Inference of Internal Loss Rates in the MBone[*]

Ramón Cáceres, N.G. Duffield
AT&T Labs – Research
180 Park Avenue
Florham Park, NJ 07932, USA
{ramon,duffield}@research.att.com

Sue B. Moon, Don Towsley
Department of Computer Science
University of Massachusetts
Amherst, MA 01003, USA
{sbmoon,towsley}@cs.umass.edu

June 25, 1999

Abstract

We present MBone experiments that validate an end-to-end measurement technique we call MINC, for Multicast Inference of Network Characteristics. MINC exploits the performance correlation experienced by multicast receivers to infer loss rates and other attributes of internal links in a multicast tree. MINC has two important advantages in the Internet context: it does not rely on network collaboration and it scales to very large measurements. In previous work, we laid the foundation for MINC using rigorous statistical analysis and packet-level simulation. Here, we further validate MINC by comparing the loss rates on internal MBone tunnels as inferred using our technique and as measured using the `mtrace` tool. Inferred values closely matched directly measured values – differences were usually well below 1%, never above 3%, while loss rates varied between 0 and 35%.

1 Introduction

As the Internet grows in size and diversity, its internal performance becomes harder to measure. Any one organization has administrative access to only a small fraction of the network's internal nodes, while commercial factors often prevent organizations from sharing internal performance data. End-to-end measurements using unicast traffic do not rely on administrative privileges, but it is difficult to infer link-level performance from them and they require large amounts of traffic to cover multiple paths. There is a need for practical and efficient procedures that can take an internal snapshot of a significant portion of the network.

We have developed a measurement technique that addresses these problems. *Multicast Inference of Network Characteristics* (MINC) [11] uses end-to-end multicast traffic as measurement probes. It exploits the inherent correlation in performance observed by multicast receivers to infer the loss rate and other attributes of paths between branch points in a multicast routing tree. These measurements do not rely on administrative access to internal nodes since they are done between end hosts. In addition, they scale to large networks because of the bandwidth efficiency of multicast traffic.

The intuition behind packet loss inference is that the event that a packet has reached a given internal node in the tree can be inferred from the packet's arrival at one or more receivers descended from that node. Conditioning on this event, we can determine the probability of successful transmission to and beyond the given node. Consider, for example, a simple multicast tree with a root node (the source), two leaf nodes (the left and right receivers), a link from the source to a branch point (the shared link), and a link from the branch point to each of the receivers (the left and right links). The source sends a stream of sequenced multicast packet through the tree to the two

[*]This work was sponsored in part by DARPA and the Air Force Research Laboratory under agreement F30602-98-2-0238.

receivers. If a packet reaches either receiver, we can infer that the packet reached the branch point. Thus the ratio of the number of packets that reach both receivers to the number that reached only the right receiver gives an estimate of the probability of successful transmission on the left link. The probability of successful transmission on the other links can be found by similar reasoning.

It is not immediately clear whether this technique applies to more than just binary trees or whether it enjoys desirable statistical properties. In previous work [2], we extended this technique to general trees and showed that the estimate is consistent, that is, it converges to the true loss rates as the number of probes grows. More specifically, we developed a *Maximum Likelihood Estimator* (MLE) for internal loss rates in a general tree assuming independent losses across links and across probes. We derived the MLE's rate of convergence and established its robustness with respect to certain violations of the independence assumption. We also validated these analytical results using the `ns` simulator [15]. We give a brief account of these results in Section 2.2.

In more recent work [3], we explored the accuracy of our packet loss estimation under a variety of network conditions. Again using `ns` simulations, we evaluated the error between inferred and actual loss rates as we varied the network topology, propagation delay, packet drop policy, background traffic mix, and probe traffic type. We found that, in all cases, MINC accurately inferred the per-link loss rates of multicast probe traffic.

In this paper, we further validate MINC through experiments under real network conditions. We used a collection of end hosts connected to the MBone, the multicast-capable subset of the Internet [10]. We chose one host as the source of multicast probes and used the rest as receivers. We then made two types of measurements simultaneously: end-to-end loss measurements between the source and each receiver, and direct loss measurements at every internal node of the multicast tree. Finally, we ran our inference algorithm on the results of the end-to-end measurements, and compared the inferred loss rates to the directly measured loss rates. Across all our experiments, the inferred values closely matched the directly measured values. The differences between the two were usually well below 1%, never above 3%, while loss rates varied between 0 and 35%. Furthermore, the inference algorithm converged well within 2-minute, 1200-probe measurement intervals.

The rest of this paper is organized as follows: Section 2 describes our experimental methodology; Section 3 presents our experimental results; Section 4 discusses our ongoing work; Section 5 surveys related work; and Section 6 offers some conclusions.

2 Experimental Methodology

During each of our MBone experiments, we had a source send a stream of sequenced packets to a collection of receivers while we made two types of measurement at each receiver. At the source, we used our `mgen` traffic generation tool to send one 40-byte packet every 100 milliseconds to a specific multicast group. The resulting traffic stream placed less than 4 Kbps of load on any one MBone link. We reserved multicast address 224.2.130.64 and port 22778 for our experiments using the `sdr` session directory tool [21]. At each receiver, we ran the `mtrace` [13] and `mbat` [9] tools to gather statistics about traffic on this multicast group. Below we describe our use of `mtrace` and `mbat` in more detail.

2.1 Direct measurements

`mtrace` traces the *reverse* path from a multicast source to a receiver. It runs at the receiver and issues trace queries that travel hop-by-hop up the multicast tree towards the source. Each router along the path responds to these queries with information about traffic on the specified multicast group as seen by that router, including counts of incoming and outgoing packets. `mtrace` calculates packet losses on a link by comparing the packet counts returned by the two routers at either end of the link.

In each of our experiments, we collected `mtrace` statistics for consecutive two-minute intervals over the course of one hour. We ran a separate instance of `mtrace` for each interval. Each `mtrace` run issued a trace query at the beginning of the interval and another query at the end. We thus measured link-level loss rates for all thirty intervals in one hour as shown in Figures 2 – 4. These intervals are not exactly two

Physical location	**Abbreviation**
AT&T Labs – Research, Florham Park, New Jersey	AT&T
Carnegie Mellon University, Pittsburgh, Pennsylvania	CMU
Georgia Institute of Technology, Atlanta, Georgia	GaTech
University of California, Berkeley, California	UCB
University of Kentucky, Lexington, Kentucky	UKy
University of Massachusetts, Amherst, Massachusetts	UMass
University of Southern California, Los Angeles, California	USC
University of Washington, Seattle, Washington	UWash

Table 1: End hosts used during our MBone experiments.

Physical location	**Abbreviation**
Atlanta, Georgia	GA
Cambridge, Massachusetts	MA
San Francisco, California	CA
West Orange, New Jersey	NJ

Table 2: Routers at multicast branch points during our representative MBone experiment.

minutes long due to delays incurred in collecting responses to the queries. We recorded timestamps for the actual beginning and end of each `mtrace` run to help synchronize our inference calculations to these direct measurements.

We chose to measure two-minute intervals based on our previous experience with MINC. Our simulations have shown that the statistical inference algorithm at the heart of MINC converges to true loss rates after roughly 1,000 observations [2]. Given the 100 milliseconds between probes in our MBone experiments, two minutes allow for 1,200 probes between measurements. As shown in Figure 5, 1,200 probes were indeed enough for MINC to converge.

It is important to note that `mtrace` does not scale to measurements of large multicast groups if used in parallel from all receivers as we describe here. Parallel `mtrace` queries come together as they travel up the tree. Enough such queries will overload routers and links with measurement traffic. We used `mtrace` in this way only to validate MINC on relatively small multicast groups before we move on to use MINC alone on larger groups.

2.2 Statistical inference

MINC works on *logical* multicast trees. A logical tree is one where all nodes, except the root and the leaves, have at least two children. A physical tree can be converted into a logical tree by deleting all nodes, other than the root, that have only one child and then collapsing the links accordingly. A link in a logical tree may thus represent multiple physical links. This conversion is necessary because inference based on correlation among receivers cannot distinguish between two physical links unless these links lead to two different receivers. Henceforth when we speak of trees we will be speaking of logical trees.

2.2.1 Inference algorithm

Our model for loss on a multicast tree assumes that packet loss is independent across different links of the tree, and independent between different probes. With these assumptions, the loss model is specified by associating a probability α_k with each node k in the tree. α_k is the probability that a packet is transmitted successfully across the link terminating at node k, given that it reaches the parent node $p(k)$ of k.

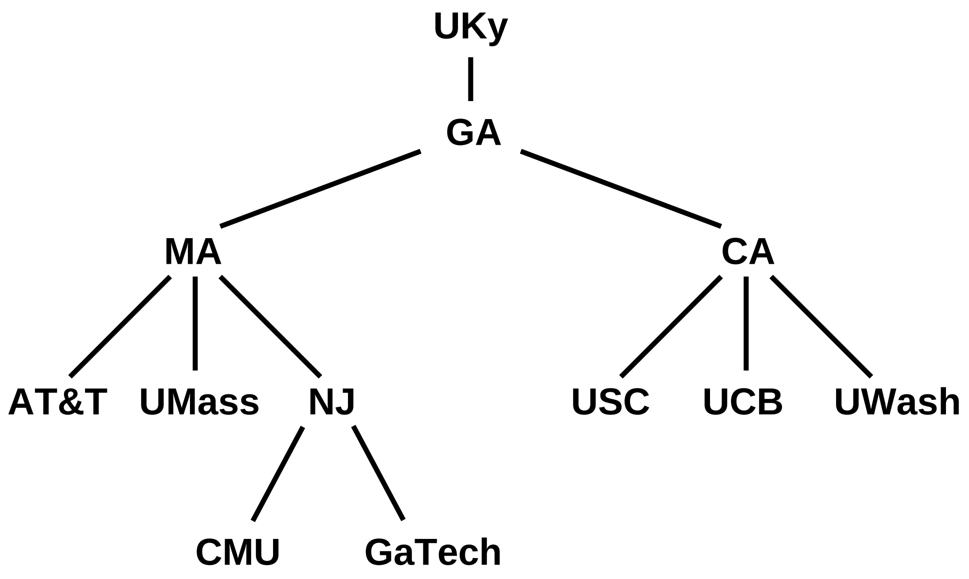


Figure 1: Multicast routing tree during our representative MBone experiment.

When a probe is transmitted from the source, we can record the outcome as the set of receivers the probe reached. The loss inference algorithm is based on probabilistic analysis that allows us to express the α_k directly in terms of the *expected* frequencies of such outcomes. More precisely, for each node k let γ_k denote the probability of the outcome that a given packet reaches at least one receiver that has k as an ancestor in the tree. Let A_k denote the probability that a given packet reaches the node k, i.e., $A_k = \alpha_k \alpha_{k_1} \alpha_{k_2} \dots \alpha_{k_m}$ where $k_1, k_2, \dots, k_m$ is the chain of m adjacent nodes leading back from node k to the root of the tree. Then it can be shown that A_k satisfies

$$(1 - \gamma_k / A_k) = \prod_{j \in c(k)} (1 - \gamma_j / A_k) \tag{1}$$

where the product is taken over all nodes j in $c(k)$, the set of children of the node k. It was shown in [2] that under generic conditions the A_k can be recovered uniquely through (1) if the γ are known. The α_k can in turn be recovered since $\alpha_k = A_k / A_{p(k)}$. Generally, finding A_k requires numerical root-finding for (1). In the special case of a node k with two offspring j and j', (1) can be solved explicitly:

$$A_k = \frac{\gamma_j \gamma_{j'}}{\gamma_j + \gamma_{j'} - \gamma_k} \tag{2}$$

Suppose that in place of the γ_k in (1), we use the *actual* frequencies $\widehat{\gamma}_k$ with which n probes reach at least one receiver with ancestor k. We denote the corresponding solutions to (1) by $\widehat{A}_k$ and estimate the link probabilities by $\widehat{\alpha}_k = \widehat{A}_k / \widehat{A}_{p(k)}$. The calculation of the $\widehat{\gamma}_k$ is achieved though a simple recursion as follows. Define new variables $Y_k(i)$ as function of the measured outcomes of n probes by

$$Y_k(i) = \begin{cases} 1 & \text{if probe } i \text{ reaches node } k \\ 0 & \text{otherwise} \end{cases} \tag{3}$$

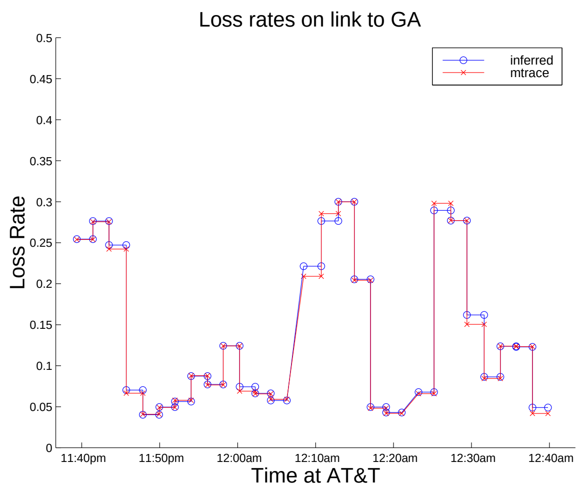
if k is a leaf node, and

$$Y_k(i) = \max_{j \in c(k)} Y_j(i) \tag{4}$$

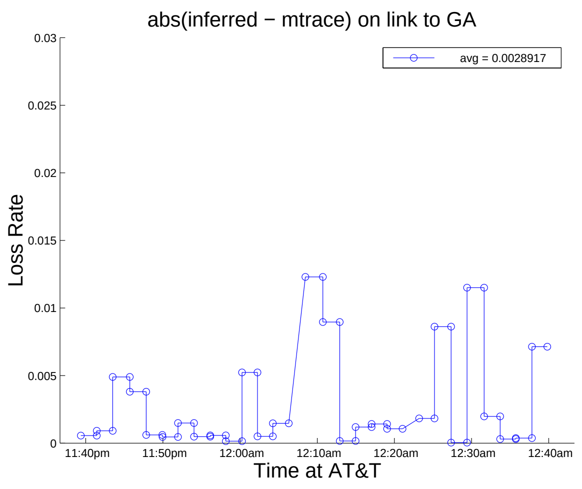
otherwise. Then

$$\widehat{\gamma}_k = \frac{1}{n} \sum_{i=1}^{n} Y_k(i) \tag{5}$$

We showed in [2] that the estimator $\widehat{\alpha}_k$ enjoys two useful properties: (i) *consistency*: $\widehat{\alpha}_k$ converges to the true value α_k almost surely as the number of probes n grows to infinity, and (ii) *asymptotic normality*: the distribution of the normalized difference $\sqrt{n}(\widehat{\alpha}_k - \alpha_k)$ converges to a normal distribution as n grows to infinity. We also investigated in [2] the effects of correlations that violate the independent



(a) Inferred vs. directly measured loss rates



(b) Difference between the two

Figure 2: Loss rates on link to GA when running `mtrace` from AT&T. The two sets of loss rates agreed closely over a wide range of values. Differences remained below 1.5% while loss rates varied between 4 and 30%.

loss assumptions. Consistency is preserved under a large class of temporal correlations, although convergence of the estimates with n can be slower. Spatial correlations perturb the estimate continuously, in that small correlations lead to small inconsistencies. When losses on sibling links are correlated the perturbation is a second-order effect, in that the degree of inconsistency depends not on the size of the correlations, but on the degree to which they change across the tree.

Our earlier papers on MINC [2, 3] contain a detailed description and analysis of the above inference algorithm, including rules to handle special cases of the data in which the generic conditions required for the existence of solutions to (1) fail. In the interests of brevity, we omit these details from this paper.

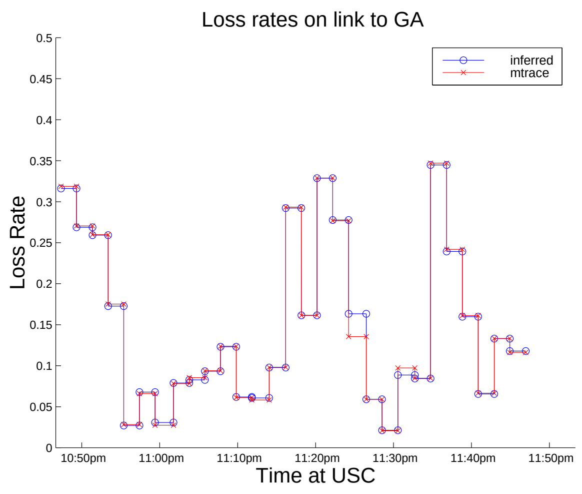
2.2.2 Inference calculations

We encoded our loss inference algorithm in a program called `infer`. `infer` takes two inputs: a description of the tree topology and a description of the end-to-end losses experienced by each receiver. It produces as output the estimated loss rates on every link in the tree.

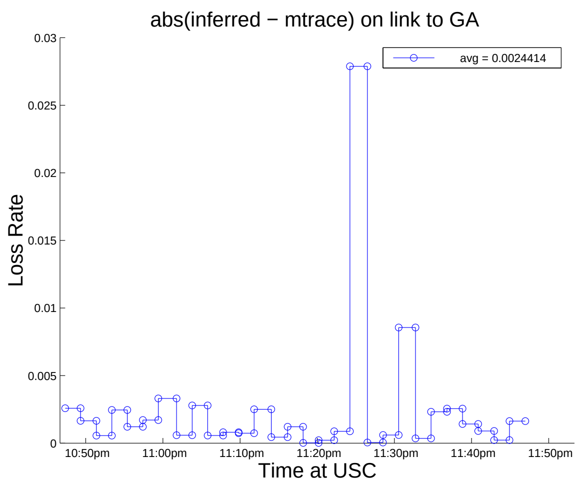
We determined the tree topology by combining the `mtrace` output from all the receivers. Along with packet counts, `mtrace` reports the domain name and IP address of each router on the path from the source to a receiver. We built a complete multicast tree by looking for common routers and branch points on the paths to all the receivers. The topology of the MBone is relatively static due to that network's current reliance on manually configured IP-over-IP tunnels. These tunnels are themselves logical links that may each contain multiple physical links. We verified that the topology remained constant during our experiments by inspecting the path information we obtained every two minutes from `mtrace`.

We measured end-to-end losses using the `mbat` tool. `mbat` runs at a receiver, subscribes to a specified multicast group, and collects a trace of the incoming packet stream, including the sequence number and arrival time of each packet. We ran `mbat` at each receiver for the duration of each experiment. At the conclusion of an experiment, we transferred the `mbat` traces and `mtrace` output from all the receivers to a single location.

There we ran the loss inference algorithm on the same two-minute intervals on which we collected `mtrace` measurements. For each receiver, we used the timestamps for the beginning and end of `mtrace` measurements to segment the `mbat` traces into corresponding two-minute subtraces. Then we ran `infer` on each two-minute interval and compared



(a) Inferred vs. directly measured loss rates



(b) Difference between the two

Figure 3: Loss rates on link to GA when running `mtrace` from USC. These measurements span different two-minute intervals than those from AT&T (see Fig. 2) because of clock asynchrony. Nevertheless, inferred and directly measured loss rates agreed closely. Differences were usually below 0.5%, never above 3%, while loss rates varied between 2 and 35%.

the inferred loss rates with the directly measured loss rates. We discuss the results in the next section.

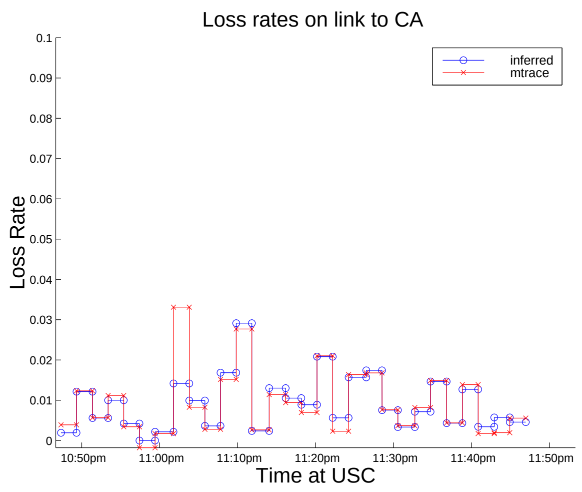
3 Experimental Results

We performed a number of MBone experiments using different multicast sources and receivers, and thus different multicast trees. Inferred loss rates agreed closely with directly measured loss rates throughout our experiments. Here we discuss results from a representative experiment on August 26, 1998. Tables 1 and 2 list the end hosts and branch routers involved in this experiment, while Figure 1 shows the resulting multicast tree.

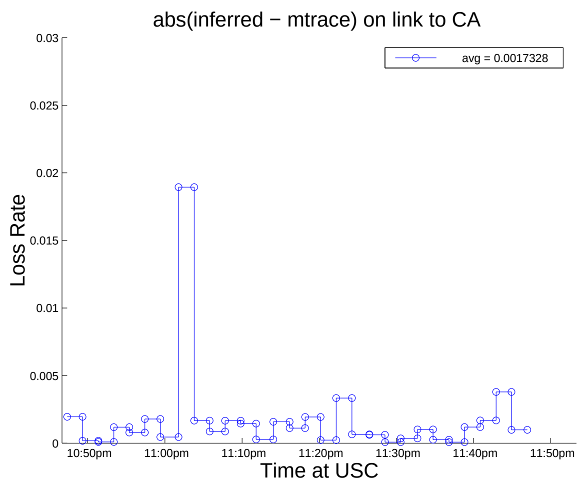
Figure 2 shows that inferred and directly measured loss rates agreed closely despite a link experiencing a wide range of loss rates. In this case, loss rates as measured by `mtrace` varied between 4 and 30%. Nevertheless, differences between inferred and directly measured loss rates remained below 1.5%.

Figures 2 – 4 all show that inferred and directly measured loss rates agreed closely despite imperfect synchronization between `infer` and `mtrace` intervals. The two sets of intervals do not always match because of variable network delays. The timestamps for the beginning and end of `mtrace` intervals are recorded before a trace query is issued and after a trace query returns, both according to the clock at the relevant receiver. However, the corresponding packet counts are recorded at the time the trace query arrives at each router. Therefore, although the `infer` intervals are derived from the `mtrace` intervals using the same receiver clock, the inference is not always applied to exactly the same 1,200 probe packets as the direct loss measurement. Nevertheless, differences between inferred and directly measured loss rates across Figures 2 – 4 were usually well below 1%, never above 3%.

Along the same lines, Figures 2 and 3 together show that inferred and directly measured loss rates agreed closely for different two-minute intervals on the same link. We have multiple sets of `mtrace` measurements for links shared by multiple receivers, one set for each receiver. In these cases, we can run `infer` on different sets of intervals corresponding to the different sets of `mtrace` intervals. `mtrace` intervals are different for each receiver because of clock asynchrony between receivers and because of the variable network delays discussed above. Nevertheless, differences between inferred and directly measured loss rates across Figures 2 and 3 remained below 3%.



(a) Inferred vs. directly measured loss rates



(b) Difference between the two

Figure 4: Loss rates on link to CA when running `mtrace` from USC. CA experienced an order of magnitude lower loss rates than GA (see Figs. 2 and 3). Nevertheless, inferred and directly measured loss rates agreed closely. Differences were usually below 0.5%, never above 2%, while loss rates varied between 0 and 4%.

Figure 4 shows that inferred and directly measured loss rates agreed closely even for links with very low loss rates. In this case, loss rates varied between 0 and 4%, an order of magnitude lower than the loss rates in Figure 2. Nevertheless, differences between inferred and directly measured loss rates were usually below 0.5%, never above 2%.

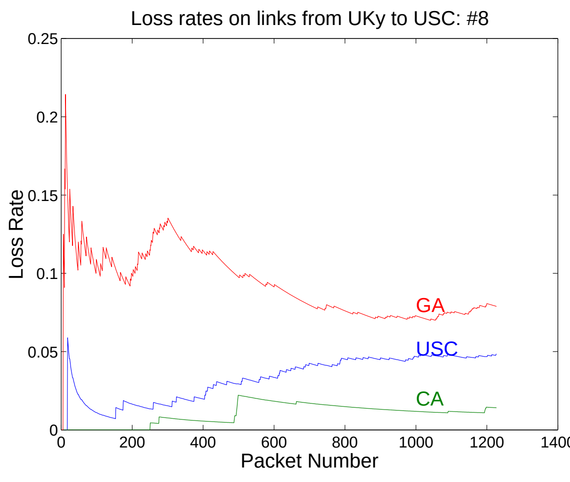
Finally, Figure 5 shows that the inference algorithm converged quickly to the desired loss rates. Each inferred loss rate reported in Figures 2 – 4 is the value calculated by `infer` at the end of the corresponding 2-minute, 1200-probe measurement interval. However, `infer` outputs a loss rate value for every probe. Figure 5 reports these intermediate values. As shown, inferred loss rates stabilized well before a measurement intervals ends. Our algorithm converged after fewer than 800 probes for all links and all measurement intervals in our experiments.

4 Ongoing Work

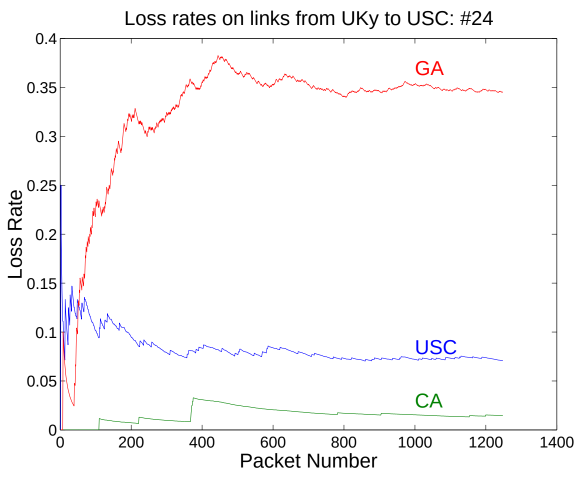
The results reported in the previous section suggest that it is possible to characterize link-level loss based on end-to-end multicast measurements. However, a number of issues need to be resolved before the technology can be deployed and made available for general use.

First, there is the question of how end-to-end multicast measurements are to be generated and collected. We are pursuing two approaches for addressing this question. The first is to add a multicast probe capability to an existing measurement infrastructure. We are working with the National Internet Measurement Infrastructure (NIMI) [14] project to do exactly this. Currently NIMI permits users to schedule a variety of unicast end-to-end measurements between NIMI platforms and to download the traces to a site of their choosing. We are augmenting this capability to permit the scheduled execution of multicast end-to-end measurements followed by a distribution of the traces. Once we have accomplished this, we will also be able to design and execute a more extensive set of experiments to validate the inference techniques against `mtrace`.

There is one disadvantage with the above approach, namely that the set of links that can be covered is limited by the number and placement of NIMI nodes. We are investigating a second approach that has the potential of addressing that problem. The basic idea is to gather end-to-end loss information for multicast applications, such as teleconferencing and continuous media streaming, that already exist in the network. This is possible when the application uses the Real-Time Transport Protocol (RTP) and its as-



(a) Eighth measurement interval



(b) Twenty-fourth measurement interval

Figure 5: Inferred loss rates on the three links between UKy and USC (i.e., the links to GA, CA, and USC) during individual 2-minute, 1200-probe measurement intervals. The inference algorithm converged well before the measurement interval ended for all links during all measurement intervals.

sociated control protocol, RTCP [20]. Currently, applications using these protocols require receivers to multicast loss and delay information to each other. Currently the loss information is limited, consisting of short-term and long-term loss rates, and is not adequate for our inference techniques.

We plan to evaluate different ways of augmenting these RTCP loss reports to provide sufficient information to infer link-level loss behavior. This would allow a measurement node anywhere in the network to monitor and record the RTCP loss reports for various applications. This measurement node could identify the topology of an application using the third-party measurement feature of `mtrace` [13], then apply the inference methodology to obtain the link-level behavior. The hope is that the number of participants in these applications will be sufficiently large to allow a measurement node to estimate the loss behavior of a large portion of the links in the network.

A second important deployment issue concerns the need to know the topology of the multicast tree in order to apply our techniques. Our current experiments use the multicast tree topology discovered from executing `mtrace`. Recent work has shown that algorithms based on link-level loss estimators for binary trees can be used to infer the topology of multicast trees. Topology inference of binary and general trees was proposed in [19] and [4], respectively. Although not reported here, the latter algorithms are able to infer the trees described in Section 3 with reasonable accuracy. Any general purpose inference infrastructure, whether built on top of NIMI or RTCP, should have the flexibility to use both `mtrace` and the topology inference algorithms reported in [19, 4].

In addition to addressing the above deployment issues, we are also exploring new application areas for MINC. One, we are investigating extensions to our inference methodology to estimate link-level *delay* behavior. We have developed prototype estimators for the delay distribution and delay variance on internal links of a multicast tree based on end-to-end delay measurements. We will describe these results in future papers. Two, we believe our inference techniques will prove useful in reliable multicast applications. These applications need to aggregate receivers to achieve scalable loss recovery. MINC could be used to group receivers that are topologically close and share loss performance.

5 Related Work

A growing number of measurement infrastructure projects (e.g., AMP [1], Felix [6], IPMA [7], NIMI [14], Surveyor [22], and Test Traffic [23]) aim to collect and analyze end-to-end performance data for a mesh of unicast paths between a set of participating hosts. We believe our multicast-based inference techniques would be a valuable addition to these measurement platforms. As mentioned in the previous section, we are working to incorporate MINC capabilities into NIMI.

A lot of recent experimental work has sought to understand internal network behavior from end-to-end performance measurements (e.g., see [5, 12, 17, 18]). In particular, `pathchar` [16] is under evaluation as a tool for inferring link-level statistics from end-to-end unicast measurements. Much work remains to be done in this area and with MINC we are contributing a novel multicast-based methodology.

Regarding multicast-based measurements, we have already described the `mtrace` tool [13]. In addition, the `tracer` tool [8] performs topology discovery through the use of `mtrace`. However, `mtrace` suffers from performance and applicability problems in the context of large-scale Internet measurements. First, as mentioned earlier in this paper, `mtrace` needs to run once for each receiver in order to cover a complete multicast tree. This behavior does not scale well to large numbers of receivers. In contrast, MINC covers the complete tree in a single pass. Second, `mtrace` relies on multicast routers to respond to explicit measurement queries. Although current routers support these queries, Internet Service Providers (ISPs) may choose to disable this feature since it gives anyone access to detailed delay and loss information about paths inside their networks. In contrast, MINC does not rely on cooperation from any network-internal elements.

6 Conclusions

We have presented experimental results that validate the MINC approach to inferring link-level loss rates from end-to-end multicast measurements. We compared loss rates in MBone tunnels as inferred using our technique and as measured by `mtrace`. Inferred values closely matched directly measured values – differences were usually well below 1%, never above 3%, while loss rates varied between 0 and 35%. In addition, our inference algorithm quickly converged to the true loss rates – inferred values stabilized well within 2-minute, 1200-probe measurement intervals.

We feel that MINC is an important new methodology for network measurement, particularly Internet measurement. It does not rely on network cooperation and it scales to very large networks. MINC is firmly grounded in statistical analysis that is backed up by packet-level simulations and now experiments under real network conditions. We are continuing to extend MINC along both analytical and experimental fronts.

Acknowledgements

Tian Bu wrote the `infer` program. Mark Handley suggested using RTCP receiver reports to carry MINC loss traces.

References

[1] AMP: Active Measurement Project. `http://amp.nlanr.net`

[2] R. Cáceres, N.G. Duffield, J. Horowitz, D. Towsley, "Multicast-Based Inference of Network-Internal Loss Characteristics," Comp. Sci. Tech. Rep. 98-17, University of Massachusetts at Amherst, February 1998. `ftp://gaia.cs.umass.edu/pub/CDHT98:MINC.ps.Z`

[3] R. Cáceres, N.G. Duffield, J. Horowitz, D. Towsley, T. Bu, "Multicast-Based Inference of Network-Internal Characteristics: Accuracy of Packet Loss Estimation," *Proc. IEEE Infocom '99*, March 1999.

[4] R. Cáceres, N.G. Duffield, J. Horowitz, F. Lo Presti, D. Towsley, "Loss-Based Inference of Multicast Network Topology," submitted for publication, February 1999.

[5] R. L. Carter, M. E. Crovella, "Measuring Bottleneck Link Speed in Packet-Switched Networks," *Proc. PERFORMANCE '96*, October 1996.

[6] Felix: Independent Monitoring for Network Survivability project. `ftp://ftp.bellcore.com/pub/mwg/felix/index.html`

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

[8] B.N. Levine, S. Paul, J.J. Garcia-Luna-Aceves, "Organizing Multicast Receivers Deterministically According to Packet-Loss Correlation," *Proc. ACM Multimedia 98*, September 1998.

[9] `mbat`: MBone Analysis Tool. `ftp://gaia.cs.umass.edu/pub/mist`

[10] M. R. Macedonia, D. P. Brutzman, "MBone Provides Audio and Video across the Internet," *IEEE Computer*, April 1994.

[11] MINC: Multicast Inference of Network Characteristics project. `http://gaia.cs.umass.edu/minc`

[12] M. Mathis, J. Mahdavi, "Diagnosing Internet Congestion with a Transport Layer Performance Tool," *Proc. INET '96*, June 1996.

[13] `mtrace`: Print multicast path from a source to a receiver. `ftp://ftp.parc.xerox.com/pub/net-research/ipmulti`

[14] V. Paxson, J. Mahdavi, A. Adams, M. Mathis, "An Architecture for Large-Scale Internet Measurement," *IEEE Communications*, Vol. 36, No. 8, pp. 48-54, August 1998.

[15] `ns`: Network simulator. `http://www-mash.cs.berkeley.edu/ns`

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

[17] V. Paxson, "End-to-End Routing Behavior in the Internet," *Proc. ACM SIGCOMM '96*, August 1996.

[18] V. Paxson, "End-to-End Internet Packet Dynamics," *Proc. ACM SIGCOMM 1997*, September 1997.

[19] S. Ratnasamy, S. McCanne, "Inference of Multicast Routing Tree Topologies and Bottleneck Bandwidths Using End-to-End Measurements," *Proc. IEEE Infocom '99*, March 1999.

[20] H. Schulzrinne, S. Casner, R. Frederick, V. Jacobson, "RTP: A Transport Protocol for Real-Time Applications," RFC 1889, January 1996.

[21] `sdr`: Session Directory tool. `http://www-mice.cs.ucl.ac.uk/multimedia/software/sdr`

[22] Surveyor project. `http://io.advanced.org/surveyor`

[23] Test Traffic project. `http://www.ripe.net/test-traffic`