Server-based Inference of Internet Link Lossiness

Venkata N. Padmanabhan, Lili Qiu, and Helen J. Wang
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. 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. We also perform inference based on Internet traces gathered at the busy *microsoft.com* Web site. However, validating these inferences is a challenging problem. We present a method for indirect validation that suggests that the false positive rate is manageable.

Subject keywords: Network tomography; Network Measurement; Bayesian Inference

Method keywords: Network measurements; Simulations; Mathematical programming/optimization; Statistics

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 lossiness of interior links in the network.

There are a number of ways in which the server could benefit from such inference. 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 link lossiness, not on its applications.

There are, of course, other performance metrics that may be important, depending on the application of interest. 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 on packet loss rate because it is the most direct indicator of network congestion.[1] We view packet loss rate together with RTT 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 [17].

Here is an overview of the rest of this paper. In Section II, we discuss related work. In Section III, we present the key findings from our analysis of end-to-end packet loss rate based on traces gathered at the *microsoft.com* site. We find that end-to-end packet loss rate correlates poorly with the server-to-client hop count, is stable for up to tens of minutes, and exhibits a limited degree of spatial locality. These findings suggest that it would be interesting to identify the few lossy links, whether shared or non-shared, that dominate the end-to-end loss rate.

This sets the stage for our main focus, *Passive Network Tomography*, which we present in Section IV. The goal here is to identify the lossy links in the interior of the network based on *passive* observation at a server of existing traffic between the server and its clients. This is in contrast to much of the previous work on network tomography (e.g., [5]) that has been based on active probing, and hence cannot directly be applied to our problem. We develop three techniques for passive network tomography: Random Sampling, Linear Optimization, and Bayesian Inference using Gibbs Sampling. These techniques depend only on knowing the number of lost and successful packets sent to each client, which is much less information than the exact loss sequence needed by previous techniques such as [5].

In Section V, we evaluate our techniques for passive tomography 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%. In Section VI, 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.

Finally, we present our conclusions in Section VII.

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, such as ours, analyze existing traffic obtained from server logs, packet sniffers, etc.

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

Several studies have examined the temporal stability of Internet performance metrics through active measurements. [19] 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. [23] examines the stationarity of packet loss rate and available bandwidth. It reports that the correlation in the loss process arises mainly 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 a period 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 based on how far apart they were 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) over 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. [15] uses packet-level traces to study the effectiveness of delta compression for HTTP.

Our study is similar to [2] in that it is based on packet 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. More importantly, we go beyond simply characterizing the end-to-end loss rate and use this information to identify the lossy links in the network.

This 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 [5], [4] and [20] base their inference on loss experienced by multicast probe packets while [7], [8] 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). Also, these techniques depend on knowing the exact loss sequence observed at each client, while our passive techniques only require the number of lost and successful packets sent to each client.

[21] and [14] present passive approaches to 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 may not contain enough such packet-pairs to enable 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.

III. Analysis of End-to-End Loss Rate

We analyzed the end-to-end loss rate information derived from traffic traces gathered at the *microsoft.com* site. Due to space limitations, we only present a sketch of our experimental methodology and our key findings here. The technical report version of this paper [18] includes a detailed description of our experimental setup, methodology, and results.

The traces were gathered by running the *tcpdump* tool on a machine connected to the replication port on a Cisco Catalyst 6509 switch. The packet sniffer was thus able to listen on all communication between the servers connected to the same switch and their clients located anywhere in the Internet. We inferred packet loss based on TCP retransmissions by the servers, the assumption being that the conservative TCP retransmission strategy results in few spurious retransmissions. This assumption would clearly be violated, for instance, if the incidence of packet reordering in the Internet were significant enough to frequently overwhelm TCP's threshold of 3 duplicate ACKs for fast retransmissions. The findings regarding packet reordering in the Internet, however, are mixed [3], [12]. Another potential cause of spurious retransmissions (and consequently inaccuracy in our estimation of the packet loss rate) is the loss of ACKs. The cumulative nature of TCP ACKs would mitigate, although not eliminate, this problem.

We gathered multiple traces, each over 2 hours long and containing over 100 million packets. One trace in particular that we use for the analysis presented in Section VI was gathered on 20 Dec 2000. This trace was 2.12 hours long and contained 100 million packets to or from 134,475 clients.

The key findings of our analysis of end-to-end loss rate are:

- The correlation between the end-to-end loss rate and the server-to-client hop count is weak, regardless of whether hop count is quantified at the granularity of routers, autonomous systems (AS), or address prefix (AP) clusters. The coefficient of correlation between the loss rate and the three hop count metrics is 0.05, 0.03, and 0, respectively. That hop count is a poor indicator of end-to-end loss rate suggests that a few lossy links are likely to be responsible for much of the packet loss.
- Loss rate tends to be stable over a period ranging from several minutes to tens of minutes, where stability refers to the notion of "operational stationarity" described in [23].
- Clients that are topologically close to each other experience more similar loss rates than clients picked at random, but in general there is only a limited degree of spatial locality in loss rate. The correlation is strongest at the granularity of /24 subnets and is much weaker at the level of AP clusters or ASes.

These findings suggest that it would be interesting to identify the few lossy links, whether shared or non-shared, that dominate the end-to-end loss rate. This sets the stage for our work on passive network tomography, where we develop techniques to provide greater insight into some of these conjectures.

IV. Passive Network Tomography

In this section, we develop techniques 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*.

Figure 1 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.

To determine the network path from the server to each client, we use the *traceroute* tool [13]. For security reasons, our packet sniffing machine was configured to be in "listen only" mode, so the traceroutes were run from a different machine 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.

While running traceroute does 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 to determine the server-to-client path "pseudo-passively" by invoking the record route option (IPv4) or extension header (IPv6) on a small subset of the packets[2] It may also be possible to discover the topology using (active) end-point delay measurements [6]. Hence, in the rest of our discussion, we assume that the network path from the server to each client is known.

For ease of exposition, we refer to the network topology from the vantage point of the server as a "tree" (as depicted in Figure 1) and couch our discussion in tree-specific terminology. *However, our techniques do not assume that the topology is a tree*. We elaborate on this point in Section IV-B.

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. To make the problem tractable, we focus on estimating

[2]The frequency of invocation could be set adaptively based on the observed frequency of route changes. However, in the extreme case where each packet to a client can potentially be routed via a different path, it might be hard to determine which path a lost packet took.

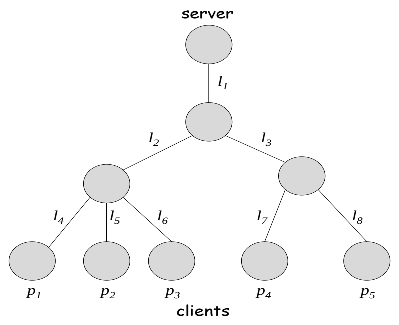

Fig. 1. A sample network topology as viewed from a server. The link loss rates are denoted by l_i and the end-to-end loss rate at the clients are denoted by p_j. Note that although the topology depicted here is a tree, our techniques do not assume that the topology is a tree.

the average link loss rate. Although this is not perfect, 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 an hour. Our temporal locality analysis based on the *microsoft.com* traces indicates a stability duration ranging from several minutes to tens of minutes (Section III and [18]). So it is reasonable to perform inference based on end-to-end packet loss information gathered over such time scales.

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.

We address this issue in several ways. First, we collapse a linear section of a network path with no branches into a single *virtual link*[3]. 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. 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 idea is to find a common cause to the extent possible for the observed packet losses. (Otherwise, we could end up at the other extreme where all of the losses experienced by each client is ascribed to its last-hop link. This trivial "inference" is unlikely to be of much use.) This bias towards parsimony is implicit in the case of Random Sampling and explicit in the case of Linear Optimization. On the other hand, our Gibbs Sampling technique uses a uniform prior and so is unbiased (or, to be more precise, it loses its bias as the Markov chain converges to the steady state).

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

Finally, we set our goal to primarily be the identification of links that are likely to have a high loss rate (i.e., the "lossy" links) rather than inferring a specific loss rate for each link. We believe that the identification of the lossiest 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 indicate, even the simplest technique, yields good results.

B. Random Sampling

The set of constraints mentioned in Section IV-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. (This is akin to the Monte Carlo method.) The solution space is sampled as follows. We first assign a loss rate of zero to each link of the tree (Figure 1). 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., closer to the server) are picked early in the process, and tend to get assigned a higher loss rate. (This bias, however, is consistent with the goal of parsimony, as discussed in Section IV-A.) On the other hand, 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.

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-to-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 IV-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.

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 objective 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 minimizing 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 VI). 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 [11].

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 is $P(D, \theta) = P(D|\theta)P(\theta)$. We can then compute the posterior distribution of θ as follows:

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

In general, it is hard to compute $P(\theta)|D$ directly because of the complex integrations involved, especially when θ is a vector (as it is in our case). An alternative approach is to construct a Markov chain whose stationary distribution exactly equals the posterior distribution of interest ($P(\theta|D)$). 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 to construct an approximation of the posterior distribution. Hence the name *Markov Chain Monte Carlo (MCMC)* [10], [11] is given to this class of techniques.

The challenge then is to construct a Markov chain (i.e., define its transition probabilities) whose stationary distribution matches $P(\theta|D)$. *Gibbs sampling* [10] is a widely used technique to accomplish this. The basic idea is 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.

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 IV-B). The likelihood function can then be written as:[4]

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

Recall from Section IV-A that $p_j = 1 - \prod_{i \in T_j} (1 - l_i)$ and represents the loss rate observed at client C_j. Note that equation 1 assumes a Bernoulli loss process, where the probability of a packet getting lost is independent of the fate of other packets.

[4]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.

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., $P(l_L) = 1$, since we do not have information, such as the type or nature of individual links, that could serve as the basis of a prior.

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

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

Since $P(l_L)$ is a uniform distribution, and $\{l_i\} \cup \{\bar{l_i}\} = l_L$, we have

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

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[5]. 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.

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.

V. Simulation Results

In this section, we show results of our experimental evaluation of the three passive network tomography techniques presented on Section IV. We present simulation results here; Internet results are presented in Section VI. The main advantage of simulation is that the true link loss rates are known, so validating the inferences of the tomography techniques is easy.

[5]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.

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 non-leaf 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 network topology constructed from our traceroute data set. This topology spans 123,166 clients drawn from the Dec 2000 trace gathered at the *microsoft.com* site (the number of clients is somewhat smaller than that reported in Section III because we ignore clients to whom traceroute failed).

A fraction, f, of the links were classified as "good" and the rest as "bad". We use 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 is picked uniformly at random in the 0-1% range and that for bad links is picked in the 5-10% range. In the second model (LM_2), the loss rate ranges for good and bad links are 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. In the good state, no packets are dropped while in 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 probabilities are 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 [23], these models offer a reasonable approximation.

We repeat 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 IV-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 (drawn from 500 to 2000 iterations) 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.

A. Random Topologies

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

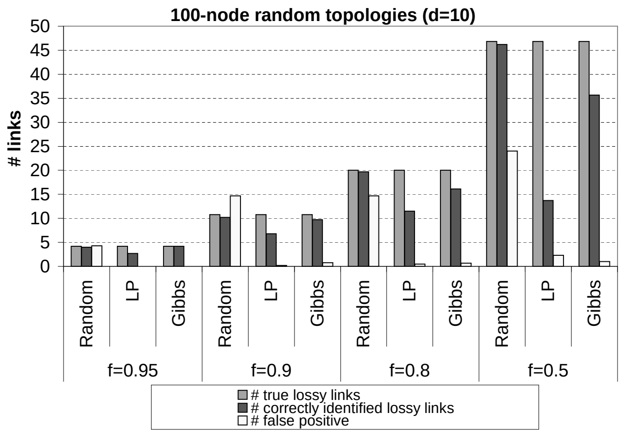

Fig. 2. Varying f: 100-node random topologies with maximum degree = 10.

Figure 2 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 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 IV-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 2 is hard to see because the number of false positives is close to or equal to zero.) As explained in Section IV-C, LP is less susceptible to statistical fluctuations in the end-to-end loss rates since it allows some slack in the constraints. This reduces the false positive rate. However, the slack in the constraints and the fact that the objective function assigns equal weights to 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-C, we examine the impact of 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 the *number* of lost and successful packets, and not on the (noisy) computation of packet loss *rates*.

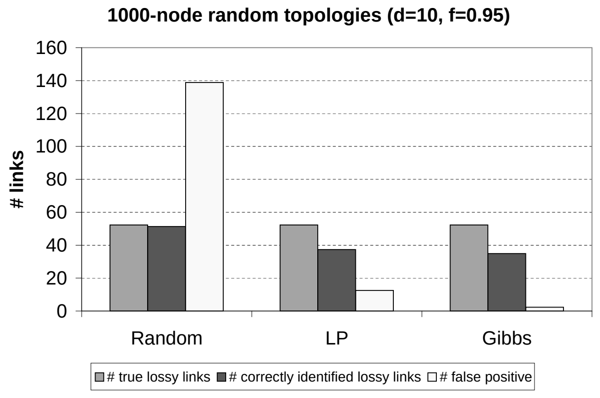

Fig. 3. 1000-node random topologies with maximum $degree = 10$ and $f = 0.95$.

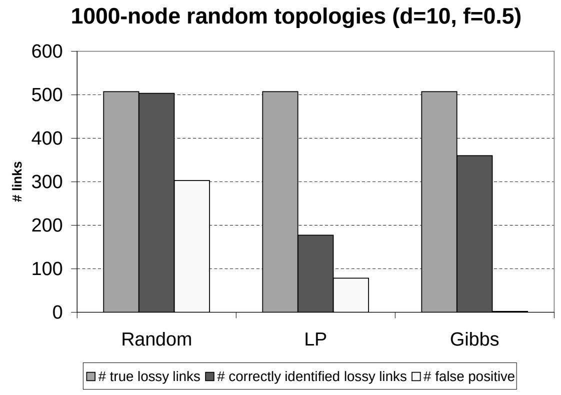

Fig. 4. 1000-node random topologies with maximum $degree = 10$ and $f = 0.5$.

Figures 3 and 4 show the corresponding results for experiments on 1000-node topologies. Figure 5 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 a false positive rate less than 5%.

Figure 6 shows 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 the order of decreasing confidence. We

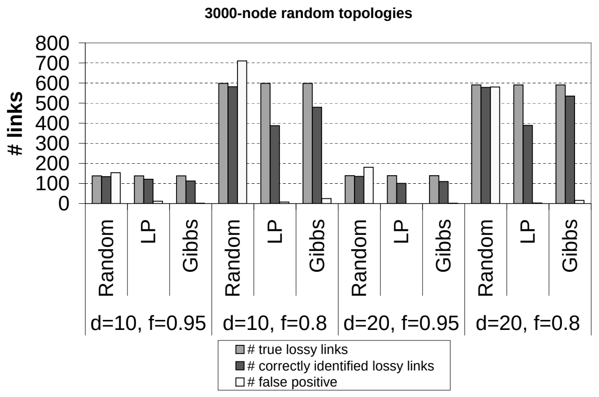

Fig. 5. 3000-node random topologies.

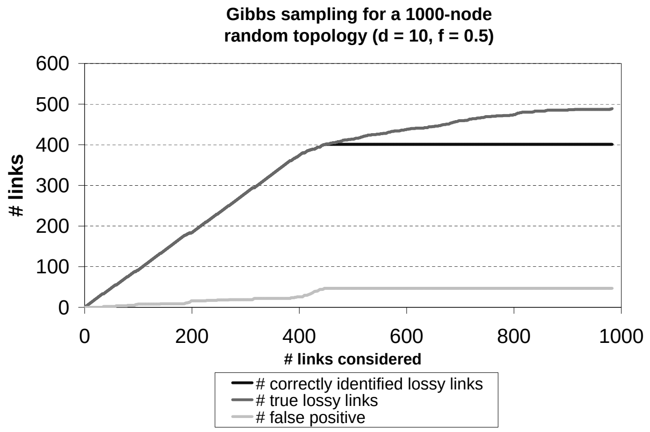

Fig. 6. The performance of Gibbs sampling when the inferences are rank ordered based on a confidence estimate. (1000-node random topology, maximum $degree = 10$, and $f = 0.5$)

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 correctly 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.

B. 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 7 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 three techniques. 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.

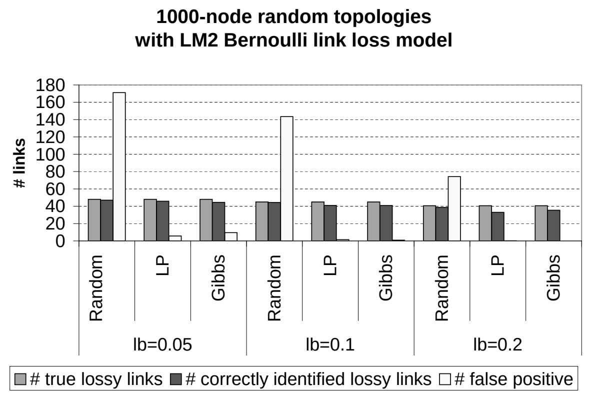

Fig. 7. A LM_2 Bernoulli loss model for 1000-node random topologies with maximum $degree = 10$ and $f = 0.95$. We vary the loss threshold lb, and only the links with loss rate higher than lb are considered lossy.

LM_1 **and** LM_2 **Gilbert loss models:** Figure 8 and Figure 9 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 technique continues to be the best performer: it has a coverage of around 90% with the lowest false positive rate among all the schemes. This good performance is despite the underlying likelihood computation being based on a different (i.e., Bernoulli) loss model (equation 1 in Section IV-D.2). The insensitivity to the loss model is in part because we are only evaluating the accuracy of identifying lossy links, not computing the actual link loss rates.

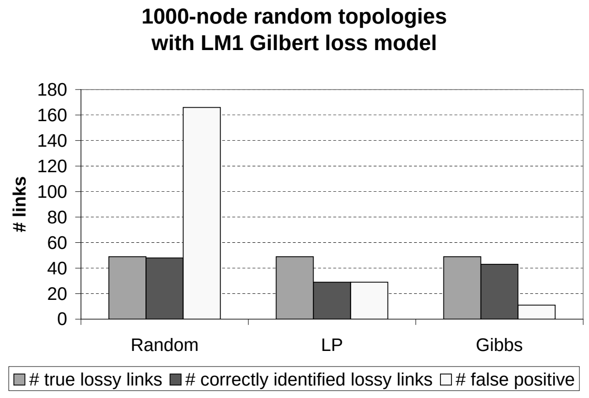

Fig. 8. A LM_1 Gilbert loss model for 1000-node random topologies with maximum $degree = 10$ and $f = 0.95$.

C. Different Weights in LP

As discussed in Section IV-C, the linear optimization technique seeks 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 10 and Figure 11 show the LP performance for 1000-node random topologies under Gilbert LM_1 and LM_2 loss models, respectively. As we can see, the smaller w is, the better is coverage that the inference achieves, but at the cost of a higher false positive rate. This is because when w is decreased, a greater emphasis is placed on satisfying the constraints than on finding a parsimonious solution; as a result, we are more likely to attribute 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.

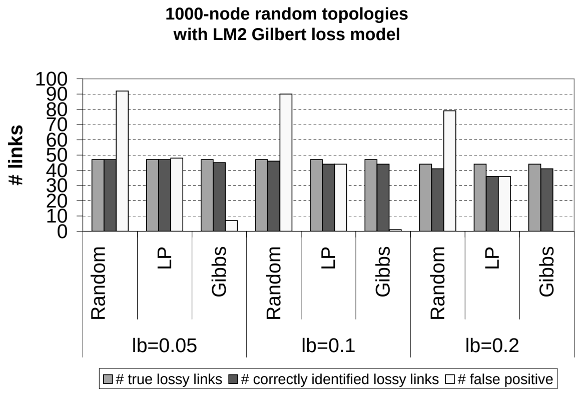

Fig. 9. A LM_2 Gilbert loss model for 1000-node random topologies with maximum $degree = 10$ and $f = 0.95$. We vary the loss threshold lb, and only the links with loss rate higher than lb are considered lossy.

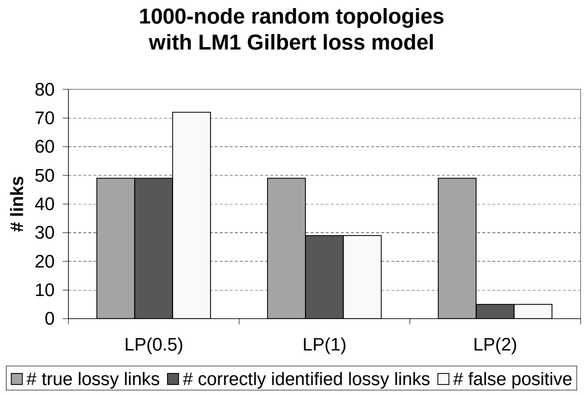

Fig. 10. Effects of different weights in LP: A LM_1 Gilbert loss model for 1000-node random topologies with maximum $degree = 10$ and $f = 0.95$.

D. Real Topology

We also evaluate the effectiveness of inference using a real topology (constructed from traceroute data) spanning 123,166 clients. We assign a loss rate to each link based on the LM_1 Bernoulli loss model with different settings of f. Figure 12 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.

VI. 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

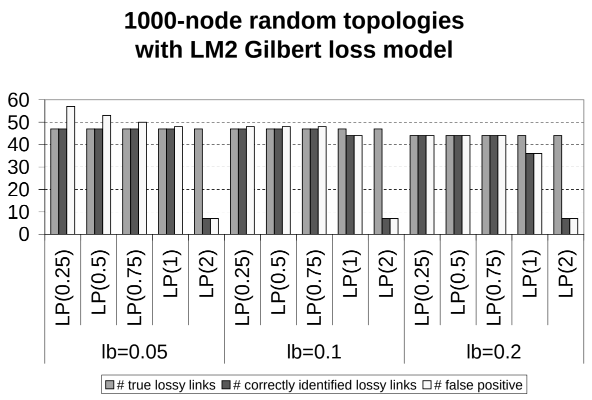

Fig. 11. Effects of different weights in LP: A LM_2 Gilbert loss model for 1000-node random topologies with maximum $degree = 10$ and $f = 0.95$. We vary the loss threshold lb, and only the links with loss rate higher than lb are considered lossy.

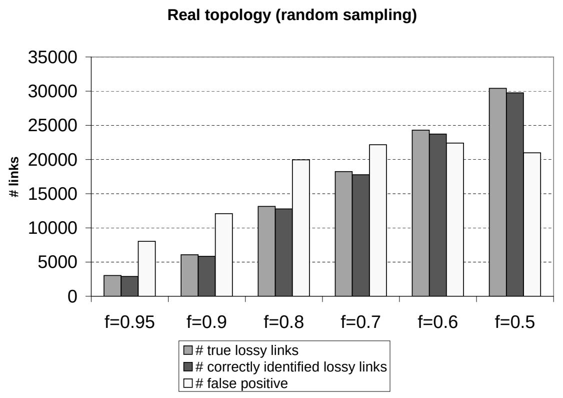

Fig. 12. Real topology from the Dec 2000 traceroute.

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

The evaluation we present here is based on the first hour of the Dec 2000 trace mentioned in Section III. To compute the end-to-end 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. The results presented are based on about 4000 such clients.

A. Consistency Across the Different Schemes

First, we examine the consistency in the lossy links identified by the three tomography techniques. Figure 13 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. [6] The overlap between LP and the other techniques is also significant — over 60%.

[6]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 VI-B, most of the 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.

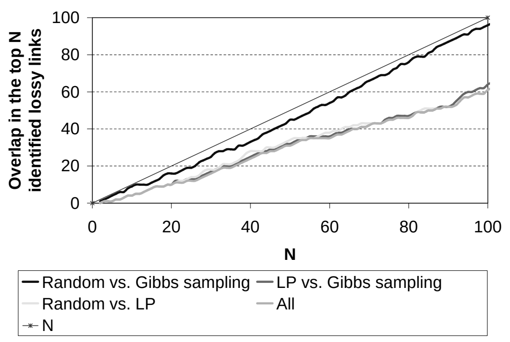

Fig. 13. Overlap in the top N lossy links identified by different schemes.

B. 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 inferred lossy links in the Internet topology. As shown in Figure 14, 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 IV-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 [9]. 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 III.

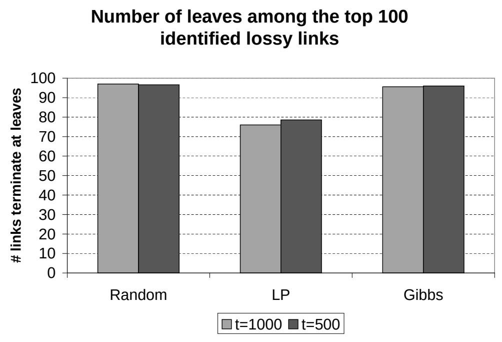

Fig. 14. Number of lossy links that terminate at leaf nodes.

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 virtual links inferred to be lossy, 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

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 I
TRACE-DRIVEN VALIDATION FOR RANDOM SAMPLING AND LINEAR OPTIMIZATION.

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 the U.S. to China.

C. *Trace-driven Validation*

We now consider the problem of validating our inferences more directly than the intuitive arguments made in Section VI-B. 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: the tomography set and the 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 VI-B.

Table I 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 to 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.

VII. CONCLUSIONS

In this paper, we investigate the problem of inferring the loss characteristics of Internet links based on passive observation at a server of existing end-to-end, client-server traffic. Based on our analysis of traffic traces gathered at the busy *microsoft.com* Web site, we find that the end-to-end packet loss rate correlates poorly with the server-to-client hop count, remains stable for up to tens of minutes, and exhibits a limited degree of spatial locality. These findings suggest that it would be interesting to identify the few lossy links that dominate the end-to-end loss rate.

We develop and evaluate three 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), 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 gathered at the *microsoft.com* site. 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 that the last-mile is often the bottleneck. Finally we develop an indirect validation scheme, and show the false positive rate is manageable (below 30% in most cases and often much lower).

Although the finding that most of the lossy links are non-shared may appear to weaken our original motivation of identifying lossy links in the interior of the network, we would like to note that our simulation results do indicate that the techniques we have developed are effective in finding shared lossy links where they exist. This makes us optimistic about the effectiveness of these techniques in instances where significant losses happen in the interior of the network, say due to a major outage or failure, as well as in the future Internet where the last-mile problems may be alleviated with the increasing deployment of broadband connectivity.

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 educated us about Gibbs sampling. Dimitris Achlioptas provided useful feedback in the early stages of this work. The anonymous Infocom 2003 reviewers provided useful feedback on an earlier version of this paper. 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] J. Bennett, C. Partridge, and N. Shectman. Packet reordering is not pathological network behavior. *IEEE/ACM Transactions on Networking*, December 1999.

[4] R. Caceres, N. G. Duffield, J. Horowitz, and D. Towsley. Multicast-based inference of network-internal loss characteristics. *IEEE Transactions in Information Theory*, November 1999.

[5] 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.

[6] M. Coates, R. Castro, and R. Nowak. Maximum likelihood network topology identification from edge-based unicast measurements. In *Proceedings of ACM SIGMETRICS*, June 2002.

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

[8] 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.

[9] 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.

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

[11] W. R. Gilks, S. Richardson, and D. J. Spiegelhalter. *Markov Chain Monte Carlo in Practice*. Chapman and Hall, London, U.K., 1996.

[12] G. Iannaccone, S. Jaiswal, and C. Diot. Packet reordering inside the sprint backbone. In *Sprint ATL Technical Report TR01-ATL-062917*, June 2001.

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

[14] D. Katabi, I. Bazzi, and X. Yang. A passive approach for detecting shared bottlenecks. In *IEEE ICCCN*, October 2001.

[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. N. Padmanabhan, L. Qiu, and H. J. Wang. Server-based inference of internet performance. In *Microsoft Research Technical Report MSR-TR-2002-39*, May 2002.

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

[20] 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.

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

[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.