GRE Encapsulated Multicast Probing: A Scalable Technique for Measuring One-Way Loss

Yu Gu[‡], Lee Breslau[*], Nick Duffield[*], Subhabrata Sen[*]
[*]AT&T Labs – Research,Florham Park, NJ 07932
[‡]Dept. of Comp. Science,University of Massachusetts, Amherst
{breslau,duffield,sen}@research.att.com, yugu@cs.umass.edu

ABSTRACT

Internet service providers increasingly wish to monitor the performance of customer traffic within their networks. This paper addresses the problem of scalably performing one-way loss measurements across specific network paths. Our solution addresses the issue of scale by exploiting measurement features of the deployed network infrastructure to a large degree. There are three components. Firstly, GRE tunneling is used to steer measurement traffic from a small number of measurement hosts down the paths followed by customer traffic. Secondly, innovative probing methods, coupled with standard measurement capabilities, such as NetFlow, are used to isolate the performance of groups of measurement packets. Thirdly, we exploit and extend tomographic inference methods in order to extract the performance of probe traffic on customer paths within the network. This combination yields a powerful yet lightweight method to determine customer performance within the network. We motivate and illustrate the method with a timely example: performance measurement in multicast Virtual Private Networks.

1. INTRODUCTION

Provider-based VPNs provide a scalable and secure way for a service provider to support many enterprise customers across its backbone. In such a VPN, each customer site connects to one or more edge routers in the provider network. Customer traffic is encapsulated across the provider network, decapsulated by a remote provider edge router and handed off to the customer router (see Figure 1). The provider backbone can offer either a point-to-point unicast delivery [19] (for unicast VPN applications) or a point-to-multipoint [18] delivery (for multicast applications).

These VPNs can be both large and complex, comprised of large numbers of routers in the provider backbone, hundreds of customer locations, and use complex, new and not-well-studied protocols and features such as MVPN [18] and MPLS. This environment presents significant management challenges for network providers. In order to support these VPNs, providers require management tools to provide a wide range of functionality, including performance anomaly detection, troubleshooting, and Service Level Agreement (SLA) monitoring. These functions depend on accurate, scalable and reliable monitoring of the service provider's network.

One particularly sought-after performance capability is that of measuring or estimating the one-way performance

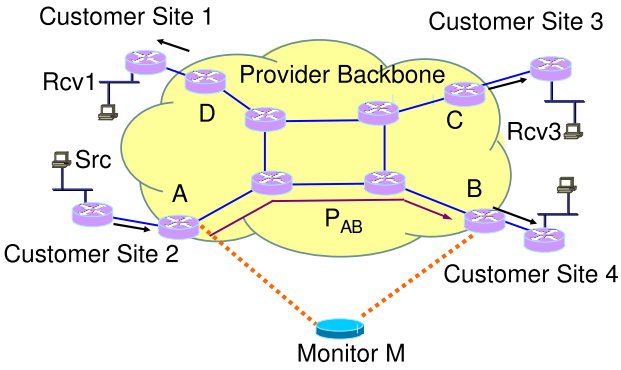

Figure 1: Monitoring one-way performance in VPNs

along a specific path (or subpath) from a network element A to an element B in the provider backbone part of the VPN (Figure 1). For instance, in the context of a multicast VPN, we want to measure the loss experienced along the multicast tree from A, the point at which customer traffic enters the provider network, to B and C, the edge routers at which the traffic exits the provider network. This loss information can be immensely useful to inform a range of management and decision processes. In this paper, we develop lightweight, scalable deployable techniques for estimating one-way loss.

One-way performance measurements are easy to realize when the starting and ending points in the path (e.g., A and B) have specialized measurement capabilities. For instance, A would be capable of launching customized probe sequences towards B, and the latter is able to terminate active measurements by receiving the probe packets and compiling and dispatching reports on them. However, such functionality is not routinely provided by ordinary production routers, and so for the above situation to occur, both would need to be a special purpose measurement hosts. However the introduction of an additional measurement host carries equipment, management and administrative costs which can be substantial if many such hosts are required to be deployed in a large network. This is particularly true in a large scale provider network where there can be a large number of paths and combination of end-points, making it infeasible to have specialized probe sources and sinks at the endpoints of every path that needs to be measured. For these reasons it is appealing if the measurement technique requires, apart from the probe source, only the presence of ordinary production routers using standard capabilities.

To address the challenges of realizing scalable performance

Submitted to SIGMETRICS, 2007 for review

monitoring with low management and deployment costs, recent work has proposed using *remote virtual monitoring* to monitor unicast and multicast VPNs. Two such systems have been designed and built for unicast [4] and multicast [3] VPNs, respectively. The key idea behind this approach is that a *single* centralized monitoring host establishes virtual connections (e.g., via encapsulation) to routers in the network to enable remote monitoring of customer performance. In the context of VPNs, the monitor can send and receive either unicast (for unicast VPN) or multicast (for multicast VPN) traffic across the provider backbone. Within the provider network, this traffic appears like, and is treated the same as, the corresponding traffic sent by the customer into the VPN. Thus, the approach enables the provider to measure customer performance across the VPN and provides a critical performance monitoring and troubleshooting tool. As an example, consider Figure 1. The monitor M *virtually attaches* to points A and B. M transmits probe packets to A along path P_{MA}. From A to B, the probes traverse the path P_{AB} that we want to monitor. Finally they are sent back from B to M along the path P_{BM}. By comparing what was sent to what was eventually recieved, M can measure the one-way performance along the directed path P_{MABM}. The key advantage of this approach is that it requires a *single* specialized monitor located anywhere, making it easy to deploy and manage such a system.

While the use of a centralized and remote monitor makes this architecture deployable and scalable, these benefits do not come without a cost. Specifically, the systems measures the path P_{MABM}, while the path of interest is P_{AB}. Thus, a critical final component of these systems is the ability to factor out the performance contributions due to the path segments P_{MA} and P_{BM}, from the overall performance observed on P_{MABM}. The measurement problem then devolves to measuring one-way loss for paths P_{MA} and P_{BM}, where one end of the path has a specialized measurement device and the other end is a production router (A or B). The existing remote virtual monitoring systems [3, 4] make the simplifying assumption that performance on the paths between the monitoring device and the network routers is symmetric. These systems make continual round trip measurements on P_{MA} and P_{MB} from M (using the *ping* utility), and assign half of the measured loss to each direction of the path. Given that the assumption of symmetric performance does not hold in the Internet (due to both the existence of assymmetric routing and unequal loading of duplex links), we are motivated to explore more accurate techniques for factoring out the performance of these one-way paths.

In this paper we develop techniques for estimating one-way loss between a host and a production router, in which the former is a fully programmable measurement device and the latter supports only standard router features. These techniques provide capabilities not supported by previously feasible round trip measurements between a host and router, and they are far more usable than techniques that depend on two measurement hosts located at each endpoint of a path to provide these measurements. The techniques use tunneling to virtualize the IP layer topology, conduct a set of measurements over this topology which are received either at network routers or a second measurement host, and use the resulting measurements as inputs into inferencing algorithms. While our work is motivated by the remote virtual performance monitoring systems that have been proposed for VPNs [3, 4], we believe our results have wider applicability. As an example, the architecture in the remote virtual monitoring systems can be applied to generic IP network monitoring, which would also require a general solution to the problem of estimating one-way loss.

1.1 Contributions

We next overview the main contributions of this paper. We develop novel techniques for estimating one-way loss from a measurement host to network routers which exploit commonly implemented features on commercial routers and do not require any new router capabilities. By exploiting these features, the expense of deploying special purpose measurement hosts can be avoided while overcoming the shortcomings of two-way measurements between a source measurement host and a router.

Our solution addresses the issue of scale by exploiting measurement features of the deployed network infrastructure to a large degree. There are three components. Firstly, GRE tunneling is used to steer measurement traffic from a small number of measurement hosts down the paths followed by customer traffic. Secondly, innovative probing methods, coupled with standard measurement capabilities such as NetFlow and SNMP, are used to isolate the performance of groups or even individual measurement packets. Thirdly, we exploit and extend tomographic inference methods in order to extract the performance of probe traffic on customer paths within the network. This combination yields a powerful yet lightweight method to determine customer performance within the network.

The remainder of the paper is organized as follows. Section 2 describe the use of GRE tunnels to steer measurement traffic down customer paths, and describes how to structure traffic probing processes in order that they can be captured by standard passive measurement technologies, in particular NetFlow, in a way that allow the correlation of packet or groups of packets between the probes and their measurements. It describes three specific measurement configuration in which these probes may be deployed. Section 3 describes two new statistical methods for inferring one-way loss measurements from these measurement, and analyses their relative performance under model conditions. Section 4 reports evaluations of the methods under network simulations which recreate the most likely violations of the model assumptions behind the estimators. The results indicate that under those conditions, and except for some potential topological biases, estimator performance in sufficient to estimate loss rates of between 5% and a fraction of 1% with no worse than at 7% error on average. Section 5 describes a laboratory implementation of our methods, so establishing a proof of concept for our measurement architecture. We conclude in Section 6 with a summary of our finding and outline additional technical issues and future research directions.

1.2 Related Work

The inference methods developed in this paper build on and extend multicast inference methods developed in the MINC project; see [5, 9], and related unicast approaches [15, 20]. The pathchar tool [8] was designed for one way measurement along a path starting from a measurement host, although may require too much bandwidth for widespread use. A number of measurement infrastructure systems conducted measurements across a comprehensive mesh of paths between network hosts; see e.g. [14]. Finally, fault isolation using scoped multicast traceroute has been proposed in [16]. Unlike the tomographic methods, this requires participation

by network routers (to respond to mtrace requests).

2. MEASUREMENT METHODS

The measurement techniques we developed to solve the one-way loss problem can be decomposed into three components. The first consists of a probing method which determines what probe packets are sent into the network. The second concerns data how are the results of the probes are measured and collected? The final component uses the probe results to compute the metric of interest.

Before describing the specific components we employ, we first motivate our solution. In our motivating application (see Figure 1), we have a fully programmable active measurement device at the edge of the network. However, our ability to leverage the routers inside the network is constrained. These devices support a standard set of router features (packet forwarding, standard management interface, etc.), but they do not provide a general programming environment. Thus, one-way measurements using active measurement devices at *both* ends of the path of interest, as could be provided by a dedicated measurement infrastructure [14], are not feasible. Instead, we have a programmable device at the edge of the network along with a limited ability to measure directly inside the network, and we are trying to learn something about a piece of an end-to-end path.

This bears some similarity to environments in which tomographical inferencing, such as MINC [5], can be applied. In the case of MINC, a measurement host at the edge of the network sends multicast probe packets into the network which are delivered to multiple receiving hosts. The loss rates on components of the multicast tree can then be inferred. The components for which loss rate is inferred consist of the links in the logical multicast tree whose nodes are the branch points in the actual multicast tree. The shortcoming of this technique, as it applies to the problem at hand, is that it provides no control over which segments of the end-to-end path are measured. Rather, these are solely a function of the multicast routing protocol used in the network.

2.1 Multicast Probing Over GRE

We overcome the inability of MINC to control which components of the end-to-end path it reports on by manipulating the IP layer topology. To do this, we employ Generic Routing Encapsulation (GRE) [12] tunnels between measurement hosts and the routers at the end of the paths we want to measure. With a GRE tunnel, an IP packet, referred to as the *inner* packet, is encapsulated in another IP packet, referred to as the *outer* packet. The source and destination IP addresses in the outer packet are the addresses of the local and remote tunnel endpoints, respectively. The outer packet is transmitted from the local to the remote tunnel endpoint, and the inner packet is not processed at the intervening routers. At the remote tunnel endpoint, the outer packet header is removed and the inner packet is processed according to normal IP processing (i.e., it is handed to a higher layer protocol if it is destined for the remote endpoint, or it is forwarded towards its eventual destination.) GRE tunneling provides IP-in-IP encapsulation such that, from the standpoint of the IP layer topology, the two tunnel endpoints appear directly connected.

We exploit the functionality of GRE tunnels in the following way. The measurement host sends a multicast packet across a GRE tunnel to the router at the other end. This multicast packet is encapsulated in a unicast packet addressed to the router. The encapsulation guarantees that the multicast packet will reach the remote router even if the intervening infrastructure does not support multicast routing. Further, and as importantly, encapsulation ensures that even if the intervening infrastructure is multicast enabled, the tunneled multicast packet will reach the remote tunnel endpoint (i.e., the tunnel bypasses normal multicast routing at the intervening routers.) We use this encapsulation of multicast packets in GRE tunnels as the first component in our measurement framework.[1]

2.2 Data Collection – Individual Packets

GRE tunnels are used to support the data collection component for individual packets. Since a measurement host and router are directly connected in the (virtual) IP topology via the GRE tunnel, the host can send IGMP [13] messages to the router. IGMP is a protocol between hosts and adjacent routers used to announce a host's desire to join a multicast group. IGMP Join messages will be forwarded along the tunnel, bypassing processing (other than IP forwarding of the outer packet) at the intervening routers. Once the host has joined a multicast group, packets received by the router and addressed to the multicast group will be encapsulated in the tunnel and delivered to the host. The unicast GRE encapsulation ensures that the packets are forwarded to the host. Thus, by establishing a tunnel from a router to a measurement host, multicast probe packets that are received by the router can be forwarded to the host. The host can produce packet reports that can be used in estimating network performance measures.

It is possible for the packet to be received at the original host that sent it or at a different host. In the former case, two parallel GRE tunnels must be configured between the host and router. The requirement for two tunnels is needed to overcome multicast routing loop prevention mechanisms which otherwise prevent a multicast packet from being transmitted over an interface on which it was received.

2.3 Data Collection – Aggregates

Utilizing individual packet reports, as described above, requires packets to be delivered to a measurement host over a GRE tunnel. As an alternative data collection method, we leverage standard data reporting capabilities of routers, thereby avoiding the need to involve hosts in the data collection process. Commercial routers generally provide two data collection and reporting mechanisms – NetFlow and SNMP. We describe aggregate data collection using Netflow.[2]

Flows and NetFlow. NetFlow [7] is a feature introduced on Cisco routers that is supported by other router vendors. NetFlow collects and exports data on packet aggregates traversing a router. A NetFlow record contains a summary of information about all packets belonging to a *flow* received by a router during a particular interval. All packets of a flow match the same key, where the key is specified by the following information in a packet: the source and destination IP addresses, the IP protocol field, and the source and destination port numbers (if pertinent to the higher layer protocol.) A NetFlow record contains, among other things, the time that the first and last packets included in

[1] While we focus our discussion on GRE encapsulation, other tunneling mechanisms would also suffice.

[2] A description of how SNMP could be used as an alternative, as well as a discussion of the tradeoffs between the two data collection methods, is omitted for space constraints.

the record were received, the number of packets received, and the total number of bytes contained in these packets.

Flow Termination. When NetFlow is enabled on an interface, a NetFlow record is instantiated whenever a packet is received on the interface and no NetFlow record already exists for that flow. When the NetFlow record is subsequently terminated, it is exported to a NetFlow collector, a special purpose host that may collect NetFlow records from many routers simultaneously. Several events can cause a NetFlow record to be terminated. First, if no packets are received for the flow for a configured interval, referred to as the *inactive timeout*, the NetFlow record is terminated. Second, records are also terminated if they have been active for longer than a second configured interval, known as the *active timeout*. Third, there is a maximum number of active NetFlow records that a router can store concurrently. If this number is exceeded when a new NetFlow record is instantiated, an existing NetFlow record is terminated. Finally, protocol specific events (e.g., the receipt of a TCP FIN packet) can cause a NetFlow record to be terminated.

Measuring Multicast with NetFlow. NetFlow provides information about multicast packets if the multicast tree over which the packets are transmitted reaches a NetFlow-enabled router. This can be accomplished in one of two ways. First, the router can explicitly join the multicast group, ensuring that the multicast tree will be extended to the router. Router command line interfaces generally provide this capability. Alternatively, a host that is downstream (relative to the source of the probes) of a NetFlow-enabled router can issue an IGMP Join for the multicast group used for probing. The challenge in this second method is in ensuring that the host joining the group is the right place topologically in the network. A GRE tunnel between the host and the router provides a trivial solution to this challenge. Once the multicast tree is extended to the router, a NetFlow record for the probe packets is created, exported to the NetFlow collector when the NetFlow record terminates, and can be used in packet loss computations.

Impact of NetFlow Sampling. The higher overhead required to support NetFlow has been addressed in three ways, all of which impact its ability to serve as a data collection function in our framework. First, because of the overhead of data collection and storage, and even the cost of licenses, NetFlow may only be enabled on a subset of network routers and interfaces. Second, on high speed links, packets may be sampled prior to the formation of flow statistics, with 1 in N packets being selected either randomly or deterministically. Third, completed flow records may be sampled within the measurement collection infrastructure, after they have been exported by the router; see e.g. [11].

Our current analysis assumes that no sampling takes place. However, availability of only partial data after sampling is not in principle an obstacle to our general framework. There exists a substantial literature on inference with partial data, some specifically in network performance tomography [10]. Its application to the variant problem with sampled flow data is a topic for future work. In general, the presence of sampling will increase the number of probe packets needed to perform estimates with a given target accuracy.

2.4 Correlating Aggregate Measurements.

The inference methods used in this paper require the ability to associate measurements taken at different points in the network, of a given multicast probe packet, or given group

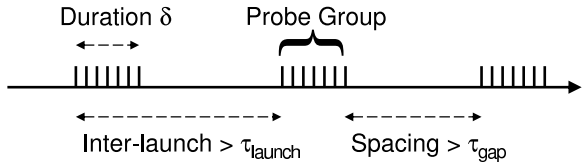

Figure 2: Probe groups and timing constraints

of probe packets. With fully programmable measurement points, association can be accomplished by including sequence numbers or other identifiers in probe packets. However, NetFlow does not report such sequence numbers. Instead, we propose two mechanisms by which association may be performed on aggregate measurements that they produce.

Temporal Association. NetFlow possesses an inherent temporal granularity. The inactive flow timeout can be used to separate groups of probe packets. Measurement traffic will constitute groups of probe packets in which the gap between the last packet of one group and the first packet of the next group is chosen in order that no flow record will report packets from both groups. This is done by choosing the gap to be at least equal to the sum $\tau_{\text{gap}} = \tau_i + \tau_p$, where τ_i is the inactive flow timeout and τ_p is the largest possible packet propagation time through the network (including transmission delay and possible queueing). Suppose that clocks at the probe source and measuring router are synchronized to within a tolerance τ_s. Then provided the time between the launch of successive probe groups exceeds $2\tau_s + \tau_p$ we can unambiguously associate a given NetFlow report with the set of probe packets that generated it.[3] Let δ be the duration of an individual probe group. Then the time between probe launches must also exceed $\delta + \tau_{\text{gap}}$. Thus the time between probe launches must exceed

$$\tau_{\text{launch}} = \max\{\delta + \tau_{\text{gap}}, 2\tau_s + \tau_p\} \quad (1)$$

Acheiving sub-second clock synchronization in today's networks is not difficult, maximum propagation time is also typically less than a second, and the Netflow inactive timeout is often set to 30 or 60 seconds. Thus, we would expect the first term in the maximum to dominate.

It is possible for multiple flow records to be reported for a given probe group due to flow termination prior to active timeout. However, this can be detected from the timestamps of the flows' first and last packets, and the corresponding flow records amalgamated in the collector.

Spatial Association. With NetFlow measurement, packets are distinguished by their flow key. Using a single key, we are confined to measurement sequences that will contain gaps on the order of the inactive timeout, τ_i, which is inadequate for measurement functions requiring continuous estimates of loss. We overcome this problem by using multiple key values. By using the same key for each packet in a probe group, but changing the flow key for each successive group of probe packets, we cause them to generate distinct flow records. This may be accomplished by changing the port numbers in UDP probe packets. The effect is to remove the constraint that probe starts be separated by at least the inactive timeout τ_i. Suppose we have k key values at our disposal, used round robin on the probes groups. Probe groups in a given key must obey the conditions described above (times between launch $> \tau_{\text{launch}}$, gaps between

[3]The factor of 2 accounts for clock skew in either direction.

groups $> \tau_{\text{gap}}$). However, for the purpose of distinguishing groups, the time between successive launches of probes of all types need only be greater than τ_{launch}/k. Less time between probes will cause increased occupation of the flow cache; assuming the smallest possible interprobe time τ_{launch}/k there would be up to k concurrent flows.

We next describe specific measurement scenarios that enable us to collect the data necessary to estimate the one-way loss rates of interest.

2.5 Measurement Configurations

We propose three variant configurations of the basic measurement topology; these are illustrated in Figure 3. In each figure we show the packet transmission probabilities α_1 or β_i associated with tunnels. In all cases it is assumed that the router $R1$ does not report traffic measurements.

Configuration A: Probe Collection at Two Measurement Hosts. Two GRE tunnels r and s are created from $M1$ to $R1$, using distinct IP addresses at $M1$, denoted $M1(a)$ and $M1(b)$. A GRE tunnel t is created from $M2$ to $R1$ using IP address c on $M2$. Multicast routing is enabled on $R1$, and on each tunnel interface at $R1$. Host $M1$ joins a multicast group through the tunnel s and host $M2$ join the same multicast group through tunnel t. Host $M1$ launches probes into tunnel r, copies of which are multicast down tunnels s and t, to be collected at $M1(a)$ and $M2(c)$.

In this configuration we exploit the flexibility of $M1$ and $M2$ to measure individual packets; therefore inference can be performed using the individial packet MINC estimator.

Configuration B: Probe Collection at a Measurement Host and a Router. There are two variants of this configuration both developed from Configuration A. In the first variant, the additional router $R2$ is positioned in the path between $R1$ and $M2$ in order to observe packets in transit. In the second variation, $R2$ plays substitutes for $M2$, and joins the multicast group using a static IGMP join, then observes packets destined to it.

In this configuration, the ability to distinguish individual packets is limited by the router level measurement at $R2$, which is assumed to report only aggregates. Therefore we must use the aggregate MINC estimator (described later) on packets aggregate reported by $R2$.

Configuration C: Observation at a Measurement Host and Reflection. This configuration is based upon configuration A, except that host $M2$, instead of measuring each packet, responds to its receipt by multicasting a second packet into tunnel t, subsequently to be received by $M1$. Host M2 may introduce a delay before responding, to break temporal correlations. The configuration advantage of this approach is that only M1 collects data, thus avoiding coordination of measurement collection between multiple hosts. In Figure 3 β_3 is the transmission probability on the round-trip path $R1 \rightarrow M2 \rightarrow R1$.

In this configuration we can exploit the flexibility of $M1$ to measure individual packets, and therefore inference can be performed using the reflected packet MINC estimator (described later.)

3. INFERENCE METHODOLGY

3.1 Tree Model and Probe Process

In this section we detail three algorithms used to perform inference in the network measurement configurations discussed in Section 2.5: the individual packet MINC estimator, the aggregate MINC estimator, and the reflected packet MINC estimator. We abstract the logical topologies in Figure 3 as follows: considering a two leaf multicast tree in which the root node 0 has a single child 1 (the branch point) which in turn has two children 2 and 3. The link terminating at node k will be called link k. A set of probes labelled by $i = 1, 2, \ldots, n$ is multicast from 0 to the leaves. A packet attempting to cross link k is transmitted independently with probability α_k. X_k^i is a random variable taking the value 1 is packet i reaches node k, and 0 otherwise. The observed outcome from probe i is the pair (X_2^i, X_3^i). We will write $\overline{\alpha} = 1 - \alpha$.

3.2 Individual Packet MINC Estimator

Let $n(xy)$ denote the number of probes for which $(X_2^i, X_3^i) = (x, y)$. Let $n(x*) = n(x1) + n(x0)$. The log-likelihood function is

$$\begin{aligned} L(\alpha) &= n(11)\log(\alpha_1\alpha_2\alpha_3) + (n(1*) - n(11))\log(\alpha_1\alpha_2\overline{\alpha}_3) \\ &\quad + (n(*1) - n(11))\log(\alpha_1\overline{\alpha}_2\alpha_3) \\ &\quad + (n + n(11) - n(1*) - n(*1))\log(\overline{\alpha}_1 + \alpha_1\overline{\alpha}_2\overline{\alpha}_3) \end{aligned} \tag{2}$$

In [5] it is shown that the MLE for α is

$$\widehat{\alpha}_1 = \frac{n(1*)n(*1)}{n(11)n}, \quad \widehat{\alpha}_2 = \frac{n(11)}{n(*1)}, \quad \widehat{\alpha}_3 = \frac{n(11)}{n(1*)} \tag{3}$$

3.3 Reflected Packet MINC Estimator

This estimator is used in Configuration C, where the packet dispatched to M_2 is reflected back to R_1, with roundtrip transmission probability β_3 between M_2 and R_1, and then forwarded to M_1. As indicated in the logical topology for Configuration C in Figure 3, we can map the link parameters β into the link parameters α of Configuration A by:

$$(\alpha_1, \alpha_2, \alpha_3) = f(\beta_1, \beta_2, \beta_3) = (\beta_1, \beta_2, \beta_2\beta_3) \tag{4}$$

The likelihood function is $K(\beta) = L(f(\beta))$ and hence the MLE for β is

$$\widehat{\beta} = f^{-1}(\widehat{\alpha}) = (\widehat{\alpha}_1, \widehat{\alpha}_2, \widehat{\alpha}_3/\widehat{\alpha}_2) \tag{5}$$

where $\widehat{\alpha}$ is the MLE on the MINC model. In particular $\widehat{\beta}_3 = n(1*)/n(*1)$.

3.4 Packet Aggregate MINC Estimator

In this MINC variant, the observed outcomes are not on individual packets, but on blocks of consecutive packets. Specifically, given a block size s, the block aggregates at link k are block aggregates

$$Y_k^i = \sum_{i'=s(i-1)+1}^{is} X_k^i \tag{6}$$

The likelihood to observe the data $Y = \{Y^i\} = \{Y_2^i, Y_3^i\}$ is $P_\alpha(Y) = \prod_i P_\alpha^i(Y^i)$ where

$$P_\alpha^i(Y^i) = \sum_{r=\max\{Y_2^i, Y_3^i\}}^{s^i} P_\alpha^i(r, Y_2^i, Y_3^i) \tag{7}$$

and

$$\begin{aligned} P_\alpha^i(r, Y_2^i, Y_3^i) &= \alpha_1^r\overline{\alpha}_1^{(s^i - r)} \binom{s^i}{r} \alpha_2^{Y_2^i}\overline{\alpha}_2^{(r - Y_2^i)} \binom{r}{Y_2^i} \\ &\quad \times \alpha_3^{Y_3^i}\overline{\alpha}_3^{(r - Y_3^i)} \binom{r}{Y_3^i} \end{aligned} \tag{8}$$

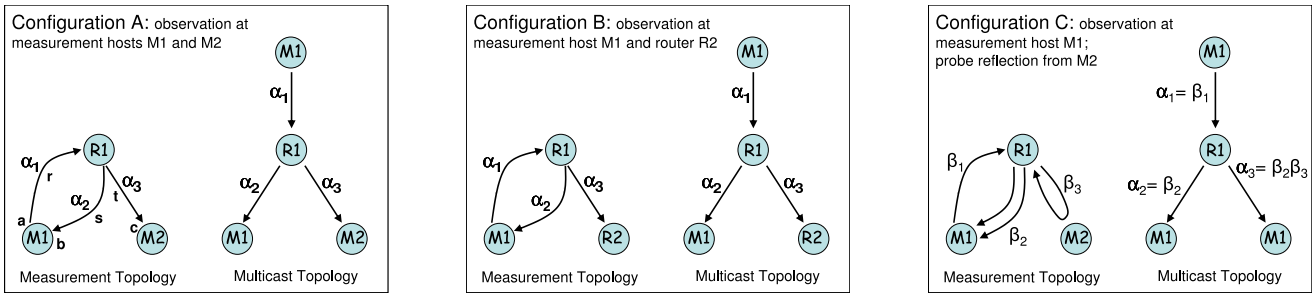

Figure 3: Three measurement configurations and associated logical multicast topologies

is the complete data likelihood, i.e., the probability for r packets in group i to reach node 1, then Y_2^i and Y_3^i of these to reach nodes 2 and 3 respectively.

The maximum likelihood estimator (MLE) is that which maximizes P, i.e.,

$$\widehat{\alpha}^{\text{mle}} = \arg\max_{\alpha} P_\alpha(Y) \tag{9}$$

Candidates for the MLE are the stationary points of P_α. These may, in principle, be found by differentiation and numerical route finding; in practice this will be computationally very intensive.

3.5 EM Algorithm for Aggregate MLE

Let Y_1^i denote the (unknown) number of packets of group i that reach node 1. The complete data likelihood to obtain the complete data $Y^{\text{cmp}} = \{Y_1^i, Y_2^i, Y_3^i\}$ is

$$Q_\alpha(Y^{\text{cmp}}) = \prod_i P_\alpha^i(Y_1^i, Y_2^i, Y_3^i) \tag{10}$$

In the EM algorithm, we find an iterative sequence $\{\widehat{\alpha}^{(j)}\}$ which, under certain conditions, approximates the MLE $\widehat{\alpha}^{\text{mle}}$. The procedure is given a $\widehat{\alpha}^{(j)}$, find $\widehat{\alpha}^{(j+1)}$ as the maximizer:

$$\widehat{\alpha}^{(j+1)} = \arg\max_{\alpha} \mathsf{E}_{\widehat{\alpha}^{(j)}}[\log Q_\alpha(\{Y_1^i, Y_2^i, Y_3^i\}) \mid \{Y_2^i, Y_3^i\}] \tag{11}$$

The solution of this can be shown to be

$$\widehat{\alpha}_1^{(j+1)} = \sum_i \mathsf{E}_{\widehat{\alpha}^{(j)}}[Y_1^i \mid Y_2^i, Y_3^i]/S \tag{12}$$

$$\widehat{\alpha}_2^{(j+1)} = Y_2 / \sum_i \mathsf{E}_{\widehat{\alpha}^{(j)}}[Y_1^i \mid Y_2^i, Y_3^i] \tag{13}$$

$$\widehat{\alpha}_3^{(j+1)} = Y_3 / \sum_i \mathsf{E}_{\widehat{\alpha}^{(j)}}[Y_1^i \mid Y_2^i, Y_3^i] \tag{14}$$

Here $S = \sum_i s$ is the total number of packets and $Y_k = \sum_i Y_k^i$. The required conditional expectation is

$$E_{\widehat{\alpha}^{(j)}}[Y_1^i \mid Y_2^i, Y_3^i] = \frac{\sum_{r=\max\{Y_2^i, Y_3^i\}}^{s^i} r P^i_{\widehat{\alpha}^{(j)}}(r, Y_2^i, Y_3^i)}{P^i_{\widehat{\alpha}^{(j)}}(Y^i)} \tag{15}$$

Starting form some initial state, we terminate the iteration when successive iterates are sufficiently close, e.g., when $\|\widehat{\alpha}^{(j)} - \widehat{\alpha}^{(j+1)}\| < \varepsilon$ for some $\varepsilon > 0$. (Here $\|\cdot\|$ is the standard Euclidean norm).

In [9] a moment estimator was derived for α based on the same aggregate measurements. The moment estimator has the advantage that it can be computed explicitly; however, the estimator we present here was noticeably more accurate. This is to be expected since MLEs in general enjoy a minimal variance property. The moment estimator could be used as the initial state $\widehat{\alpha}^{(0)}$ our iteration.

3.6 Estimation Variance

Maximum Likelihood Estimators enjoy the properties of being consistent (i.e., they converge to the true value as the number n of observation grows) and being asymptotically efficient (i.e., they have minimal variance, as the number of observations grows.) In particular, we have the property that $\sqrt{n} \cdot (\widehat{\alpha} - \alpha)$ converges in distribution to a mean zero Gaussian random variable whose covariance matrix σ^2 is $-(\mathsf{E}_\alpha[L_1''(\alpha)])^{-1}$ where L_1 is the single observation log-likelihood function, and L_1'' is the matrix of partial second derivatives. For approximation purposes we have that the the covariance matrix of $\widehat{\alpha}$ is aproximately σ^2/n for large n.

3.6.1 *Variance of MINC Estimator*

Using the above approach It has been shown in [5] that the asymptotic covariance of the MINC Estimator for $\{\widehat{\alpha}_1, \widehat{\alpha}_2, \widehat{\alpha}_3\}$ is

$$\sigma_\alpha^2 = \begin{pmatrix} \frac{\alpha_1(\overline{\alpha}_2 - \alpha_3(1+\alpha_2(\alpha_1-2)))}{\alpha_2\alpha_3} & \frac{-\overline{\alpha}_2\overline{\alpha}_3}{\alpha_3} & \frac{-\overline{\alpha}_2\overline{\alpha}_3}{\alpha_2} \\ \frac{-\overline{\alpha}_2\overline{\alpha}_3}{\alpha_3} & \frac{\overline{\alpha}_2\alpha_2}{\alpha_1\alpha_3} & \frac{-\overline{\alpha}_2\overline{\alpha}_3}{\alpha_1} \\ \frac{-\overline{\alpha}_2\overline{\alpha}_3}{\alpha_2} & \frac{-\overline{\alpha}_2\overline{\alpha}_3}{\alpha_1} & \frac{\overline{\alpha}_3\alpha_3}{\alpha_1\alpha_2} \end{pmatrix} \tag{16}$$

Note that to first order in the loss rates, this is the diagonal matrix with entries $\{\overline{\alpha}_1, \overline{\alpha}_2, \overline{\alpha}_3\}$.

3.6.2 *Variance of Reflected Packet Estimator*

Since $K(\beta) = L(f(\beta))$,

$$K''(\beta) = \nabla f(\beta) \cdot L''(f(\beta)) \nabla f(\beta) \tag{17}$$

where ∇f is the matrix of partial derivatives of f, namely

$$\nabla f(\beta) = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & \beta_3 \\ 0 & 0 & \beta_2 \end{pmatrix} \tag{18}$$

Combining (16), (17) and (18) we obtain the asymptotic covariance for $\widehat{\beta}$ as

$$\sigma_\beta^2 = (-\mathsf{E}[K_1''])^{-1} = (-\nabla f(\beta) \cdot \mathsf{E}_\alpha[L''(f(\beta))] \nabla f(\beta))^{-1} =$$

$$\begin{pmatrix} \frac{\beta_1(\overline{\beta}_2 - \beta_2\beta_3(1+\beta_2(\beta_1-2)))}{\beta_2^2\beta_3} & \frac{\overline{\beta}_2(\beta_2\beta_3-1)}{\beta_2\beta_3} & 0 \\ \frac{\overline{\beta}_2(\beta_2\beta_3-1)}{\beta_2\beta_3} & \frac{\overline{\beta}_2}{\beta_1\beta_3} & \frac{-\overline{\beta}_2\beta_3}{\beta_1} \\ 0 & \frac{-\overline{\beta}_2\beta_3}{\beta_1} & \frac{\beta_3(1+\beta_3-2\beta_2\beta_3)}{\beta_1\beta_2} \end{pmatrix} \tag{19}$$

To first order in the loss rates, this is the matrix

$$\begin{pmatrix} \overline{\beta}_1 & 0 & 0 \\ 0 & \overline{\beta}_2 & -\overline{\beta}_2 \\ 0 & -\overline{\beta}_2 & 2\overline{\beta}_2 + \overline{\beta}_3 \end{pmatrix} \tag{20}$$

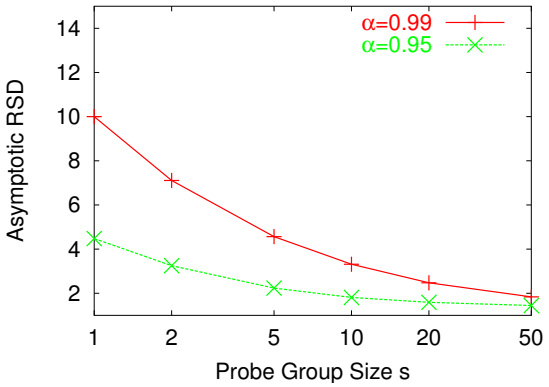

Figure 4: Asymptotic Variance for Aggregate MLE. Asymptotic variance $\sigma^2(s)$ in the scale of many probe groups, as a function of group size s

The estimate variance that concern us most, those with the labels 1 and 2, have the same leading order behavior for both the individual packet and reflected packet MINC estimators.

3.6.3 *Variance of the Packet Aggregate MLE*

The same methods apply straightforwardly; we omit the expresion of the variance due to its complexity. In this case L_1 is the likelihood function for observation of a group of s packets. $\sigma^2(s)$ will denote the corresponding asymptotic covariance $-(\mathsf{E}_\alpha[L_1''(\alpha)])^{-1}$. If there are n total packets in $m = n/s$ groups, then the variance of the MLE is approximately $\sigma^2(s)/m$.

Figure 4 shows the asymptotic Relative Standard Deviation (RSD) $\sigma(s)/\sqrt{1-\alpha}$ of estimation of the shared link loss for loss rates $\overline{\alpha} = 1\%$ and 5%, as the number of probe groups grows. That is, dividing the asymptotic RSD by the square root $\sqrt{m}$ of the number of probe groups approximates the RSD. We plot the asymptotic RSD as a function of the group size s; the individual packet MINC estimator corresponds to $s = 1$. The figure illustrates the benefits of increasing the number of packets in a group; even though the packet level information is aggregated, increasing the number of packets per group increases the amount of measurement information, reducing the RSD. We believe this is a natural regime to consider, since in practice the main constraint on aggregate measurements is the frequency at which probes groups can be measured. Consequently, each probe group may contain a relatively large number of probes without the average rate being high.

Although we have no analytic results concerning the asymptotic variance of the aggregate MLE, as an efficient estimator, its asymptotic variance is bounded above by the variance $\sigma_m^2(s)$ of the moment estimator of [9]. In the case of equal loss rates on each link, $\sigma_m^2(s)$ is approximated for small $\overline{\alpha}$ as $\sigma_m^2(s) \approx \overline{\alpha}/s + 3\overline{\alpha}^2$. The qualitative behavior is similar to that displayed in Figure 4.

To get an idea of likely values of the aggregate size s, in NetFlow-based measurements, consider group of probes with width δ, and assume the probes have packet rate equivalent to VoIP call transmitting one packet every 20ms. Then the number of packets per group is $s = 50\delta$. Taking $\delta \ll \tau_i$ the inactive timeout which is say, 30s, then suppose each probe group uses a new flow key. Then $\tau_{\text{launch}} \approx \tau_i$ the number of concurrent flows is $k \approx \tau_i/\delta$. Consider a backbone router handling 10^5 flows concurrently. If we which to use up no more than 0.1% of the flow cache, then $k < 100$, so $\delta > 0.3$ and hence $s > 60$.

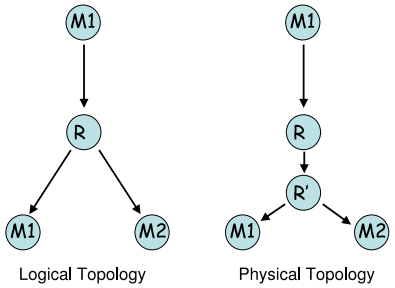

Figure 5: Topological Bias. Left: logical topology with branch point R. Right: physical topology has branch point at R'. Estimate of shared link loss rate reflects loss on $M1 \to R'$ not $M1 \to R$.

4. EXPERIMENTAL EVALUATION

A potential pitfall for the inference method is the presence of spatial correlation of loss on different links. This would violate the model assumptions, and, if not accounted for, would bias the estimators of probe loss rates. We envisage two mechanisms by which spatial correlation could arise:

- *Topological Bias:* the branch point of the logical multicast tree is not a branch point in the underlying physical topology; see Figure 5. Denote the actual branch point by R'. Inference of the loss rate on the common path will yield the loss rate of the path $M_1 \to R'$ rather than that of $M_1 \to R$. Any packet loss on the path $R \to R'$ will hence cause bias.

- *Load Synchronization:* Synchronization amongst traffic of different TCP connections sharing a common bottleneck link has been identified as a possible consequence of the TCP control algorithm [21]. This would result in synchronization of the rates of the TCP connections on the non-shared portion of their paths. Regarding such TCP traffic as background traffic, this would manifest itself as synchronization in the time-varying utilization of distinct links, potentially causing correlated loss of probe traffic.

For topological bias it is relatively simple to determine the likely magnitude of this effect. If α_1 is the transmission rate of $M_1 \to R$ and α' the transmission rate on $R \to R'$, then estimation of the common path transmission rate would yield an unbiased estimator of $\alpha_1\alpha'$ rather than α_1.

Possible bias due to load synchronization is difficult to model explicitly, for while potential mechanisms are understood, the likely magnitude of their effect is not. For this reason we elected to perform packet level simulations in which we could explicitly represent the TCP protocol and have any synchronization follow from the operation of the protocol rather than a model. We describe the layout and results from these simulations in the remainder of this section.

We stress here that the aim of these simulations is to determine the level of bias in estimation of probe loss rates. In this study we do not test the degree to which inferred loss rates agree with the loss rates of background or other application traffic. Indeed, it has been determined that the directly measured loss rates of simple periodic or Poisson

probe streams do *not* accurately represent those of some applications or protocols. This is not surprising, since traffic streams with different temporal properties (e.g. constant vs. bursty bitrate) are not expected to experience the network in an identical manner, especially in a complex and time-varying congestion environment. Our response to these concerns is that inference of general applications would require an extension of our work that encompasses more complex probe streams or probe statistics that are better matched, for inference purposes, to the traffic under study. We intend to investigate this in future work.

4.1 Network and traffic layout

We use four scenarios to study the impacts from various packet loss correlation on the accuracy of the inference methods. These scenarios are devised to have different temporal and spatial correlation in packet loss patterns over multiple links. By using long-lived TCP flows, short-lived HTTP traffic, and on-off traffic sources as background traffic, temporal correlation is introduced. For HTTP traffic, we are using the web traffic generator PackMime-HTTP package included in ns-2 [6]. In different scenarios, the paths of the traffic are deliberately constructed to induce different spatial correlation on the packet loss patterns over multiple links.

Figure 6 shows the topology and background traffic of the four scenarios. Scenario 1 studies the case where there is no spatial correlation but only temporal correlation on the packet loss patterns. In this scenario, all traffic source destination pairs cross only one link and do not affect the packet loss patterns on other links. Therefore, there is no spatial correlation among packet loss patterns over different links. Scenario 2 studies the cases where there is correlated loss between the path from M1 to R and the path from R to M2. The traffic from S4 to T4 traverses the link n1 to R and the link n2 to M2, which are in the previous two paths. This creates two bottlenecks together with traffic from S1 to T1 on link n1 to R and from S3 to T3 on link n2 to M2. Therefore, spatial correlation is induced between the packet loss patterns on the two paths. Traffic generated from S2 to T2 has only local impact on the packet loss patterns on the path from R to M1. Scenario 3 studies the cases where there are correlated losses between the downstream paths from R to M1 and from R to M2. In this scenario, S4 consists of 10 on-off traffic sources. Traffic originated from S4 is multicasted to two T4 destinations and creates packet loss correlation between the two downstream paths. Scenario 4 studies the impact of the topological bias on the inference algorithms. In this case, the 'branch point' is located at node n2 instead of node R. Probing packets sent from R to M1 and from R to M2 share a common path from R to n2 and experience the same packet loss pattern caused by the traffic from S3 to T3.

In each experiment, 50000 probes are sent with an exponentially distributed mean inter-arrival time of $10ms$. For the aggregate MINC method, the 50000 probes are divided into 5000 groups with 10 probes in each group.

The links are configured to have a capacity of $45Mbps$ and propagation delay of $10ms$. A drop-tail queue management is used. The router buffers are 55 packets, assuming an average round trip time of $100ms$ and 100 TCP flows sharing the bottlenecks [2]. Each source destination pair is connected with both long-lived TCP flows and short lived HTTP traffic. For each scenario, we vary the number of long lived TCP flows and the arrival rate of the HTTP requests and perform a set of experiments. In each experiment, the number of TCP flows is chosen from $\{10, 25, 50, 75, 100\}$ and the HTTP request arrival rate is chosen from $\{10, 25, 50, 75, 100\}$ per second. In scenario 3, the on and off periods of the on-off traffic sources are both set to be exponentially distributed with an average of 1 second. When the source is on, it sends constant bit rate traffic at 1Mbps.

In summary, a total of 100 experiments are performed and each experiment is duplicated 10 times. No link loss rates observed in the simulation exceeded 5%. The smallest average rate over a set of 10 runs was about 0.3%. The performance of each inference algorithm is studied using the metric described in the next subsection.

4.2 Performance Measures

4.2.1 Measures of Estimation Error

We wish to determine whether estimation errors have magnitude consistent with the underlying statistical model. Consider a set of N experiments. In each experiment j the actual probe transmission rate for link k is $\alpha_k(j)$: this is the actual proportion of probes that were successfully transmitted across link k, and hence varies between experiments. For each of the inference methods m, $\widehat{\alpha}_k(j,m)$ is the transmission rate inferred according to the method. The associated estimation error $\widehat{\alpha}_k(j,m) - \alpha_k(j)$.

We evaluate the estimation error in two ways. Firstly, the relative error $|1 - \widehat{\alpha}_k(j,m)/\alpha_k(j)|$ expresses the error as a fraction of the true loss rate; we think of it as an measure of the intrinsic accuracy of the method. Our summary statistic over all runs is the RMS relative error:

$$R_k(m) = \left(\frac{1}{N} \sum_{j=1}^{N} \left(1 - \widehat{\alpha}_k(j,m)/\alpha_k(j)\right)^2 \right)^{1/2} \tag{21}$$

We omit from the sum any terms with true loss rate $\alpha_k(j) = 0$ and reduce the count N accordingly.

The expected scale for the error is the standard deviation associated with the inference method, which we denote by $\sigma_k(\alpha(j), m)$. This is just the square root of the variance calculated in Section 3.6 for the various methods m. Note the estimation variance for link k is generally a function of the complete set of actual transmission rates $\{\alpha_{k'}(j) : k' = 1, 2, 3\}$ in experiment j. By normalizing the error by the associated standard deviation we can assess for each method, the error relative to expectations. We summarize this statistic over all runs with a quality measure which is the RMS normalized error, namely,

$$Q_k(m) = \left(\frac{1}{N} \sum_{j=1}^{N} \frac{\left(\widehat{\alpha}_k(j,m) - \alpha_k(j)\right)^2}{\sigma_k^2(\alpha(j), m)} \right)^{1/2} \tag{22}$$

If the errors did follow a Gaussian distribution with the expected variance, then each term in $Q_k(m)$ should follow a standard normal distribution and Q_k^2 would have expected value 1. Thus the value of Q_k allows us to measure departure of actual error from the model.

4.2.2 Measures of Correlation.

We expect departures from the underlying model, in the form of spatial and temporal correlations, to be the source of estimation inaccuracy beyond what is expected on the basis on normal statistical variability. For this reason we measured the spatial and temporal correlation in the underlying probe process in each scenario in each experiment.

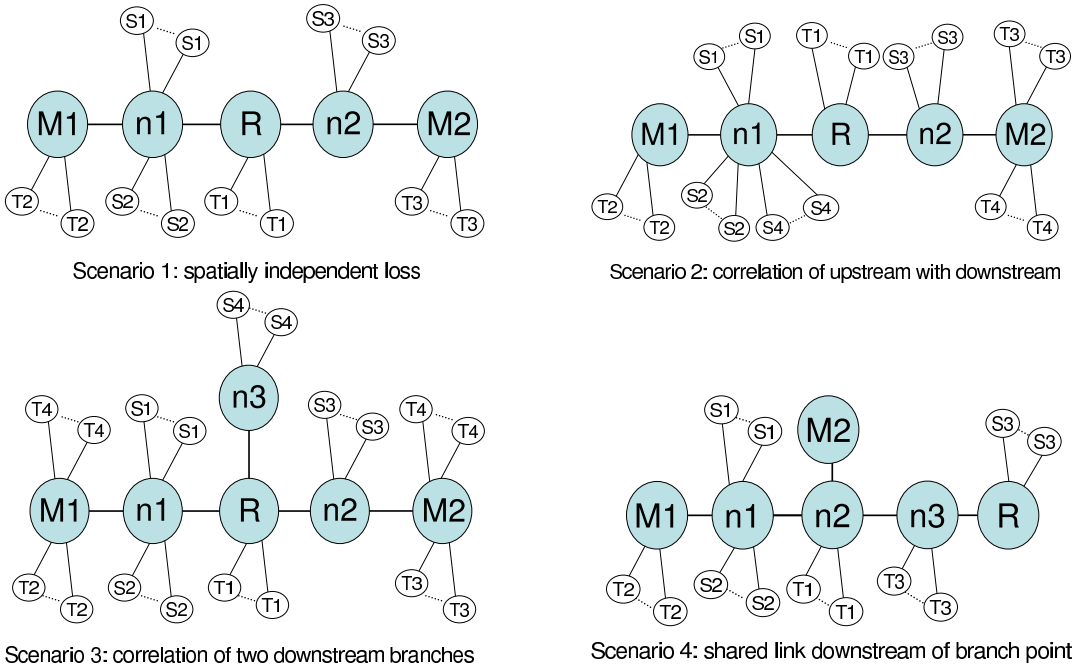

Figure 6: Layouts for Network Level Simulations. M = measurement host. R = router. S = traffic source. T = traffic sink.

Spatial Correlation: We wish to determine the extent of correlations between loss on the child links. Our simulation was instrumented to determine whether probe packets reached R from M1. According to the model, loss on the leaf links is independent when conditioned on having reached R. In each run, for this subset of packets we recorded the number $n(ij)$ in each of the outcomes in $\Omega = \{00, 01, 10, 11\}$. Let $n = \sum_{ij\in\Omega} n(ij)$ be the total number of probes considered. Then under the independence assumption, we expect $n(ij)$ to be close to $m(ij) = np(i)q(j)$ where $p(i) = (n(i1) + n(i0))/n$ and $q(j) = (n(1j) + n(0j))/n$. In fact, we can compute the corresponding χ-square statistic

$$\widehat{\chi}^2 = \sum_{ij\in\Omega} \frac{(n(ij) - m(ij))^2}{m(ij)} \tag{23}$$

Under the independence hypothesis, $\widehat{\chi}^2$ should follow a χ-square distribution with 1 degree of freedom. We can assess the significance of a given value by calculating the so-called P-value corresponding to $\widehat{\chi}^2$, i.e., the corresponding quantile of the χ-square distribution. This can be thought of as the highest significance level at which the independence hypothesis would be rejected on a one-sided significance test for that run. Thus the higher the P-value, the larger the correlation. One attractive feature of this approach is that, it provides a uniform way to assess loss in runs which can have differing underlying loss rates.

Temporal Correlation: We use the above method to assess temporal correlations. For any node k, for any link or path, we let $n(ij)$ be the number of pairs of successive probes $\{t, t+1\}$ for which $(X_k^t, X_k^{t+1}) = (i, j)$.

4.3 Experimental Results

Figure 7 displays the relative error measure R_1 for the common link in Scenario 2 for all three methods. In each plot, each curve corresponds to a certain number *ntcp* of TCP background flows: 10, 25, 50, 75, 100. Each point on the horizontal axis corresponds to an arrival rate for HTTP flows: 10, 25, 50, 75, 100. For space reasons we omit the figures for other Scenarios, but the broad behavior which we describe was the same for Scenarios 1 and 3.

- Relative Error broadly increases with the number *ntcp* of background TCP flows, although there is less dependence on *ntcp* for the Individual MINC method.
- Relative Error is less dependent of the number of HTTP connections
- Relative Errors are smallest for the MINC method, being about twice as large in the Reflected MINC method, and about 5 to 10 times larger for the aggregate MLE method.

However, in each case, the errors would seem to be manageable for the purpose of estimation, being less than 7% even in the worst case. Launching 5,500 probes with a mean interprobe time of 10ms (comparable with a single VoIP call) would take less than 1 minute, which is a reasonably short time over which to measure link behavior, being, for example, far less than the normal timescale of routing changes.

The corresponding graphs for the quality measure Q_1 are shown in Figure 8. Observe the relation between the quality values follow closely the trends describe above for the relative error. The quality values for individual and reflected MINC are all somewhat less than 1, indicating errors within expected limits. For the aggregate method the error are up to about twice the expected size.

The quality measure normalizes out the effects of accuracy of different ambient loss rates. So the similarity of trends between Figures 7 and 8 indicates that the variation in accuracy is due to departures from response expected under the uncorrelated loss model, rather than expected variation under conformance with the uncorrelated loss model.

We examined the correlations in the probe traffic loss to try to understand the way these influence departures in estimator behavior from that expected from the model. We found that almost all of the temporal P-values were close to 1, indicating significant departure from independence. We believe the temporal correlation account for the different quality measures for the three methods. This is because the presence of temporal correlations modifies the joint distribution of probe loss at different times, relative to the product distribution expected under the independent model. This in turn perturbs those estimators which use measured values of these distribution for estimation. This is clearly the case for the aggregate MLE, since the aggregate counts groups of packets. It is also the case for the reflected MINC estimator, since temporal correlations will perturb the joint distribution of transmission of the original and reflected packet on the link $R \to M1$. The mean interprobe time, 10ms, is comparable with the roundtrip times on the path $R \to M2 \to R$ in the reflected MINC estimator, so the level of correlation in the reflected MINC estimator would be comparable with that seen by the aggregate MLE. On the other hand, the marginal, single packet distribution used by the individual packet MINC estimator are not changed by correlations, although the convergence of the estimator as the number of probes grows may be slower.

To what extent can variation within each figure be explained by spatial correlations? To gauge this we plotted (see Figure 9), for each number of TCP connections, the sorted P-values for spatial correlation in Scenario 3; behavior in Scenarios 1 and 2 was similar. Under the null hypothesis, n of these should lie roughly on a straight line of slope $1/n$, i.e., the probability for any P-value to exceed a level p is $1-p$. In fact, this behavior is more closely followed for *larger ntcp* values: 50, 75, 100, whereas for fewer *ntcp* values, 10 and 25, there are substantially smaller large correlations. This seems to be reflected in Figures 7 and 8, where, especially for the reflected and aggregate methods, the errors for *ntcp* value 10 and 25 fall below the others. It could be said that theirs is the abnormal behavior, since it reflects unusually small correlations. The origin of this behavior is not clear, though we conjecture that is may be due to some form of TCP synchronization [21], that can occur for smaller numbers of TCP connections, but which is generally dissipated by the increased complexity in behavior that ensues when more TCP connections are present.

In Figure 10 we display the estimator quality metric Q_k for the aggregate MLE methods under the three Scenarios 1, 2 and 3. The quality across scenarios is similar, indicating that temporal correlations are the main factor driving Q_k to be somewhat greater than 1. Finally, our experiments with Scenario 4 gave far larger errors: relative errors were of the order 1, showing that, as expected, the method has failed to provide useful estimates in the presence of topological bias.

5. IMPLEMENTATION

We have implemented the inference algorithms in a prototype system and tested the implementation in a small scale testbed. The implementation and testbed include all the components of our system. As a complement to the performance evaluations described in the previous sections, our purpose here is to build a prototype system to assess the practical feasibility of implementing and deploying such a system.

The packet probing and data collection modules, as well as the inferencing algorithms, were implemented in a single application. We implemented GRE tunnels (over raw sockets) in the application to support sending and receiving probes.[4] While the application level implementation of GRE tunnels improves the portability of our system, it required that we also implement the IGMP and UDP functionality needed by the application. The probing module sends UDP multicast packets. These packets include an application layer header containing a sequence number, to match received with sent packets. Packets belonging to different probe groups are identified by their destination port numbers. To support the reflected MINC method, the application is capable of "reflecting" probe packets when they are received. All information about the transmission of probes and their subsequent receipt is stored in a database. The inference engine uses these probing records to execute the requested inference alogirthm. Parameters of the experiments (probing rate, inferencing technique used, etc) are controlled by scripts running on the measurement hosts.

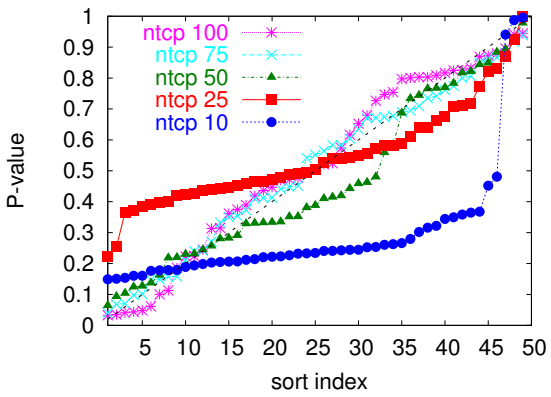

Figure 9: P-values: Sorted for Scenario 3, broken out by number of tcp connections.

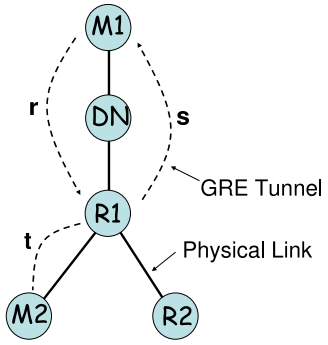

Figure 11: Testbed Topology

The experimental testbed consists of three PCs ($M1$, $M2$ and DN) and two Cisco 7200 routers ($R1$ and $R2$) connected as shown in Figure 11. $M1$ and $M2$ act as probing and monitoring hosts. The third PC, DN, connects $M1$ and router $R1$. Dummynet [17] is configured on DN to induce random loss on packets sent between $M1$ and $R1$. $M2$ and $R2$ are both connected to $R1$ and are used to collect probe packets in the individual packet and aggregate packet experiments.

[4]We omit a detailed discussion of the tradeoffs motivating our design decisions here. A discussion of these issues can be found in [1].

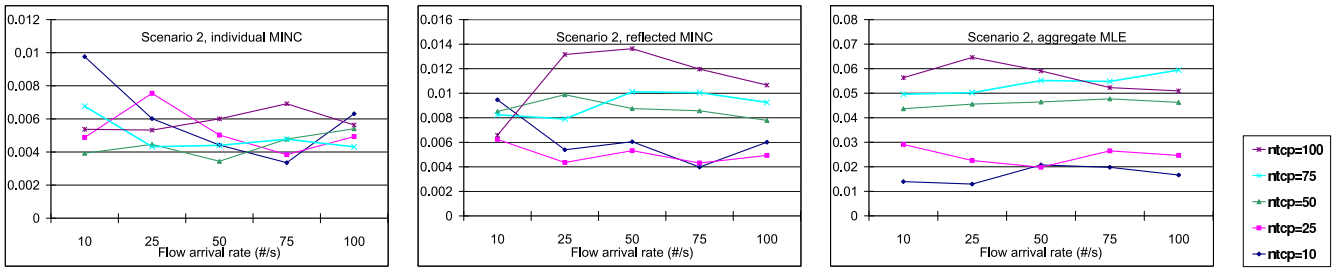

Figure 7: Relative Error: RMS relative estimation error R_1 on shared path for three estimation methods under Scenario 2.

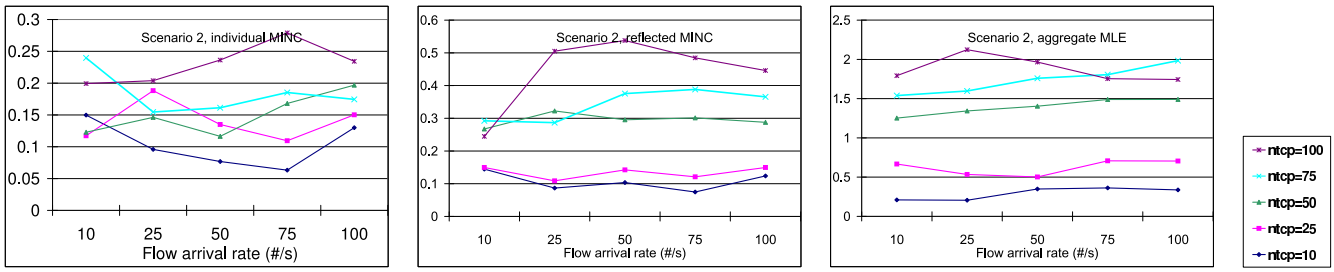

Figure 8: Quality Measure Error: RMS scaled error Q_1 on shared path for three estimation methods under Scenario 2.

A pair of GRE tunnels, r and s, is configured between $M1$ and $R1$. (Recall from Section 2 that two parallel tunnels are needed so that multicast packets received by $R1$ from $M1$ can be sent back to $M1$.) A third GRE tunnel, t, is configured between $R1$ and $M2$. Since $R1$ is directly connected to $M2$, this GRE tunnel is not required for our experiments. However, we include this tunnel to more closely emulate the network environment for which the system is targeted.

We performed experiments to test the inidividual MINC, reflected MINC and aggregate MINC methods. In the individual MINC case, $M1$ sends multicast packets addressed to group $G1$ over tunnel r. $M2$ joins group $G1$ over tunnel t and $M1$ joins group $G1$ over tunnel s. Thus, $R1$ forwards a copy of each multicast packet it receives to $M1$ and $M2$. In the reflected MINC experiments, $M1$ and $M2$ join group $G1$ over tunnels s and t, respectively, and receive packets transmitted by $M1$ over tunnel r. After receiving a packet, $M2$ transmits a second packet to group $G1$ over tunnel t. This packet is delivered to $M1$ over tunnel s. Finally, in the aggregate experiments, $R2$ is used in place of $M2$. $M1$ again joins group $G1$ over tunnel s. $R2$ also joins the multicast group. Packets transmitted by $M1$ over tunnel r are delivered to $M1$ and to $R2$. Netflow is enabled on $R2$ and produces summaries that are exported to a Netflow collector running on another host.

In each experiment, packets to and from $M1$ and $R1$ are subjected to loss at DN. For each algorithm, we run tests with a 5% target loss rate on the path $M1 \rightarrow R1$, a 5% target loss rate on the reverse link, and a 5% loss rate on both links simultaneously. Probe packets are sent every second. A total of 1100 probes are transmitted in all experiments; in the aggregate MINC experiments, this was done with 100 groups of 11 probes each. In each experiments the receipt of packets at the monitoring hosts and/or netflow records from the router are stored to a database and used to estimate the loss rate. These estimated values are compared to the actual probe loss rates. The actual probe loss rates are computed by using the *tcpdump* to capture all incoming and outgoing probe packets at each host network interface.

The testbed results are shown in Table 1. Each row contains results for a particular setting of the per-link loss rate probabilities in Dummynet. The first two columns show loss probabilities on the link from $M1$ to $R1$ and from $R1$ to $M1$, respectively. Each subsequent set of four columns shows the results for one of the MINC methods. Within each set of four columns, the first two columns show the actual loss rate on the two directions of the link, and the last two columns show the corresponding estimated loss rates. In all cases, the estimated loss rates match the actual loss rates very well.

These results are not suprising. After all, there is no correlation in the packet loss patterns generated by Dummynet. However, our prototype implementation and testbed results have demonstrated that we are able to build a system including all the components we specified and which produces results that are consistent with our earlier analytic results.

6. CONCLUSIONS

In this paper, we presented techniques for estimating one-way loss from a measurement host to network routers which exploit commonly implemented features on commercial routers and do not require any new router capabilities. There are three components to these techniques. Firstly, GRE tunneling is used to steer measurement traffic from a small number of measurement hosts down the paths followed by customer traffic. Secondly, innovative probing methods, coupled with standard measurement capabilities, such as NetFlow, are used to isolate the performance of groups of measurement packets. Thirdly, we exploit and extend tomographic inference methods in order to extract the performance of probe traffic on customer paths within the network. This combination yields a powerful yet lightweight method to determine customer performance within the network. We have implemented the measurement and inference algorithms in a prototype system and have tested the implementation in a research testbed.

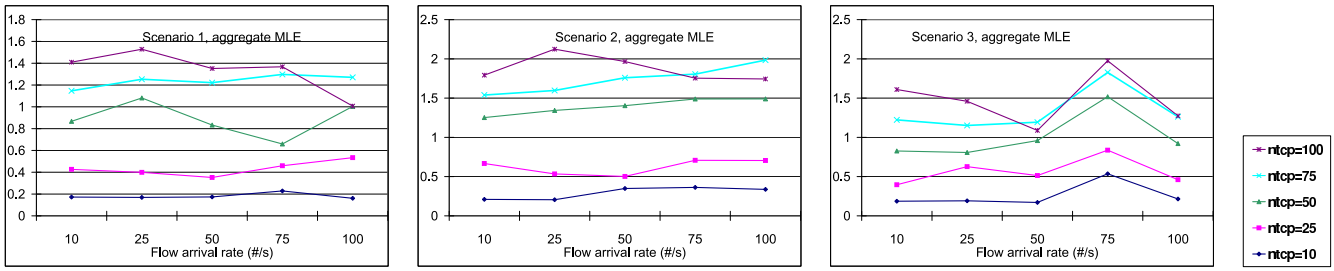

Figure 10: Quality Measure Error: RMS scaled error Q_1 on shared path for aggregate MLE method under three Scenarios 1,2,3.

Dummynet target loss settings		Individual MINC				Reflected MINC				Aggregate MINC			
		measured loss rate		estimated loss rate		measured loss rate		estimated loss rate		measured loss rate		estimated loss rate	
α_1	α_2	α_1	α_2	α_1	α_2	α_1	α_2	α_1	α_2	α_1	α_2	α_1	α_2
0.05	0	0.040	0	0.040	0	0.06	0	0.06	0	0.048	0	0.048	0
0	0.05	0	0.070	0	0.070	0	0.045	0.002	0.044	0	0.054	0	0.054
0.05	0.05	0.038	0.058	0.038	0.058	0.053	0.047	0.053	0.039	0.052	0.059	0.052	0.058

Table 1: Testbed experimental results. α_1 denotes path $M1 \rightarrow R1$. α_2 denotes path $R1 \rightarrow M1$.

Concering the performance of our estimators: the relative error of all methods seem acceptable for a feasible measurement bandwidth over a period of about a minute. The presence of temporal correlation in loss due to queueing and/or TCP source behavior does increase the variance of estimators than use multipacket events frequencies for estimation, but the resulting estimator accuracy is no worse than twice as bad expected. Spatial correlations due to background traffic processes, although having some impact on estimation accuracy, likewise did not unduly increase estimation variance. By contrast, topological bias can prevent all the estimators from providing any useful information.

Finally, we conclude by returning to a deployment challenge raised earlier. Recall from Section 4 that our loss estimates are vulnerable to topological bias resulting from a lack of congruence between the branch points in the logical and physical multicast trees. By exploiting information about network topology and routing, a network provider can place measurement hosts in such a way to avoid bias. Scaling concerns make deploying a measurement host for each router for which measurements are required impractical. However, we believe that a small number of measurement hosts can be positioned appropriately in a large service provider network, such that for all target routers, suitable measurement hosts which enable unbiased estimates of loss metrics are possible. Developiong methods and algorithms to achieve this, a subject for futher study, will enable general deployment and use of our measurement techniques in operational networks.

7. REFERENCES

[1] Anonymous.

[2] G. Appenzeller, I. Keslassy, and N. McKeown. Sizing router buffers, 2004.

[3] L. Breslau, C. Chase, N. Duffield, B. Fenner, Y. Mao, and S. Sen. Vmscope – a virtual multicast VPN performance monitor. *ACM SIGCOMM Workshop on Internet Network Management (INM)*, 2006.

[4] H. Burch and C. Chase. Monitoring link delays with one measurement host. *SIGMETRICS Performance Evaluation Review*, 33(3):10–17, 2005.

[5] R. Caceres, N. Duffield, J. Horowitz, and D. Towsley. Multicast-based inference of network-internal loss characteristic. *IEEE Transactions in Information Theory*, 45:2462–2480, 1999.

[6] J. Cao, W. Cleveland, Y. Gao, K. Jeffay, F. Smith, and M. Weigle. Stochastic models for generating synthetic http source traffic, 2004.

[7] Cisco Systems. Netflow. http://www.cisco.com/warp/public/732/netflow/index.html.

[8] A. B. Downey. Using pathchar to estimate internet link characteristics. In *ACM SIGCOMM 1999*, pages 241–250, Cambridge, MA, USA, September 1999.

[9] N. Duffield, V. Arya, R. Bellino, T. Friedman, J. Horowitz, D. Towsley, and T. Turletti. Network tomography from aggregate loss reports. In *Performance 2005*, 2005.

[10] N. Duffield, J. Horowitz, D. Towsley, W. Wei, and T. Friedman. Multicast-based loss inference with missing data. *IEEE Journal on Selected Areas in Communications*, 20:700–713, 2002.

[11] N. Duffield, C. Lund, and M. Thorup. Learn more, sample less: control of volume and variance in network measurement. *IEEE Transactions in Information Theory*, 51(5):1756–1775, 2005.

[12] D. Farinacci, T. Li, S. Hanks, D. Meyer, and P. Traina. Generic Routing Encapsulation (GRE). RFC 2784, 2000.

[13] W. Fenner. Internet Group Management Protocol, Version 2. RFC 2236, Nov. 1997.

[14] J. Mahdavi, V. Paxson, A. Adams, and M. Mathis. Creating a scalable architecture for internet measurement. In *INET '98*, Geneva, Switzerland, July 1998.

[15] V. N. Padmanabhan, L. Qiu, and H. Wang. Server-based inference of internet link lossiness. In *IEEE Infocom 2003*, San Francisco, CA, USA, April 2003.

[16] A. Reddy, D. Estrin, and R. Govindan. Fault isolation in multicast trees. In *ACM SIGCOMM 2000*, Stockholm, Sweden, August 2000.

[17] L. Rizzo. Dummynet: a simple approach to the evaluation of network protocols. *ACM Computer Communication Review*, 27(1):31–41, 1997.

[18] E. Rosen, Y. Cai, and I. Wijnands. Multicast in MPLS/BGP IP VPNs. Internet draft, Dec. 2004.

[19] E. Rosen and Y. Rekhter. BGP/MPLS Virtual Private Networks (VPNs). RFC 4364, Feb. 2006.

[20] Y. Tsang, M. Coates, and R. Nowak. Passive unicast network tomography using em algorithms. In *IEEE International Conference on Acoustics, Speech, and Signal Processing*, volume 3, pages 1469–1472, Salt Lake City, Utah, USA, May 2001.

[21] L. Zhang, S. Shenker, and D. D. Clark. Observations on the dynamics of a congestion control algorithm: the effects of two-way traffic. *SIGCOMM Comput. Commun. Rev.*, 21(4):133–147, 1991.