An Algebraic Approach to Adaptive Scalable Overlay Network Monitoring

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/failures. Our earlier extended abstract [1] briefly proposes an algebraic approach that selects and monitors k linearly independent paths that can fully describe all the $O(n^2)$ paths. The loss rates (this can also be extended to latency) of these paths can be used to estimate the loss rates 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 which shows that for reasonably large n (*e.g.*, 100), k is only in the range of $O(n \log n)$, ii) efficient adaptation algorithms for topology changes, such as the addition/removal of end hosts and interconnections, iii) measurement load balancing scheme for better scalability, iv) topology measurement error handling for better accuracy. Both simulation and Internet experiments demonstrate we achieve high path loss rate estimation accuracy while adapting to topology changes within seconds and handling topology errors. We can also continuously update the loss rate estimates online. For example, in the Internet experiments, the average update time is 0.16 second for all 2550 paths, the absolute error of loss rate estimation is 0.0027 and the average error factor is 1.1.

1. 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, thus can benefit from end-to-end network distance estimation (*e.g.*, latency and loss rate).

Accurate loss rate 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. For instance, existing streaming media systems treat the underlying network as a best-effort black box, thus adaptations are limited and can only be performed at the transmission end-points. In contrast, the monitoring service can enable these systems to bypass faulty or slow links through overlay routing for continuous media transmission.

Thus it is desirable to have a scalable overlay loss rate monitoring system which is accurate and incrementally deployable. However, existing network distance estimation systems are insufficient for this end. They can be grouped into two categories based on the targeted metrics: *general metrics* [2] and *latency only* [3, 4, 5, 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 Sec. 2).

We formulate the targeted 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 an earlier extended abstract [1], we sketched the idea of 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.

In [1], we only briefly introduce the basic formulation and model. Many questions remain as follows:

- How scalable is the system? In other words, how will k grow as a function of n? It is our conjecture that for reasonably large n (say 100), $k = O(n \log n)$ [1].
- In an overlay network, end hosts frequently join/leave the overlay and routing changes can happen from time to time, how to adapt to these efficiently without setting up the system from scratch?
- How to distribute the measurement load evenly among the end hosts to improve the scalability? How to handle topology measurement errors for better accuracy?
- How dose TOM perform under various topologies and loss conditions as well as the real Internet?

To address these issues, in this paper, we first improve TOM in several directions to enhance its accuracy, scalability, and make it dynamically adaptive to network changes. Then we implement and extensively evaluate the system. In particular, we make the following contributions.

- To demonstrate its scalability, we show that k does grow as $O(n \log n)$ through extensive linear regression tests on both synthetic and real topologies.
- To adapt to changing network topology, efficient algorithms are designed so that for each path addition and deletion, it only costs $O(k^2)$ time instead of $O(n^2k^2)$ to completely reinitialize.
- To improve the scalability and accuracy, we propose schemes for measurement load balancing and topology measurement error handling.

- We evaluate TOM through extensive simulation, as well as Internet experiments to validate the simulation.

Both simulation and PlanetLab experiments results show that we achieve high accuracy when estimating path loss rates with $O(n \log n)$ 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, although about 10% of the paths have no or incomplete routing information. The average setup (monitoring path selection) time is 0.75 second, and the online update of the loss rates for all 2550 paths takes only 0.16 second. In addition, we adapt to topology changes within seconds without sacrificing accuracy. The measurement load balancing reduces the load variation and the maximum vs. mean load ratio significantly, by up to a factor of 7.3.

The rest of the paper is organized as follows. We survey related work in Sec. 2, describe our model and basic static algorithms in Sec. 3, and evaluate its scalability in Sec. 4. we extend the algorithms to adapt to topology changes in Sec. 5, and to handle overloading and topology measurement errors in Sec. 6. The methodology and results of simulation are described in Sec. 7, and those of Internet experiments are presented in Sec. 8. Finally, we conclude in Sec. 9.

2. RELATED WORK

There is a lot of prior work on scalable end-to-end latency estimation. They are either *clustering-based* [5, 6] or *coordinate-based* [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 congestions 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 tree-like structure), and try to infer link characteristics [8, 9] or shared congestion [10] in the middle of the network. However, the problem is under-constrained: 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. Neither do they examine the topology measurement errors or the topology change problems.

Recently, Ozmutlu *et al.* selected a minimal subset of paths to cover all links for monitoring, assuming link-by-link latency is available via end-to-end measurement [12]. Their approach has the following limitations: 1) Traceroute cannot give accurate link-by-link latency. Many routers in the Internet hide their identities. 2) It is not applicable for loss rate, because it is difficult to estimate link-by-link loss rates from end-to-end measurements. Many applications, like streaming media and multiplayer gaming, are more sensitive to loss rate than latency. 3) It assumes static routing paths and does not consider topology changes.

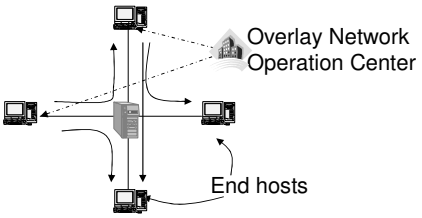


Figure 1: Architecture of a TOM system.

3. THE ALGEBRAIC MODEL AND BASIC ALGORITHMS

In this section, we briefly describe the algebraic model of TOM and its basic static algorithms.

3.1 Algebraic Model and Intuition

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 used 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 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.

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))

Table 1: Table of notations

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

The Internet topology can be much more complicated than the example in Fig. 1. Here we introduce the algebraic model that works with any topology. 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 any path represented by v has its loss rate p as

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

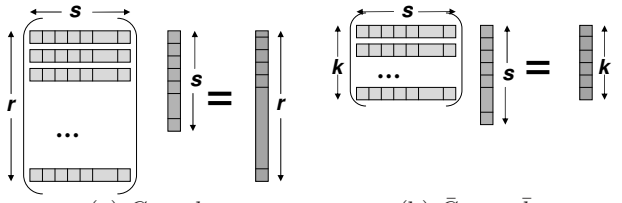
We take logarithms on both sides of (1). Then by defining a column vector $x \in \mathbb{R}^s$ with elements $x_j = \log(1 - l_j)$, and writing v^T as the transpose of the row vector v, we can rewrite (1) as follows:

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

There are $r = O(n^2)$ paths in the overlay network, thus r linear equations of the form (2). Putting them together, we form a rectangular matrix $G \in \{0,1\}^{r \times s}$ 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 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. Thus equation (3) can be visually represented as Fig. 2(a). This suggests that if we can select s paths to monitor, and then solve the link loss rate variables x, the loss rate of other paths can be inferred from (3).



(a) $Gx = b$ (b) $\bar{G}x_G = \bar{b}$

Figure 2: Matrix size representations.

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].

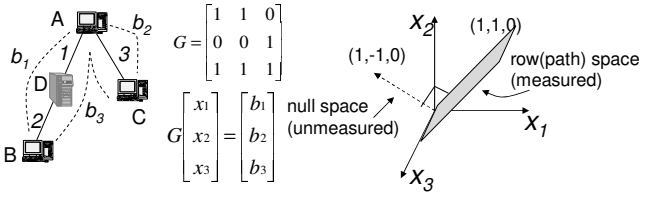


Figure 3: Sample overlay network.

We illustrate how rank deficiency can occur in Fig. 3. 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 can not uniquely solve x_1 and x_2 because links 1 and 2 always appear together. We know their sum, but not their difference.

When viewing the geometry of the linear system in Fig. 3 with each variable x_i as a dimension, there is a set of vectors $\alpha \begin{bmatrix} 1 & -1 & 0 \end{bmatrix}^T$ which are not defined by (3). This set of vectors is the *null space* of G ($\mathcal{N}(G)$). Meanwhile, there is an orthogonal *row(path) space* of G ($\mathcal{R}(G^T)$), which for this example is a plane determined by $x_1 + x_2$ and x_3. Unlike the null space, any vector on the row space can be uniquely determined by (3).

To separate the identifiable and unidentifiable components of x, we decompose x as $x = x_G + x_N$, 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 $[x_1 \quad x_2 \quad x_3]^T$ 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. Next, we examine x_G closely for the intuition.

From (4), there are two components in x_G: $x_1 + x_2$ and x_3. This suggests we can transform links 1 and 2 to a single *virtual link* with an identifiable loss rate $x_1 + x_2$ as in Fig. 4. Basically, the virtual links are the minimal path segments whose loss rates can be uniquely identified. Any end-to-end path can be represented as an unique linear combinations of the virtual links. In other words, the loss rates of the virtual links are sufficient to describe all the path loss rates.

A more interesting example is shown in the bottom figure of Fig. 4. There are four paths, but they are linearly dependent, so $\text{rank}(G) = 3$, and none of the link loss rates 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.

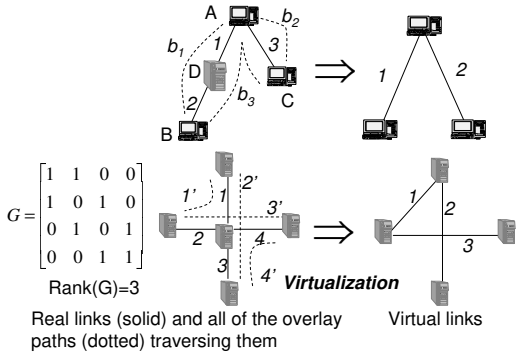


Figure 4: Virtualization of sample overlay paths.

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.

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. Then 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. 3, we only need to measure b_1 and b_2 to compute x_G, from which we can calculate b_3. Next, we will give more detailed algorithms.

3.2 Basic Algorithms

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

3.2.1 Measurement Paths Selection

To select k linearly independent paths from G, we use standard rank-revealing decomposition techniques [13], and obtain a reduced system:

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

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 [13, 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.

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

Algorithm 1: Path (row) selection algorithm

Note that the $\bar{G}$ selected may not be unique - it depends on the order of path scanning in Algorithm 1. Take the sample overlay network in Fig. 3 for example, if the order of path scan in step 1 of Algorithm 1 is AB, AC, BC, $\bar{G} = \begin{bmatrix} 1 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$. If the order is AB, BC, AC, $\bar{G} = \begin{bmatrix} 1 & 1 & 0 \\ 1 & 1 & 1 \end{bmatrix}$. These basis sets are equivalent on describing all end-to-end paths G. In Sec. 6, we make use of it for distributed load balancing.

In general, the G matrix is very sparse; that is, there are only a few nonzeros per row. We leverage that for speedup. We further uses optimized routines from the LAPACK library [14] 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 (see the running time results in Sec. 7.4 and 8.2). 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.

3.2.2 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 [15], so these measurements need not be taken simultaneously.

Given measured values for $\bar{b}$, we can compute one solution using the QR decomposition we constructed during measurement path selection [13, 16]. We choose the unique solution x_G with minimum possible norm by imposing the constraint that $x_G = \bar{G}^T y$ 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 Sec. 7.4 and 8.2.

4. SCALABILITY ANALYSIS

Such 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 [17], it is our conjecture that $k = O(n \log n)$ [1].

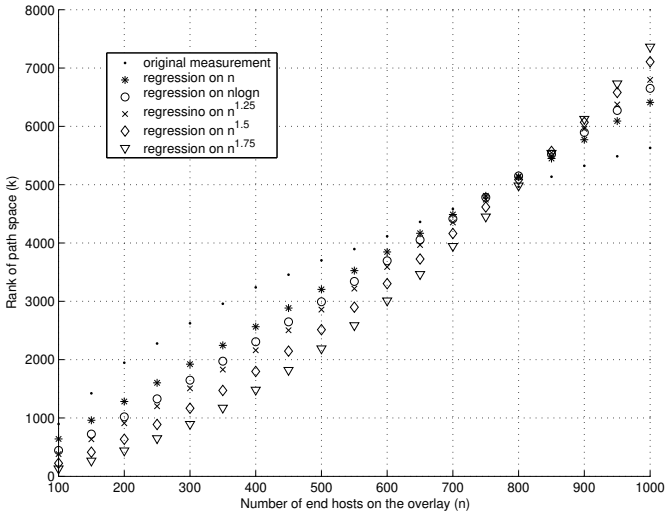
In this section, we will show through linear regression on both synthetic and real topologies that k is indeed bounded by $O(n \log n)$ for reasonably large n (*e.g*, 100). We experiment with three types of BRITE router-level topologies - Barabasi-Albert, Waxman and hierarchical models - as well as with a real router topology with 284,805 nodes [18]. 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. We show three of them except the Barabasi-Albert model for both AS and router levels which has very similar trend to the Waxman model for both AS and router levels.

We randomly select end hosts which have the least degree, 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. 5, 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 - 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. Conservatively speaking, we have $k = O(n \log n)$.

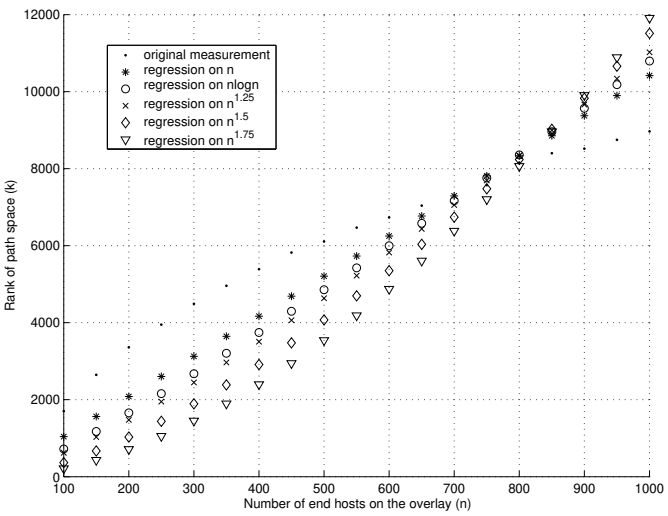
Note that such trend still holds when each end host is in a different access network. One extreme case is the "star" topology - each end host is connected to the same center router via its own access network. In such a topology, there are only n links. Thus $k = O(n)$. Only topologies with very dense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

5. DYNAMIC ALGORITHMS FOR TOPOLOGY CHANGES

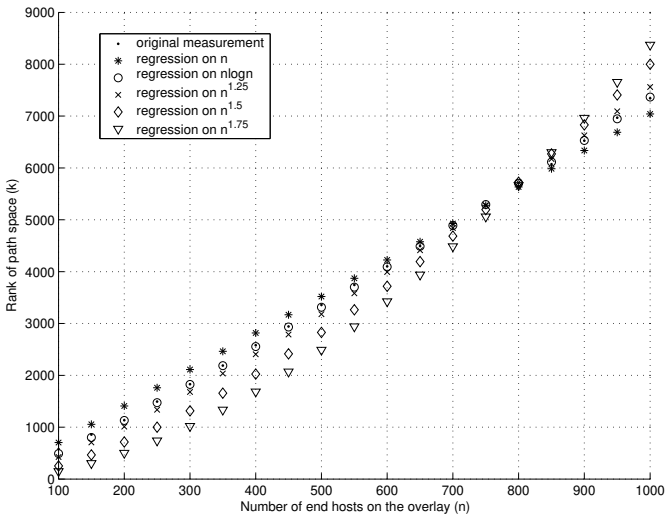
During normal operation, new links may appear or disappear, routing paths between end hosts may change, and hosts may enter or exit the overlay network. These changes may cause rows or columns to be added to or removed from



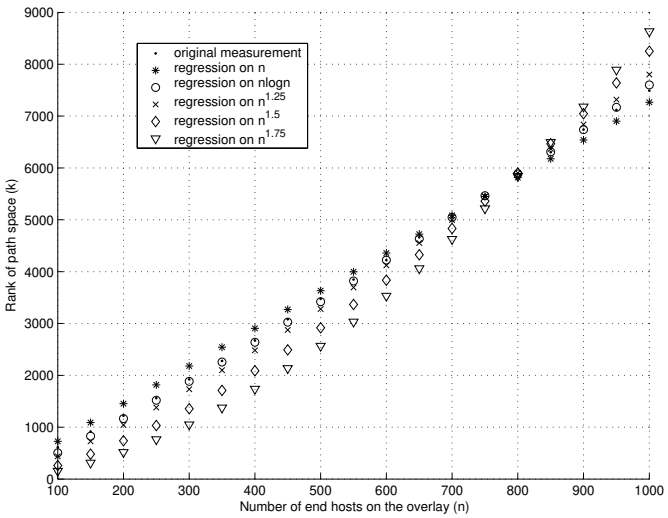
Barabasi-Albert model of 20K nodes



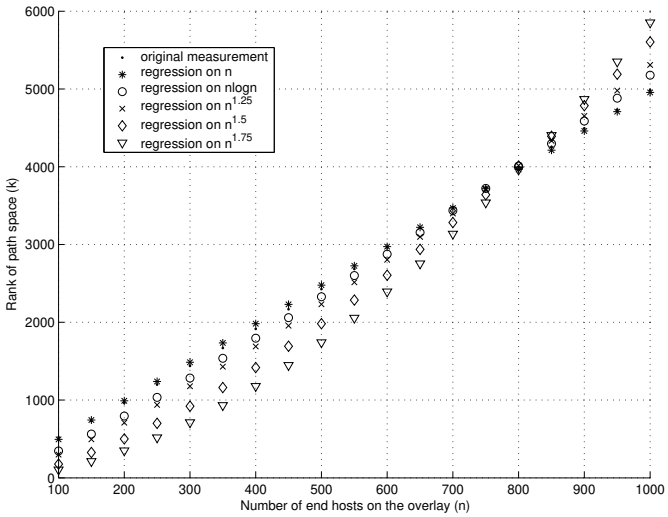
Waxman model of 20K nodes



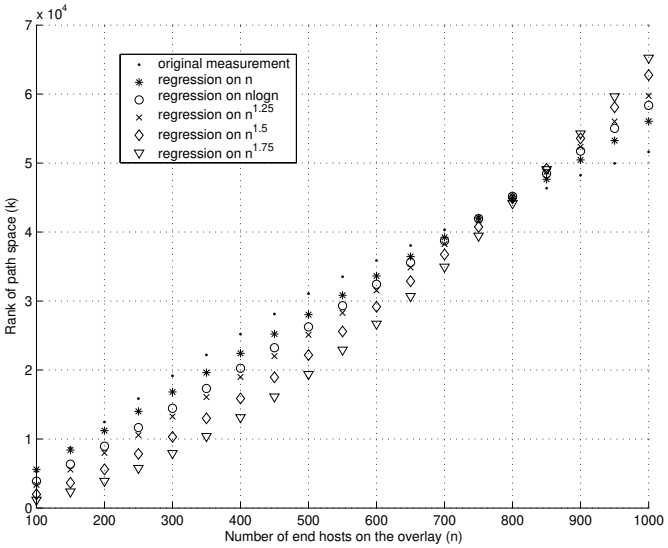
Hierarchical model of 20K nodes
(AS-level: Barabasi-Albert and router level: Waxman)



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



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



A real topology of 284K routers

Figure 5: Regression of k in various functions of n under different router-level topologies.

G, or entries in G may change. In this section, we design efficient algorithms to incrementally adapt to these changes.

5.1 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 [19, 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 $O(rk^2)$.

```
   procedure DeletePath(v, G, Ḡ, R)
1  if deleted path v is measured then
2      j = index of v in Ḡ
3      y = Ḡ^T R^{-1} R^{-T} e_j
4      Remove v from G and Ḡ
5      Update R (Algorithm 3.4 in [19, p.338])
6      r = Gy
7      if ∃ i such that r_i ≠ 0 then
8          Add the ith path from G to Ḡ (Algorithm 1,
           steps 2-6)
       end
   end
9  else Remove v from G
```

Algorithm 2: Path deletion algorithm

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 non-zero 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 view such path deletion with 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 $O(n^2 \times$ (average path length)) operations. Since most routing paths are short, the dominant cost will typically be the update of R. So the complexity of Algorithm 2 is $O(k^2)$.

5.2 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, to delete an end host h, to simply use Algorithm 2 to delete each path involving h is not efficient. These paths are often dependent to each other. Thus deleting one path form $\bar{G}$ often causes adding another to-be-removed path into $\bar{G}$. To avoid that, we remove all these paths from G first, then use the updated G in Algorithm 2 to remove each h-related path in $\bar{G}$. Each path in $\bar{G}$ can be removed in $O(k^2)$ time; the total cost of end host deletion is then at most $O(nk^2)$.

5.3 Routing Changes

In the network, routing changes or link failures can affect multiple paths in G. Most Internet paths remain stable over days [20]. 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.

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.

6. 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.

6.1 Measurement Load Balancing

To avoid overloading any single node or its access link, we 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 to be 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.

6.2 Handling Topology Measurement Errors

Because our goal is to estimate the end-to-end path loss rate instead of any interior link loss rate, we can handle 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 that only gives partial routes, 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 to the end. For instance, if the measured route is (src, ip_1, "* * *", ip_2, $dest$), 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 which may present one physical link as several links, we do not really have 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 is higher. For these links, their corresponding entries in x_G will be assigned similar values because they are really a single link. Thus the path loss rate estimation accuracy is not affected, as verified by Internet experiments in Sec. 8. In addition, our system is robust to measurement node failures and node changes by providing bounds on the estimated loss rates.

7. EVALUATION

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

7.1 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:

$$F_\varepsilon(p, \hat{p}) = \max\left\{\frac{p(\varepsilon)}{\hat{p}(\varepsilon)}, \frac{\hat{p}(\varepsilon)}{p(\varepsilon)}\right\} \quad (7)$$

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 [15]. 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 vs. 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.

$$CV(x) = \frac{standard\ deviation(x)}{mean(x)} \quad (8)$$

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.

7.2 Simulation Methodology

We consider the following dimensions for simulation.

- Topology type: three types of synthetic topologies from BRITE (see Sec. 7.3) and a real router-level topology from [18]. All the hierarchical models have similar results, we just use Barabasi-Albert at the AS level and Waxman at the router level as the representative.

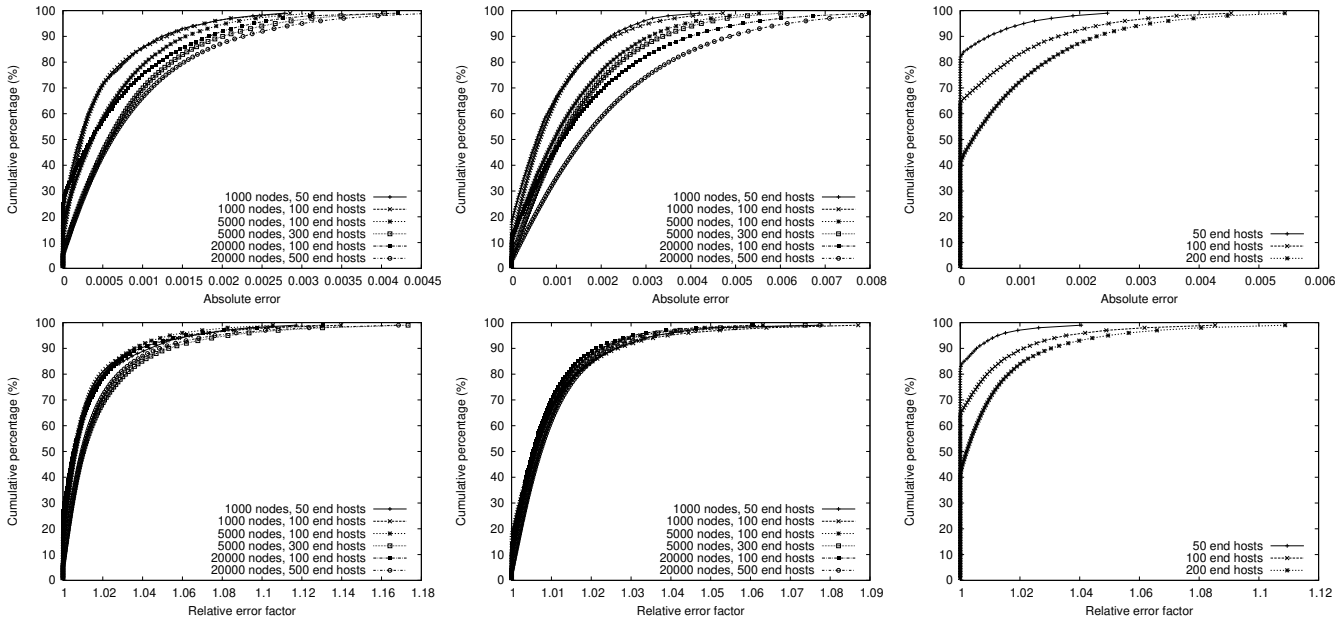
- Topology size: the number of nodes ranges from 1000 to 20000 [2]. Note that the node count includes both internal nodes (i.e., routers) and end hosts.

- 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 model $LLRD_1$ except for Sec. 7.4.

- 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 [21], 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

[2] 20000 is the largest topology we can simulate on a 1.5GHz Pentium 4 machine with 512M memory.



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

Figure 6: Cumulative distribution of absolute errors (top) and error factors (bottom) under Gilbert loss model for various topologies.

# 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 2: 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 (*i.e.*, remove the nodes and links that are not on the overlay paths). MPR is the monitored path ratio between our approach and the pair-wise scheme. FP is the false positive rate.

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.

7.3 Results for Different Topologies

For all topologies in Sec. 7.2, 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. 6. 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 2. 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. 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 [18]. There are 65,801 end host routers and 860,683 links. We get the same trend of results as illustrated in Fig. 6 and Table 3. 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 Sec. 4, the growth of k is still in the range of $O(n)$.

7.4 Results for Different Link Loss Rate Distribution and Running Time

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 4.

The running time for $LLRD_1$ and $LLRD_2$ are similar, as in Table 4. All speed results in this paper are based on a 1.5 GHz Pentium 4 machine with 512M memory. Note that it takes about 20 minutes to setup (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 seconds for 124,750 paths. Thus it is feasible to update online.

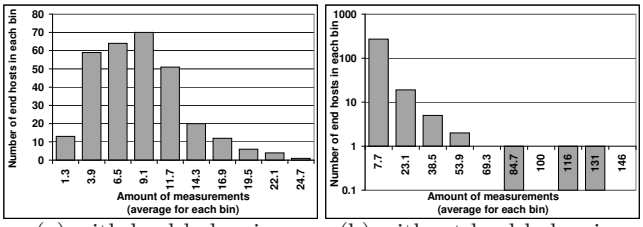
7.5 Results for Measurement Load Balancing

We examine the measurement load distribution for both synthetic and real topologies, and the results are shown in Table 5. 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 20000-node network of 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 10 equally spaced bins, and counting the number of nodes in each bin as y-axis. The x-axis denotes the center of each bin, as illustrated in Fig. 7. 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.

# of nodes	# of end hosts		lossy paths (Gilbert)			speed (second)	
	total	overlay	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

Table 4: Simulation results with model $LLRD_2$. Use the same Barabasi-Albert topologies as in Table 2. Refer to Table 2 for statistics like rank. FP is the false positive rate.



(a) with load balancing (b) without load balancing

Figure 7: Histogram of the measurement load distribution (as sender) for an overlay of 300 end hosts on a 5000-node Barabasi-Albert topology.

7.6 Results for Topology Changes

We study two common scenarios in P2P and 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.

7.6.1 End hosts join/leave

# 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 6: Simulation results for adding end hosts on a real router topology. FP is the false positive rate. Denoted as "+added_value (total_value)".

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 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 3: Simulation results for a real router topology. MPR and FP are defined the same as in Table 2.

# of nodes	OL size (n)	Barabasi-Albert model								hierarchical model							
		CV				MMR				CV				MMR			
		sender		recver		sender		recver		sender		recver		sender		recver	
		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 5: Measurement load (as sender or receiver) distribution for various BRITE topologies. OL Size is the number of end hosts on overlay. "LB" means with load balancing, and "NLB" means without load balancing.

# 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 7: Simulation results for deleting end hosts on a real router topology. FP is the false positive rate. Denoted as "-reduced_value (total_value)".

of the overlay reaches 45 and 50. Averaged over three runs, the results in Table 6 show that there is no obvious accuracy degradation caused by accumulated numerical errors. The average running time for adding a path is 125 msec, and for adding a node, 1.18 second. Notice that we add a block of paths together to speedup adding node (Sec. 3.2).

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 7. As shown in Sec. 5, 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 msec, and for deleting a node, 16.9 seconds. 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 [22]. 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 re-initialization for larger n.

7.6.2 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 re-route the affected path(s). Algorithms in Sec. 5 incrementally incorporate each path change. Averaged over three runs, results in Table 8 show that we adapt quickly, and still have accurate path loss rate estimation.

# 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

Table 8: Simulation results for removing a link from a real router topology.

8. INTERNET EXPERIMENTS

We implemented our system on the PlanetLab [23] testbed. In this section, we present the implementation results.

8.1 Methodology

We choose 51 PlanetLab hosts, each from a different organization as shown in Table 9. All the international PlanetLab 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 9: Distribution of selected PlanetLab 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 msec.

During a trial, each host sends a 40-byte UDP packet [3] 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. This is 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 performed sensitivity analysis on sending frequency as shown in Fig. 8. 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 balance quick loss rate statistics collection with moderate bandwidth consumption.

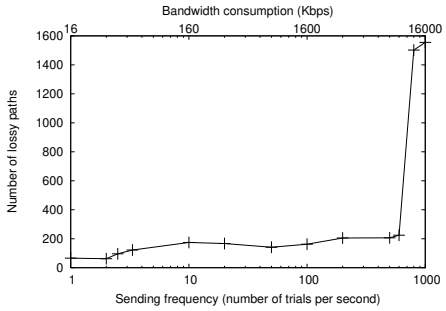


Figure 8: Sensitivity test of sending frequency

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. 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 284K-node real topology used before, k = 18668. If we reduce the measurement frequency to one trial per second, the traffic consumption for each host is 18668/400× 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 [2] to further reduce the overheads.

8.2 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 × 50× 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 10 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. 8.2.2.

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 10: Loss rate distribution: lossy vs. non-lossy and the sub-percentage of lossy paths.

8.2.1 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. 9 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%. 58 runs have no false positives and 90 runs have false positive rates less than 10%.

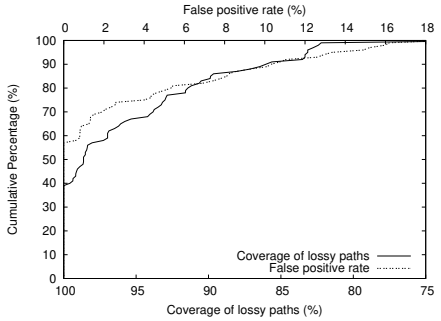


Figure 9: Cumulative percentage of the coverage and the false positive rates for lossy path inference in the 100 experiments.

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 0.0058 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. 10. 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 error factor calculation. The average error factor is only 1.1 for all paths.

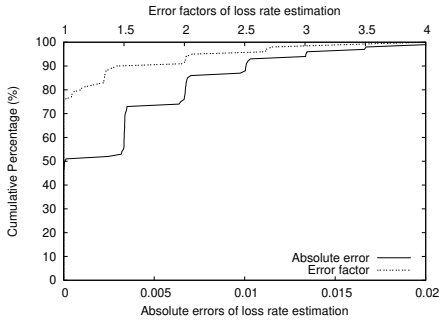


Figure 10: Cumulative percentage of the absolute errors and error factors for the experiment with the worst accuracy in coverage.

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

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. 11. 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.

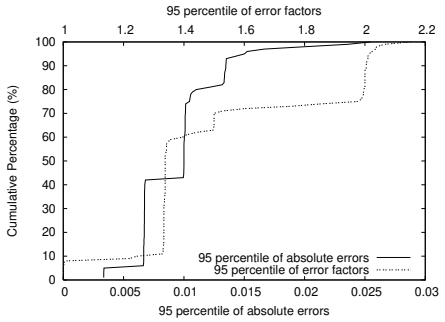


Figure 11: Cumulative percentage of the 95 percentile of absolute errors and error factors for the 100 experiments.

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

8.2.2 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 [24], many of the 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. But we are still 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. The accurate loss rate estimation results show that our topology error handling is successful.

9. CONCLUSIONS

In this paper, we improve, implement and evaluate an algebraic approach [1] 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 real Internet implementation show promising results.

For more efficient monitored path selection, we plan to investigate the use of iterative methods [22], [25] 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. Furthermore, we are building applications, such as streaming media, on top of such monitoring systems.

10. REFERENCES

[1] Extended abstract, ," anonymized for double-blind review.

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

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

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

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

[6] 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.

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

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

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

[10] 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.

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

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

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

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

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

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

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

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

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

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

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

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

[23] PlanetLab, "http://www.planet-lab.org/," .

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

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