Unicast-based Characterization of Network Loss Topologies

Khaled Harfoush harfoush@cs.bu.edu
Azer Bestavros best@cs.bu.edu
John Byers byers@cs.bu.edu

Computer Science Department
Boston University

Abstract— **Current Internet transport protocols make end-to-end measurements and maintain per-connection state to regulate the use of shared network resources. When a number of such connections share a common endpoint, that endpoint has the opportunity to correlate these end-to-end measurements to better diagnose and control the use of shared resources. A valuable characterization of such shared resources is the "loss topology". From the perspective of a server with concurrent connections to multiple clients, the loss topology is a logical tree rooted at the server in which edges represent lossy paths between a pair of internal network nodes. We develop an end-to-end unicast packet probing technique and an associated analytical framework to: (1) infer loss topologies, (2) identify loss rates of links in an existing loss topology, and (3) augment a topology to incorporate the arrival of a new connection. Correct, efficient inference of loss topology information enables new techniques for aggregate congestion control, QoS admission control, connection scheduling and mirror site selection. Our extensive simulation results demonstrate that our approach is robust in terms of its accuracy and convergence over a wide range of network conditions.**

I. Introduction

Motivation: A popular Internet server may potentially command a large number of concurrent unicast connections. As such, these servers are likely to contribute a significant portion of the data traffic on the Internet.[1] We use the term *Mass Servers* to refer to this class of Internet servers—namely, *Mass*ively Accessed *Servers* that command a large number of concurrent unicast connections. While most of the connections at a Mass server are likely to be to *different* clients, many may in fact be traversing the *same* set of congested resources.

This work was partially supported by NSF research grants CCR-9706685 and ANIR-9986397.

[1]As an anecdotal evidence, recent analysis of campus traffic showed that 3% of all accessed Internet servers were responsible for over 50% of all flows and that 0.1% (only 6 servers) were responsible for over 10% of all flows.

If flows sharing common congested resources can be identified by a Mass server, then improved network resource usage can be achieved through judicious allocation of bandwidth. In particular, rather than controlling connections traversing congested network resources independently, a Mass server could apply an aggregate control mechanism to such connections [2], [1]. Applications of this technique could extend well beyond the domain of congestion control to QoS admission control, improved connection scheduling at webservers and to the application of selecting multiple mirror sites in parallel [4]. But in order for any of these control strategies to be practical, an endpoint must be able to quickly and accurately infer the internal loss characteristics of the network connecting it to its peers at the other endpoints.

Network Loss Topology: Previous Internet characterization studies have focused on the discovery of characteristics of Internet structure that are tightly related to physical attributes of the network (e.g. routers, link speeds, AS topology) [10]. For the real-time resource management problems that face Mass servers, an accurate characterization of the physical resources between the server and its clients is not necessary. Rather, an abstraction that captures the shared resources to be judiciously managed is sufficient. In the case of Mass servers, the key resources which must be managed are those with high utilization—namely, congested network paths, or more specifically "lossy paths".

Given a set of network endpoints, we define the *loss topology* to be a graph. The vertices of this graph represent the set of network endpoints as well as a set of internal network nodes. An edge in the loss topology graph represents a sequence of (i.e. one or more) physical network hops (or links) that, collectively, exhibit packet loss rates of at least c, for some constant $c > 0$. Notice that the loss topology graph can be obtained by merely collapsing the "physical" topology graph—by reducing any sequence of links with insignificant loss rates in the physical topology graph to an internal node in the loss topology graph.

Paper Contributions: We consider three problem statements related to loss topology characterization. In each of these problems, we consider a scenario in which there is a single server which has active connections to a number of distinct clients.

+ *Inferring a Loss Topology:* For an existing fixed set of clients, obtain the loss topology connecting the server to the set of clients.
+ *Labeling a Loss Topology:* Given a loss topology known *a priori*, determine the loss rates on each link in the loss topology.
+ *Augmenting a Loss Topology:* Given an existing, labeled loss topology and a new client, augment the existing topology to produce a new labeled loss topology that includes the client.

This paper proposes solutions to the above three problems. The cornerstone of these solutions is an analytical technique for the estimation of key loss statistics for a simple queuing abstraction of lossy paths in a loss topology. Our solutions to these problems is evaluated through extensive simulations. There is a significant amount of related work on this topic; we defer the presentation of this work to Section VI.

II. Loss Topologies as Effective Representations of Shared Resources

Consider the set of links used to route unicast traffic between a server and a large number of clients. Together these links form a tree $\mathcal{T}$ rooted at the server, with the clients at the leaves and routers at the internal nodes. The flows of packets sent from a server to its clients share some of $\mathcal{T}$'s links and then continue on separate links en route to the different clients. A link L_i is the link whose downstream node is node i. A link terminating at a client is called a *terminal* link; a link terminating at an internal node is called an *internal* (or *non-terminal*) link.

As we initially motivated in the introduction, for many applications, the only resources that need to be managed by a Mass server are the "lossy paths" between that server and its clients. Given a set of network endpoints, we define the *loss topology* to be a graph. The vertices of this graph represent the set of network endpoints as well as a set of internal network nodes. Our techniques will be unable to assign loss probabilities on a link-by-link basis to a sequence of physical links in a chain with no branching. This implies that any internal node in the loss topology will have at least two children. Thus, an edge in the loss topology graph represents a sequence of one or more physical network hops that collectively exhibit "observable" packet loss rates, i.e. above a preset threshold as defined below.

Definition 1: The sensitivity constant $0 \leq c \leq 1$ for a loss topology $\mathcal{T}_c$ is the minimum loss probability for any internal link in $\mathcal{T}_c$.

Obviously, the larger the value of the sensitivity constant c, the smaller the number of links present in the loss topology $\mathcal{T}_c$. In this sense, $\mathcal{T}_c$ represents a "condensed" version of the original topology, and increasing the value of the sensitivity constant c has the effect of increasing the level of condensation. For example, a loss topology for which $c = 0$ (i.e. $\mathcal{T}_0$) amounts to a compressed version of the physical topology where chains of links in the original tree are collapsed into a single node. A loss topology for which $c > 0$ can result in more extensive condensation, where arbitrary connected subgraphs of internal nodes can collapse into a single node. We illustrate this with an example.

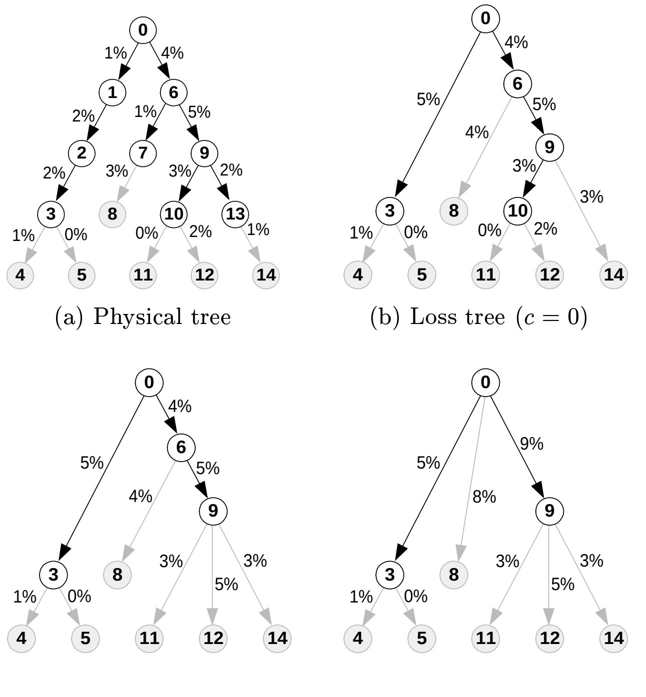
Consider the physical tree shown in Figure 1(a). This tree has 15 nodes. Node 0 is the server (root). Nodes 4, 5, 8, 11, 12, and 14 are the clients (leaves), and the remaining nodes constitute the internal nodes (routers) of that topology. The links in Figure 1(a) are labeled with the actual loss rates on these links. Figure 1(b) shows the loss topology $\mathcal{T}_0$ for this physical tree when $c = 0$. Notice that in $\mathcal{T}_0$, paths with intermediate nodes of unit out-degree (e.g. the path between nodes 0 and 3 and the path between nodes 6 and 8) are collapsed into a single (logical) link. Figures 1(c) and 1(d) show the loss topologies for this physical tree when $0.03 < c \leq 0.04$ and when $0.04 < c \leq 0.05$, respectively. These topologies are obtained from T_0 by eliminating internal links with loss probabilities that are less than the sensitivity constant c.

III. Loss Topology Identification Using Unicast Probing

Our solutions to the problems spelled out in the previous section make use of the Bayesian Probing (BP) technique detailed in [12]. In this section, we start with a summary of the BP technique and then proceed to the solution of the loss topology inference, labeling, and augmentation problems.

A. Overview of Bayesian Probing

Consider clients 11 and 14 in the topology shown in Figure 1(a). The two paths from the server (node 0) to each of these clients can be partitioned into two portions: a portion that is *shared* between the two paths and a portion that is unique for each path. Specifically, L_6L_9 is a shared segment on both paths, whereas $L_{10}L_{11}$ and $L_{13}L_{14}$ are unique segments on each path. The BP technique provides us with a simple probing methodology that enables the characterization of each one of these three segments using end-to-end unicast probes from the server (node 0) to its



(a) Physical tree (b) Loss tree ($c = 0$)

(c) Loss tree ($.03 < c \leq .04$) (d) Loss tree ($.04 < c \leq .05$)

Fig. 1. Illustration of relationship between physical and loss topologies under various sensitivity constants.

clients (nodes 11 and 14). To that end the BP technique uses two types of probe sequences:

Definition 2: A *1-packet probe sequence* $S_i(\Delta)$ is a sequence of packets destined to client i such that any two packets in $S_i(\Delta)$ are separated by at least Δ time units.

Definition 3: A *2-packet probe sequence* $S_{i,j}(\Delta, \epsilon)$ is a sequence of packet-pairs where one packet in each packet-pair is destined to i and the other is destined to j, and where the intra-pair packet spacing is at most ϵ and the inter-pair spacing is at least Δ time units.

1-packet probe sequences provide a baseline loss rate over end-to-end paths while 2-packet probe sequences enable measurement of loss rates over shared links. The key insight is that because of their temporal proximity, packets within a packet pair have a high probability of experiencing a shared fate on the shared links. If the incidence of shared loss on the shared links is high, this leads to an increased probability of witnessing coupled losses within a packet pair. While we will describe appropriate settings of Δ and ϵ in our experimental section, we will generally require ϵ to be on the order of a millisecond and Δ to be on the order of a second, to achieve high dependence and ensure independence, respectively.

Consider two clients A and B (e.g. nodes 11 and 14 as illustrated before). With our packet probe sequences, there are four experimental outcomes which we use in our analysis: successful probes in the 1-probe sequences, successful packet-pair probes in the 2-probe sequence, and unsuccessful probes in the 2-probe sequence in which *both* packets in a pair are lost. The following notation will be useful throughout our analysis. Let g_A and g_B denote the fraction of the 1-packet probes in $S_A(\Delta)$ and $S_B(\Delta)$ respectively which were successfully received. Similarly, let $g_{A,B}$ denote the fraction of the 2-packet probes in $S_{A,B}(\Delta, \epsilon)$ that were successfully received by *both* clients A and B and let $b_{A,B}$ denote the fraction of the 2-packet probes that were lost en route to *both* clients A and B. Note that $g_{A,B} + b_{A,B}$ may be less than 1 due to pairs of probes in which one probe is lost en route to one client while the other probe arrives successfully at the other client.

To establish a relationship between outcomes of probes and network queues, we use the following terminology and notation. Any individual queue can accomodate zero, one, or more than one fixed-size probe packets at any time instant. In general, we define p_i^k be the steady-state probability that the queue at L_i can store exactly k probe packets, and p_i^{k+} be the probability that the queue at L_i can store k or more probe packets. From this definition, p_i^{1+} is the probability that a single probe packet sent over L_i at an instant chosen at random will successfully traverse L_i and p_i^0 is the probability that such a probe will be lost over L_i. With this notation, we can present the following relationships (from [12]) between probe sequences and queue sizes.

Fact 1: The quantities g_A and g_B are unbiased estimators for $\prod_{i \in L_A} p_i^{1+}$ and $\prod_{i \in L_B} p_i^{1+}$, respectively.

Fact 2: The quantity $g_{A,B}$ is an unbiased estimator for $\prod_{i \in L_S} p_i^{2+} \prod_{i \in (L_A \bigcup L_B) \setminus L_S} p_i^{1+}$

Fact 3: The quantity $b_{A,B}$ is an unbiased estimator for $\left(1 - \prod_{i \in L_S} p_i^{1+}\right) + \left(\prod_{i \in L_S} p_i^{1+} - \prod_{i \in L_S} p_i^{2+}\right)(q_A + q_B) + \prod_{i \in L_S} p_i^{2+} q_A q_B$, where $q_A = 1 - \prod_{i \in L_A \setminus L_S} p_i^{1+}$, and $q_B = 1 - \prod_{i \in L_B \setminus L_S} p_i^{1+}$,

From these three facts, we can obtain an unbiased estimate for a quantity which occupies a central location in Fact 3 and which we define as follows:

$$X = \prod_{i \in L_S} p_i^{1+} - \prod_{i \in L_S} p_i^{2+} \qquad (1)$$

X can be interpreted as the probability of a packet pair encountering a situation on the shared links in which all shared queues have space for one probe packet, but not all queues have space for two packets. The following Lemma shows that we can obtain a (surprisingly simple) estimate for X:

Lemma 1: The quantity $g_A + g_B + b_{A,B} - g_{A,B} - 1$ is an unbiased estimator for X.

Proof: The complete proof is available in [12]. ■

Obtaining an unbiased estimate of the value of X is valuable because the magnitude of X is highly correlated with the magnitude of packet losses on the shared segment of the paths from a server to two clients. In particular, one can show (through M/M/1/k analysis and also through simulations with realistic background traffic) that X is an upper bound on the shared loss probability [12]. The bound is tight for light loads (generating losses of 1% or less) and is looser for heavier loads.[2]

B. Loss Topology Inference

The BP technique provides us with a mechanism via which the loss probabilities on the shared and independent segments of paths to two clients can be estimated. Our proposed technique for loss topology identification relies on using the amount of shared losses between pairs of clients as an indication of how closely located the receivers are in the tree. This is akain to the approach used by Ratnasamy and McCanne in [19] for multicast routing trees inference. Clients having the highest shared losses are aggregated together and represented by a single node. The aggregated nodes are then regarded as a single node for further aggregation. This recursive approach is terminated when all nodes have been aggregated into a single tree.

Notice that lemma 1 estimates the value of X between two clients. In order to compute an estimate of the X value between aggregate nodes, we could pick a client from each aggregate node as a representative of this node, and compute the X estimate between these representatives. Another approach would be to compute the average X estimate over all possible representatives of the aggregate nodes. The latter approach was adopted in our simulations. Also, in order to reconstruct loss topologies of arbitrary node degrees, all pairs of nodes having their X values close to each other (to within the value of the sensitivity constant c) are aggregated together.

To summarize, we can infer the loss topology using 1-packet and 2-packet probe sequences from the server to each client and to each pair of clients respectively. Using lemma 1, the X estimates are computed for all pairs and the loss topology inference technique is applied recursively.

[2]Empirically, and through M/M/1/k analysis we found that X is a good estimator for twice the shared loss probability under heavy loads (i.e. losses between 2% and 10%) [12].

C. Loss Topology Labeling

Consider the loss topology connecting a server and two clients A and B. Such a topology has three links: one corresponding to the shared links L_S, one corresponding to the links $L_A \setminus L_S$ and one corresponding to the links $L_B \setminus L_S$. The loss rates estimates of these links are $1 - \prod_{i \in L_S} p_i^{1+}$, $1 - \prod_{i \in L_A \setminus L_S} p_i^{1+}$ and $1 - \prod_{i \in L_B \setminus L_S} p_i^{1+}$, respectively. Thus, in order to label the loss topology, we need unbiased estimates for $\prod_{i \in L_S} p_i^{1+}$, $\prod_{i \in L_A \setminus L_S} p_i^{1+}$ and $\prod_{i \in L_B \setminus L_S} p_i^{1+}$. Next, we show that these estimates could be obtained using the g_A, g_B and $g_{A,B}$ estimates, together with the X estimate from lemma 1.

Lemma 2:

◇ $R = \frac{1}{2}(K \pm \sqrt{K^2 - 4XK})$ is an unbiased estimator for $\prod_{i \in L_S} p_i^{1+}$, where $K = \frac{g_A g_B}{g_{A,B}}$

◇ $\frac{g_A}{R}$ is an unbiased estimator for $\prod_{i \in L_A \setminus L_S} p_i^{1+}$

◇ $\frac{g_B}{R}$ is an unbiased estimator for $\prod_{i \in L_B \setminus L_S} p_i^{1+}$

Proof: By rewriting the g_A, g_B and $g_{A,B}$ equations in terms of $\prod_{i \in L_S} p_i^{1+}$, $\prod_{i \in L_A \setminus L_S} p_i^{1+}$ and $\prod_{i \in L_B \setminus L_S} p_i^{1+}$, we get the following equations:

$$g_A = \prod_{i \in L_S} p_i^{1+} \prod_{i \in L_A \setminus L_S} p_i^{1+} \quad (2)$$

$$g_B = \prod_{i \in L_S} p_i^{1+} \prod_{i \in L_B \setminus L_S} p_i^{1+} \quad (3)$$

$$\begin{aligned} g_{A,B} &= \prod_{i \in L_S} p_i^{2+} \prod_{i \in (L_A \bigcup L_B) \setminus L_S} p_i^{1+} \\ &= (\prod_{i \in L_S} p_i^{1+} - X) \prod_{i \in L_A \setminus L_S} p_i^{1+} \prod_{i \in L_B \setminus L_S} \quad (4) \end{aligned}$$

The proof follows directly by solving the three equations 2, 3, and 4 for the three unknown quantities $\prod_{i \in L_S} p_i^{1+}$, $\prod_{i \in L_A \setminus L_S} p_i^{1+}$, and $\prod_{i \in L_B \setminus L_S} p_i^{1+}$ in terms of g_A, g_B, $g_{A,B}$ and X. ■

Lemma 2 shows that we can get the unbiased estimates needed to label the loss topology by using the g_A, g_B, $g_{A,B}$ and X estimates. Thus, by using the 1-packet and 2-packet probe sequences between the server and any pair of clients, we can obtain the g_A, g_B, $g_{A,B}$ and X estimates and label the underlying loss topology links. Note that if the clients are not sharing any links the loss rate along L_S will be negligible and the shared link could be removed from the loss topology. This means that the model is general enough to deal with clients whether they are sharing physical links or not.

In the case of arbitrarily large loss topologies, the same labeling strategy could be applied recursively in a bottom-up (or top-down) fashion as shown in the algorithm of figure 2.

```
Procedure LossTopologyLabel($d$) {
  for all nodes $w$ of depth $d$ do
    if $w$ has at least 2 children $a$ and $b$ then
      Pick 2 leaves $i, j$, where $i$ and $j$ are
        descendants of $a$ and $b$, respectively
      $K = \frac{g_i g_j}{g_{i,j}}$;
      $X = g_i + g_j + b_{i,j} - g_{i,j} - 1$;
      $Label(L_w) = \frac{1}{2}(K \pm \sqrt{K^2 - 4XK})$;
      $Label(L_a) = \frac{Label(L_a)}{Label(L_w)}$;
      $Label(L_b) = \frac{Label(L_b)}{Label(L_w)}$;
    fi
  od
  if ($d > 1$) LossTopologyLabel($d - 1$);
}
```

Fig. 2. The Unicast Loss Topology Labeling Algorithm

The input to the labeling algorithm is the loss topology as well as the g_A, g_B, $g_{A,B}$ and $b_{A,B}$ estimates for all pairs of clients A and B computed using 1-packet and 2-packet probe sequences. The resulting labels of the loss topology are stored in the variables $Label(L_i), \forall L_i$. Notice that the resulting labels are the pass-through rates and not the loss rates (which could be easily obtained by subtracting the pass-through rates from 1). Also note that the value of $Label(L_i)$, for a client i should be initialized to g_i. Also, since the algorithm is working in a bottom-up fashion, it should be called initially using `LossTopologyLabel`$(n-1)$ where n is the depth of the tree.

The loss topology labeling algorithm shown in Figure 2 gives *two* solutions for link loss rates (corresponding to the two values of R in Lemma 2.) Thus, in order to use this algorithm, one must devise a procedure that identifies the *correct* solution out of these two candidate solutions. Space limitations prohibit us from presenting the specifics of such a procedure (which we use in our ns simulations in Section IV). Interested readers are referred to [13] for details.

D. Basic Assumptions

A basic premise of our work is that while we assume the loss rate on all links in our topology may have substantial short-term variability, the mean packet loss rate on each link is stationary over longer time scales. This stationarity requirement is needed to allow our inference/labeling procedures to converge. Thus, stationarity is required only over time scales that are comparable to the time it takes our procedures to converge, which (as shown in the next section) is quite fast. This renders our technique quite effective even when stationarity can only be assumed for short intervals, on the order of a few seconds.

In the analysis presented above, we have made the following additional assumptions:

1. Link losses are only due to queue overflows.

2. Losses on L_i and L_j are independent, $\forall i, j : i \neq j$.

3. A reliable feedback mechanism enables the sender to determine whether a probe packet was lost.

4. The temporal constraints imposed on probe sequences are preserved throughout the journey of the probes from sender to receivers.

Assumption 1 reflects the current DropTail behavior present in most Internet routers today. Assumption 2 allows us to ignore any spatial correlation between link losses, and thus ignore any additional correlation terms. Assumption 3 enables us to assume that the server is able to accurately identify the outcome of the probing process, i.e. which packets of a 1-packet or 2-packet probe sequences were lost.

Assumption 4 is our most significant assumption, since it ensures that the individual packets within each packet-pair of a 2-packet probe sequence $S_{A,B}(\Delta, \epsilon)$ are separated by at most ϵ time units on *all* traversed links. Moreover, we must be assured that ϵ is sufficiently small that two packets of a packet-pair are close enough to each other on all traversed links to enable an accurate sampling of the state of a queue at the time the 2-packet probe reaches that queue. In particular, we need to use p_i^{2+} as the probability that the two packets of a packet-pair have traversed link i. Ideally, we would desire that the two packets reach the queue with an inter-arrival time $\epsilon = 0$. If the packets in a pair become substantially separated from one another in flight, our estimates $g_{A,B}$ and $b_{A,B}$ will be biased. In [12], we have studied the effects of $\epsilon > 0$ on the performance of our BP technique. Our findings confirm that the bias introduced by small amounts of separation (up to few milliseconds) and/or long paths is not excessive.

IV. Performance Evaluation

In this section we present results of extensive simulations that demonstrate the accuracy, convergence and robustness of our approach.

A. Simulation Environment

Bayesian Probing Technique: Recall from our presentation in section III, the BP technique requires the specification of the Δ and ϵ parameters of the temporal constraints imposed on 1-packet and 2-packet probe sequences.

In the experiments we present in this section, each probe sequence was generated using independent Poisson processes. The probing rate for each such process was set to 5 probes/sec. This means that the value of Δ is not strictly larger than a specific value but is 200 msec on average. For 2-packet probing processes, the value of ϵ was set to 0; that is packets within a packet-pair were sent back-to-back, with no time separation. Also, to normalize the losses on the shared links experienced by both receivers, the 2-packet probes in $S_{A,B}(\Delta, \epsilon)$ alternate between the two possible packet orderings.

Another parameter of our topology inference is the value of the sensitivity constant (c). Recall that the value of c determines what constitutes "detectable" loss probabilities. In our experiments, the value of c was fixed at 0.04. This value was chosen empirically. The effect of changing c on the performance of our inference strategy is discussed later.

Link Baseline Model: Each one of the links used in our simulations is modeled by a single DropTail queue. The link delays were all set to 40ms and the link buffer sizes were all set to 20 packets. Each of these links was subjected to background traffic resulting from a set of Pareto ON/OFF UDP sources with a constant bit rate of 36Kbps during the ON times with a packet size of 200 bytes. The average ON and OFF times were set to 2 seconds and 1 second, respectively. The Pareto shape parameter (α) was set to 1.2. After a "warm-up" period of 10 seconds, the probing processes (and associated loss topology inference/labeling processes) are started.

To represent the various levels of congestion that any of these links may exhibit, we have chosen three sets of parameters that result in "High", "Mild", and "Low" levels of congestion. The baseline parameter settings for these congestion levels (and the resulting loss rates) are tabulated in Table I.

Parameter Setting	Congestion Level		
	High	Mild	Low
Link bandwidth	1Mb/sec	1Mb/sec	100Mb/sec
Background flows	60	56	44
Observed loss rate	7-15%	$< 7\%$	$< 1\%$

Table I. Settings used (and resulting loss rates) for the three levels of congestion considered

Topology (Tree) Baseline Model: In order to test our solutions of the inference, labeling, and augmentation problems, we had to generate random topologies (trees) to use as a baseline test set. Each one of the trees in the baseline set has a total of 5 "base" clients.

In order to enrich the topology between the server and these 5 clients, we have added 8 more "dummy" clients. These 13 (base + dummy) clients represented the leaf nodes of the tree. To generate the internal nodes of the tree, an iterative process was used, whereby a set of d nodes (leaves or internal nodes) were selected at random, and connected to a single new parent node. This process of adding internal parent nodes was continued until only one node remained—namely the root of the entire tree—representing the server. Once this process is completed, the dummy clients were trimmed off, while keeping the internal nodes induced by these clients.

Clearly, the random variable d used in our construction determines the distribution of node degrees. To that end, we used a distribution that resulted in 65% of the nodes having a degree of 2, 30% of the nodes having a degree of 3, and 5% of the nodes having a degree of 4.

Once a tree topology was generated (as described above), the congestion level for each of the links of that tree was selected randomly from one of the three link baseline models described earlier. Specifically, we used a distribution with 50% Low, 30% Mild and 20% High congestion.

B. Performance Metrics

Three metrics are used to evaluate our unicast-based loss topology inference and labeling techniques—namely, *inference accuracy*, *inference discrepancy*, and *labeling error*. The first two metrics are used to evaluate the goodness of a loss topology inference technique, whereas the third is used to evaluate the goodness of a loss topology labeling technique.

In each of the definitions below, we assume that the inference/labeling process starts at time $t = 0$, that $1 \leq k \leq N$ refers to the experiment under consideration, that $0 < i, j \leq M$ refer to clients (or end-points), and that $0 < l \leq L$ refers to a link of a given loss topology (tree).

Definition 4: The inference Accuracy $A(t)$ of a loss topology inference strategy at time t is defined as the probability that the strategy yields the correct loss topology at time t.

To measure accuracy at time t, we calculate the percentage of the simulation experiments in which the correct loss topology was identified at time t. The accuracy metric is an absolute metric, in the sense that it does not allow for a quantification of how "close" an inferred loss topology is to the exact loss topology in the event that it is inaccurate. The discrepancy metric provides such a quantification.

Definition 5: The inference Discrepancy $D(t)$ of a loss topology inference strategy on a tree T at time t is given by:

$$D(t) = \sqrt{\frac{\sum_{i,j:i\neq j}(\hat{d}_{i,j}(t) - d_{i,j})^2}{\binom{M}{2}}}$$

where $d_{i,j}$ denotes the depth of the least common ancestor of a pair of clients i and j in the correct loss topology induced by T and $\hat{d}_{i,j}(t)$ denotes the depth of least common ancestor of a pair of clients i and j in the inferred loss topology at time t.

To give an intuition for the discrepancy metric, consider the topology shown in Figure 1(a) and assume that as a result of applying a topology inference procedure with $c = 0.05$, the topology shown in Figure 3(left) is obtained. Obviously, the inferred topology is not identical to the correct $\mathcal{T}_{0.05}$ loss topology shown in Figure 1(d). The *discrepancy* between the inferred topology and $\mathcal{T}_{0.05}$ can be calculated to be $\frac{\sqrt{3}}{30} = 0.0577$.

Definition 6: The labeling Error $E(t)$ of a loss topology labeling process on tree T at time t is given by:

$$E(t) = \sqrt{\frac{\sum_{l=1}^{L}(\hat{e}_l(t) - e_l)^2}{L}}$$

where e_l denotes the correct loss probability (i.e. label) for link l and $\hat{e}_l(t)$ denotes the measured loss probability for link l at time t.

To give an intuition for the labeling error metric, consider the topology shown in Figure 1(a) and assume that as a result of applying a topology labeling procedure with $c = 0.05$, the labeled topology shown in Figure 3(right) is obtained. Obviously, the labels on that topology are not identical to the labels on the $\mathcal{T}_{0.05}$ loss topology shown in Figure 1(d). The *labeling error* can be calculated to be $\sqrt{0.02^2 + 0.01^2 + 0.01^2}/8 = 0.0033$. This quantity can be viewed as the average deviation in the labeling of an arbitrary link.

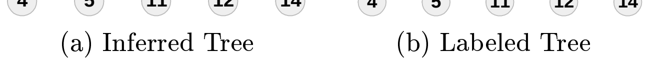


Fig. 3. Illustration of the use of the Inference Discrepancy and Labeling Error metrics.

The metrics presented above are computed by running experiments over a representative set of similar trees, computing the metrics over those inputs, then averaging the results to derive an estimate.

C. Loss Topology Inference Experiments

In order to determine the accuracy of our topology inference technique, we generated 20 5-client baseline trees at random (as described earlier). Our objective is to infer the loss topology between a server and the 5 clients in each one of the baseline trees. We ran the loss topology inference technique from the server (root node) by creating an ns "agent" that sends the probes, collects statistics about these probes, calculates the needed estimates, and executes our topology inference procedure. For each one of the 20 randomly-generated trees, we ran this experiment 20 times; each time with a different random seed. The results were then averaged over all these 400 experiments.

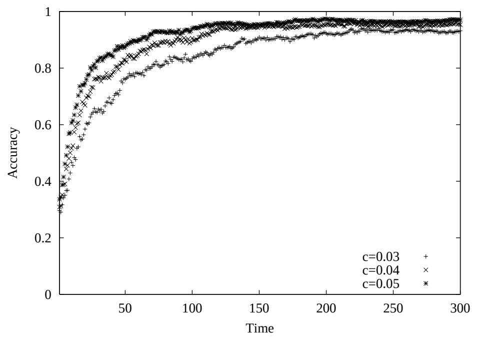


Fig. 4. Accuracy of Loss Topology Inference over Time

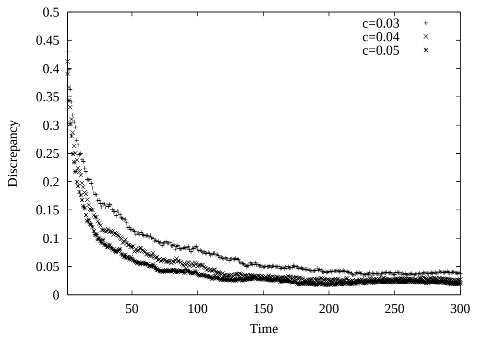


Fig. 5. Discrepancy of Loss Topology Inference over Time

Figures 4 and 5 show the accuracy and discrepancy metrics for our loss topology inference experiments as functions of time for three specific values of the sensitivity constant (c). Clearly, our inference technique converges rapidly. Specifically, both the accuracy and discrepancy metrics settle to within 10% of their steady-state values within 40 seconds (or 200 probing rounds at the rate of 5 probes/sec).

Figures 4 and 5 indicate that both the accuracy and discrepancy metrics improve as the value of the sensitivity constant (c) increases. We quantified this dependence by measuring both accuracy and discrepancy at three different points in time for various values of c. Figures 6 and 7 show that there is indeed an "inflection point", after which both accuracy and discrepancy start deteriorating. In our experiments the best value for c was around 0.10 (i.e. 10%). Notice that this value is related to our choices of what constituted "High", "Mild", and "Low" levels of congestion (see Table I). Specifically, an optimal value of c allows a correct disambiguation between significant losses (due to mild or high congestion) versus insignificant losses (due to low congestion).

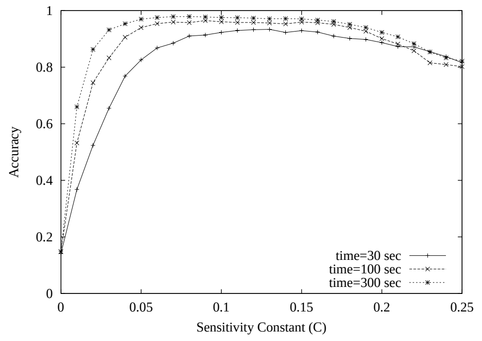


Fig. 6. Effect of sensitivity constant on accuracy

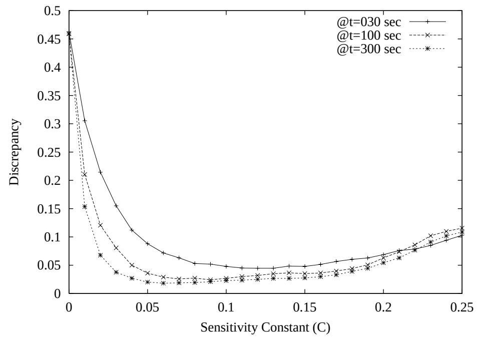


Fig. 7. Effect of sensitivity constant on discrepancy

D. Loss Topology Labeling Experiments

The setup for the loss topology labeling experiments was the same as the loss topology inference experiments; except that the server "agent" has the exact loss topology and runs the labeling algorithm to label this topology. The results are still averaged over 400 experiment runs for 20 randomly-generated trees.

Figure 8 shows the labeling error as a function of time. The labeling error converges to within 1% of the actual links loss rates. We noted that in most of the cases the labeling results are very close to the actual losses, except for the cases where the shared links between 2 clients contain more than one highly congested bottleneck. In this case, probes within a packet-pair are separated after the first congested bottleneck and become uncorrelated when passing through the second bottleneck. This leads to an accurate assessment of the first bottleneck loss rate, but inaccurately assigns most of the second shared bottleneck losses to the independent links.

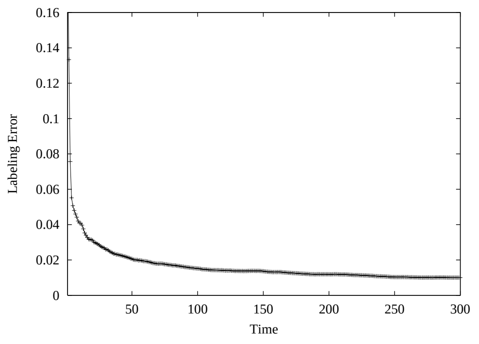


Fig. 8. Error in loss topology labeling over time

E. Loss Topology Augmentation Experiments

The setup for the loss topology augmentation experiments used the same 20 randomly-generated trees of the previous 2 experiments. However, the topology (and associated link loss rates) made known *a priori* to the server is the tree connecting the server to a random set of 4 out of the 5 clients. The loss topology augmentation agent uses probe sequences involving the (missing) fifth client to augment the *a-priori*-known 4-client loss topology to include the fifth client. As before, the results were averaged over the 400 experiments on the 20 randomly-generated trees.

Figures 9 and 10, show the accuracy and discrepancy metrics for loss topology augmentation over time. The results are slightly better than those obtained for the loss topology inference experiment. This is expected since the opportunity for "errors" is constrained by the *a priori* known 4-client topology.[3]

V. Scalability Considerations

A. Complexity of Inference and Labeling

Time Complexity: Let n be the number of clients. We have $\frac{n(n-1)}{2}$ client pairs for which to compute X estimates. To perform the loss topology inference, we must sort these estimates. Thus, the time complexity of the loss topology inference $O(n^2 log(n))$. On the other hand, both the labeling algorithm and the

[3]However, note that the convergence time did not improve since both inference and augmentation experiments use the same mean number of probe rounds (on average 5 probes/sec).

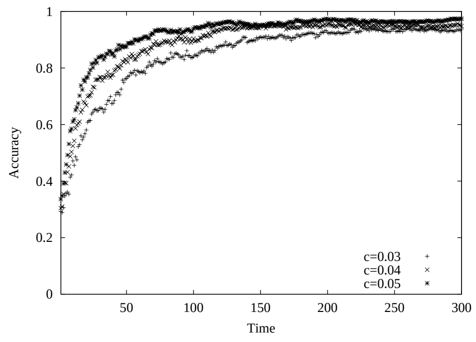


Fig. 9. Accuracy of topology augmentation vs time

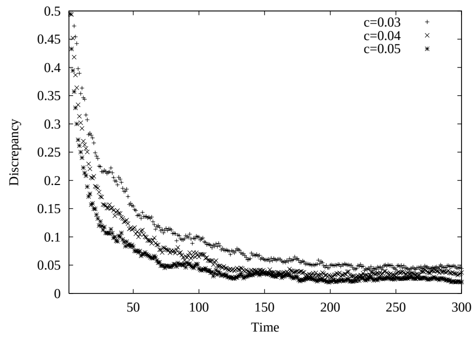


Fig. 10. Discrepancy of topology augmentation vs time

augmentation procedure require only a constant number of operations per internal node in the loss topology. We have at most $n-1$ such nodes. Thus, the time complexity of the loss topology labeling and augmentation procedures is $O(n)$. Later in this section we sketch efficient methods for faster inference of key portions of the loss topology.

Space Complexity: The space needed for the inference and labeling algorithms is basically the space needed to store the routing tree in addition to the counters used to keep the estimates for each tree node. It is not hard to verify that a simple tree representation—keeping for each node its required counters, pointers to its children and its parent—needs $O(n)$ space in the worst case.

B. Simulation of Large-Scale Topologies

The performance evaluation experiments we have presented in the previous section were restricted to relatively small loss topologies (in terms of the total number of endpoints and the total number of internal nodes). This choice was motivated by our desire to keep the simulation times relatively short.

There are a number of issues that may negatively impact the performance of our approach to loss topology inference. For example, a key condition for our BP probing technique to be effective in identifying shared losses is the requirement that the two packets in a 2-packet probe be separated by no more that ϵ units of time, for a small value of ϵ.[4] While a server can ensure that packets used in 2-packet probes satisfy such a constraint, it cannot "guarantee" that such a constraint is satisfied throughtout the network. In particular, the separation between packets in a packet-pair may increase as the number of "hops" traversed increases. Thus, to assess the impact of such conditions, it is necessary to evaluate topologies that are much larger than the topologies considered in the previous section.

To assess the robustness of our approach and its effectiveness for larger loss topologies, we have conducted experiments on a relatively large tree. We used the same tree generation procedure described in Section IV, except that the number of base clients was increased to 50 and the number of dummy clients was increased to 200. The resulting trees had an average depth of over 8 levels and in excess of 400 nodes. As before, we ran 20 inference and labeling experiments on that tree. Figure 11 shows the inference accuracy for this large-scale simulation. While the convergence of loss topology inference for this large tree seems slightly slower than that presented in Figure 4, the accuracy of the inference (at stead-state) remains robust (over 90% after 150 seconds). The results for labeling were equally robust (average labeling error was less than 1.5% after 150 seconds).

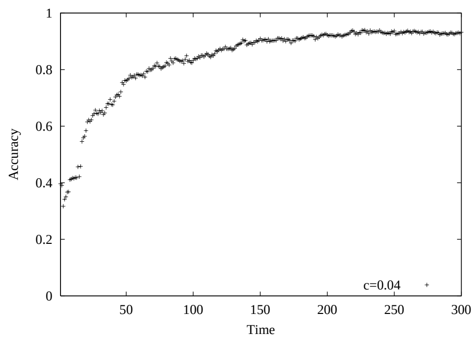


Fig. 11. Inference accuracy for a 250-client tree over time

C. Scalable Management of Probing Processes

As the number of clients scales, the number of all-pairs unicast probes grows quadratically, thus it is often infeasible or unacceptable to perform pairwise probes with respect to all clients already in the tree. However, this is not an inherent limitation of our approach, as we now demonstrate.

In practice, resource constraints may limit the number of probe-pairs we can transmit in a given time interval. In such a constrained environment, it is best

[4]In [12], the value of ϵ was found to be around 1 msec.

	Network Measurement		End-to-End Measurement			
	Active	**Passive**	**Active**		**Passive**	
Multicast	[17]	[11], [14]		[6], [5], [19], [22]		
Unicast	[15], [8]	[11], [14]	[16], [3], [20], [X]	[18], [22], [20], [12]	[20], [12],[X]	[20]
			Sender	**Receiver**	**Sender**	**Receiver**

Table II. A Taxonomy of Network Characterization Efforts

to employ a top-down approach, i.e. by identifying those links in the loss topology closest to the server which are thus likely to be shared by larger numbers of clients. Fortunately, even choosing clients to probe at random gives us considerable leverage in this regard. Assuming our algorithm eventually derives a correct inference from probing a pair of clients, probing a given pair of destinations has the capability of identifying shared loss above their least common ancestor. Since the location of the least common ancestor of a randomly chosen pair of clients is unlikely to be deep in the tree, correlated measurements across pairs enable us to focus intensively on the portion of the loss topology near the source. Empirical validation of mechanisms for robust tree identification under resource constraints and the importance of such an activity is outlined further in the section on future work.

VI. Related Work

Table II presents a taxonomy of network characterization efforts along four dimensions: (1) network vs end-to-end, (2) unicast vs multicast, (3) active vs passive, and (4) sender-oriented vs receiver-oriented. The work we present in this paper is identified by [X]; it is sender-based and is targeted for unicast environments. It works under both passive and active probing assumptions, albeit with different accuracy and convergence properties. Below, we single out specific efforts that are closely related to ours.

Estimation of Network Parameters Using End-to-End Measurements: The specific problem of identifying and characterizing loss topologies is motivated in part by recent work on topological inference over *multicast* sessions [7], [5], [9], [19]. By making purely end-to-end observations of packet loss at endpoints of multicast sessions, Ratnasamy and McCanne [19] and Cáceres *et al.* [5] have demonstrated how to make unbiased, maximum likelihood estimation inferences of (a) the multicast tree topology and (b) the packet loss rates on the edges of the tree, respectively. They demonstrate that an observer with access to a complete record of packet arrivals and losses at each destination can make unbiased inferences about the underlying tree from that record. Their work is made possible by the fact that only one copy of a packet traverses any edge of the multicast tree. Thus, if two receivers share a common edge in the multicast tree, and the packet is dropped in the queue prior to traversing that shared edge, *both* downstream receivers will lose that particular packet. With sufficiently many measurements, this correlated behavior makes the inferences above possible.

Diagnosis of Shared Losses: In [20], Rubenstein, Kurose, and Towsley propose the use of end-to-end probing to detect shared points of congestion (POCs). By their definition, a point of congestion is shared when a set of routers are dropping and/or delaying packets from both flows. Their Markovian Probing (MP) technique for identifying POCs uses a Poisson probe traffic to both remote endpoints and cross-correlation measures computed between pairs of packets from these flows. In [12], we have presented an alternative technique for the identification of shared losses using a Bayesian Probing (BP) approach. The BP approach relies on the use of two types of unicast probes—namely, 1-packet and 2-packet probes. The use of 1-packet probes provides a baseline loss rate over each of the two end-to-end paths between a server and one of its clients. The use of 2-packet probes provides a distinguishing mechanism to measure correlated loss over shared links between a server and two of its clients. The work presented in this paper uses the BP approach to not only identify shared losses, but also to infer and label the loss topology between a server and a set of clients.

VII. Conclusion

Summary: One of the defining principles of the network protocols used in the Internet lies in their ability to manage and share network resources fairly across competing connections. This is a notable engineering achievement, especially in light of the fact that individual connections exert distributed control over their transmission rates. But this fine-grained autonomy that connections exert coupled with our limited understanding of the interactions that multiple (TCP) connections impose limits the degree to which network resources can be tightly controlled. In our ongoing work as part of the Mass project [21], we investigate circumstances in which better diagnosis of network resources can be obtained, which we hope will lead to improved control mechanisms.

In this paper, we have proposed the use of network "loss topologies" as abstractions that enable a compact representation of the shared congested network resources that need to be managed by Mass servers. Three specific problems related to loss topologies were singled out—namely, (1) Inferring a loss topology, (2) Labeling a loss topology, and (3) Augmenting a loss topology. Solutions to these three problems were proposed using a novel end-to-end unicast probing technique and associated analyses. We have demonstrated the viability, robustness, and scalability of our proposed solutions using extensive ns simulations.

Future Work: Our ongoing work focuses on extending the results presented in this paper for deployment within the Mass server project [21]. Specifically, in this paper we have focused on the problem of characterizing the *entire* loss topology for a *fixed* set of end-clients, over a *short time scale*; i.e. for a duration on the order of seconds spanned by the lifetimes of the connections to these clients. In the context of our Mass project, an equally important characterization is the *top portion* [5] of the loss topology from the Mass server to a *variable* set of end clients, over a *long time scale*—namely over a duration far longer than that of a typical connection lifetime, i.e. hours. To that end, the following are extensions we are pursuing:

1. Pruning a Loss Topology: For a continuously changing set of flows to a population of clients, identify the "top portion" of the loss topology connecting the server the client population, subject to constraints on the volume of aggregated traffic through the leaves of the pruned loss topology.

2. Labeling a Pruned Loss Topology: Given a pruned loss topology and a continuously changing set of flows going through leaves of that topology, label the links of the pruned loss topology.

3. Attaching a Client to a Pruned Loss Topology: Given a labeled, pruned loss topology and a new client, identify the leaf of the pruned loss topology to which the client should be attached.

References

[1] H. Balakrishnan, H. Rahul, and S. Seshan. An Integrated Congestion Management Architecture For Internet Hosts. In *SIGCOMM '99*, Cambridge, MA, September 1999.

[2] Azer Bestavros and Olivier Hartmann. Aggregating Congestion Information Over Sequences of TCP Connections. Technical Report TR-98-001, Boston Univ, CS Dept, January 1998.

[3] J. C. Bolot. End-to-end Packet Delay and Loss Behavior in the Internet. In *SIGCOMM '93*, pages 289–298, September 1993.

[4] John W. Byers, Michael Luby, and Michael Mitzenmacher. Accessing multiple mirror sites in parallel: Using tornado codes to speed up downloads. In *Proceedings of IEEE INFOCOM '99*, pages 275–83, March 1999.

[5] R. Cáceres, N. G. Duffield, J. Horowitz, D. Towsley, and T. Bu. Multicast Based Inference of Network-Internal Characteristics: Accuracy of Packet-Loss Estimation. In *INFOCOM '99*, March 1999.

[6] R. Cáceres, N. G. Duffield, S. B. Moon, and D. Towsley. Inference of Internal Loss Rates in the MBone. In *IEEE Global Internet (Globecom)*, Rio de Janeiro, Brazil, 1999.

[7] R. Cáceres, N.G. Duffield, S.B. Moon, and D. Towsley. Inferring Link-level Performance from End-to-End Multicast Measurements. Internal report. Available at ftp://gaia.cs.umass.edu/pub/Caceres99gi99.ps.Z.

[8] Robert L. Carter and Mark E. Crovella. Measuring bottleneck link speed in packet switched networks. In *PERFORMANCE '96, the International Conference on Performance Theory, Measurement and Evaluation of Computer and Communication Systems*, October 1996.

[9] N. Duffield, V. Paxson, and D. Towsley. MINC: Multicast-based Inference of Network-Internal Characteristics. http://www-net.cs.umass.edu/minc/.

[10] Michalis Faloutsos, Petros Faloutsos, and Christos Faloutsos. On power-law relationships of the internet topology. In *Proceedings of ACM SIGCOMM '99*, September 1999.

[11] Felix: Independent Monitoring for Network Survivability. http://ftp.bellcore.com/pub/mwg/felix/index.html.

[12] K. Harfoush, A. Bestavros, and J. Byers. Robust Identification of Shared Losses Using End-to-End Unicast Probes. Technical Report TR-2000-013, Boston Univ, CS Dept, May 2000.

[13] K. Harfoush, A. Bestavros, and J. Byers. Unicast-based Characterization of Network Loss Topologies. Technical Report TR-2000-016, Boston Univ, CS Dept, July 2000.

[14] IPMA : Internet Performance Measurement and Analysis. http://www.merit.edu/ipma.

[15] V. Jacobson. `Pathchar`: A Tool to Infer Characteristics of Internet Paths. ftp://ftp.ee.lbl.gov/pathchar.

[16] S. Keshav. *Congestion Control in Computer Networks.* PhD thesis, University of California at Berkeley, September 1991.

[17] Mtrace: Tracing multicast path between a source and a receiver. ftp://ftp.parc.xerox.com/pub/netsearch/ipmulti.

[18] V. Paxson. *Measurements and Analysis of End-to-end Internet Dynamics.* PhD thesis, U.C. Berkeley and Lawrence Berkeley Laboratory, 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 '99*, pages 353–60, March 1999.

[20] D. Rubenstein, J. Kurose, and D. Towsley. Detecting Shared Congestion of Flows Via End-to-end Measurement. In *ACM SIGMETRICS '00*, Santa Clara, Ca, June 2000.

[21] The BU MASS Project. Diagnosis and Control of Network Variability by MASS Servers. `http://www.cs.bu.edu/groups/mass`, 2000.

[22] Maya Yajnik, Sue Moon, Jim Kurose, and Don Towsley. Measurement and modelling of the temporal dependence in packet loss. In *Proceedings of IEEE INFOCOM '99*, pages 345–52, March 1999.

[5] By "top portion" we mean the portion of the loss topology that is most proximate to the server, and to which the server contributes traffic continuously over a long period of time.