and evaluate an algebraic approach for adaptive scalable overlay network monitoring. For an overlay of n end hosts, we selectively monitor a basis set of $O(n \log n)$ paths, which can fully describe all the $O(n^2)$ paths. Then the measurements of the basis set are used to infer the loss rates of all other paths. Our approach works in real time, offers fast adaptation to topology changes, distributes balanced load to end hosts, and handles topology measurement errors. Both simulation and Internet implementation yield promising results.

The algebraic framework can inspire many future work. For instance, although we experimentally find that the rank of path matrix grows very slow as $O(n \log n)$ or even sublinearly for most topologies, it remains unknown how to model it stringently. How is such model related to the general Internet topology? In addition, Chua *et al.* have developed follow-up work to further select a smaller set of paths from $\bar{G}$ for monitoring with small sacrifice for the accuracy [43]. More recently, we investigated the link-level property inference problem under the algebraic framework [44].

REFERENCES

[1] Y. Chen, D. Bindel, and R. H. Katz, "Tomography-based overlay network monitoring," in *Proc. ACM SIGCOMM Internet Measurement Conf. (IMC)*, 2003.

[2] D. G. Andersen, "Resilient overlay networks," in *Proc. ACMSOSP*, 2001.

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

[4] S. Ratnasamy, "Topologically-aware overlay construction and server selection," in *Proc. IEEE INFOCOM*, 2002.

[5] P. Francis, "IDMaps: A global Internet host distance estimation service," *IEEE/ACM Trans. Netw.*, vol. 9, no. 5, pp. 525–540, Oct. 2001.

[6] Y. Chen, "On the stability of network distance estimation," in *ACM SIGMETRICS Performance Evaluation Rev. (PER)*, Sep. 2002.

[7] M. Coates, A. Hero, R. Nowak, and B. Yu, "Internet tomography," *IEEE Signal Process. Mag.*, vol. 19, no. 3, pp. 47–65, 2002.

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

[9] V. Padmanabhan, L. Qiu, and H. Wang, "Server-based inference of Internet link lossiness," in *Proc. IEEE INFOCOM*, 2003.

[10] D. Rubenstein, J. F. Kurose, and D. F. Towsley, "Detecting shared congestion of flows via end-to-end measurement," *ACM Trans. Netw.*, vol. 10, no. 3, 2002.

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

[12] H. C. Ozmutlu, "Managing end-to-end network performance via optimized monitoring strategies," *J. Netw. Syst. Manag.*, vol. 10, no. 1, 2002.

[13] C. Tang and P. McKinley, "On the cost-quality tradeoff in topology-aware overlay path probing," in *IEEE Proc. ICNP*, 2003.

[14] R. Mahajan, N. Spring, D. Wetherall, and T. Anderson, "User-level internet path diagnosis," in *Proc. ACMSOSP*, 2003.

[15] K. Anagnostakis, M. Greenwald, and R. Ryger, "Cing: Measuring network-internal delays using only existing infrastructure," in *Proc. IEEE INFOCOM*, 2003.

[16] Y. Chen, "Towards a scalable, adaptive and network-aware content distribution network," Ph.D. dissertation, Univ. of California at Berkeley, Berkeley, CA, Nov. 2003.

[17] R. Caceres, N. Duffield, J. Horowitz, and D. Towsley, "Multicast-based inference of network-internal loss characteristics," *IEEE Trans. Inf. Theory*, vol. 45, no. 7, pp. 2462–2480, Nov. 1999.

[18] S. Floyd and V. Jacobson, "Random early detection gateways for congestion avoidance," *IEEE/ACM Trans. Netw.*, vol. 1, no. 4, pp. 397–413, Aug. 1993.

[19] N. Duffield, "Multicast-based loss inference with missing data," *IEEE J. Sel. Areas Commun.*, vol. 20, no. 4, 2002.

[20] G. H. Golub and C. F. Van Loan*, Matrix Computations*. Baltimore, MD: The Johns Hopkins Univ. Press, 1989.

[21] E. Anderson*, LAPACK Users' Guide, Society for Industrial and Applied Mathematics*, 3rd ed. Philadelphia, PA: , 1999.

[22] Y. Zhang, "On the constancy of Internet path properties," in *Proc. SIGCOMMIMW*, 2001.

[23] J. W. Demmel*, Applied Numerical Linear Algebra*. Philadelphia, PA: SIAM, 1997.

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

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

[26] A. Medina, I. Matta, and J. Byers, "On the origin of power laws in Internet topologies," *ACM Comput. Commun. Rev.*, vol. 30, no. 2, Apr. 2000.

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

[28] N. Spring, R. Mahajan, and D. Wetherall, "Measuring ISP topologies with rocket fuel," in *Proc. ACM SIGCOMM*, 2002.

[29] L. Subrmanian, S. Agarwal, J. Rexford, and R. H. Katz, "Characterizing the Internet hierarchy from multiple vantage points," in *Proc. IEEE INFOCOM*, 2002.

[30] G. W. Stewart*, Matrix Algorithms: Basic Decompositions*. Philadelphia, PA: SIAM, 1998.

[31] V. Paxon, "End-to-end routing behavior in the Internet," *IEEE/ACM Trans. Netw.*, vol. 5, no. 5, pp. 601–615, Oct. 1997.

[32] Y. Zhang, V. Paxson, and S. Shenker, "The stationarity of internet path properties: Routing, loss, and throughput," ACIRI Tech. Rep., May 2000.

[33] T. Wiegand, "Overview of the H.264/AVC video coding standard," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 13, no. 7, pp. 560–576, 2003.

[34] S. Gringeri, R. Egorov, K. Shuaib, A. Lewis, and B. Basch, "Robust compression and transmission of MPEG-4 video," in *ACM MultiMedia*, 1999.

[35] *Winamp*, [Online]. Available: http://www.winamp.com, Nullsoft

[36] *Shoutcast*, [Online]. Available: http://www.shoutcast.com, Nullsoft

[37] *PlanetLab*, [Online]. Available: http://www.planet-lab.org

[38] *PacketShaper: Bandwidth Control and IP Management*, [Online]. Available: http://www.bandwidth-management.org, Workgroup Solutions

[39] *Pre-processing and Encoding: Making the Most of Your Bandwidth*, [Online]. Available: http://service.real.corn/learnnav/pptht.html, RealOne Player

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

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

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

[43] D. B. Chua, E. D. Kolaczyk, and M. Crovella, "Efficient monitoring of end-to-end network properties," in *Proc . IEEE INFOCOM*, 2005.

[44] Y. Zhao, Y. Chen, and D. Bindel, "Towards deterministic network diagnosis," in *Proc. ACM SIGCOMM*, 2006.

Yan Chen (M'03) received the Ph.D. degree in computer science from the University of California at Berkeley in 2003.

He is currently an Assistant Professor in the Department of Electrical Engineering and Computer Science at Northwestern University, Evanston, IL. His research interests include network measurement, monitoring, and security, for both wired and wireless networks.

Dr. Chen has been a Member of the Association for Computing Machinery (ACM) since 2003. He won the DOE Early CAREER award in 2005 and the Microsoft Trustworthy Computing Awards in 2004 and 2005.

David Bindel received his the Ph.D. degree in computer science from the University of California at Berkeley.

He joined the Courant Institute at New York University, New York, as a Courant Instructor in 2006. His research interests include numerical linear algebra, finite-element analysis, and modeling of microelectromechanical systems and of computer networks.

Han Hee Song received the Bachelor's degree in computer science from Yonsei University, Seoul, Korea, in 2004 and the Master's degree in computer science from the University of Texas at Austin (UT Austin) in 2006. He is currently working towards the Ph.D. degree at UT Austin.

His research interests are in the areas of network measurement, traffic engineering, and wireless networks.

Randy H. Katz (F'96) is the United Microelectronics Corporation Distinguished Professor in Electrical Engineering and Computer Science at the University of California at Berkeley. He has published over 250 refereed papers, book chapters, and books. His current research interests are reliable, adaptive distributed systems supported by new services deployed in the network.

Prof. Katz is a Fellow of the Association for Computing Machinery (ACM) and a member of the National Academy of Engineering and the American Academy of Arts and Sciences.