Server-based Inference of Internet Performance

Venkata N. Padmanabhan
Lili Qiu
Helen J. Wang

May 2002

Technical Report
MSR-TR-2002-39

Microsoft Research
Microsoft Corporation
One Microsoft Way
Redmond, WA 98052

Server-based Inference of Internet Performance

Venkata N. Padmanabhan, Lili Qiu, and Helen J. Wang
{*padmanab,liliq,helenw*}*@microsoft.com*
Microsoft Research

Abstract—

We investigate the problem of inferring the packet loss characteristics of Internet links using server-based measurements. Unlike much of existing work on network tomography that is based on active probing, we make inferences based on *passive* observation of end-to-end client-server traffic.

We start with a brief analysis of end-to-end packet loss rate over wide-area Internet paths, as observed from a busy Web site. We find that the end-to-end packet loss rate correlates poorly with topological distance (i.e., hop count), remains stable for several minutes, and exhibits a limited degree of spatial locality. These findings suggest that passive network tomography would be both interesting and feasible.

Our work on passive network tomography focuses on *identifying* lossy links (i.e., the trouble spots in the network). We have developed three techniques for this purpose based on Random Sampling, Linear Optimization, and Bayesian Inference using Gibbs Sampling, respectively. We evaluate the accuracy of these techniques using both simulations and Internet packet traces. We find that these techniques can identify most of the lossy links in the network with a manageable false positive rate. For instance, simulation results indicate that the Gibbs sampling technique has over 80% coverage with a false positive rate under 5%. Furthermore, this technique provides a confidence indicator on its inference. In the case of Internet traces, validating the inferences is a challenging problem. We present a method for indirect validation, which suggests that the false positive rate is manageable.

I. Introduction

The Internet has grown rapidly in terms of size and heterogeneity in recent years. The set of hosts, links, and networks that comprise the Internet is diverse. This presents interesting challenges from the viewpoint of an Internet server, such as a Web site, whose goal is to provide the best possible service to its clients. A significant factor that the server must contend with is the dissimilar and changeable network performance experienced by clients.

The goal of our work is to investigate ways to infer the performance of the Internet by *passively* monitoring existing network traffic between a server and its clients. Our goal is to go beyond characterizing end-to-end network performance by developing techniques to infer the performance of interior links in the network.

There are a number of ways in which the server could benefit from such characterization and inference. Information on the stability or predictability of network performance to one or more clients could be used to adapt content for speedy delivery to the client(s) [21]. Information on bottlenecks or other hot spots within the network could be used to direct clients to replica servers so that they avoid the hot spot. Such information could also be used by a Web site operator to have the hotspot problem resolved in cooperation with the concerned ISP(s). The focus of this paper, however, is on the inference of network performance, not on its applications.

One question is what "network performance" means. Clearly, the performance metrics that matter depend on the application. Latency may be most critical in the case of game servers while throughput may be the most important metric for software download servers. In our study, we primarily focus on the packet loss rate because it is the most direct indicator of network congestion.[1] We view the packet loss rate and RTT metrics as being more fundamental than throughput since the latter is affected by factors such as the workload (e.g., bulk transfers versus short Web transfers) and the transport protocol (e.g., the specific variant of TCP). Furthermore, it is possible to obtain a rough estimate of throughput knowing the packet loss rate and RTT, using an analytical model of TCP such as [17].

Here is an overview of the rest of this paper. In Section II, we discuss related work. In Section III, we describe our experimental setup and methodology. We present details of the packet traces we gathered at the busy *microsoft.com* Web site.

We begin our analysis in Section IV by seeking answers to three questions pertaining to the end-to-end packet loss rate experienced by clients: (a) how well loss rate correlates with topological distance between the server and client, and (b) how stable the loss rate is over time, and (c) how strong the spatial locality in loss rate is. We find the correlation between end-to-end loss rate and hop count is weak, which suggests that the end-to-end path is dominated by a few lossy links. The end-to-end loss rate is stable for several minutes, suggesting that lossy links remain so for several minutes. Finally, there is a limited degree of spatial locality in end-to-end loss rate (especially at the subnet level), but in general the locality is not very strong. This suggests that while in some cases lossy links may be shared (i.e., such links may lie on the path from the server to multiple clients), often they are non-shared (e.g., the "last-hop" link to clients). These findings suggest that it would be interesting to try and identify the lossy links.

This sets the stage for our main focus, *Passive Network Tomography*, which we present in Section V. The goal here is to identify the lossy links in the interior of the network by *passively* observing the end-to-end performance of existing traffic between a server and its clients. This is in contrast to the previous work on network tomography (e.g., [4]) that has been based on active probing. We develop three techniques for passive network tomography: Random Sampling, Linear Optimization, and Bayesian Inference using Gibbs Sampling. We evaluate these techniques using extensive simulations and find that we are able to identify more than 80% of the lossy links with a false positive rate under 5%. We also apply these techniques to the traffic traces gathered at the *microsoft.com* site. Validation is challenging in this setting since we do not know the true loss rate of Internet links. We present a method for indirect validation, which suggests that the false positive rate is manageable.

[1] We have also done some characterization of the round-trip time (RTT) metric, but we do not present those results in this paper.

Finally, we present our conclusions in Section VI.

II. Related Work

There have been numerous studies of Internet performance. We can broadly classify these studies as either *active* or *passive*. Active studies involve measuring Internet performance by injecting traffic (in the form of pings, traceroutes, TCP connections, etc.) into the network. In contrast, passive studies analyze existing traffic obtained from server logs, packet sniffers, etc. Our study is a passive one.

Several studies (e.g., [18], [23]) have examined the temporal stability of Internet performance metrics through active measurements. In [18] Paxson reports that observing (no) packet loss along a path is a good predictor that we will continue to observe (no) packet loss along the path. However, the magnitude of the packet loss rate is a lot less predictable. Zhang *et. al.* examines the stationarity of packet loss rate and available bandwidth [23]. They find that the correlation in the loss process mainly comes only from back-to-back loss episodes, and not from "nearby" losses. Throughput has a close coupling with the loss process, and can often be modeled as a stationary IID process for periods of hours.

Several studies have also examined similar issues by studying traces gathered passively using a packet sniffer. The authors in [2] used traces from the 1996 Olympic Games Web site to analyze the spatial and temporal stability of TCP throughput. Using traceroute data, they constructed a tree rooted at the server and extending out to the client hosts. Clients were clustered together based on how far apart they are in the tree. The authors report that clients within 2-4 tree-hops of each other tend to have similar probability distributions of TCP throughput. They also report that throughput to a client host tends to remain stable (i.e., within a factor of 2) for time scales of many tens of minutes.

Packet-level traces have also been used to characterize other aspects of network traffic. In [1] Allman uses traces gathered at the NASA Glenn Research Center Web server to study issues such as TCP and HTTP option usage, RTT and packet size distributions, etc. Mogul *et al.* uses packet-level traces to study the effectiveness of delta compression for HTTP [15].

Our study is similar to [2] in that it is based on packet sniffer traces gathered passively at a busy server. However, our analysis is different in many ways. We focus on packet loss rate rather than TCP throughput for the reasons mentioned previously. Our analysis of spatial locality considers operationally meaningful entities such as autonomous systems (ASes) and BGP prefix clusters [14] rather than treating the network as an undifferentiated tree. And, most importantly, we try to infer the characteristics of internal links in the network rather than just the end-to-end characteristics.

This last aspect of our work lies in the area of *Network Tomography*, which is concerned with the inference of the internal network characteristics based on end-to-end observations. The observations can be made through *active* probing (either unicast or multicast probing) or *passive* monitoring. MINC [4] and [19] base their inference on loss experienced by multicast probe packets while [6], [7] use closely-spaced unicast probe packets striped across multiple destinations. A common feature of the above techniques is that they are based on *active* injection of probe packets into the network. Such active probing imposes an overhead on the network and runs the risk of altering the link characteristics, especially when applied on a large scale (e.g., on the path from a busy server to all of its clients).

In [20] and [13], the authors take a passive approach in detecting shared bottlenecks. The former requires senders to cooperate by time stamping the packets while the latter requires an observer that receives more than 20% of the output traffic of the bottleneck (i.e., light background traffic). Tsang et al. [22] estimate loss rate for each link by passively observing closely spaced packet-pairs. A problem, however, is that existing traffic often may not contain enough such packet-pairs to make an inference. Furthermore, their evaluation is based on very small topologies containing a dozen (simulated) nodes, and it is not clear how well their technique would scale to large topologies.

Date	Duration	# packets	# clients
Dec 20, 2000	2.12 hours	100.0 million	134,475
Jan 11, 2002	2.21 hours	125.0 million	945,986

TABLE I
SUMMARY OF THE TWO TRACES ANALYZED IN THIS PAPER.

III. Experimental Setup and Methodology

We now describe the experimental setup and methodology used in our study. The packet traces were gathered at the *microsoft.com* site, which is a busy corporate Web site that sees a large number of long TCP connections (e.g., software downloads) from users across the globe. We used the *tcpdump* tool [12] on a Pentium-III 550 MHz PC running the Windows 2000 Server OS to do the packet capture. This machine was connected to the spanning (replication) port of a Cisco Catalyst 6509 switch via a 100 Mbps Ethernet link. With port replication turned on, our packet sniffer saw traffic to/from several server nodes that were connected either to the same Catalyst switch or to a sister switch in a two-switch cluster. The packet drop rate on the replicated port on the switch was no more than 0.3%.

For our study, we only captured (the headers of) TCP packets since we are able to estimate packet loss rate by observing TCP data packets and the corresponding ACKs. With the setup described above, we were able to capture a portion of the traffic entering and leaving the *microsoft.com* site. This included Web traffic, software download traffic, and streaming media traffic (including streaming media over TCP to traverse firewalls). Due to cluster-level load balancing, our packet sniffer did not necessarily see all connections to/from a particular client. For a particular connection, however, it either saw *all* of the packets or none at all. So we are able to derive meaningful estimates of packet loss rate from the subset of connections that were captured. Table I summarizes the two traces we analyze in this paper.

We used the *traceroute* [11] tool to determine the network path from the *microsoft.com* site to each of the clients seen in the traces. The traceroute data was collected in the few days following the trace capture. Due to security and administrative concerns, the packet sniffer machine located in the data center was

configured to be in "listen-only" mode. So the traceroutes were run from a FreeBSD PC located on a separate *microsoft.com* network. While the first few hops within the corporate network were different, the entire external path was identical to the path that packets from the server nodes located in the data center would have taken. So these traceroutes help us determine the wide-area Internet path from the server cluster to the clients.

Since our packet sniffer is located very close to the server nodes, we detect packet losses by looking for packet retransmissions by the sender. The underlying assumption is that the TCP sender only retransmits a packet if the original transmission was lost, which is reasonable since TCP is conservative about retransmissions. We compute the loss rate for client node as the ratio of the number of retransmitted packets to the total number of packets sent to it within a window of time. As explained in Section IV, we varied the window size when studying temporal locality.

Our analysis shows that clients experience widely different loss rates. In the Dec. 2000 trace, around 40% of the clients experience no loss, and 6% of the clients suffer from more than 20% loss, when the loss rate is computed using the entire trace period, while in the Jan. 2002 trace, half the clients have no loss, and above 10% of the clients encounter over 20% loss.

IV. Analysis of End-to-End Loss Rate

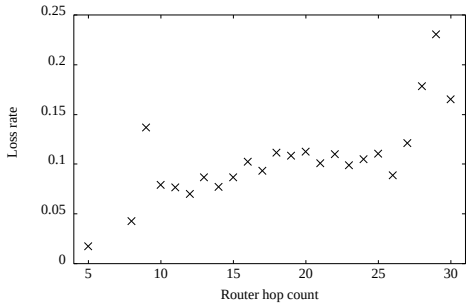
In this section, we analyze the end-to-end loss rate information derived from the traffic traces with a view to answering three questions: (a) how well loss rate correlates with topological distance between the server and client, and (b) how stable the loss rate is over time, and (c) how strong spatial locality in loss rate is. This analysis is not the primary objective of this paper; indeed, some of our findings confirm previous findings albeit in a different setting (i.e., server-based passive measurements rather than active or client-based measurements). Our goal here is to motivate the primary focus of our work, Passive Network Tomography, which is presented in Section V.

A. Correlation between Topological Distance and Loss Rate

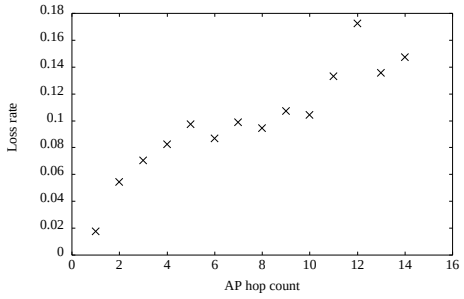
We begin by studying the correlation between the topological distance between the server and a client and the end-to-end loss rate experienced by the client. We consider several different notions of topological distance between the server and client: (i) router hop count, (ii) AS hop count, and (iii) address prefix (AP) hop count. To compute these hop counts from a traceroute path, the AS associated with a router is determined by querying *Whois* and the AP is determined using address prefix information [14] from a BGP routing table dump [3] obtained on Jan 24, 2001. Our goal is to understand if such static topological metrics correlate well with end-to-end loss rate. For instance, can such distance metrics be used to select the best replica for a particular client as is done by the Cisco Distributed Director [5]?

As explained in Section III, we determined the network path from the server to each client using traceroute. For each client, we determined the number of router hops. We determined the AS number corresponding to each router by querying the *Whois* database and thereby computed the AS hop count of each path. (We ignored paths for which we were unable to determine the AS number for even one router. 31.6% of paths were ignored as a result of this.) Likewise, we used BGP prefix information to determine the address prefix (AP) corresponding to each router and thus computed the AP hop count of each path. Since BGP prefixes tend to be more fine-grained than ASes, the AP hop count for a path is generally larger than the AS hop count but smaller than the router hop count.

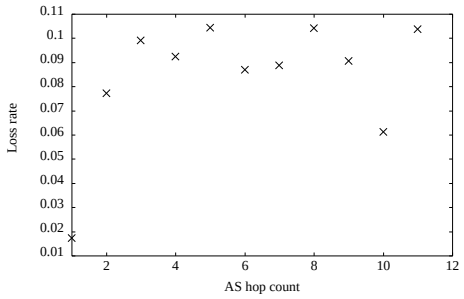
Figure 1 shows the average end-to-end loss rate experienced by clients for various values of router hop count, AP hop count, and AS hop count (Dec 2000 trace). We find that there is little correlation between loss rate and hop count. The correlation coefficients between loss rate and router hop count, AP hop count, and AS hop count are 0.05, 0.03, and 0, respectively. (Note that the correlation coefficient is computed using the entire raw data set whereas Figure 1 only plots the average loss rate for legibility.) Filtering out the IP addresses that have loss rate below 10% improves the correlation coefficient somewhat. The corresponding correlation coefficients become 0.112, 0.104, and 0.011, respectively. The correlation is still pretty weak, which implies that the hop count is not a reliable indicator of end-to-end packet loss rate.



(a) Loss rate versus router hop count.



(b) Loss rate versus AP hop count.



(c) Loss rate versus AS hop count.

Fig. 1. The correlation between loss rate and hop counts.

One reason why the correlation between topological distance and loss rate is weak is that all links are not "equal" (which is in contrast to the implicit assumption made in metrics such as router hop count that all hops are the same). In other words, it is likely that poor end-to-end performance is caused by a few lossy links. If the network path from the server to a client traverses one or more of the lossy links, then it is likely that the client would see poor performance (e.g., a high packet loss rate) even if the number of hops in the path is small. Therefore it is important to identify the lossy links. This motivates our network tomography work presented in Section V.

B. Temporal Locality of Loss Rate

We now turn to the question of how long the end-to-end packet loss rate experienced by clients remains stable. We use the methodology developed in [23] to analyze the constancy of Internet path properties based on active measurements in the NIMI testbed. Basically, we partition loss rates into the following categories: 0 - 0.5%, 0.5 - 2%, 2 - 5%, 5 - 10%, 10 - 20%, and 20+%. According to the classification in [23], these categories correspond to "no loss", "minor loss", "tolerable loss", "serious loss", "very serious loss", and "unacceptable loss". We study how long a client's loss rate remains stable, i.e., remains in the same category. In our analysis, from each trace we pick the top 1000 hosts in terms of the total number of packets sent and received to avoid biasing the result due to lack of data samples.

Figure 2 plots the cumulative distribution (CDF) of the maximum duration for which loss rate remains stable based on the Dec 2000 trace. The curve is weighted by the size of the stationary interval as in [23]. When computing the loss rate, we only consider clients that have received at least a threshold number of packets within a 10-second time interval. We use two thresholds: 100 and 1000 packets. As we can see, about 50% of time, the stability period is less than 10 minutes using 1000 as the threshold, and about 70% of time, the stability period is less than 10 minutes using 100 as the threshold. To test for possibility of binning effects, we also use a different set of cutpoints for the loss categories, each falling in the middle of the above cutpoints, as suggested in [23]. The results are very similar and are omitted in the interest of space.

The above analysis results suggest that loss rate is stable on the time scale of several minutes, which is consistent with the findings in [23]. This consistency is interesting considering that our data set is different from that studied in [23] in several respects: passive, server-based measurements of a much larger and more heterogeneous set of end hosts.

The stability of end-to-end loss rate over a time scale of several minutes suggests that the underlying cause of packet loss — i.e., the lossiness of network links — is also likely to persist for a significant length of time. So trying to identify the lossy links using network tomography would be a worthwhile goal.

C. Spatial Locality of Loss Rate

Finally, we analyze the spatial locality of end-to-end packet loss rate. We would like to answer the question of whether clients that are topologically close to each other experience similar loss rates. Toward this end, we cluster clients using various schemes and examine whether clients belonging to the same cluster see similar loss rates.

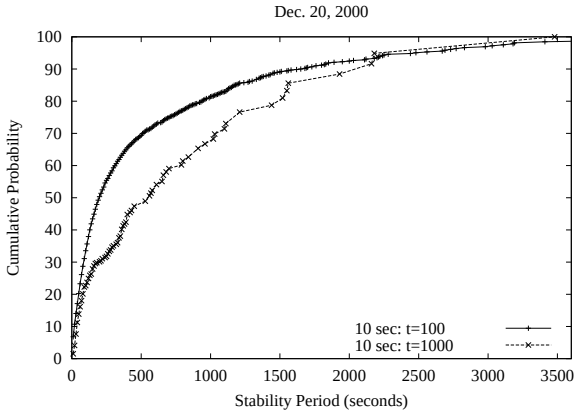


Fig. 2. CDF of the time period in which a host's loss rate remains in the same loss category.

We consider the following clustering schemes: (i) clients clustered by subnet address assuming a 24-bit subnet prefix, (ii) clients clustered by the address prefix (AP) in BGP routing tables [14], (iii) clients clustered by autonomous system (AS) number, (iv) clients clustered randomly.

In addition, we also examine a new clustering scheme, which we call DNS-based clustering. The basic idea is to cluster the clients that share the same set of authoritative name servers. Such clustering is cheap to perform: we just need to do a DNS zone transfer, and obtain the mapping between IP addresses and authoritative name servers. It is likely to be more coarse-gained than BGP address prefix clustering, as different BGP prefixes may share the same authoritative name server. An alterative approach is to cluster the clients that share the same local name server. This may be better in terms of network locality. However, the mapping between client IP addresses and their local name servers is not easily obtainable. In our study, we use the data obtained from a DNS zone transfer done in October 2001 to perform DNS-based clustering.

In our analysis, we use the top 20000 clients from the Dec 2000 trace, and the top 50000 clients from the Jan 2002 trace. We cluster them using the above schemes. Then from each cluster we randomly pick two groups, each containing 3 clients, and compare the difference in their average loss rates (we only consider groups that received at least 1000 packets to keep the loss rate computation meaningful). We repeat this five times for each cluster.

We use the same categorization of loss rates as detailed in Section IV-B. Instead of comparing the absolute difference in average loss rate between the two groups, we compare the number of loss categories the two groups are away from each other. So, for example, the difference between the loss rate values 0.3% ("no loss") and 4% ("tolerable loss") is quantified as a difference of 2 loss rate categories.

Figure 3 and Figure 4 plot CDF of the loss rate difference within clusters for the Dec 2000 and Jan 2002 traces, respectively. We find that spatial locality is greatest in the case of subnet-based clustering. The probability of the loss rates of two groups drawn from the same cluster being in the same loss rate category is 40-60%. The corresponding probability for the AP

and AS based clustering schemes is around 30%. However, we also find that even random clustering yields a 25% probability of a match in loss rate categories. So the question is how significant the apparent spatial locality in the case of subset, AP, and AS based clustering is.

Random clustering exhibits a significant degree of spatial locality because many clients see little or no loss (i.e., lie in the "no loss" category) and thereby skew our results. However, our main interest is in determining how much locality exists when there is packet loss. Therefore we introduce a loss threshold and only consider samples of loss rate difference between the two groups when the loss rate of *either* group exceeds the threshold. We vary the loss threshold from 0 to 10%.

With a loss threshold of 10%, we find that probability of loss rates within a subnet lying within the same category is nearly 80% in the Dec 2000 trace and about 30% in the Jan 2002 trace. However, the correponding probability for random clustering is almost zero. This suggests that there is indeed some spatial locality, especially at the subnet level. However, the degree of spatial locality varies and is in some cases (e.g., Jan 2002 trace) quite limited.

We repeat the same analysis using a set of cutpoints for the loss categories that are in the middle of the above cutpoints, as was done in [23], and the results are very similar.

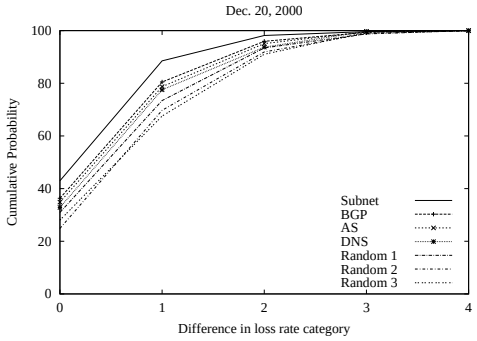
Our observation of a limited degree of spatial locality in loss rate, especially at the subnet level, suggests that there may in some cases be a shared cause (i.e., lossy link(s) shared by all clients in the cluster) for the end-to-end packet loss experienced by clients. At the same time, however, our results suggest that often there is low degree of locality in loss rate among clients in a cluster. This leads us to believe that often the cause of packet loss is a non-shared link (e.g., the "last-hop" link). It would be interesting to see if network tomography could shed more light on this issue.

D. Summary

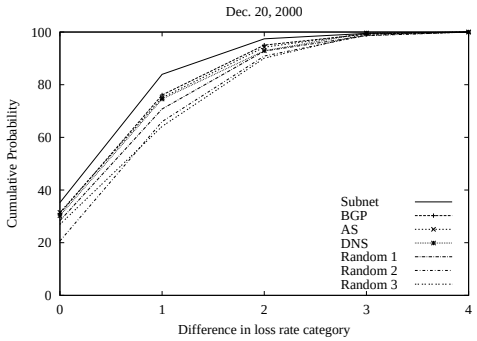
In summary, our analysis of end-to-end loss rate indicates that (a) the correlation between loss rate and hop count is weak, (b) loss rate tends to be stable over a period of several minutes, and (c) clients that are topologically close to each other experience more similar loss rate than clients picked at random, which indicates that there is a limited degree of spatial locality in loss rate. These finding suggest that a few lossy links, whether shared or non-shared, dominate the end-to-end loss rate and that the link loss rate tends to be stable for a significant length of time. This sets the stage for our work on passive network tomography, where we develop techniques to provide greater insight into some of these conjectures.

V. Passive Network Tomography

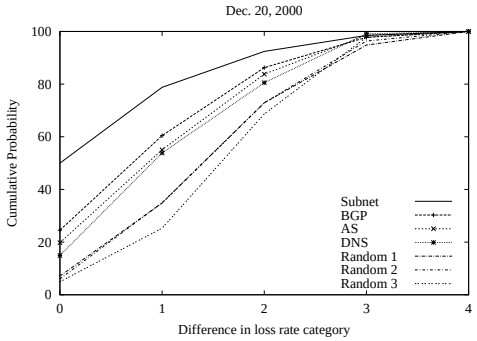
In this section, we attempt to identify lossy links in the network based on observations made at the server of end-to-end packet loss rates to different clients. As noted in Section II, much of the prior work on estimating the loss rate of network links has been based on the active injection of probe packets into the network. In contrast, our goal here is to base the inference on *passive* observation of existing network traffic. We term this *passive network tomography*.



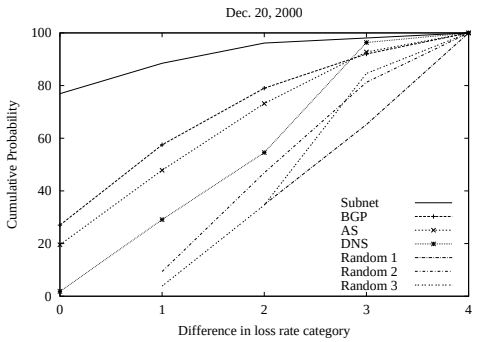
(a) Loss threshold = 0



(b) Loss threshold = 0.01

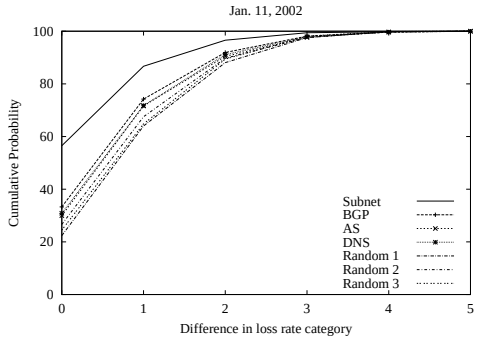


(c) Loss threshold = 0.05

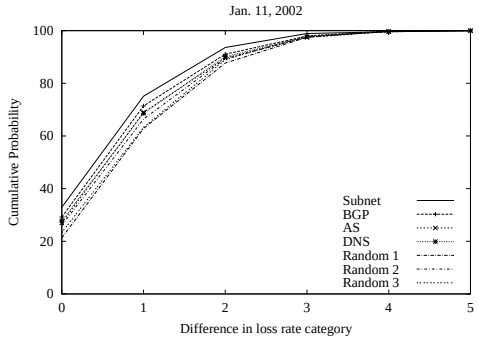


(d) Loss threshold = 0.1.

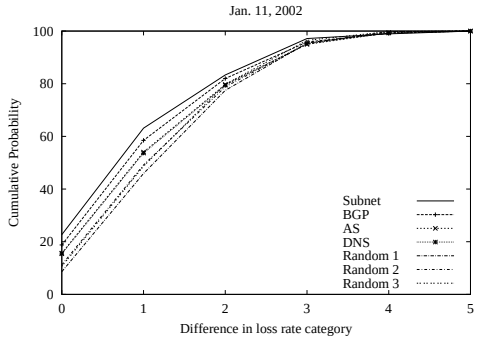
Fig. 3. CDF of the loss rate difference within clusters (Dec 2000).



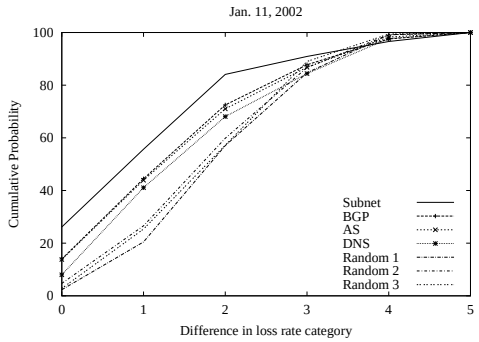
(a) Loss threshold = 0.



(b) Loss threshold = 0.01.



(c) Loss threshold = 0.05.



(d) Loss threshold = 0.1.

Fig. 4. CDF of the loss rate difference within clusters (Jan 2002).

Figure 5 depicts the scenario of interest: a server transmitting data to a distributed set of clients. By passively observing the client-server traffic, we can determine the number of packets transmitted by the server to each client. Based on the feedback from the clients (e.g., TCP ACKs, RTCP receiver reports), we can also determine how many of those packets were lost in the network.

We assume that the network path from the server to each client is known. In the experiments reported in this paper, the path to each client was determined using the *traceroute* tool [11]. While these traceroutes do constitute *active* measurement, this need not be done very frequently or in real time. (Indeed previous studies have shown that end-to-end Internet paths generally tend to be stable for significant lengths of time. For instance, [24] indicates that very often paths remain stable for at least a day.) Moreover, it may be possible determine the server-to-client path "pseudo-passively" by invoking the record route option (IPv4) or extension header (IPv6) for a small subset of the packets.

The set of paths from the server to its clients is likely to form a tree (as depicted in Figure 5) and so our explanations here are couched in terms of tree-specific terminology. However, we do recognize that the topology may not strictly be a tree (for instance, because of transient route fluctuations), so our techniques do not depend on the topology being a tree. We elaborate on this point in Section V-B.

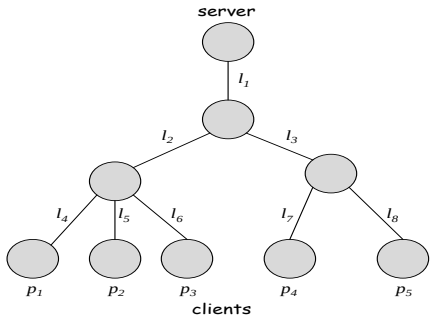


Fig. 5. A sample network topology as viewed from a server. The link loss rates are denoted by l_i and the end-to-end loss rate at the clients are denoted by p_j.

A. Challenges

Identifying lossy links is challenging for the following reasons. First, network characteristics change over time. Without knowing the temporal variation of the network link performance, it is hard to correlate performance observed by different clients. Second, even when the loss rate of each link is constant, it may not be possible to definitively identify the loss rate of each link. Given M clients and N links, we have M constraints (corresponding to each server→client path) defined over N variables (corresponding to the loss rate of the individual links). For each client C_j, there is a constraint of the form $1 - \prod_{i \in T_j}(1 - l_i) = p_j$ where T_j is the set of links on the path from the server to client C_j, l_i is the loss rate of link i, and p_j is the end-to-end loss rate between the server and client C_j. There is not a unique solution to this set of constraints if $M < N$, as is often the case.

To make the problem tractable, we make the simplifying assumption that the loss rate of each link is constant. Although this is not a very realistic assumption, it is a reasonable simplification in the sense that some links consistently tend to have high loss rates whereas other links consistently tend to have low loss rates. Zhang et al. [23] reported that the loss rate remains operationally stable on the time scale of several minutes. Our temporal locality analysis in Section IV-B also confirms it.

There is still the problem that we may not, in general, be able to determine a unique assignment of loss rate to network links. We address this issue in several ways.

First, we collapse a linear sections of a network path with no branches into a single *virtual link*[2]. This is appropriate since it would be impossible to determine the loss rates of the constituent physical links of such a linear section using end-to-end measurements.

Second, although there may not be a unique assignment of loss rate to network links, two of our techniques seek a parsimonious explanation for the observed end-to-end loss rates. (This bias is implicit in the case of Random Sampling and explicit in the case of Linear Optimization.) So given a choice between an assignment of high loss rates to many links and an assignment of high loss rates to a small number of links, they would prefer the latter. The underlying assumption is that a lossy link is relatively uncommon. If most of the links are lossy, network tomography may not be very useful any way since there are not specific trouble spots to pinpoint. On the other hand, our Gibbs Sampling technique uses a uniform prior and so is *unbiased*.

Finally, we set our goal to primarily be the identification of links that are likely to have a high loss rate rather than inferring a specific loss rate for each link. We believe that the identification of the most lossy links in itself would be very useful for applications such as network diagnosis and server selection.

We now describe the three different techniques we have explored and developed for passive network tomography. We present these in roughly increasing order of sophistication. However, as the experimental results in Section V-E indicate, even the simplest technique, yields good results.

B. Random Sampling

The set of constraints mentioned in Section V-A define a space of feasible solutions for the set of link loss rates. (We denote a specific solution as $l_L = \bigcup_{i \in L} l_i$ where L is the set of all links in the topology.) The basic idea of random sampling is to repeatedly sample the solution space at random and make inferences based on the statistics of the sampled solutions. The solution space is sampled as follows. We first assign a loss rate of zero to each link of the tree (Figure 5). The loss rate of link i is bounded by the minimum (say l_i^{min}) of the observed loss rate at the clients downstream of the link. We pick the loss rate, l_i, of the link i to be a random number between 0 and l_i^{min}. We define the residual loss rates of a client to be the loss rate that is not accounted for by the links whose loss rates have already been assigned. We update the residual loss rate of a client C_j to $1 - \frac{1-p_j}{\prod_{i \in T'_j}(1-l_i)}$ where T'_j is the subset of links along the path from the server to the client C_j for which a loss rate has been assigned. Then we repeat the procedure to compute the loss rate at the next level of the tree by considering the residual loss rate of each client in place of its original loss rate. At the end, we have one sample solution for l_L.

We iterate R times to produce R random solutions for l_L. We draw conclusions based on the statistics of the individual link loss rates, l_i, across the R random solutions. For instance, if the average loss rate assigned to a link across all samples is higher than a threshold, we conclude that the link is lossy.

Note that we compute a loss rate only for those clients to whom the server has transmitted at least a threshold number of packets. Only this subset of the clients and the topology induced by them is considered in the random sampling algorithm.

The sampling procedure outlined above is biased because the order in which links are picked matters. As we assign loss rates to an increasing number of links, the loss rate bound on the remaining links gets tighter. So links that are picked early in an iteration are likely to be assigned a higher loss rate than ones picked later. Thus in the above algorithm, links higher up in the tree (i.e., close to the server), which are picked early in the process, tend to get assigned a higher loss rate. Of course, the loss rate bound on a link higher up in the tree might be tighter to begin with because of there is a greater chance that one or more downstream clients will have experienced a low loss rate.

This bias, however, has a positive side-effect in that it favors parsimonious solutions (i.e., ones in which the observed client loss rates can be accounted for by assigning a high loss rate to fewer links). If many clients are experiencing a high loss rate, an explanation that involves one shared, lossy link higher up in the tree is more plausible than one that involves a large number of independently lossy links.

Note that our random sampling algorithm would work the same way even if the topology were not a tree. In fact, at any stage in an iteration, we can pick an arbitrary link, determine the bounds on its loss rate by examining all server→client paths that traverse the link, and then randomly assign it a loss rate. Just like in a tree topology, we could start by picking links close to the server and then working our way towards the clients.

The random sampling algorithm has the advantage of being simple. However, it is quite susceptible to estimation errors in the client loss rate. Due to a statistical variation, a single client that is downstream of a true lossy link could experience a low loss rate. This would cause the random sampling algorithm to assign a low loss rate to the link even if all of the other downstream clients experience a high loss rate. The alternative algorithms for passive network tomography that we describe below are robust to such errors.

C. Linear Optimization

We formulate the network tomography problem as a linear program (LP). As noted in Section V-A, we have a constraint of the form $1 - \prod_{i \in T_j}(1-l_i) = p_j$ corresponding to each client C_j. We can turn this into a linear constraint $\sum_{i \in T_j} L_i = P_j$ where $L_i = log(1/(1-l_i))$ and $P_j = log(1/(1-p_j))$. Note that the transformed variables L_i and P_j are monotonic functions of l_i and p_j, respectively.

[2] In the rest of the paper, we use the term "link" to refer to both physical links and virtual links.

To be robust to errors or aberrations in client loss rate estimates, we allow the above constraints to be violated (a little). We do so by introducing a slack variable, S_j, in the constraint corresponding to client C_j yielding a modified constraint: $\sum_{i \in T_j} L_i + S_j = P_j$. In addition, we have the constraints $L_i \geq 0$.

The objection function to minimize is $w \sum_i L_i + \sum_j |S_j|$. This reflects the objectives of finding a parsimonious solution (hence the $\sum_i L_i$ term) and minizing the extent to which the original constraints are violated (hence the $\sum_j |S_j|$ term). The weight, w, allows us to control the relative importance of finding a parsimonious solution versus satisfying the original constraints well; we set w to 1 by default. Note that the $|S_j|$ term means that this is not strictly a linear program in its present form. However, it is trivial to transform it into one by defining auxiliary variables, S'_j and adding constraints of the form $S'_j \geq S_j$ and $S'_j \geq -S_j$. The objective function to minimize is then $w \sum_i L_i + \sum_j S'_j$.

The linear optimization approach also has its drawbacks. First, like the random sampling approach, it depends on the client loss *rates*, p_j, to be computed. However, the loss rate may be meaningfully computed only when a sufficiently large number of packets are sent to the client (we use a minimum threshold of 500 or 1000 packets in the experiments presented in Section V-F). This limits the applicability of this technique. Second, while the objective function listed above intuitively conforms to our goals, there is no fundamental justification for its specific form. Indeed the solution obtained would, in general, be different if the objective function were modified. This then motivates the statistically rigorous technique we describe next.

D. Bayesian Inference using Gibbs Sampling

We model passive network tomography as a Bayesian inference problem. We begin by presenting some brief background information; for details, please refer to [10].

D.1 Background

Let D denote the observed data and θ denote the (unknown) model parameters. (In the context of network tomography, D represents the observations of packet transmission and loss, and θ represents the ensemble of loss rates of links in the network.) The goal of Bayesian inference is to determine the *posterior* distribution of θ, $P(\theta|D)$, based on the observed data, D. The inference is based on knowing a *prior* distribution $P(\theta)$ and a *likelihood* $P(D|\theta)$. The *joint* distribution $P(D, \theta) = P(D|\theta)P(\theta)$. We can then compute the posterior distribution of θ as follows:

$$P(\theta|D) = \frac{P(\theta)P(D|\theta)}{\int_\theta P(\theta)P(D|\theta)d\theta}$$

Any features of the posterior distribution are legitimate for Bayesian inference: moments, quantiles, etc. All of these quantities can be expressed as posterior expectations of functions of θ:

$$E(f(\theta)|D) = \frac{f(\theta)P(\theta)P(D|\theta)}{\int_\theta P(\theta)P(D|\theta)d\theta}$$

In general, it is hard to compute $E(f(\theta)|D)$ directly because of the complex integrations, especially when θ is a vector (as it is in our case). An indirect approach is to use *Monte Carlo integration*. The idea here is to sample underlying posterior distribution and use the sample mean as an approximation of $E(f(\theta)|D)$. One way of doing the appropriate sampling is to construct a Markov chain whose stationary distribution exactly equals the posterior distribution of interest ($P(\theta|D)$). (Hence the name *Markov Chain Monte Carlo (MCMC)* [9], [10] was given to this class of techniques.) When such a Markov chain is run for a sufficiently large number of steps (termed the *burn-in* period), it "forgets" its initial state and converges to its stationary distribution. It is then straightforward to obtain samples from this stationary distribution.

The challenge then is then to construct a Markov chain (i.e., define its transition probabilities) whose stationary distribution matches $P(\theta|D)$. *Gibbs sampling* [9] is a widely used technique to accomplish this. The basic idea that at each transition of the Markov chain, only a single variable (i.e., only one component of the vector θ) is varied. Rather than explain Gibbs sampling in general, we now switch to modeling network tomography as a Bayesian inference problem and explain how Gibbs sampling works in this context.

D.2 Application to Network Tomography

To model network tomography as a Bayesian inference problem, we define D and θ as follows. The observed data, D, is defined as the number of successful packet transmissions to each client (s_j) and the number of failed (i.e., lost) transmissions (f_j). (Note that it is easy to compute s_j by subtracting f_j from the total number of packets transmitted to the client.) Thus $D = \bigcup_j (s_j, f_j)$. The unknown parameter θ is defined as the set of links' loss rates, i.e., $\theta = l_L = \bigcup_{i \in L} l_i$ (Section V-B). The likelihood function can then be written as:[3]

$$P(D|l_L) = \prod_{j \in clients} (1 - p_j)^{s_j} p_j^{f_j} \tag{1}$$

Recall from Section V-A that $p_j = 1 - \prod_{i \in T_j}(1 - l_i)$ and represents the loss rate observed at client C_j.

The prior distribution, $P(l_L)$, would indicate prior knowledge about the lossiness of the links. For instance, the prior could be defined differently for links that are known to be lossy dialup links as compared to links that are known to be highly reliable OC-192 pipes. However, in our study here, we only use a uniform prior, i.e., $P(l_L) = 1$.

The object of network tomography is the posterior distribution, $P(l_L|D)$. To this end, we use MCMC with Gibbs sampling as follows. We start with an arbitrary initial assignment of link loss rates, l_L. At each step, we pick one of the links, say i, and compute the posterior distribution of loss rate for that link alone conditioned on the observed data D and the loss rates assigned to all other links (i.e., $\{\bar{l_i}\} = \bigcup_{k \neq i} l_k$). Note that $\{l_i\} \cup \{\bar{l_i}\} = l_L$. Thus we have

[3]Note that we are only computing the likelihood of the specific observation we made. We are *not* interested in counting all possible ways in which client j could have had s_j successes and f_j failures, so the equation does not include such a combinatorial term. We offer this clarification since a few readers have been confused at first blush.

$$P(l_i|D,\{\bar{l_i}\}) = \frac{P(D|\{l_i\}\cup\{\bar{l_i}\})P(l_i)}{\int_{l_i} P(D|\{l_i\}\cup\{\bar{l_i}\})P(l_i)dl_i}$$

Since $P(l_L) = 1$ and $\{l_i\}\cup\{\bar{l_i}\} = l_L$, we have

$$P(l_i|D,\{\bar{l_i}\}) = \frac{P(D|l_L)}{\int_{l_i} P(D|l_L)dl_i} \quad (2)$$

Using equations 1 and 2, we numerically compute the posterior distribution $P(l_i|D,\{\bar{l_i}\})$ and draw a sample from this distribution[4]. This then gives us the new value, $l_i^{'}$, for the loss rate of link i. In this way, we cycle through all the links and assign each a new loss rate. We then iterate this procedure several times. After the burn-in period (which in our experiments lasts a few hundred iterations), we obtain samples from the desired distribution, $P(l_L|D)$. We use these samples to determine which links are likely to be lossy.

D.3 Discussion

The Bayesian approach outlined above is based on solid theoretical foundations. Another advantage of this approach over the random sampling and the linear optimization approaches is that it only requires the *number* of packets sent to and lost at each client, *not* the loss rate. So it can be applied even when the number of packets sent to a client is not large enough for the packet loss rate to be meaningfully computed.

E. Simulation Results

In this section, we show results of our experimental evaluation of the three passive network tomography techniques discussed above. We begin with a discussion of our simulation experiments and results. The main advantage of simulation is that the true link loss rates are known, so validation of the inferences of the tomography algorithm is easy to do.

The simulation experiments are performed on topologies of different sizes using multiple link loss models. The topologies considered are randomly constructed trees with the number of nodes (n) ranging from 20 to 3000. (Note that the node count includes both interior nodes (i.e., routers) and leaves (i.e., clients).) The number of links in each topology is roughly equal to the number of nodes (modulo the slight reduction in link count caused by the collapsing of linear chains, if any, into virtual links). The degree of each node (i.e., the number of children) was picked at random between 1 and an upper bound (d) which was varied from 5 to 50.

In addition, we also consider a real server→clients topology constructed from our traceroute data set. This topology spans 123166 clients drawn from the Dec 20, 2000 data set.

A fraction f of the links were classified as "good" and the rest as "bad". We used two different models for assigning loss rates to links in these two categories. In the first loss model (LM_1), the loss rate for good links was picked uniformly at random in the 0-1% range and that for bad links was picked in the 5-10% range. In the second model (LM_2), the loss rate ranges for good and bad links were 0-1% and 1-100%, respectively.

Once each link has been assigned a loss rate, we use one of two alternative loss processes at each link: Bernoulli and Gilbert. In the Bernoulli case, each packet traversing a link is dropped with a fixed probability determined by the loss rate of the link. In the Gilbert case, the link fluctuates between a good state and a bad state. At the good state, no packets are dropped while at the bad state all packets are dropped. As in [16], we chose the probability of remaining in the bad state to be 35% based on Paxson's observed measurements of the Internet. The other state transition probability is picked so that the average loss rate matches the loss rate assigned to the link. Thus the Gilbert loss process is likely to generate more bursty losses than the Bernoulli loss process. In both cases, the end-to-end loss rate is computed based on the transmission of 1000 packets from the root (server) to each leaf (client). Unless otherwise indicated, our simulation experiments use the LM_1 loss model together with the Bernoulli loss process.

We have chosen these somewhat simplistic loss models over simulating real congestion losses because it gives us greater flexibility in terms of being able to explicitly control the loss rate of each link. Furthermore, to the extent that the loss rate of Internet paths is operationally stationary for significant lengths of time, these models offer a reasonable approximation.

We repeated our experiment 6 times for each simulation configuration, where each repetition has a new topology and loss rate assignments. In each repetition of an experiment, a link is inferred to be lossy as follows. For random sampling, we compute the mean loss rate of the link over 500 iterations (Section V-B). We infer the link to be lossy if the mean exceeds a loss rate threshold. Likewise, for the linear optimization (LP) approach, we compare the (unique) inferred link loss rate to the loss rate threshold. In the case of Gibbs sampling, since we numerically compute the posterior *distribution*, we apply a somewhat more sophisticated test. We infer a link to be lossy if more than 99% of the loss rate samples for the link exceed the loss rate threshold. For the LM_1 model, the loss rate threshold was set to 3% (i.e., the midpoint between the 1% and 5% range delimiters discussed above) while for the LM_2 model it was varied in the range of 5-20%.

We report the true number of lossy links, and the number of correctly inferred lossy links (*coverage*) and the number of incorrectly inferred lossy links (*false positives*), all being averaged over the 6 runs of the experiment for each configuration.

E.1 Random Topologies

We present simulation results for different settings of tree size (n), maximum node degree (d), and fraction of good links (f). We repeated our experiments 6 times for each setting of n, d, and f. The results presented in this sub-section are based on the LM_1 loss model with the Bernoulli loss process.

Figure 6 shows the simulation results for 100-node topologies and $d = 10$, and f varying from 0.5 to 0.95. We note that in general, random sampling has the best coverage. In most cases, it is able to identify over 90-95% of the lossy links. However, the high coverage comes at the cost of a very high false positive rate — ranging from 50-140%. Such a high false positive

[4]Since the probabilities involved may be very small and could well cause floating point underflow if computed directly, we do all our computations in the logarithmic domain.

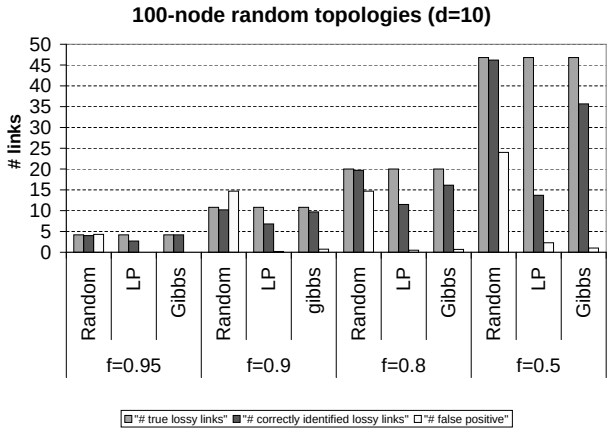


Fig. 6. Varying f: 100-node random topologies with maximum degree = 10.

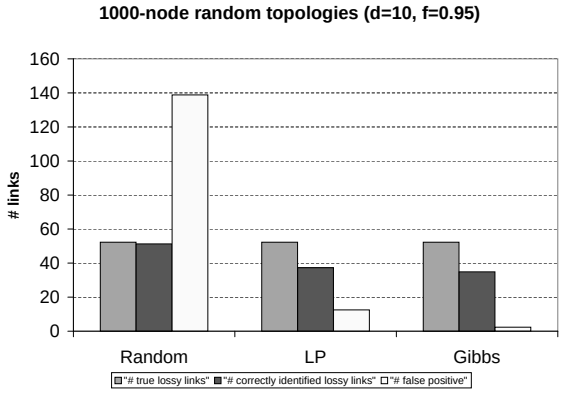


Fig. 7. 1000-node random topologies with maximum $degree = 10$ and $f = 0.95$.

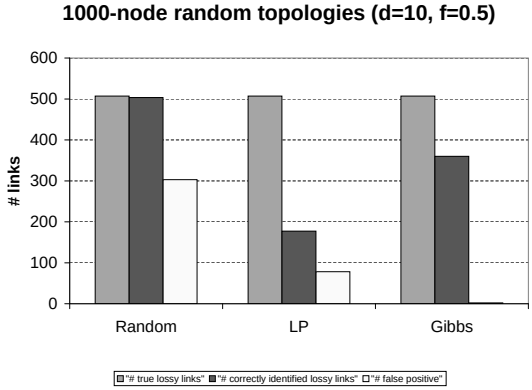


Fig. 8. 1000-node random topologies with maximum $degree = 10$ and $f = 0.5$.

rate may be manageable when there are few lossy links in the network (i.e., f is large) since we can afford to run more expensive tests (e.g., active probing) selectively on the small number of lossy links inferred. However, the large false positive rate is unacceptable when there are a large number of lossy links in the network. For instance, when $f = 0.5$, random sampling correctly identifies 46 of the 47 lossy links. In addition, however, it generates 24 false positives, which makes the inference almost worthless since there are only about 100 links in all.

One reason why random sampling generates a large number of false positives is its susceptibility to statistical fluctuations in the end-to-end loss rate experienced by clients (Section V-B). For instance, instead of correctly identifying a lossy link high up in the tree, random sampling may incorrectly identify a large number of links close to individual clients as lossy.

In contrast to random sampling, LP has relatively poor coverage (30-60%) but an excellent false positive rate (rarely over 5%). (In some cases, the false positive bar in Figure 6 is hard to see because the number of false positives is close to or equal to zero.) As explained in Section V-C, LP is less susceptible to statistical fluctuations in the end-to-end loss rates since it allows some slack in the constraints. This cuts down the false positive rate. However, the slack in the constraints and the fact that the objective function assigns equal weights the link loss variables (L_i) and the slack variables (S_j) causes a reduction in coverage. Basically, a true lossy link (especially one near the leaves) may not be inferred as such because the constraint was slackened sufficiently to obviate the need to assign a high loss rate to the link. In Section V-E.3, we will examine different weights in LP on the inference.

Finally, we observe that Gibbs sampling has a very good coverage (over 80%) and also an excellent false positive rate (well under 5%). We believe that the excellent performance of this technique arises, in part, because the Bayesian approach is based on observations of packet loss *events* and not the (noisy) computation of packet loss *rates*.

Figures 7 and 8 show the corresponding results for experiments on 1000-node topologies. Figure 9 shows the results for 3000-node topologies. We observe that the trends remain qualitatively the same even for these larger topologies. Gibbs sampling continues to have good coverage with less than 5% false positive.

Figure 10 shows the how accurate the inference based on Gibbs sampling is when the links inferred as lossy are rank ordered based on our "confidence" in the inference. We quantify the confidence as the fraction of Gibbs samples that exceed the loss rate threshold set for lossy links. The 983 links in the topology are considered in order of decreasing confidence. We plot 3 curves: the true number of lossy links in the set of links considered up to that point, the number of correct inferences, and the number of false positives. We note that the confidence rating assigned by Gibbs sampling works very well. There are zero false positives for the top 33 rank ordered links. Moreover, each of the first 401 true lossy links in the rank ordered list is cor-

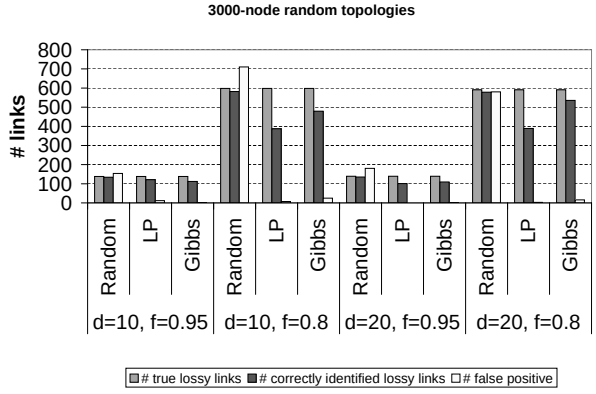


Fig. 9. 3000-node random topologies.

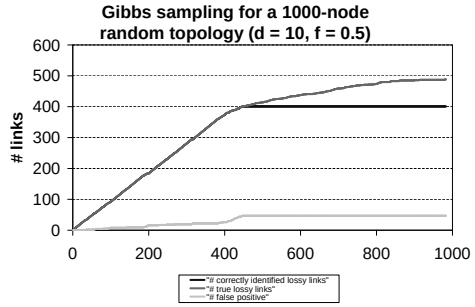


Fig. 10. The performance of Gibbs sampling when the inferences are rank ordered based on a confidence estimate. (1000-node random topology, maximum $degree = 10$, and $f = 0.5$)

rectly identified as lossy (i.e., none of these true lossy links is "missed"). These results suggest that the confidence estimate for Gibbs sampling can be used to rank the order of the inferred lossy links so that the top few inferences are (almost) perfectly accurate. This is likely to be useful in a practical setting where we may want to identify at least a small number of lossy links with certainty so that corrective action can be taken.

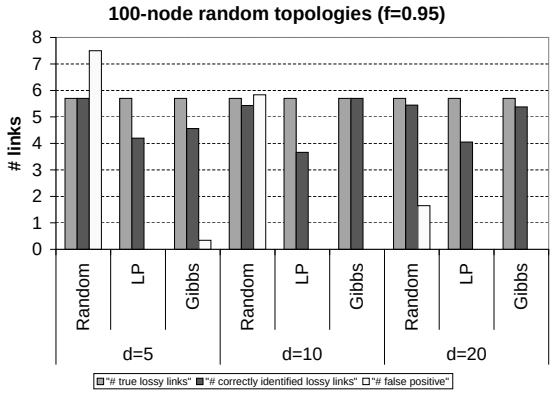


Fig. 11. Varying degree: 100-node random topologies with $f = 0.95$.

Finally, we evaluate the impact of node degree on the accuracy of the inference. Figure 11 shows the results for a 100-node topology and $f = 0.95$. We consider 3 settings for the maximum node degree, d: 5, 10, and 20. The qualitative trends across the three techniques are the same as discussed above. However, it is interesting to observe that the false positive rate for random sampling decreases as the node degree increases. We believe this happens because a larger node degree implies that a larger proportion of the nodes are leaves. So end-to-end loss rate information imposes a larger number of constraints on the link loss rates. This results in a smaller feasible solution space, which makes random sampling of this space more accurate.

E.2 Alternative Loss Model

So far, we have considered LM_1 loss model with the Bernoulli loss process. In this section, we evaluate the effectiveness of inference using alternatives for both (i.e., the LM2 loss model and the Gilbert loss process) in various combinations.

LM_2 Bernoulli loss model: Figure 12 shows the results for 1000-node random topologies with $d = 10$ and $f = 0.95$ using the LM_2 Bernoulli loss model. We vary the loss rate threshold, lb, used to decide whether a link is lossy. We observe that the coverage is well over 80% for all the algorithms. As the loss threshold is increased, the false positive rate decreases while the coverage remains high. This suggests that the inference can be more accurate if we are only interested in highly lossy links.

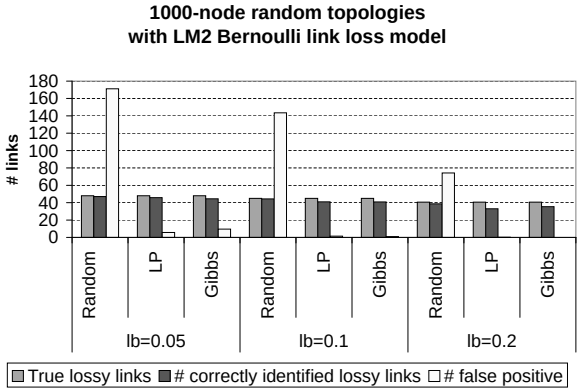


Fig. 12. A LM_2 Bernoulli loss model for 1000-node random topologies with maximum $degree = 10$ and $f = 0.95$. We vary the loss threshold lb, and only the links with loss rate higher than lb are considered lossy.

LM_1 and LM_2 Gilbert loss models: Figure 13 and Figure 14 show the performance of inference for LM_1 and LM_2 Gilbert loss models. The relative performance of different inference schemes remains the same. The Gibbs sampling continues to be the best performer: it has around 90% coverage with the lowest false positive among all the schemes.

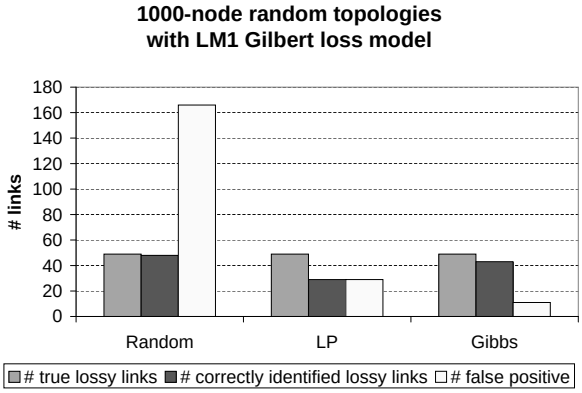


Fig. 13. A LM_1 Gilbert loss model for 1000-node random topologies with maximum $degree = 10$ and $f = 0.95$.

E.3 Different Weights in LP

The linear optimization algorithm aims to minimize $w \sum_i L_i + \sum_j |S_j|$, where the weight, w, reflects the relative importance between finding a parsimonious solution versus satisfying the end-to-end loss constraints. So far in our experiments, we use $w = 1$. In this section, we vary w and examine its effect on the performance of the inference.

Figure 15 and Figure 16 show the LP performance for 1000-node random topologies under Gilbert LM_1 and LM_2 loss models, respectively. As we can see, the lower the w, the better coverage the inference achieves, but at the cost of higher false positive rates. This is because when we decrease the weight, we emphasize more on satisfying the constraints than finding a parsimonious solution; as a result, we are more likely to attribute

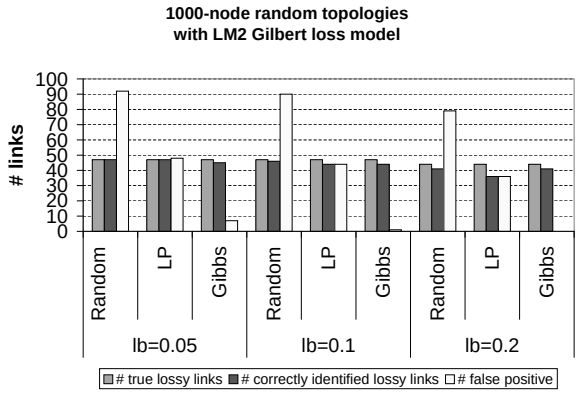


Fig. 14. A LM_2 Gilbert loss model for 1000-node random topologies with maximum $degree = 10$ and $f = 0.95$. We vary the loss threshold lb, and only the links with loss rate higher than lb are considered lossy.

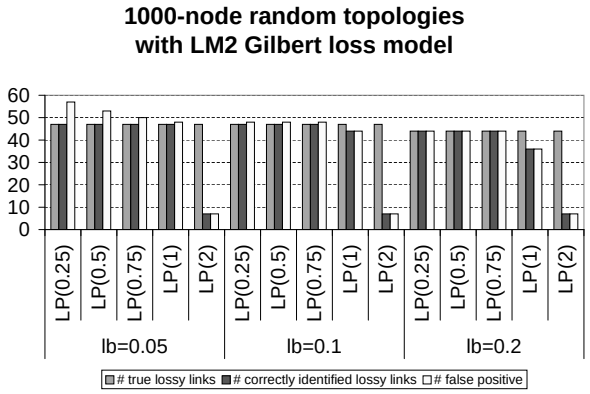


Fig. 16. Effects of different weights in LP: A LM_2 Gilbert loss model for 1000-node random topologies with maximum $degree = 10$ and $f = 0.95$. We vary the loss threshold lb, and only the links with loss rate higher than lb are considered lossy.

loss to several non-shared links than a single shared link in order to satisfy the constraints more closely. Moreover it is interesting that the performance of LP is less sensitive to the weights in the LM_2 loss model than in the LM_1 loss model.

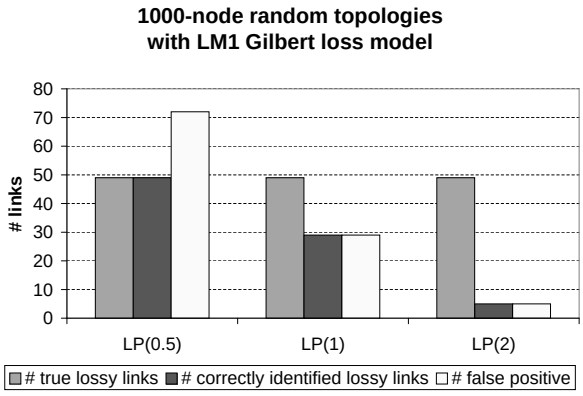


Fig. 15. Effects of different weights in LP: A LM_1 Gilbert loss model for 1000-node random topologies with maximum $degree = 10$ and $f = 0.95$.

E.4 Real Topology

We also evaluate the effectiveness of inference using a real topology (constructed from traceroute data) spanning 123166 clients. We assign a loss rate to each link based on the LM_1 Bernoulli loss model with different settings of f. Figure 17 shows the performance of random sampling. As with the random topologies, random sampling has very good coverage but a significant false positive rate.

We were unable to evaluate the performance of LP and Gibbs sampling over the real topology because of computational complexity.

F. Internet Results

In this section, we evaluate the passive tomography techniques using the Internet traffic traces from *microsoft.com*. Validating our inferences is challenging since we only have end-to-end performance information and do not know the true link loss rates. The validation approach we use is to (i) check consistency in the inferences made by the three techniques, (ii) look at the characteristics of inferred lossy links, and (iii) examine

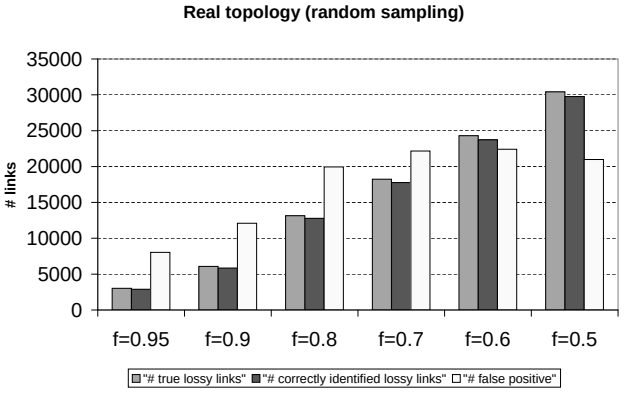


Fig. 17. Real topology from the Dec 2000 traceroute.

whether clients downstream of an inferred lossy link do in fact experience high loss rates.

The evaluation we present here is based on the Dec 2000 trace. To compute loss rate, we only consider clients that receive at least a threshold number of packets, t, which is set to 500 or 1000 packets in our evaluation.

F.1 Consistency across different schemes

First, we examine the consistency in the lossy links identified by the three tomography techniques. Figure 18 shows the amount of overlap when we consider the top N lossy links found by different schemes. Gibbs sampling and random sampling yield very similar inferences, with an overlap that is consistently above 95% when N is varied from 1 to 100. [5] The overlap between LP and the other techniques is also significant — over 60%.

F.2 Characteristics of Inferred Lossy Links

In this section, we examine the characteristics of the inferred lossy links. We are interested in knowing the location of the

[5]This overlap is higher than we had expected, since random sampling has a relatively high false positive rate in our simulations. As we describe in Section V-F.2, most lossy links terminate at leaves and most internal links are not lossy. So clients whose last hop links are not lossy experience little or no loss. This places tighter constraints on the space of feasible solutions, which makes random sampling more accurate.

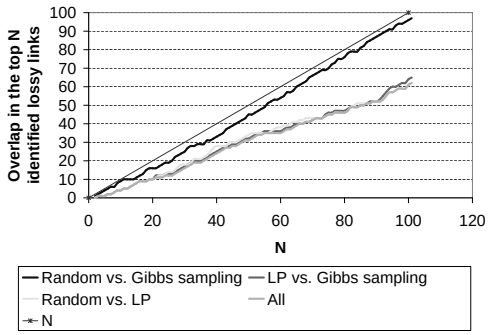


Fig. 18. Overlap in the top N lossy links identified by different schemes.

inferred lossy links in the Internet topology. As shown in Figure 19, more than 95% of lossy links detected through random sampling and Gibbs sampling terminate at leaves (i.e., clients). In other words, these are non-shared links that include the physical last-hop link to clients. (Recall from Section V-A that the tomography techniques operate on virtual links, which may span multiple physical links.) Even though the linear optimization technique is biased toward ascribing lossiness to shared links, more than 75% of the inferred lossy links are non-shared links terminating at clients. These findings are consistent with the common belief that the last mile to clients is often the bottleneck in Internet paths [8]. Since many losses happen at non-shared links, it is not surprising that there is only a limited degree of spatial locality in end-to-end loss rate, as reported in Section IV-C

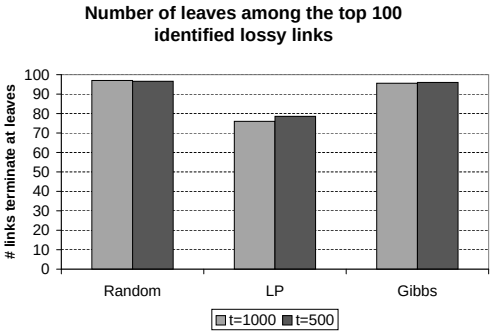


Fig. 19. Number of lossy links that terminate at leaf nodes.

We also examine how many of the links inferred to be lossy cross AS boundaries since such crossings (such as peering points) are thought to be points of congestion. We find that among all the virtual links in our topology (each of which may include multiple physical links), around 45% cross AS boundaries, and 45% have roundtrip delay (i.e., the delay between the two ends of the virtual link as determined from the traceroute data) over 100 ms. When we consider only the inferred lossy virtual links, the percentage of links that cross AS boundaries or have long delay is considerably higher. For example, if we only consider those links with an inferred loss rate above 10%, 70% cross AS boundaries, and 80% have one-way delay over 100 ms. Some examples of such links we found include the connection from AT&T in San Francisco to IndoInternet in Indonesia (inter-ISP and transcontinental), from Sprint to Trivalent (inter-ISP), and an international link in ChinaNet from U.S. to China.

L_l	Method	t	N_i	N_c
4%	Rand	1000	5	5
	Rand	500	5	4
	LP	1000	8	5
	LP	500	11	6
2%	Rand	1000	11	10
	Rand	500	14	13
	LP	1000	22	14
	LP	500	24	20
1%	Rand	1000	24	17
	Rand	500	23	19
	LP	1000	46	28
	LP	500	106	77

TABLE II
TRACE-DRIVEN VALIDATION FOR RANDOM SAMPLING AND LINEAR OPTIMIZATION.

F.3 Trace-driven Validation

We now consider the problem of validating our inferences more directly than the intuitive arguments made in Section V-F.2. This is a challenging problem since we do not know the true loss rates of Internet links. (All the inferences were made offline. So we could not validate the results using active probing.)

We have developed the following approach for validation. We partition the clients in the trace into two groups: tomography set and validation set. The partitioning is done by clustering all clients according to BGP address prefixes and dividing each cluster into two sets. One set is included in the tomography set and the other in the validation set. This partitioning scheme ensures that there is a significant overlap in the end-to-end path to clients in the two sets.

We apply the inference techniques to the tomography set to identify lossy links. For each lossy link that is identified, we examine whether clients in the validation set that are downstream of that link experience a high loss rate on average. If they do, we deem our inference to be correct. Otherwise, we count it as a false positive. Clearly, this validation method can only be applied to shared lossy links. We cannot use this method to validate the many "last-hop" lossy links reported in Section V-F.2.

Table II shows our validation results for random sampling and linear optimization, where L_l is the loss rate threshold we used to deem a link to be lossy, t is the minimum number of packets a client should have received to be considered in the tomography computation, N_i is the number of inferred (shared) lossy links, and N_c is the number of correct inferences according our validation method. In most cases random sampling and linear optimization have a false positive rate under 30%. Gibbs sampling identified only 2 shared lossy links, both of which are deemed correct according to our validation method.

VI. CONCLUSIONS

In this paper, we present a study of wide-area Internet performance as observed from the busy *microsoft.com* Web site. We characterize the end-to-end loss rate experienced by clients

and present techniques to identify lossy links within the network based on passive monitoring of existing client-server traffic.

We find that the end-to-end packet loss rate correlates poorly with topological distance (i.e., hop count), remains stable for several minutes, and exhibits a limited degree of spatial locality.

These findings suggest that passive network tomography would be both useful and feasible. We develop and evaluate three different techniques for passive network tomography: random sampling, linear optimization, and Bayesian inference using Gibbs sampling. In general, we find that random sampling has the best coverage but also a high false positive rate, which can be problematic when the number of lossy links is large. Linear optimization has a very low false positive rate but only a modest coverage. Gibbs sampling offers the best of both worlds: a high coverage (over 80%) and a low false positive rate (below 5%).

On the flip side, however, Gibbs sampling is computationally the most expensive of our techniques. On the other hand, random sampling is the quickest one. Therefore, we believe that random sampling may still be useful in practice despite its high false positive rate. For instance, when the number of lossy links in (the portion of) the network of interest is small, it may be fine to apply random sampling since the number of false positives (in absolute terms) is likely to be small. Furthermore, if the number of lossy links is large (for instance, the $f = 0.5$ configurations in Section V-E), it is a moot question as to whether network tomography will be very useful.

In addition to simulation, we have applied some of our tomography techniques to Internet packet traces. The main challenge is in validating our inferences. We validate the inference by first checking consistency across the results from different schemes. We find over 95% overlap between the top 100 lossy links identified by random sampling and Gibbs sampling, and over 60% overlap between LP and the other two techniques. We also find that most of the links identified as lossy are non-shared links terminating at clients, which is consistent with common belief. Finally we develop an indirect validation scheme, and show the false positive rate is manageable (below 30% in most cases and often much lower).

We are presently investigating an approach based on selective active probing to validate the findings of our passive tomography techniques. To this end, we are working on making inferences in real time.

ACKNOWLEDGEMENTS

Rob Emanuel, Scott Hogan, Chris Darling, and Al Lee helped us set up the packet sniffer at the *microsoft.com* site. Chris Meek, David Wilson, Christian Borgs, Jennifer Chayes, and David Heckerman suggested and helped us develop the Gibbs sampling technique. Dimitris Achlioptas provided useful feedback in the early stages of this work. We would like to thank them all.

REFERENCES

[1] M. Allman. A web server's view of the transport layer. *ACM Computer Communication Review*, October 2000.

[2] H. Balakrishnan, S. Seshan, M. Stemm, and R. H. Katz. Analyzing stability in wide-area network performance. In *Proceedings of ACM SIGMETRICS'98*, June 1997.

[3] telnet://ner-routes.bbnplanet.net.

[4] R. Caceres, N. G. Duffield, J. Horowitz, D. Towsley, and T. Bu. Multicast-based inference of network internal characteristics: Accuracy of packet loss estimation. In *Proceedings of IEEE INFOCOM'99*, March 1999.

[5] http://www.cisco.com/warp/public/cc/pd/cxsr/dd/index.shtml.

[6] A. Downey. Using pathchar to estimate internet link characteristics. In *ACM SIGCOMM*, 1999.

[7] N. G. Duffield, F. Lo Presti, V. Paxson, and D. Towsley. Inferring link loss using striped unicast probes. In *Proceedings of IEEE INFOCOM'2001*, April 2001.

[8] C. Fraleigh, S. Moon, C. Diot, B. Lyles, and F. Tobagi. Packet-level traffic measurements from a tier-1 ip backbone. In *Under submission*, November 2001.

[9] S. Geman and D. Geman. Stochastic relaxation, gibbs distributions and the bayesian restoration of images. In *IEEE Trans. Pattn. Anal. Mach. Intel.*, 1984.

[10] W. R. Gilks, S. Richardson, and D. J. Spiegelhalter. *Markov Chain Monte Carlo in Practice*. 1996.

[11] V. Jacobson. Traceroute software. 1989.

[12] V. Jacobson, C. Leres, and S. McCanne. tcpdump - dump traffic on a network.

[13] D. Katabi, I. Bazzi, and X. Yang. A passive approach for detecting shared bottlenecks, October 2001.

[14] B. Krishnamurthy and J. Wang. On network-aware clustering of web clients. In *Proceedings of ACM SIGCOMM'2001*, August 2000.

[15] J.C. Mogul, F. Douglis, A. Feldman, and B. Krishnamurthy. Potential benefits of delta-encoding and data compression for http. In *Proceedings of ACM SIGCOMM '97*, Sept. 1997.

[16] E. M. Nahum, C.-C. Rosu, S. Seshan, and J. Almeida. The effects of wide-area conditions on www server performance. In *Proceedings of ACM SIGMETRICS*, June 2001.

[17] J. Padhye, V. Firoiu, D. Towsley, and J. Kurose. Modeling tcp throughput: A simple model and its empirical validation. In *Proceedings of ACM SIGCOMM'98*, August 1998.

[18] V. Paxson. Measurements and analysis of end-to-end internet dynamics. In *Proceedings of ACM SIGCOMM '97*, Sept. 1997.

[19] S. Ratnasamy and S. McCanne. Inference of multicast routing trees and bottleneck bandwidths using end-to-end measurements. In *Proceedings of IEEE INFOCOM 1999*, 1999.

[20] D. Rubenstein, J. Kurose, and D. Towsley. Detecting shared congestion of flows via end-to-end measurement. In *Proceedings of ACM SIGMETRICS*, 2000.

[21] S. Seshan, M. Stemm, and R. H. Katz. SPAND: Shared passive network performance discovery. In *Proceedings of 1st Usenix Symposium on Internet Technologies and Systems (USITS '97)*, December 1997.

[22] Y. Tsang, M. J. Coates, and R. Nowak. Passive network tomography using em algorithms. In *Proceedings of the IEEE International Conferenceon Acoustics, Speech, and Signal Processing*, May 2001.

[23] Y. Zhang, N. Duffield, V. Paxson, and S. Shenker. On the constancy of internet path properties. In *Proceedings of ACM SIGCOMM Internet Measurement Workshop*, November 2001.

[24] Y. Zhang, V. Paxson, and S. Shenker. The stationarity of internet path properties: Routing, loss, and throughput. In *ACIRI Technical Report*, May 2000.