Detecting Shared Congestion of Flows Via End-to-End Measurement

Dan Rubenstein, *Member, IEEE*, Jim Kurose, *Fellow, IEEE*, and Don Towsley, *Fellow, IEEE*

***Abstract*—Current Internet congestion control protocols operate independently on a per-flow basis. Recent work has demonstrated that cooperative congestion control strategies between flows can improve performance for a variety of applications, ranging from aggregated TCP transmissions to multiple-sender multicast applications. However, in order for this cooperation to be effective, one must first identify the flows that are congested at the same set of resources. In this paper, we present techniques based on loss or delay observations at end hosts to infer whether or not two flows experiencing congestion are congested at the same network resources. Our novel result is that such detection can be achieved for unicast flows, but the techniques can also be applied to multicast flows. We validate these techniques via queueing analysis, simulation, and experimentation within the Internet. In addition, we demonstrate preliminary simulation results that show that the delay-based technique can determine whether two TCP flows are congested at the same set of resources. We also propose metrics that can be used as a measure of the amount of congestion sharing between two flows.**

***Index Terms*—Hypothesis testing, inference, network congestion, queueing analysis.**

I. Introduction

THE RECENT success of the Internet arguably stems from the philosophy that complexity should be relegated to the endpoints of the network. In the Internet, data is transmitted using only best-effort service, with reliability and congestion control being implemented only within the Internet's end systems. Current approaches to congestion control, such as those incorporated into TCP and those proposed for multicast congestion control, have a sender regulate its transmission rate *independently* from other senders, based on feedback (typically, loss indications) received from its receiver(s).

Recent work has demonstrated that *cooperative* congestion control strategies among different sessions or among different senders in a single session (in the case of multicast) can improve performance for a variety of applications, ranging from aggregated TCP transmissions to multiple-sender multicast applications.

- The benefits of performing congestion control over *flow aggregates* are explored in [1], [2]. Here, an aggregate consists of a set of flows that are treated as a single, virtual flow for the purposes of congestion control. For example, in the presence of contention, a WWW session with multiple on-going (TCP and/or continuous media) streams that interfere with each other over a common bottleneck might choose to optimize session utility by more drastically reducing the rate of one session in the face of congestion, while only slightly decreasing the rate of another. The server's aggregate session rate remains the same as if each session was treated as an isolated TCP session, but the rate of the individual sessions within the aggregate can vary (from what would be achieved under vanilla TCP) according to server policy.
- In many-to-one or many-to-many applications, a receiver within a single "session" may receive data from multiple senders. When a receiver detects congestion, the specific actions taken by the senders to reduce their transmission rates should depend upon whether or not the senders share a common resource bottleneck on the path to that receiver. Distributed gaming [3], teleconferencing, and accessing data in parallel from multiple mirror sites simultaneously [4], [5] are examples of such applications.

A key technical issue underlying both of these scenarios is the ability to detect whether two "flows" (whether individual unicast sessions or different senders within a single multicast session) share a common resource bottleneck. In this paper, we address the fundamental issue of detecting shared points of congestion among flows. Informally, the *point of congestion* (POC) for two flows is the same when the same set of resources (e.g., routers) are dropping or excessively delaying packets from both flows due to backup and/or overflowing of queues. We present two techniques that operate on an end-to-end basis and use only end-system observations to detect whether or not a pair of flows experiences a common POC, also referred to as a shared point of congestion (SPOC). One technique uses observations of packet losses to identify whether or not packets are being dropped at the same POC. A second uses observations of end-to-end delays, computed between end hosts whose clocks need not be synchronized, to identify whether or not packets are experiencing significant delays at the same POC. These techniques assume that the flows share a common end point, i.e., it is either the case that flow sources are co-located, or that flow receivers are co-located.

Manuscript received August 2, 2001; revised November 23, 2001; approved by IEEE/ACM TRANSACTIONS ON NETWORKING Editor S. Jamin. This work was supported in part by the National Science Foundation under Grant ANI-9805185, CDA-9502639, NCR 9523807, by the Defense Advanced Research Projects Agency (DARPA) under Grant N66001-97-C-8513, and by a gift from Sprint. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the National Science Foundation.

D. Rubenstein is with the Department of Electrical Engineering, Columbia University, New York, NY 10027 USA (e-mail: danr@ee.columbia.edu).

J. Kurose and D. Towsley are with the Department of Computer Science, University of Massachusetts, Amherst, MA 01003 USA (e-mail: kurose@cs.umass.edu; towsley@cs.umass.edu).

Publisher Item Identifier S 1063-6692(02)05226-3.

The key idea underlying the techniques presented in this paper is the fact that adjacent packets in the same flow experience some amount of positive correlation in loss and delay as they necessarily share any POCs. It follows that if two flows have the same POC, then adjacent packets in the two flows should similarly experience some amount of positive correlation. However, values of standard quantitative measures of correlation, such as correlation coefficients, depend on several factors, such as the rate of the flows, the amount of background (cross) traffic that passes through the flows' POCs, and the POCs' processing capabilities. Hence, the standard measures of correlation exhibited both within a flow and between flows that have the same POC differ under different network conditions. This makes it difficult to use these values directly to determine whether or not two flows share a common POC. Our novel insight is to construct a measure of correlation between flows and a measure of correlation within a flow with the following property: the measure between flows is greater than the measure within a flow if and only if the flows share the same POC. We call this method of identifying whether or not two flows share a POC a comparison test, and demonstrate how measures similar to those used within our comparison tests can also be used to estimate the "level" of sharing between two flows in cases where flows can have multiple POCs, some of which are shared, and some of which are not.

We first use traditional queueing models to prove that, in theory, our comparison tests can identify whether or not a POC is shared. Next, we use simulation to examine the performance of the comparison tests in more practical settings, where background traffic in the network consists of TCP and exponential on-off sources. We show that over time, (as the number of packet samples increases), the comparison tests always correctly identify whether or not the POC is shared, and that the techniques based on delay converge an order of magnitude faster than those based on loss. We also explore through simulation the accuracy of the techniques in detecting SPOCs when the network's routers deploy random early dropping (RED) [6], and where the probing flows are TCP flows. We find that the delay-based technique can still correctly infer whether the POC is shared by the flows, though the test must be run for a longer time to guarantee an answer within the same degree of accuracy. We also find that the loss-based tests do not perform well in such environments. Last, we demonstrate the performance of the tests in practice using actual network traces over simple topology configurations.

The work that most closely resembles our work presented here is that of Harfoush *et al.* [7], which presents an alternative loss-based technique to identify whether two flows share a common POC. The technique relies on the senders ability to transmit packet pairs, which restricts applicability of their technique to the case where the flow sources are co-located. They demonstrate that their technique converges to the correct result faster than an improved version of the technique presented here, but do not compare their technique to our delay-based technique. It can be inferred from their simulation results that our delay-based technique still converges faster than their loss-based technique. More recent work has looked at somewhat similar approaches to infer network tomography from unicast probes [8], [9]. In [10], the authors identify potential benefits of having separate end systems share locally observed statistics, such as available bandwidth and loss rate. It is observed in [11] that a comparison of IP addresses might be of assistance in determining which flows share bottlenecks, but the work subsequently states, "Determining a better estimate of which flows share a bottleneck is an open problem." While [1] and [2] demonstrate the value of performing congestion control over flow aggregates, [2] considers the detection of shared POCs to be future work, while the aggregated flows in [1] are limited to those having identical source-to-destination network paths. This significantly restricts the set of flows that can be aggregated. At a recent workshop, Padmanabhan [12] demonstrated that only flows sharing a point of congestion exhibit high correlation in packet delay, and hypothesized that this correlation could be used to make such a detection. An unpublished project report by Katabi *et al.* [13] presents a clever entropy-based technique to partition a set of unicast receivers at the same end system into clusters that share a common bottleneck. Their technique is very efficient in the number of packets needed to accurately perform the clustering, and is robust when the bandwidth to the end host constitutes at least 20% of the bandwidth at the bottleneck (i.e., light background traffic). In comparison, our loss-based techniques require more packet transmissions, but our delay-based techniques require a similar number of packet transmissions to Katabi's technique. Our techniques do not scale as easily to large receiver sets. However, our techniques remain robust under heavier background traffic loads, and can also detect shared POCs among flows in which the senders and not the receivers are co-located.

Our work differs significantly from previous work in that using multicast loss traces infers network characteristics, such as multicast tree topology and the loss rates on individual links within the network. The work by Ratnasamy *et al.* [14] and that of the MINC project [15] require transmission of multicast probes. Their approaches identify a shared POC among receivers receiving from a single source, relying on the fact that a multicast router forwards a packet on either all or none of the downstream links that are requesting the multicast transmission. These approaches are not designed for the case when flow senders are not co-located. Furthermore, because the end-to-end multicast route between a source and receiver can differ substantially from the unicast route between the same end points, results pertaining to shared POCs based on the multicast route need not apply to unicast traffic. More recently, they have extended their work to unicast using techniques [16] that are able to reconstruct multicast session topologies based on end-system observations of losses and delays experienced by unicast probes. The intuition as to why they could extend their work to within a unicast environment follows from the intuition as to why our techniques work.

There are several practical issues that we identify in this paper as open areas of research and do not solve; these require further consideration before our techniques can or should be applied within an operational network for the purposes of congestion control. Our goal in this paper is to make a fundamental first step in solving the problem of congestion control for aggregated streams.

The remainder of the paper proceeds as follows. Section II overviews the two testing techniques for performing the detection of a shared POC, and provides a high-level intuition as to why the techniques work. Section III presents queueing analyses that demonstrate the effectiveness of the tests using theoretical models of the POCs. Section IV presents simulation results that demonstrate the performance of the techniques under more realistic traffic conditions. Section V presents results of experiments conducted over the Internet. Section VI briefly discusses some open issues. Finally, Section VII concludes the paper.

II. Technique Description

In this section, we present two techniques, the *loss-corr* technique and the *delay-corr* technique, that use loss and delay measurements, respectively, at receivers to determine whether or not a pair of sessions (also called flows) have the same POC. The POC for a flow is the set of locations (routers) at which the flow's packets are lost or experience excessive queueing delay. We say we are *testing* two flows when we are trying to identify whether or not they have the same POC. For conciseness, we say that two flows *share congestion* if their POCs are identical, and that flows *do not share congestion* if the intersection of their POCs is empty. In this section, we assume that the flows' POCs are either identical or mutually exclusive, which means that the question, "Do flow A and flow B share congestion?" can be answered with a simple "yes" or "no." Later in the paper, we address how to handle cases where two flows' POCs can partially overlap.

We emphasize that we assume *a priori* that both sessions are experiencing congestion. We assume that another testing method is first used to determine that each of the pair of sessions being considered is congested, such as observing the loss rate or expected delay exceeding a threshold. Once a conclusion has been reached that both sessions are congested, our test can be applied to determine whether or not this congestion emanates from the same set of network points.

Our findings are that the delay-corr technique converges in much less time to the correct hypothesis than the loss-corr technique. However, there are two reasons why an application might prefer to use a technique that generates estimates using only loss statistics.

- The delay-corr technique requires time stamping of packets. We have noticed in our experimental results that performing the time stamping at the user level is sufficient, but becomes less reliable if the hosts are heavily loaded. Thus, the delay-corr technique requires more resources than the loss-corr technique.
- Heavy delay congestion is likely to manifest itself in routers with larger queues, whereas heavy loss congestion is likely to manifest itself in routers with smaller queues. While we suspect that the POC is often the same for both forms of congestion, this need not be the case. Thus, the best way to determine that the POC that causes loss is shared is to apply the loss-corr technique (and wait the extra time). Similarly, the best way to ensure that the POC that causes delay is shared is to apply the delay-corr technique (and use the additional resources).

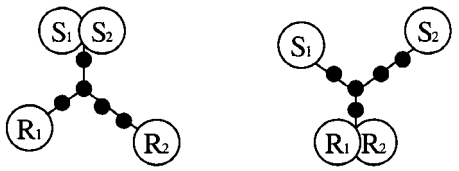

Fig. 1. Virtual topologies. (a) Inverted-Y topology. (b) Y topology.

We consider only topologies in which either the pair of senders or the pair of receivers of both flows are co-located at the same host. This assumption does restrict the set of pairs that can be considered. However, as compared to a randomly chosen pair of flows for which neither the senders nor the receivers are co-located, flows that have at least one set of co-located hosts 1) are easily located from the point of co-location, 2) are more likely to share congestion, since portions of their paths are guaranteed to overlap, and 3) require less communication overhead (i.e., they can communicate over a LAN) to perform aggregated congestion control.

Fig. 1 gives a pictorial representation of sample topologies formed from the paths of the two flows with co-located hosts. S_1 and S_2 are the senders of the two flows, R_1 and R_2 are the two receivers, and the filled circles are routers at the intermediate hops. In the *Inverted-Y* topology [Fig. 1(a)], the senders are co-located. Packets transmitted by the senders traverse a set of common links up to some point in the network, after which the flows travel along separate paths. In the Y topology [Fig. 1(b)], the receivers are co-located. Packets transmitted by the senders initially traverse a separate set of links. At some point along each flow's data-path, the flows meet and the remaining path to the receivers is identical.

A shared POC exists if congestion occurs along the top portion of the inverted-Y topology, or along the bottom portion of the Y topology. We assume that in the Y (Inverted-Y) topology, after the flows' paths are joined (deviate), they do not deviate (re-join). Otherwise, the order of packet arrivals (departures) could differ substantially from what is observed at a shared POC. Note that if a pair of flows can be mapped onto either of these two topologies, then (barring reordering) we can observe, from the point of co-location, the order in which packets pass through the shared POC, if it exists. This allows us to infer whether or not the flows share congestion using only information that can easily be monitored at the three end-system locations. Hence, the techniques do not require any information pertaining to router processing rates, link speeds, or traffic patterns of any background traffic.

Let us now formalize the notation that will be used throughout the paper to refer to the packet flows. Let f_1 and f_1 represent the two flows that we are testing. Each of these flows is referred to as a *foreground flow*, and we refer to the packets within the flows as *foreground transmissions*. Any other traffic/packet in the network that does not belong to either of these flows is referred to as *background* traffic. Let $p_{1,i}$ represent the ith packet transmitted by f_1, and $p_{2,i}$ represent the ith packet transmitted by f_2. We write the jth foreground packet transmitted (counting packets in both flows) as p_j, i.e., for each p_j, there is some i where either $p_j = p_{1,i}$, or $p_j = p_{2,i}$.

Input:	Trace information from the two flows
Step 1:	Compute the *cross-measure*, M_x, between pairs of packets in both flows, spaced apart by time t.
Step 2:	Compute the *auto-measure*, M_a from packets within a flow, spaced apart by time $T > t$.
Step 3:	If $M_x > M_a$, then the flows share a POC. Else, the flows do not share a POC.

Fig. 2. Comparison test.

Finally, we define a function that allows us to identify the *adjacency* of two packets in the foreground. For any two packets, p_a and p_b, from either flow, f_1 or f_2, we define the function $a(p_a, p_b) = 1$ if $b = a+1$, and 0 otherwise. $a(p_a, p_b)$ indicates whether or not two foreground packets are adjacent with respect to the other foreground packets. In other words, $a(p_{1,i}, p_{2,j}) = 1$ implies that there is some k for which $p_{1,i} = p_k$ and $p_{2,j} = p_{k+1}$.

A. Comparison Tests

Our techniques for detecting whether or not a pair of flows share congestion are based on two fundamental observations of Internet congestion.

- Losses or delays experienced by two packets passing through the same POC exhibit some degree of positive correlation (i.e., a loss or excessive delay observed by a packet increases the likelihood that a later packet will be lost or experience a large delay). However, in general, the degree of correlation decreases as the time between the packets' transmissions is increased [17], [18].
- The losses or delays experienced by two packets that do not share the same POC will exhibit little or no correlation.

Our idea is to measure the correlation between pairs of packets both within a flow, and between flows. We choose the pairs between flows such that if the POC for the flows is shared, then on average, the time between arrivals at the POC of packets in the between-flow pair is less than the time between arrivals at the POC of packets of a single flow. Hence, the between-flow pairs will experience higher levels of (positive) correlation if the POC for the flows is shared. If it is not shared, then the between-flow pairs will exhibit no correlation, and the level of correlation will be higher for the single-flow pairs. We refer to this simple method of making this determination as a *comparison test*. The basic steps are reiterated in Fig. 2. We refer to M_x, the measure of correlation between the flows, as the *cross-measure* (as in cross-correlation), and M_a, the measure of correlation within a flow, as the *auto-measure* (as in auto-correlation).

The benefit of using a comparative test is that it gives a definitive answer as to whether or not the flows share, regardless of what the specific values of the cross- and auto-measures are. Alternatively, one could construct measures that indicate congestion when taking on certain values (e.g., a correlation coefficient that is larger than some fixed value, α). Often, the value for α depends on several factors, including the service rate of the queues in the network, and the rate of the probe traffic, making a unique value for α unlikely.

B. Poisson Probes

We have noted that we need a method to generate packet samples in such a way that the average time of arrival at a shared POC (if it exists) between a sample pair from separate flows is less than that between a sample pair of packets from the same flow. To simplify presentation, we consider a single method for transmitting probes that is robust over both the Inverted-Y and Y topologies. The method we use, commonly referred to as a *Poisson probe*, is a flow whose inter-packet departure times are described by a Poisson process. We represent the rate of f_1's process by λ_1, and the rate of f_2's process by λ_2. We assume in our analysis that the transmission and queueing delays between the source and the POC do not significantly change the inter-packet spacing, and thus the arrival process at the POC can be modeled as Poisson with respective arrival rates of λ_1 and λ_2. We note that the aggregate arrival process formed by combining these two Poisson processes is itself a Poisson process with rate $\lambda_1 + \lambda_2$. The length of time between the arrival at the POC of two adjacent packets, p_i and p_{i+1}, from this aggregate process of rate $\lambda_1 + \lambda_2$ is on average smaller than the time interval between two successive packets from a single flow (e.g., $p_{2,j}$ and $p_{2,j+1}$) transmitted at rate $\lambda_2 < \lambda_1 + \lambda_2$.[1] Furthermore, because the aggregate process is Poisson, the distribution of the time interval between the adjacent packets is independent of the packets' flow origins (i.e., whether they came from f_1 or f_2). It follows that the average time interval between the arrival of two adjacent packets from different flows is less than that between two successive packets within a single flow.

In the remainder of this section, we describe how to compute measures of M_x and M_a using loss and delay measurements obtained from using Poisson probes. We conjecture that these measures work for other probe distributions, and thus in many cases, the measures can be applied *in-band*, i.e., the probes can be incorporated into the underlying data stream. However, it is likely that the techniques are not robust for all possible distributions of traffic. One example is when each flow transmits packets in groups (i.e., bursty traffic), that places packets within a single flow very close together. In such cases, these techniques can still be applied by transmitting a Poisson probe *out-of-band*, alongside each of the two data flows. Results presented later in this paper demonstrate that the detection of a shared POC can be done efficiently in practice using a total probing bandwidth of one kilobyte per second.

C. Loss-Corr Technique

The loss-corr technique is based on the intuitive notion that if two packets proceed through the same bottleneck, and the first packet is dropped, then the likelihood of the second packet being dropped becomes higher as the time between the packets' arrivals to the bottleneck is decreased. Define L_i to be 0 if p_i is dropped prior to reaching the destination host to which it was sent, and 1 if it is received at its destination. Define $L_{j,i}$ similarly, to indicate whether or not packet $p_{j,i}$ reaches the receiving host of f_j, where $j = 1, 2$.

[1]Note that a pair of successive packets within a flow need not be adjacent, e.g., packets from f_1 may arrive between arrivals of successive packets $p_{2,j}$ and $p_{2,j+1}$.

For the Inverted-Y topology, the loss-corr cross-measure and auto-measure are the following conditional probabilities:

$$M_x = \Pr(L_{2,i} = 0 \,|\, L_{1,j} = 0,\, a(p_{1,j}, p_{2,i}) = 1) \quad (1)$$
$$M_a = \Pr(L_{2,i} = 0 \,|\, L_{2,i-1} = 0). \quad (2)$$

The cross-measure we use for the Inverted-Y topology is the conditional probability that a packet from f_2 is lost, given that the preceding foreground packet was from f_1 and was lost. The auto-measure is the conditional probability that a packet from f_2 is lost given that the previous packet from f_2 is lost.

In the Inverted-Y topology, we have utilized the fact that the relative order in which lost packets arrive at the POC can be identified from the co-located sending end systems: even when $L_{2,i} = 0$ and $L_{1,j} = 0$, it is always possible to determine whether or not $a(p_{1,j}, p_{2,i}) = 1$. In the Y-topology, this is not the case. For instance, a received sequence of $p_{1,j}, p_{2,i}, p_{2,i+2}, p_{1,j+2}$ implies that packets $p_{1,j+1}$ and $p_{2,i+1}$ were lost. However, one cannot determine from these measurements whether $p_{1,j+1}$ preceded $p_{2,i+1}$ (or whether $p_{1,j+1}$ preceded $p_{2,i}$, etc.)[2] It follows that co-located receiving hosts cannot determine whether or not $a(p_{1,j}, p_{2,i}) = 1$ when both $p_{1,j}$ and $p_{2,i}$ are lost. As a consequence, we cannot compute the cross-measure defined by (1).

Instead, we define another cross-measure that can be computed by end hosts configured in a Y-topology, and another auto-measure that, when compared to this cross-measure, meet the requirements of the comparison test. We define $a_R(p_i, p_j)$ such that $a_R(p_i, p_j) = 1$ if and only if $i < j$, $L_i = 1$, $L_j = 1$, and $L_k = 0$ for all $i < k < j$, and let $a_R(p_i, p_j) = 0$ otherwise. In other words, $a_R(p_i, p_j)$ is 1 if and only if p_i and p_j are adjacently received packets (i.e., p_k is lost for any $i < k < j$). The cross-measure and auto-measure for the Y topology are the following conditional probabilities:

$$M_x = \Pr(L_{2,i-1} = 0 \,|\, L_{1,j-1} = 0,\, a_R(p_{1,j}, p_{2,i}) = 1) \quad (3)$$
$$M_a = \Pr(L_{2,i} = 0). \quad (4)$$

M_x is the conditional probability that for any i, a packet, $p_{2,i-1}$, from f_2 is lost, given that 1) the subsequent packet from f_2, $p_{2,i}$, is received, 2) the nearest foreground packet that is subsequently received after $p_{2,i}$ is from f_1 ($p_{1,j}$ for some j), and 3) that the preceding packet from f_1, $p_{1,j-1}$, is lost. The reader should note that the sequence of events used in (3) can be identified at the co-located receivers in the Y-topology: the sequence "pivots" on a pair of received packets to detect a pair of lost packets that are likely to be adjacent. M_a is the loss rate experienced by f_2. We note that this version of M_a is itself not a measure of correlation, but we find that its value is smaller than that of (3) only when the POCs are shared.

D. Delay-Corr Technique

The delay-corr technique applies the *correlation coefficient* to the delays experienced by receivers. For a set of pairs of real valued numbers, $S = \{(x_i, y_i)\}$, $x_i, y_i \in \Re$, the correlation coefficient of the set is defined as

$$C(S) = \frac{E[x_i y_i] - E[x_i]E[y_i]}{\sqrt{(E[{x_i}^2] - E^2[x_i])(E[{y_i}^2] - E^2[y_i])}} \quad (5)$$

where $E[f(x_i)] \equiv \sum_{(x_i, y_i) \in S} f(x_i)/|S|$ and $E[f(y_i)] \equiv \sum_{(x_i, y_i) \in S} f(y_i)/|S|$. Define D_i to be the *observed delay* incurred by packet i. $D_i = a_i - d_i$, where d_i is the departure time of p_i according to the sender's clock, and a_i is its arrival time according to the receiver's clock. Note that because of unsynchronized clocks and/or clock drift, the observed delay we compute need not equal the true time elapsed between the packet's departure from the sender and its arrival at the receiver. The lack of time synchronization between clocks does not affect the value of the correlation coefficient: the correlation coefficient of two random variables, X and Y, is the same as that between $X + c$ and Y when c is a constant. A large skew in the clock rates can alter the effectiveness of using the correlation coefficient of delay over long traces. However, efficient algorithms for removing clock skew from long traces are known [19], [20]. Henceforth, we simply refer to the observed delay as the delay.

We similarly define $D_{j,i}$ to be the respective delays of $p_{j,i}$, $j = 1, 2$. For both the inverted-Y and Y topologies, M_x and M_a are computed as

$$M_x = C(\{(D_{1,i}, D_{2,j}) \colon a(p_{1,i}, p_{2,j}) = 1\}) \quad (6)$$
$$M_a = C(\{(D_{2,i}, D_{2,i+1})\}). \quad (7)$$

M_x is the correlation coefficient computed from the delays of pairs of packets that are adjacent with respect to the foreground flows. The previously arriving (transmitted) packet must be from f_1, and the subsequent packet must be from f_2. M_a is the correlation coefficient computed from the delays between arrivals (transmissions) within f_2 that are adjacent with respect to packets in f_2.

III. Queueing Analysis

In this section, we demonstrate the correctness of the comparison tests described in Section II in the context of various queueing models. We assume that the time between transmissions for each of the foreground flows, f_1 and f_2, are described by Poisson processes with rates of λ_1 and λ_2, respectively.

Fig. 3 depicts our models of a shared POC for flows f_1 and f_2, and separate POCs for the flows. A POC is represented by a queue. A shared POC [Fig. 3(a)] is represented by a single queue; packets from both of the foreground flows enter this queue at respective rates λ_1, and λ_2. Additionally, background traffic enters the queue at a rate of λ_b. The queue services packets at a rate of μ. Separate POCs [Fig. 3(b)] are represented by two queues. Packets from f_i enter a queue whose background traffic arrival rate is λ_{b_i} and whose service rate is μ_i, $i = 1, 2$. Each packet that proceeds through the queueing system is serviced by only one of the two queues (e.g., packets from f_1 do not previously or subsequently proceed through the queue servicing packets from f_2). There are no restrictions on

[2]It may be possible to predict the more likely case by looking at inter-packet spacing within a flow. However, packets can experience unpredictable delays (jitter) that would make such estimation less reliable.

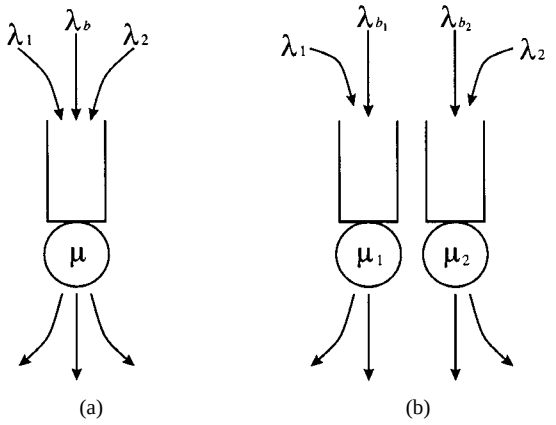

Fig. 3. Queueing models for shared and separate POCs. (a) Shared. (b) Separate.

any of the rates (foreground rates can differ from one another; in the two-queue case, background flow rates can differ in the two queues). Unless specifically stated otherwise, background traffic arrivals and queue service completions are described by any general i.i.d. distribution.

In the next subsection, we prove that, given the queues are all $M/M/1/K$ queues (where the buffer size K can differ among the various queues as well), the loss-corr technique correctly identifies whether or not the foreground flows share a POC in the inverted-Y topology. We do not have a proof that the loss-corr technique correctly identifies whether or not two flows share in the Y topology. However, we have formulated a set of recursive equations that allow us to compute the steady-state values of (3) and (4) as functions of λ_1, λ_2, λ_b, and K, when the POC is shared and behaves as an $M/M/1/K$ queue. We then compared the values of these equations for a variety of values of λ_1, λ_2, λ_b, and K, and found (3) to always be larger than (4) (the desired result). These results are presented in [21].

In the following subsection, we demonstrate that, given all queues are $M+G/G/1/\infty$ queues (foreground traffic remains Poisson, background traffic and service times satisfy any i.i.d. general distribution), the delay-corr technique successfully distinguishes between shared and separate POCs for both the Y and Inverted-Y topologies. Since the queue's capacities are unbounded, the proof requires the additional assumption that the aggregate rate of traffic into any of the queues is less than the processing rate for that queue.

A. Loss-Corr Technique, Inverted-Y Topology

We write q_i, $i = 1, 2$ to represent two $M/M/1/K$ queues. We define ω to be a sequence of insert and remove events, $\omega = \langle e_1, e_2, \ldots, e_m \rangle$, and let $Q_i(\omega, j)$ be the number of packets in q_i after in-order application of events $e_1, \ldots, e_j$ to the queue. We write $Q_i(\omega, 0)$ to be the number of packets in the queue prior to the application of ω. We assume that the system has been in operation for some time when ω is applied to the queue so that it need not be the case that $Q_i(\omega, 0) = 0$. An insert event increases the queue length by one unless already full, and a remove event decreases the queue length by one unless it is already empty.

Lemma 1: Consider two queues, q_1 and q_2, of identical buffer capacities, K. If $Q_1(\omega, 0) \leq Q_2(\omega, 0)$, then $Q_1(\omega, j) \leq Q_2(\omega, j)$ for all $j > 0$ as well.

Lemma 1 can be proven trivially by induction over the length of the sequence ω. The proof is omitted.

Lemma 2: Consider a queue q_1 of capacity K where $Q_1(\omega, 0) = K$ (the queue is full). Let ω' be a suffix sequence of ω, i.e., $\omega' = (f_1, f_2, \ldots, f_{m'})$ where for some $i \geq 1$, $m' = m - i + 1$ and $f_j = e_{j+i-1}$ where $1 \leq j \leq m'$. Then $Q_1(\omega', j) \geq Q_1(\omega, j+1)$.

Proof: Consider the application of ω to the queue. After applying the (possibly empty) prefix $(e_1, \ldots, e_{i-1})$ to the queue, it must be the case that $Q_1(\omega, i-1) \leq K$. The result then follows from Lemma 1, since the remaining sequence of ω to be applied is ω', hence $Q_1(\omega, i-1+j) \leq Q_1(\omega', j)$ for $0 \leq j \leq m - i + 1$. ■

Lemma 2 states that for arbitrary sequences of events ω and ω', the application of ω to a full queue will result in a queue whose height is less than or equal to that of a full queue to which ω' is first applied, followed by ω. Intuitively, this is because application of ω' can only reduce the height of the queue from its original full position. The result then follows from Lemma 1.

Theorem 1: In an $M/M/1/K$ in which both foreground flows enter into the same queue, $\Pr(L_{2,j} = 0 \,|\, (L_{1,i} = 0), a(p_{1,i}, p_{2,j})) > \Pr(L_{2,j+1} = 0 \,|\, L_{2,j} = 0)$ (i.e., $M_x > M_a$).

An intuitive approach to proving this theorem would be to use a sample-path argument. Consider any sequence ω that contains a pair of arrivals from f_2, such that the sequence provides a sample point used in computing the conditional probability, $\Pr(L_{2,j+1} = 0 \,|\, L_{2,j} = 0)$. We wish to construct 1–1 mapping that maps each such sequence to a sequence ω' that provides a sample point for $\Pr(L_{2,j} = 0 \,|\, (L_{1,i} = 0), a(p_{1,i}, p_{2,j}))$, such that ω' yields a "positive" sample (i.e., $L_{2,j} = 0$ and $L_{1,i} = 0$) whenever ω yields a "positive" sample (i.e., $L_{2,j} = 0$ and $L_{2,j+1} = 0$) such that ω' occurs with higher probability measure (conditioned on $L_{1,i} = 0$) than does ω (conditioned on $L_{2,j} = 0$). An intuitive mapping is one in which ω' equals suffix of ω that starts from the last arrival of a packet from f_1, when one exists. The problem with this mapping does not cover those sequences ω that do not contain an arrival from f_1, and we are unable to identify an appropriate 1–1 mapping to complete the proof in this manner. Instead, we resort to an alternative approach that uses similar intuition, though in a less straightforward manner.[3]

Proof: Let $\omega = \langle e_1, \ldots, e_{m_\omega} \rangle$ be a finite-length sequence of events, each $e_i \in \{1, 2, b, s\}$, where $e_i = 1$ means that the ith event is an arrival from f_1, $e_i = 2$ means that the ith event is an arrival from f_2, $e_i = b$ means that the ith event is a background arrival, and $e_i = s$ means that the ith event is a service completion (this event has no effect on the queue if the queue is already empty).

Let $S = \{\omega\}$ be the set of all possible finite-length sequences of events. Define a function $g_p(\omega) = \langle e_1, \ldots, e_n \rangle$ where $n \leq m_\omega$, $n = 0$ or $e_n = 1$ where $e_i \neq 1$ for $n < i \leq m_\omega$. In other words, $g_p(\omega)$ is the longest prefix of ω whose last event is an

[3]The more formal proof presented here corrects an oversight of the proof that appeared in [21], [22].

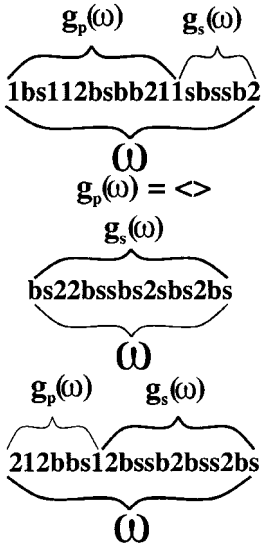

Fig. 4. Decompositions of three samples of ω into $g_p(\omega)$ and $g_s(\omega)$.

arrival from f_1. Note that if ω contains no $e_i = 1$, then $g_p(\omega)$ is the empty sequence. Let $g_s(\omega)$ be the longest suffix of ω that contains no $e_i = 1$. i.e., $g_s(\omega) = (e_{n+1}, \ldots, e_{m_\omega})$ where $n = 0$ or else $e_n = 1$, $e_i \neq 1$ for $n < i \leq m_\omega$. Note that each sequence ω has a unique decomposition as $\omega = g_p(\omega) \cdot g_s(\omega)$, where $\cdot$ is the concatenation operation. Such a decomposition is demonstrated in Fig. 4.

Define P to be the probability measure over S.[4] This is well defined since all events are generated from a Poisson process, so the measure of a sequence is independent of any previous history (previous arrivals, state of the queue). Furthermore, it follows from the Poisson assumption that the measures of prefixes and suffixes are independent and satisfy $P(\omega) = P(g_p(\omega))P(g_s(\omega))$.

We now define several random variables that will allow us to formally describe the conditional probabilities stated in the theorem over the set of sequences in S. Define X to be a random variable on S where $X(\omega) = 1$ if e_{m_ω}, the last event in ω, is the first (and only) arrival from f_2 in ω and 0 otherwise. Define X_p to be a random variable on S where $X_p(\omega) = 1$ if ω is the empty sequence or is a sequence in which $e_i \neq 2$ for all $1 \leq i \leq m_seq$ and $e_{m_seq} = 1$, and equals 0 otherwise. Define X_s to be a random variable on S where $X_s(\omega) = 1$ if ω contains no event $e_i = 1$, and only the last event, e_{m_ω}, is an arrival from f_2. Note that $\forall \omega \in S$, $X(\omega) = X_p(g_p(\omega))X_s(g_s(\omega))$. Also note that for any $\omega \in S$ where $X(\omega) = 1$, there is a unique pair, $\omega_1, \omega_2 \in S$, where $\omega = \omega_1 \cdot \omega_2$ and $X_p(\omega_1)X_s(\omega_2) = 1$. Namely, $\omega_1 = g_p(\omega)$ and $\omega_2 = g_s(\omega)$.

In addition, we note that $\sum_{\omega \in S} X(\omega)P(\omega) = 1$. This is because the set of finite sequences that yield nonzero terms in the sum are all and only those in which the last event is the first arrival from f_2. Hence, any infinite-length sampling that contains an arrival from f_2 is prefixed by exactly one member of the set, and the set of infinite-length sequences that do not contain an arrival from f_2 have measure zero. This yields

$$\begin{aligned}
&\sum_{\omega_s \in S} X_p(\omega_s)P(\omega_s) \sum_{\omega_p \in S} X_s(\omega_p)P(\omega_p) \\
&\quad = \sum_{\omega_s \in S} \sum_{\omega_p \in S} X_p(\omega_s)P(\omega_s)X_s(\omega_p)P(\omega_p) \\
&\quad = \sum_{\omega \in S} X_p(g_p(\omega))X_s(g_s(\omega))P(\omega) \\
&\quad = \sum_{\omega \in S} X(\omega)P(\omega) = 1. \qquad (8)
\end{aligned}$$

Define L_K to be random variable on S where for $\omega \in S$, $L_K(\omega) = 1$ if the last event of ω is a packet arrival, and applying ω to a queue of capacity K whose buffer is initially full causes this last arrival to be dropped (i.e., the queue is full upon its arrival). It follows from Lemma 2 that $L_K(\omega) = 1 \Rightarrow L_K(g_s(\omega)) = 1$, in other words, $\forall \omega_p, \omega_s \in S$ we have $L_K(\omega_p \cdot \omega_s) \leq L_K(\omega_s)$. We make use of the PASTA property [24] that the first event "of interest" [arrival of $p_{2,j}$ for $\Pr(L_{2,j+1} = 0, L_{2,j} = 0)$, $\Pr(L_{2,j} = 0)$, $p_{1,i}$ for $\Pr(L_{2,j} = 0 \,|\, L_{1,i} = 0, a(p_{1,i}, p_{2,j}) = 1)$] is Poisson and hence the distribution of the queue's height is described by its steady state distribution. Defining π_i to be the steady-state probability that the queue length is i, we have

$$\Pr(L_{2,j+1} = 0, L_{2,j} = 0) = \pi_K \sum_{\omega \in S} P(\omega)X(\omega)L_K(\omega) \qquad (9)$$

$$\Pr(L_{2,j} = 0) = \pi_K. \qquad (10)$$

We can rewrite the conditional probability, $\Pr(L_{2,j+1} = 0 \,|\, L_{2,j} = 0)$, as

$$\begin{aligned}
&\Pr(L_{2,j+1} = 0 \,|\, L_{2,j} = 0) \\
&\quad = \sum_{\omega \in S} P(\omega)X(\omega)L_K(\omega) \\
&\quad = \sum_{\omega_p \in S} \sum_{\omega_s \in S} P(\omega p)P(\omega s)X_p(\omega p)X_s(\omega s)L_K(\omega p \cdot \omega s) \\
&\quad \leq \sum_{\omega_p \in S} \sum_{\omega_s \in S} P(\omega p)P(\omega s)X_p(\omega p)X_s(\omega s)L_K(\omega s) \qquad (11) \\
&\quad = \left(\sum_{\omega_p \in S} P(\omega p)X_p(\omega p) \right) \\
&\qquad \cdot \left(\sum_{\omega_s \in S} P(\omega s)X_s(\omega s)L_K(\omega s) \right) \qquad (12) \\
&\quad = \frac{\sum_{\omega_s \in S} L_K(\omega s)X_s(\omega s)P(\omega s)}{\sum_{\omega_s \in S} P(\omega s)X_s(\omega s)} \qquad (13) \\
&\quad = \frac{\sum_{\omega_s \in S} \pi_K P(\omega s)L_K(\omega s)X_s(\omega s)}{\sum_{\omega_s \in S} \pi_K P(\omega s)X_s(\omega s)} \\
&\quad = \Pr(L_{2,j} = 0 \,|\, L_{1,i} = 0, a(p_{1,i}, p_{2,j} = 1) \qquad (14)
\end{aligned}$$

where we use $L_K(\omega p \cdot \omega s) \leq L_K(\omega s)$ to establish the inequality in (11), and (8) gives the equality between (12) and (13). The

[4]We emphasize that P is a probability *measure* [23] and not a probability distribution. Note also that S is a countable set, so that the measure of a set $S' \subset S$ that contains a set of sequences, where no sequence in S' is a subsequence of another $\omega \in S'$, is simply $\sum_{\omega \in S'} P(\omega)$.

inequality is strict since there exists at least one $\omega = \omega p \cdot \omega s$ where $L_K(\omega) < L_K(\omega s)$ and $X_p(\omega p)X_s(\omega s) \neq 0$. ■

Theorem 2: In two $M/M/1/K$ queues in which the foreground flows enter separate queues, it is the case that $\Pr(L_{2,j} = 0 \,|\, (L_{1,i} = 0),\, a(p_{1,i}, p_{2,j})) < \Pr(L_{2,j+1} = 0 \,|\, L_{2,j} = 0)$ (i.e., $M_x < M_a$).

Proof: Arrivals (departures) to (from) the first queue have no impact on the second queue, and can be ignored when considering the status of the second queue. Because all arrivals and departures from the queues are Poisson, by PASTA [24], $\Pr(L_{2,j} = 0 \,|\, L_{1,i} = 0,\, a(p_{1,i}, p_{2,j})) = \Pr(L_{2,j} = 0)$ for any packet in f_2. Thus, we need only prove that $\Pr(L_{2,j} = 0) < \Pr(L_{2,j+1} = 0 \,|\, L_{2,j} = 0)$.

We prove this by a sample path argument. Similar to Theorem 1, we define $S = \{\omega\}$ to be the set of all possible finite-length sequences through the queue. Since packets from f_1 pass through a separate queue, each event, e_i of $\omega = \langle e_1, e_2, \ldots, e_{m_\omega}\rangle$ is chosen from $\{2, b, s\}$. Define P to be the probability measure over S (again this is well defined due to the memoryless nature of the Poisson distribution).

Define X to be a random variable on S as in Theorem 1: $X(\omega) = 1$ when the first and only arrival from f_2 is the last event, e_{m_ω}, in the sequence, and 0 otherwise. Define Y_i to be a random variable on S where $Y_i(\omega) = 1$ if applying the sequence, $\omega = \langle e_1, \ldots, e_{m_\omega}\rangle$, to the queue with initial length $i \leq K$ causes the last event, e_{m_ω} to result in a packet drop, and 0 otherwise.

$\Pr(L_{2,j} = 0)$ can also be obtained by picking an arbitrary point in time (such that the queue is in steady state) and considering the sequence leading up to the first arrival of a packet from f_2.

$$\Pr(L_{2,j} = 0) = \sum_{\omega \in S} \sum_{i=0}^{K} \pi_i P(\omega) X(\omega) Y_i(\omega). \tag{15}$$

We compute $\Pr(L_{2,j+1} = 0 \,|\, L_{2,j} = 0)$ by considering sequences that start at the point in time of the arrival of a packet from f_2.

$$\Pr(L_{2,j+1} = 0 \,|\, L_{2,j} = 0) = \frac{\sum_{\omega \in S} \pi_K P(\omega) X(\omega) Y_k(\omega)}{\pi_K} = \sum_{\omega \in S} P(\omega) X(\omega) Y_k(\omega). \tag{16}$$

We note that for any ω where $X(\omega) = 1$ and any $i,\ j$ such that $0 \leq j < i \leq K$, it follows from Lemma 1 that $Y_j(\omega) \leq Y_i(\omega)$. In particular, there is some ω for which $X(\omega) = 1$ where for some i, $Y_i(\omega) = 0$ while $Y_k(\omega) = 1$. Also, since $\sum_{i=0}^{K} \pi_i = 1$, we get

$$\sum_{\omega \in S} \sum_{i=0}^{K} \pi_i P(\omega) X(\omega) Y_i(\omega) < \sum_{\omega \in S} \sum_{i=0}^{K} \pi_i P(\omega) X(\omega) Y_k(\omega) = \sum_{\omega \in S} P(\omega) X(\omega) Y_k(\omega).$$

Applying this inequality to (15) and (16) completes the proof.

B. Delay-Corr Technique: Inverted-Y and Y Topologies

We now demonstrate that the delay-corr technique will correctly infer whether or not the two flows share in a queueing system where the background traffic arrives according to an arbitrary, ergodic, and stationary process, and the service times are characterized by an arbitrary distribution. We do require that the random variables that represent the background traffic and service times be i.i.d. The analysis also assumes that the system has entered into the stationary regime, i.e., the system is initially in the steady state.

Our arguments rely on the following technical lemma that is established in [21].

Lemma 3: Let G be a nondecreasing function over $[0, \infty)$, where $\lim_{x\to\infty} G(x) > G(0) > 0$, and let f and g be functions such that $\int_{x=0}^{\infty} f(x)\,dx = \int_{x=0}^{\infty} g(x)\,dx$, $\int_{x=0}^{\infty} G(x)f(x)\,dx < \infty$, $\int_{x=0}^{\infty} G(x)g(x)\,dx < \infty$, and there is some γ such that for $x < \gamma$, $f(x) > g(x)$, and for $x > \gamma$, $f(x) < g(x)$. Then $\int_{x=0}^{\infty} G(x)\,f(x)\,dx < \int_{x=0}^{\infty} G(x)g(x)\,dx$. Similarly, if G is nonincreasing with $0 < \lim_{x\to\infty} G(x) < G(0)$, then $\int_{x=0}^{\infty} G(x)f(x)\,dx > \int_{x=0}^{\infty} G(x)g(x)\,dx$.

The following lemma implies that the delay correlation between two adjacent foreground packets is higher than that between two nonadjacent foreground packets. Its proof appears in the Appendix.

Lemma 4: Consider an $M+G/G/1$ server (infinite capacity queue) where background traffic arrives with an aggregate arrival rate of λ_b, foreground traffic arrives according to a Poisson process with rate λ_f, and packets are served at an average rate of $\mu > \lambda_b + \lambda_f$. Then $E[D_iD_{i+1}] > E[D_iD_{i+n}]$ for $n > 1$.

Armed with this lemma, we can now prove the result that $M_x > M_a$ when the POC for both flows is the same $M + G/G/1$ queue.

Theorem 3: Consider the same $M + G/G/1$ queue as in Lemma 4, where the foreground flow consists of packets from flows f_1 and f_2 whose arrivals to the queue are each described by Poisson processes with rates λ_1 and λ_2, respectively, $\lambda_1 + \lambda_2 = \lambda_f$. Then $M_x > M_a$.

Proof: We start by noting that $\forall i, j, k, m = 1, 2$, $E[D_{1,i}] = E[D_{1,j}] = E[D_{2,k}] = E[D_{2,m}]$. In other words, each packet has the same expected delay. Similarly, $\forall i, j, k, m = 1, 2$, $E[(D_{1,i})^2] = E[(D_{1,j})^2] = E[(D_{2,k})^2] = E[(D_{2,m})^2]$. Hence, to prove the theorem, we need only show that $E[D_{1,i}D_{2,j} \,|\, (a(p_{1,i}, p_{2,j}) = 1)] > E[D_{2,i}D_{2,i+1}]$.

A Poisson process of rate λ_1 has the same distribution as a Poisson process with rate $\lambda_1 + \lambda_2$ that has been thinned with probability $\lambda_2/(\lambda_1 + \lambda_2)$. As defined in (6), M_x computes the correlation coefficient between adjacent packets in the aggregate foreground flow. Hence, $E[D_{1,i}D_{2,j}|(a(p_{1,i}, p_{2,j}) = 1)] = E[D_iD_{i+1}]$. Alternatively, as defined in (7), M_a is the correlation coefficient between packets from f_2 that are adjacent with respect to f_2 (i.e., packets $p_{2,j}$ and $p_{2,j+1}$). Let $\Lambda_1(i, i+n)$ be a random variable that equals 1 if p_j is from f_1 for all j where $i < j < i + n$ and 0 otherwise. Let $\Lambda_2(i, i+n)$ be a random variable that equals 1 if p_i and p_{i+n} are from f_2, and 0 otherwise. Hence, p_i and p_{i+n} are adjacent packets in f_2 with respect to packets in f_2 when both $\Lambda_2(i, i+n) = 1$ and $\Lambda_1(i, i+n) = 1$. Using the fact that packet delays are independent of their marking ($E[D_iD_{i+n}|\Lambda_1(i, i+n) = 1, \Lambda_2(i, i+n) = 1] = E[D_iD_{i+n}]$) and that $\Lambda_2()$

and $\Lambda_1(\,)$ are independent random variables [such that $\Pr(\Lambda_1(i,\, i{+}n) = 1 \,|\, \Lambda_2(i,\, i{+}n) = 1) = \Pr(\Lambda_1(i,\, i{+}n) = 1]$, then

$$\begin{aligned}
&E[D_{2,\,j} D_{2,\,j+1}] \\
&\quad = \sum_{n=1}^{\infty} E[D_i D_{i+n} \,|\, \Lambda_1(i,\, i+n) = 1,\, \Lambda_2(i,\, i+n) = 1] \\
&\qquad \cdot \Pr(\Lambda_1(i,\, i+n) = 1 \,|\, \Lambda_2(i,\, i+n) = 1) \\
&\quad < \sum_{n=1}^{\infty} E[D_i D_{i+1}] \Pr(\Lambda_1(i,\, i+n) = 1 \,|\, \Lambda_2(i,\, i+n) = 1) \\
&\quad = E[D_i D_{i+1}] \sum_{n=1}^{\infty} \Pr(\Lambda_1(i,\, i+n) = 1) \\
&\quad = E[D_i D_{i+1}]
\end{aligned}$$

where Lemma 4 yields the above inequality. ∎

Thus far, we have shown that $M_x > M_a$ when the flows share POCs. We now prove that $M_x < M_a$ when the flows do not share POCs.

Lemma 5: $E[D_{2,\,i+1} | D_{2,\,i} = x]$ is an increasing function of x.

This lemma is also intuitive. It says that the expected delay of $p_{2,\,i+1}$ is an increasing function of the delay of $p_{2,\,i}$. A detailed proof is given in [21]

Theorem 4: Let f_1 and f_2 have separate queues as bottlenecks, and let f_2's queue be an $M + G/G/1$ queue as in Theorem 3 (except that f_1 does not pass through the queue). Then $M_x < M_a$.

Proof: First, note that $M_x = 0$, since the delays experienced by two packets drawn from separate foreground flows are independent. The denominator of a correlation coefficient is always larger than 0. Hence, we need only show that the numerator in the correlation coefficient of M_a is larger than 0.

$$\begin{aligned}
&E[D_{2,\,i+1} D_{2,\,i}] - E[D_{2,\,i+1}] E[D_{2,\,i}] \\
&\quad = \int_{x=0}^{\infty} x \Pr(D_{2,\,i} = x) E[D_{2,\,i+1} \,|\, D_{2,\,i} = x]\, dx \\
&\qquad - \int_{x=0}^{\infty} x \Pr(D_{2,\,i} = x) E[D_{2,\,i+1}]\, dx.
\end{aligned}$$

By Lemma 5, $E[D_{2,\,i+1} | D_{2,\,i} = x]$ is an increasing function of x. Noting that $\int_{x=0}^{\infty} \Pr(D_{2,\,i} = x) E[D_{2,\,i+1} \,|\, D_{2,\,i} = x]\, dx = E[D_{2,\,i+1}] = \int_{x=0}^{\infty} \Pr(D_{2,\,i} = x) E[D_{2,\,i+1}]\, dx$, it follows that there must exist some γ for which $x < \gamma \Leftrightarrow E[D_{2,\,i+1} | D_{2,\,i} = x] < E[D_{2,\,i+1}]$, so that we can apply Lemma 3 with $G(x) = x$, $f(x) = \Pr(D_{2,\,i} = x) E[D_{2,\,i+1}]$, and $g(x) = \Pr(D_{2,\,i} = x) E[D_{2,\,i+1} \,|\, D_{2,\,i} = x]$, we get that the right-hand side of (17) is larger than 0, which completes the proof. ∎

IV. Performance in Simulation

In this section, we use simulation to examine four scenarios. In the first two scenarios, we simulate flows that are configured in an Inverted-Y topology. In the second two scenarios, the flows are configured in a Y topology. In the first and third scenarios, the flows' POCs are independent, and in the second and fourth, the flows' POCs are shared. Fig. 5 demonstrates the topology on which we ran our simulations using the `ns-2` simulator [25]. For the Y topology, probe receivers are connected to the leftmost node, the sender for f_1 is connected to the bottom-right node, the sender for f_2 to the top-right. For the Inverted-Y topology, we simply swap the locations of each flow's sender with its receiver. We construct POCs by assigning links that we want congested to process at a rate of 1.5 Mb/s, and links that we do not want congested to process at a rate of 1000 Mb/s. The links that are assigned the 1.5-Mb/s capacity are either the set of links numbered 1 through 3 (shared POC) or the set of links numbered 4 through 8 (separate POCs). All background data traffic flows in the same direction as that of the foreground flows, and traverses a subset of links that are assigned the 1.5-Mb/s capacity (i.e., there is no background traffic on the high bandwidth links). Background flows are placed on the path of each probe. The number of such flows is chosen uniformly between 10 and 20, and each flow uses the TCP protocol with probability 0.75. Otherwise, it is a CBR flow with on-off service times. The CBR rate is chosen uniformly between 10 and 20 kb/s, and the average on time and off time is chosen uniformly between 0.2 and 3.0 s. For each of the four scenarios, we run 1000 experiments, starting the background traffic at time $t = -10$, and then starting the probes at time $t = 0$, and ending the experiment at time $t = 120$.

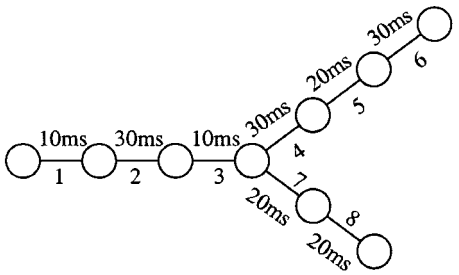

Fig. 5. Topology used in simulation experiments.

Fig. 6 plots the percentage of experiments run over the Inverted-Y topology that, using the loss-corr and delay-corr techniques, correctly infer whether or not the flows share as a function of time. As clock time progresses and additional packets arrive at the receivers, the estimates of M_x and M_a are computed over an increasing sample set size. The hope is that over time, as the estimates of M_x and M_a increase in accuracy, more tests will correctly infer whether or not the flows' POCs are shared.

Fig. 6(a) plots the results of 1000 experiments in which the flows' POCs are separate. Fig. 6(b) plots the results of 1000 other experiments in which the flows' POCs are shared. In each experiment, both foreground flows send 20-B packets at an average rate of 25 packets per second. The clock time varies exponentially on the x axis, where a time of zero indicates the time that the first probe packet arrived at either receiver. The y axis indicates the percentage of the experiments that satisfy the property being plotted. Curves labeled "no response" plot the percentage of tests that cannot form a hypothesis by the time indicated on the x axis (the test must have at least one sample that can be used to compute an estimate for both M_x and M_a before it forms a hypothesis). Curves labeled "correct" plot the percentage of tests returning a hypothesis whose hypothesis is correct at the time indicated on the x axis (i.e., tests that have not yet returned a hypothesis are omitted when computing the

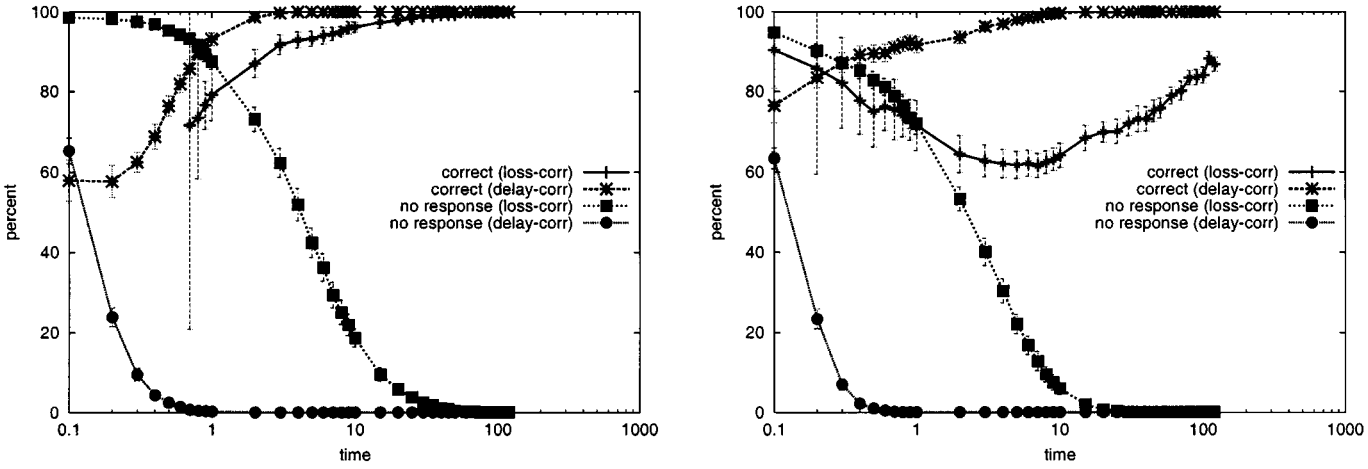

Fig. 6. Inverted-Y topology. (a) Independent congestion. (b) Shared congestion.

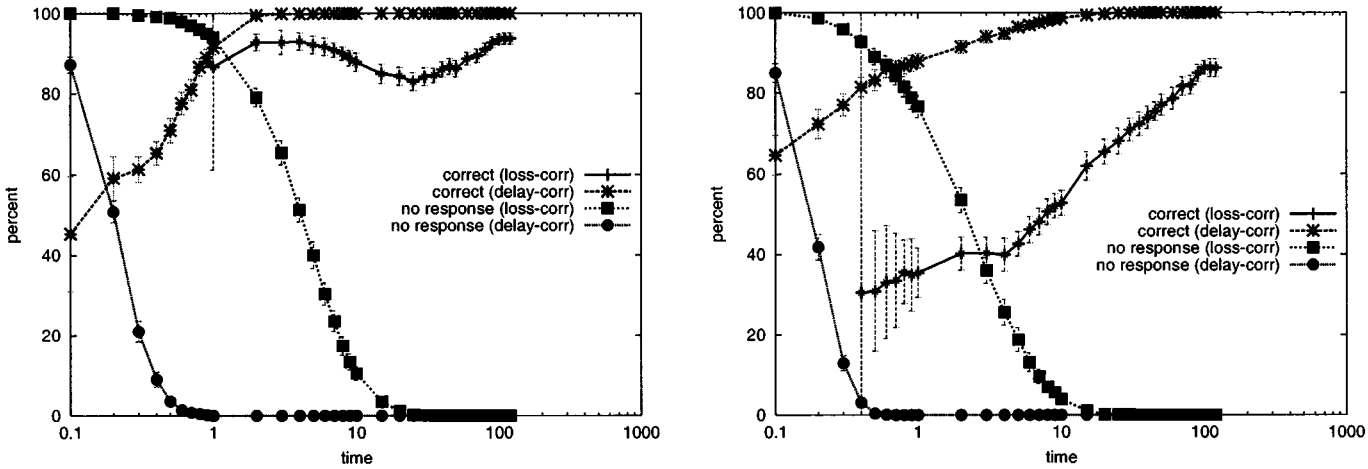

Fig. 7. Y topology. (a) Independent congestion. (b) Shared congestion.

values of the "correct" curves). Ninety-five percent level confidence intervals are generated by averaging over 20 samples at a time, such that the distribution of the average of the samples is approximately normal. Points are omitted when confidence intervals are too wide.

We can make several observations from these graphs. First, the rate at which the delay-corr technique correctly assesses whether or not a POC is shared is an order of magnitude faster than that of the loss-corr technique. For instance, for 90% of the experiments to draw a correct conclusion, the delay-corr technique obtains a sufficient number of samples within a second, whereas the loss-corr technique must proceed for between 10 and 50 s over the various experiments. This is not surprising, given the fact that the delay-corr technique is able to use almost every packet to compute its measures, whereas the loss-corr technique only uses samples that contain certain sequences of packet losses. We also note a trend that for the loss-corr technique when POCs are shared, the percentage of hypotheses that are correct initially decreases with time. This is likely to be a result of a bias caused by the fact that the samples used to compute M_x arrive at a slower rate than those used to compute M_a.

Fig. 7 plots similar results for a Y-topology as those in Fig. 6. There is little difference in the results of the delay-corr technique between the two topologies. This is not surprising, since the difference in topology does not affect the way the delay-corr experiment is executed. On the other hand, the loss-corr technique for the Y-topology converges at a slower rate than the loss-corr technique for the Inverted-Y topology. This is because in most cases, the value of M_x computed using (3) is not significantly different from the value of M_a computed using (4), so more samples are necessary to correctly assess with a given level of confidence which one is larger. Furthermore, the conditioning within (3) is stricter than that for (1), such that on average it takes longer to get the same number of samples.

A. Network Variations: RED and TCP

Our theoretical and preliminary simulation work considers a networking environment in which routers utilize drop-tail routing and in which the time between transmissions of the foreground flows is exponentially distributed. We now present a preliminary exploration through simulation on variations of this model. In particular, we consider networking environments in which 1) routers enable random early detection (RED) and 2) the foreground flows are TCP. Our findings are that the loss-corr

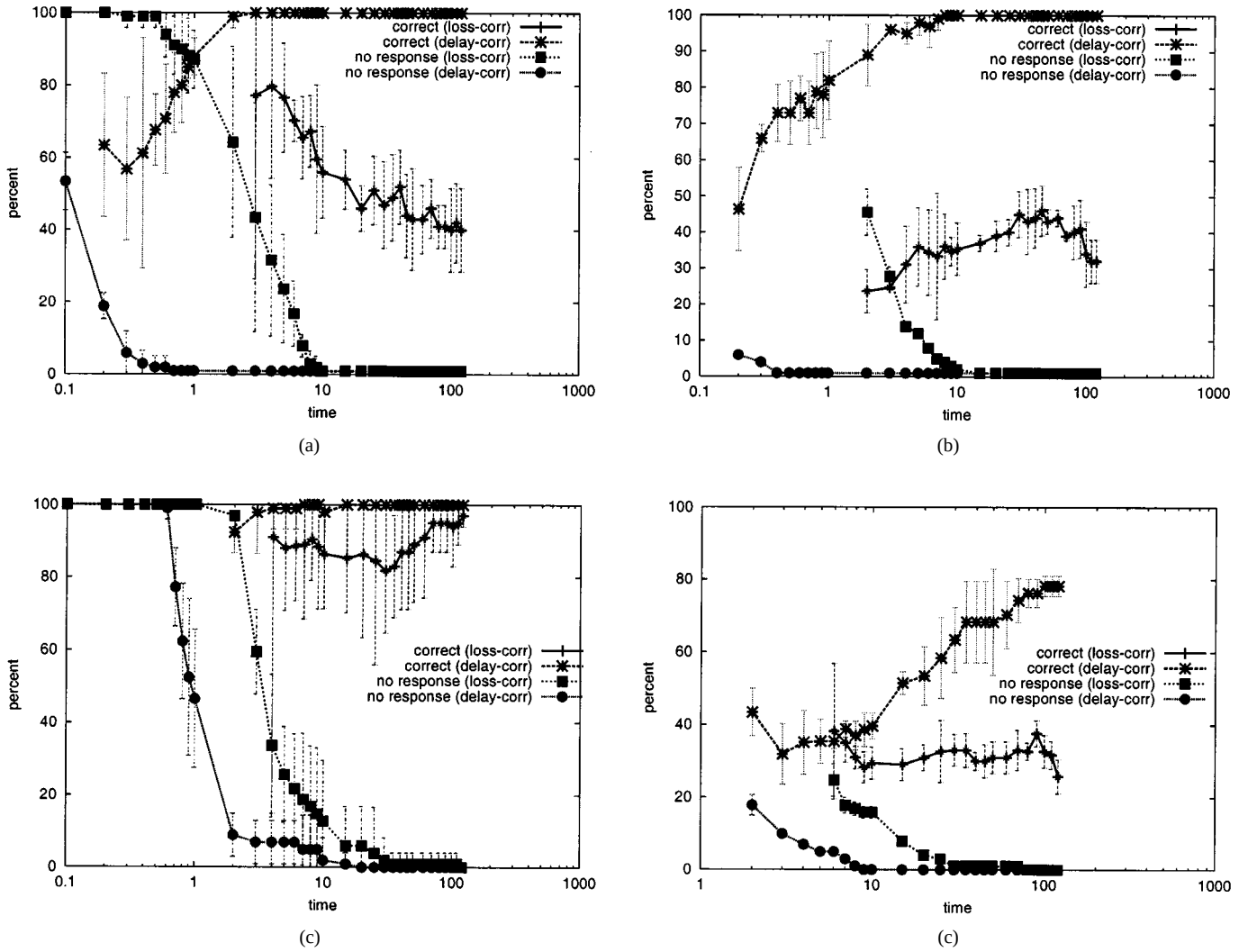

Fig. 8. Variation of queue management policy (RED) and probe type (TCP). (a) RED: independent congestion. (b) RED: shared congestion. (c) TCP: independent congestion. (d) TCP: shared congestion.

technique becomes unreliable, and that the delay-based technique converges to the correct result, but that in order to achieve a high degree of confidence, considerably more time is required.

Fig. 8 presents our results of simulation experiments applying these variations in the network environment. Fig. 8(a) and (b) presents the results of ns simulations upon a Y-topology. The experiment is similar in all respects to the experiments conducted previously in this section with the exception that the routers activate RED. We see that the probability that the loss-corr technique returns the correct result does not converge to one as the test is run for more time. However, the probability that the delay-corr technique returns the correct result does converge to one, but at a rate that is an order of magnitude slower than that when the network consists of drop-tail routers. These results are not surprising. First, RED will randomly drop probes as the queue fills; this by itself introduces noise into the test statistic. Second, RED is designed to encourage TCP sessions to "back off" prior to overflowing its bottleneck queue. This reduces the likelihood that the queue will be full and reduces the rate of packet loss. Third, RED maintains a more stable queue length, reducing the variance of the queueing delay process.

Fig. 8(c) and (d) presents the results of ns simulations upon an inverted-Y topology, similar in all respects to the previous experiments (drop-tail routers) with the exception that the foreground flows are TCP flows. Again, we see that the loss-corr technique's probability of returning the correct result does not converge to one. The delay-corr techniques probability of returning the correct result when the bottlenecks are shared converges toward 1 at a rate that is at least two orders of magnitude slower than when the probe transmissions are exponentially distributed and transmitted at an average rate of 20 per second. We suspect that with additional time, the delay-corr technique's reliability would converge to 1, but that it is unlikely that the test would be run in practice for more than two minutes. These results are not surprising, either. The bursty nature of TCP packet transmissions diminishes instances of the sequences in which packets from alternate flows arrive adjacent to one another. In addition, the bursty nature increases the likelihood that statistically, packet arrivals from the same flow will appear closer together in time to one another than packets across flows. It is this same property that provides the intuition as to why the tests work.

TABLE I
SITE NAME ABBREVIATIONS

C	Columbia (New York)	M_1	UMass-1	M_2	UMass-2	U	UCL (London)
S	AT&T-San Jose (California)	M_3	UMass-3	A	ACIRI (California)		

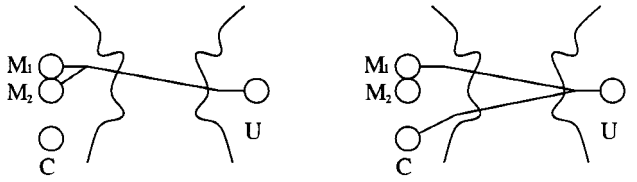

Fig. 9. Experimental topologies. (a) Shared. (b) Independent.

V. ACTUAL TRACES

We have demonstrated the robustness of our comparison tests through queueing analysis and simulation. Now, we present results of experiments used to evaluate how the tests work in practice. We apply the tests to flows that traverse the Internet, choosing end-system locations such that we can be reasonably sure as to whether or not the flows share congestion. We then examine the results of our comparison tests. The set of end systems used in the experiments consists of machines located at ACIRI (California), UCL (London, U.K.), Columbia (New York, NY), AT&T-San Jose (California), and three of our own machines, labeled UMass-1 through UMass-3. Table I presents a shorthand notation for these sites that is used in the subsequent figures and tables.

Fig. 9 demonstrates an example of a set of end-system sites for experiments such that we can be reasonably sure (without using the comparison tests) whether or not the flows share a POC. The example in Fig. 9 involves four sites, UMass-1, UMass-2, Columbia, and UCL, three of which are located in the U.S., and one in Europe. UMass-1 and UMass-2 are in fact located on the same LAN, such that the paths from (to) UMass-1 and UMass-2 to (from) UCL shared all links in common except for the initial (final) hop (this was verified using `traceroute`). We expect that in this configuration [Fig. 9(a)], the two flows will share congestion. We believe that at the time of our experiments, the path from (to) UMass-1 to (from) UCL and the path from (to) Columbia to (from) UCL were traversed separate trans-Atlantic links, and that the paths were disjoint along all links in the U.S. [Fig. 9(b)]. We came to this conclusion via an examination of `traceroute` statistics (a more detailed discussion of our use of `traceroute` is presented later in the paper). We expect that in this configuration, the flows will not share congestion. In either case, we can then apply the comparison tests and see whether or not the results of the test correctly identify whether or not the POCs are shared.

Table II summarizes the results of experiments performed during the middle of the day on November 1 and November 3, 1999, using the hosts listed in Table I. Each experiment ran for 600 s, with each foreground source sending 20-byte UDP Poisson probes (not counting bytes in the IP header) at a rate of 25 per second. Each packet contained a sequence number and a time stamp whose time was computed at the source immediately prior to the socket call that transmitted the packet. Packet arrival times at the receiver were recorded at the receiver immediately after the socket call was performed to retrieve the packet data. All time stamping was performed at the user level.

The first column in Table II indicates the date on which the experiment was performed. The second column indicates whether the topology was a Y or Inverted-Y topology. The third column indicates the hosts that participated in the experiment, using the abbreviations for the host names supplied in Table I. For the Y topology, the labeling, (A, B $\rightarrow$ C), indicates that senders at host A and host B transmitted probes to receivers co-located at host C. For the Inverted-Y topology, the labeling is of the form (A $\rightarrow$ B, C), indicating that the co-located senders at host A transmitted probes to receivers at hosts B and C.

The fourth column provides a rough approximation of the average delay experienced over the shared path of the two flows, as well as the average delay over the respective independent portions of the paths. These values were obtained through two calls to `traceroute` that were executed during the experiment from the locations of the probe sender(s), one for each source–destination pair. The shared links are the longest sequence of links, starting from the point of the co-located hosts, that contain the same sequence of IP addresses. The remaining links are unshared. The delay for a sequence of links is the average of the delays as reported by `traceroute` at one endpoint of the sequence minus the average of the delays as reported by `traceroute` at the other end.[5] If a sequence of links is assigned a delay that is less than zero, we assume that the delay on this sequence of links is negligible, and write the delay as $\sim$0.

For the Y topology, the entry, $(x, y \rightarrow z)$, $x, y, z \in \Re$ that is associated with the labeling, (A, B $\rightarrow$ C), indicates that the unshared portion of the path from host A to host C has an average delay of x ms, the unshared portion of the path from host B to host C has an average delay of y ms, and the shared portion of these paths has an average delay of z ms. For the inverted-Y topology, the entry $(x \rightarrow y, z)$ that is associated with the labeling, (A $\rightarrow$ B, C), indicates that it takes on average x ms to traverse the shared portion of the paths, and on average, y and z ms to traverse the unshared portions of the paths to B and C, respectively.

We use the relative values of these path delays to estimate whether or not the POCs are shared. If the delay over the shared portion is small with respect to the nonshared portions, we assume that the POC is not shared. Otherwise, we assume it is. A line is drawn in the middle of the table separating the experiments whose flows we assume traverse a shared POC (above the line) from those whose flows we assume traverse separate POCs (below the line). We wish to point out that these assumptions are only a "best guess" as to whether the congestion is actually shared or not. The information obtained by `traceroute` can be used to distinguish the number of links that two

[5]No more than three are reported per hop, but in all our calls, at least one was reported where necessary, allowing us to compute an average.

TABLE II
TRACE RESULTS

Date	Topology	Hosts	shared / non-shared hop ratio (msec)	loss rates (%)	loss-corr result	stable since (sec)	delay-corr result	stable since (sec)
11/3	Y	(M_1,$M_2 \rightarrow U$)	(1,1 $\rightarrow$ 440)	1.42, 1.29 : 1.36	Shared	154	Shared	0.5
11/3	Y	(M_1,$M_2 \rightarrow A$)	(1,1 $\rightarrow$ 91)	0.07, 0.01 : 0.04	Not shared	184	Shared	2
11/3	Inv-Y	($A \rightarrow M_1$, M_2)	(98 $\rightarrow \sim 0$, ~ 0)	0.04, 0.07 : 0.06	INSUF		Not shared	552
11/1	Inv-Y	($A \rightarrow M_2$, M_3)	(91 $\rightarrow \sim 0$, ~ 0)	0.03, 0.03 : 0.03	INSUF		Shared	562
11/1	Inv-Y	($U \rightarrow M_1$, M_2)	(150 $\rightarrow \sim 0$, ~ 0)	5.33, 6.10 : 5.72	Shared	23	Shared	0.8
11/3	Inv-Y	($M_1 \rightarrow U$, A)	(6 $\rightarrow$ 82, 322)	0.75, 0.17 : 0.46	INSUF		Not shared	23
11/1	Inv-Y	($M_2 \rightarrow U$, A)	(0 $\rightarrow$ 102, 447)	2.08, 0.24 : 1.25	Not shared	337	Not shared	4
11/1	Inv-Y	($U \rightarrow M_1$, A)	(3 $\rightarrow$ 313, 141)	6.08, 0.26 : 3.05	Not shared	411	Not shared	8.2
11/1	Inv-Y	($U \rightarrow M_1$, C)	(47 $\rightarrow$ 110, 75)	12.12, 0.07 : 6.12	Not shared	6	Not shared	6.2
11/1	Inv-Y	($U \rightarrow M_1$, S)	(75 $\rightarrow$ 233, 75)	8.55, 0.01 : 4.26	Not shared	249	Not shared	4
11/1	Inv-Y	($U \rightarrow M_2$, A)	(30 $\rightarrow$ 264, 193)	1.95, 0.10 : 1.03	Not shared	109	Not shared	48
11/3	Y	(U,$A \rightarrow M_1$)	(323,91 $\rightarrow \sim 0$)	7.73, 0.09 : 3.90	Shared	543	Not shared	7
11/3	Inv-Y	($A \rightarrow C$, M_1)	(4 $\rightarrow$ 65, 87)	0.05, 0.06 : 0.06	INSUF		Not shared	328
11/1	Inv-Y	($A \rightarrow U$, M_2)	(4 $\rightarrow$ 189, 91)	0.15, 3.51 : 1.82	Not shared	560	Not shared	3
11/3	Y	(C,$M_1 \rightarrow A$)	(64,87 $\rightarrow$ 4)	0.00, 0.03 : 0.02	INSUF		Shared	30
11/3	Y	(C,$M_1 \rightarrow U$)	(88,340 $\rightarrow$ 130)	1.51, 2.32 : 1.92	Shared	61	Shared	0.5

paths share and can give coarse estimates of delays experienced on those links. This information is helpful only in that without any other information, two paths that share numerous links in common are more likely to experience shared congestion than two paths that have few links in common. In addition, high delays can be an indication that either router queues are backed up or that the router is incapable of handling high loads. Hence, using these observations is by no means a definitive way to determine whether or not two flows share points of congestion, but aside from comparison tests such as those that we propose, there is little that can be done using today's technology to determine whether or not two flows share common points of congestion. Hence, we feel that the means that we use to make a "best guess" is the best approach we have available to test our techniques in a practical setting.

The fifth column presents the loss rates. An entry, a, b:c, associated with the labeling, (A, B $\rightarrow$ C), or the labeling, (C $\rightarrow$ A, B), indicates that the loss rate of the flow involving host A is a, the loss rate of the flow involving host B is b, and the average loss rate over both of the flows is c. We emphasize that the loss rates are given as percentages, so values less than one indicate that fewer than one out of every one hundred packets was lost.

The last four columns present the results of the experiments. The column labeled "loss-corr result" presents the hypothesis returned by the loss-corr technique after 600 s; to its right is the time of the experiment when the comparison test last changed its hypothesis, i.e., the time at which it "stabilized" on its final hypothesis. A hypothesis of INSUF indicates that the technique was unable to form a hypothesis due to a lack of samples. The last two columns present similar results for the delay-corr technique.

We find that five of the 16 experiments that applied the loss-corr technique were unable to construct a hypothesis. We note that in all but one of these tests in which no hypothesis was constructed, the host at ACIRI was the point of co-location. The loss rates in these traces were so low, that no samples were produced that could be used to estimate the cross-measure, M_x. Of the remaining eleven experiments, only three of eleven fail to match the assumed correct hypothesis. Except for the last experiment listed, all experiments that returned the wrong hypothesis were conducted using flows with very low loss rates, which suggests that these flows did not experience significant levels of congestion.

In more than 80% of our experiments, the delay-corr test returned the hypothesis that matched our assumption about whether or not the POCs were shared. Two of the three tests that failed consisted of sessions with very low loss rates. We hypothesize that the low loss rates are an indication that the links were in use far below their capacity, such that the level of delay congestion was insignificant.

VI. OPEN ISSUES

There are several issues that remain open with regard to detecting shared congestion that we have not considered. We touch briefly on those that we feel are the most critical to solve. First, in the Inverted-Y topology, the information necessary to compute the cross-measures is distributed at the receiving hosts. In this paper, our processing of the information is done off-line, at a centralized point to which we transmit all data. One direction for future work is to design protocols that, accounting for the fact that the information may be distributed, can efficiently construct a hypothesis. A second direction is to scale the tests such that they can detect POCs efficiently among several flows. Katabi's technique [13] is one possibility, but this technique is currently limited to the Y-topology, where the ratio of bandwidth utilized at the POC by the background traffic in relation to the foreground traffic is small. In practice, we expect POCs exist at points where many flows are being aggregated, and expect that this ratio can be quite large. A solution that scales easily to many flows over a variety of traffic conditions remains an open problem.

Finally, our work has assumed that congestion at different bottlenecks exhibit significantly lower levels of correlation of congestion events such as loss or delay between packets at a given bottleneck point than between packets at two different bottleneck points. Recent work (such as [26]) conjectures that certain events might be correlated across different parts of the network. We suspect that due to the time scale over which such

correlation phenomena are observed, such correlations are unlikely to affect our results significantly. Nonetheless, we feel that further study of the potential impact of such effects is warranted.

VII. Conclusion

We have demonstrated two techniques that, via end-to-end measurement, are able to accurately detect whether or not two flows share the same points of congestion within the network. One of our key insights is the construction of a comparison test. Rather than trying to figure out the level of correlation that indicates that two flows share a common point of congestion, we compare the correlation across flows to the correlation within a single flow to make the determination. Another insight is that the detection can be performed by transmitting probes, each of which have intra-transmission times that are described by Poisson processes. These techniques can be applied to flow topologies where the senders are co-located but the receivers are not, as well as the case where the receivers are co-located but the senders are not. We demonstrated the performance of these techniques through a mix of proofs using traditional queueing models, simulation over a wide range of controlled scenarios, and results using actual Internet traces.

Appendix
Proofs of Delay Lemmas

Proof (of Lemma 4): Define A_i to be the time of arrival of p_i at the queue. $D_{i+n} = E_{i+n} - A_{i+n}$, where E_{i+n} is the time in which p_{i+n} exits (i.e., completes being serviced by) the queue. p_{i+n}'s service is not completed until after 1) p_is service is completed, and then 2) all background packets that arrive between p_i and p_{i+n} and all foreground packets $p_{i+1}, \ldots, p_{i+n}$ are serviced. Thus, $E_{i+n} = A_i + D_i + \sum_{j=1}^{N(A_i, A_{i+n})} s_j + \sum_{j=1}^{n} S_j + \gamma(A_i, A_{i+n})$, where $N(x, y)$ is the number of (background) arrivals admitted into the queue during the time interval $[x, y)$, s_j is the time it takes to process the jth of the these arrivals, S_j is the time it takes the server to process p_{i+j}, and $\gamma(x, y)$ is the total time within the interval $[x, y)$ that the processor is idle (no jobs in queue).

By substituting the above expression in place of E_{i+n} within $D_{i+n} = E_{i+n} - A_{i+n}$, we obtain

$$D_{i+n} = D_i - t_n + \sum_{j=1}^{N(A_i, A_{i+n})} s_j + \sum_{j=1}^{n} S_j + \gamma(A_i, A_{i+n}) \tag{17}$$

where $t_n = A_{i+n} - A_i$. We make several observations that will help in proving the lemma. First, note that t_n is independent of D_i: the time spent by p_i in the queue is independent of the time it takes p_{i+n} to arrive after p_i's arrival. Second, the service time, S_j, of p_{i+j} for each $j > 0$ is independent of arrival times of foreground packets and the delay of p_i, and is therefore independent of A_{i+m} for all m and of D_i as well. Similarly, the service time, s_j, of any background packet that arrives after time A_i is independent of arrival times and of D_i. Third, since the queue has infinite capacity, $N(x, y)$ is independent of the queueing system during the time interval of length $[x, y)$. Thus, $N(x, y)$ and D_i are independent, and $E[N(x, y)]$, the expected number of background packets that arrives in the interval $[x, y)$, is simply $\lambda_b(y - x)$. It follows that $E[\sum_{j=1}^{N(x,y)} s_j] = E[N(x, y)]E[s_j] = \lambda_b(y - x)E[s]$, where s has the same distribution as each s_j (because service times are i.i.d.). The rate at which packets can be processed at the queue is $\mu = 1E[s_j] = 1E[S_j]$.[6] Finally, note that $N(x, z) = N(x, y) + N(y, z)$ and $\gamma(x, z) = \gamma(x, y) + \gamma(y, z)$ whenever $x \le y \le z$. Letting $t_n = A_{i+n} - A_{i+1}$ (the time between the first and nth arrivals of Poisson process with rate λ_f), we have that $E[t_n] = (n-1)/\lambda_f$.

We now prove the result by showing that for $n > 1$, $E[D_i D_{i+n}] - E[D_i D_{i+1}] = E[D_i(D_{i+n} - D_{i+1})] < 0$. After replacing D_i by D_{i+1} in (17), we have that $D_{i+n} - D_{i+1} = -t_n + \sum_{j=1}^{N(A_{i+1}, A_{i+n})} s_j + \sum_{j=2}^{n} S_j + \gamma(A_{i+1}, A_{i+n})$. Applying our observations of independence, we get

$$\begin{aligned} &E[D_i(D_{i+n} - D_{i+1})] \\ &\quad = E[D_i](-E[t_n] + E[N(A_{i+1}, A_{i+n})]E[s] + (n-1)E[s]) \\ &\qquad + E[D_i\gamma(A_{i+1}, A_{i+n})] \\ &\quad = E[D_i](E[t_n](-1 + \lambda_b/\mu) + (n-1)/\mu) \\ &\qquad + E[D_i\gamma(A_{i+1}, A_{i+n})]. \end{aligned} \tag{18}$$

Note that starting from time A_{i+1}, the queue cannot be empty at least until after p_{i+1} exits the queue. A simple sample-path argument can be used to demonstrate that increasing D_i decreases the likelihood that the queue is idle between arrivals of p_i and p_{i+n} for longer than any aggregate length of time x. More formally, for any x, $\Pr((\gamma(A_{i+1}, A_{i+n}) > x) | (D_i = d))$ is a monotonically decreasing function of d. It follows that $E[D_i\gamma(A_{i+1}, A_{i+n})] < E[D_i]E[\gamma(A_{i+1}, A_{i+n})]$ [apply Lemma 3 with $G(x) = x$, $f(x) = \Pr(D_i = x)E[\gamma(A_{i+1}, A_{i+n}) | D_i = x]$, and $g(x) = \Pr(D_i = x)E[\gamma(A_{i+1}, A_{i+n})]$]. Furthermore, we can show that $E[\gamma(A_{i+1}, A_{i+n})] < E[t_n](\mu - \lambda_b - \lambda_f)/\mu$ (the expected time times the idle rate of the system) as follows. If packet p_{i+1} took 0 seconds to process, because it and p_{i+n} are Poisson arrivals, we can use the PASTA property to obtain that $E[\gamma(A_{i+1}, A_{i+n})] = E[t_n](\mu - \lambda_b - \lambda_f)/\mu$. However, again via a sample-path argument, the fact that p_{i+1} has a nonnegative service time can only reduce the expected idle time.

Applying this resulting inequality into (18), and substituting $E[t_n] = (n-1)/\lambda_f$, we get

$$\begin{aligned} &E[D_i(D_{i+n} - D_{i+1})] \\ &\quad < E[D_i]\left(\frac{-(n-1)}{\lambda_f} + \frac{(n-1)\lambda_b}{\lambda_f\mu} + \frac{(n-1)}{\mu}\right. \\ &\qquad \left. + \frac{(n-1)(\mu - \lambda_b - \lambda_f)}{\mu\lambda_f}\right) = 0. \end{aligned}$$

■

References

[1] H. Balakrishnan, H. Rahul, and S. Seshan, "An integrated congestion management architecture for internet hosts," in *Proc. SIGCOMM'99*, Cambridge, MA, Sept. 1999, pp. 175–187.

[6]We assume for simplicity that the processing of all packets (foreground, background) have the same expected processing time. However, this is not necessary.

[2] V. Padmanabhan, "Coordinated congestion management and bandwidth sharing for heterogeneous data streams," in *Proc. NOSSDAV'99*, Basking Ridge, NJ, June 1999, pp. 187–190.
[3] L. Gautier, C. Diot, and J. Kurose, "End-to-end transmission control mechanisms for multiparty interactive applications in the internet," in *Proc. IEEE INFOCOM'99*, vol. 3, New York, NY, Mar. 1999, pp. 1470–1479.
[4] J. Byers, M. Luby, and M. Mitzenmacher, "Accessing multiple mirror sites in parallel: Using Tornado codes to speed up downloads," in *Proc. IEEE INFOCOM'99*, vol. 1, New York, NY, Mar. 1999, pp. 275–283.
[5] P. Rodriguez and E. Biersack, "Parallel-access for mirror sites in the internet," in *Proc. IEEE INFOCOM'00*, vol. 2, Tel-Aviv, Israel, Mar. 2000, pp. 864–873.
[6] S. Floyd and V. Jacobson, "Random early detection gateways for congestion avoidance," *IEEE/ACM Trans. Networking*, vol. 1, pp. 397–413, Aug. 1993.
[7] K. Harfosuh, J. Byers, and A. Bestavros, "Robust identification of shared losses using end-to-end unicast probes," in *Proc. ICNP'00*, Osaka, Japan, Nov. 2000, pp. 22–36.
[8] Y. Tsang, M. Coates, and R. Nowak, "Passive unicast network tomography using EM algorithms," in *IEEE Int. Conf. Acoustics, Speech, and Signal Processing*, vol. 3, Salt Lake City, UT, May 2001, pp. 1469–1472.
[9] M. Coates and R. Nowak, "Network tomography for internal delay estimation," in *IEEE Int. Conf. Acoustics, Speech, and Signal Processing*, vol. 6, Salt Lake City, UT, May 2001, pp. 3409–3412.
[10] S. Seshan, M. Stemm, and R. Katz, "SPAND: Shared passive network performance discovery," in *Proc. USITS'97*, Monterey, CA, Dec. 1997.
[11] S. Savage, N. Cardwell, and T. Anderson, "The case for informed transport protocols," in *Proc. 7th Workshop Hot Topics in Operating Systems*, Rio Rico, AZ, Mar. 1999, pp. 58–63.
[12] V. Padmanabhan, "Optimizing data dissemination and transport in the internet," presented at the BU/NSF Workshop Internet Measurement, Instrumentation and Characterization, Sept. 1999, slides.
[13] D. Katabi, I. Bazzi, and X. Yang, "An information theoretic approach for shared bottleneck inference based on end-to-end measurements," M.I.T. Lab. for Computer Science, Cambridge, MA, Class project, 1999.
[14] S. Ratnasamy and S. McCanne, "Inference of multicast routing trees and bottleneck bandwidths using end-to-end measurements," in *Proc. IEEE INFOCOM'99*, vol. 1, New York, NY, Mar. 1999, pp. 353–360.
[15] R. Caceres, N. Duffield, J. Horowitz, and D. Towsley, "Multicast-based inference of network-internal characteristics: Accuracy of packet loss estimation," *IEEE Trans. Inform. Theory*, pp. 2462–2480, Nov. 1999.
[16] N. Duffield, F. Lo Presti, V. Paxson, and D. Towsley, "Inferring link loss using striped unicast probes," in *Proc. IEEE INFOCOM'01*, vol. 1, Anchorage, AK, Apr. 2001, pp. 915–923.
[17] M. Yajnik, S. B. Moon, J. Kurose, and D. Towsley, "Measurement and modeling of the temporal dependence in packet loss," in *Proc. IEEE INFOCOM'99*, vol. 1, New York, NY, Mar. 1999, pp. 345–352.
[18] S. Moon, J. Kurose, and D. Towsley, "Correlation of packet delay and loss in the internet," Univ. Massachusetts, Amherst, MA, Tech. rep. CMPSCI 98-11, Jan. 1998.
[19] S. Moon, P. Skelly, and D. Towsley, "Estimation and removal of clock skew from network delay measurements," in *Proc. IEEE INFOCOM'99*, vol. 1, New York, NY, Mar. 1999, pp. 227–234.
[20] V. Paxson, "On calibrating measurements of packet transit times," in *Proc. ACM SIGMETRICS'98*, Madison, WI, June 1998, pp. 11–21.
[21] D. Rubenstein, J. Kurose, and D. Towsley, "Detecting shared congestion of flows via end-to-end measurement," Univ. Massachusetts, Amherst, MA, Tech. rep. CMPSCI 99-66, Nov. 1999.
[22] ——, "Detecting shared congestion of flows via end-to-end measurement," in *Proc. ACM SIGMETRICS'00*, Santa Clara, CA, May 2000.
[23] G. Folland, *Real Analysis: Modern Techniques and Their Applications*. New York: Wiley, 1984.
[24] S. Ross, *Stochastic Processes*. New York: Wiley, 1983.
[25] S. McCanne and S. Floyd. (1997) ns-LBL network simulator. [Online]. Available: http://www-nrg.ee.lbnl.gov/ns/
[26] A. Veres, Z. Kenesi, S. Molnar, and G. Vattay, "On the propagation of long-range dependence in the internet," in *Proc. ACM SIGCOMM'00*, Stockholm, Sweden, Sept. 2000, pp. 243–256.

Dan Rubenstein (S'97–M'00) received the B.S. degree in mathematics from the Massachusetts Institute of Technology, Cambridge, the M.A. degree in mathematics from the University of California at Los Angeles, and the Ph.D. degree in computer science from the University of Massachusetts, Amherst, in 1992, 1994, and 2000, respectively.

Since 2000, he has been a Member of the Department of Electrical Engineering and has a courtesy appointment in the Department of Computer Science at Columbia University, New York, NY.

Dr. Rubenstein was a recipient of the ACM SIGMETRICS Best Student Paper Award in 2000.

Jim Kurose (S'81–M'84–SM'91–F'97) received the Ph.D. degree in computer science from Columbia University, New York, NY.

He is currently a Professor in the Department of Computer Science at the University of Massachusetts, Amherst, where he is also Co-director of the Networking Research Laboratory of the Multimedia Systems Laboratory. He has been a Visiting Scientist at IBM Research, at INRIA Sophia Antipolis, and at Institut Eurecom. His research interests include real-time and multimedia communication, network and operating system support for servers, and modeling and performance evaluation. With K. Ross, he is the author of an introductory networking textbook, *Computer Networking: A Top-Down Approach Featuring the Internet* (Boston, MA: Addison Wesley, 2000).

Dr. Kurose is a past Editor-in-Chief of the IEEE TRANSACTIONS ON COMMUNICATIONS and the IEEE/ACM TRANSACTIONS ON NETWORKING. He has been active in the program committees for IEEE INFOCOM, ACM SIGCOMM, and ACM SIGMETRICS conferences for a number of years. He has received a number of awards for his teaching. He is a Fellow of the ACM, and a member of Phi Beta Kappa, Eta Kappa Nu, and Sigma Xi.

Don Towsley (M'78–SM'93–F'95) received the B.A. degree in physics and the Ph.D. degree in computer science from University of Texas, in 1971 and 1975, respectively.

From 1976 to 1985, he was a Member of the Faculty of the Department of Electrical and Computer Engineering at the University of Massachusetts, Amherst. He is currently a Distinguished Professor at the University of Massachusetts in the Department of Computer Science. He has held visiting positions at IBM T.J. Watson Research Center, Yorktown Heights, NY (1982–1983), the Laboratoire MASI, Paris, France (1989–1990), INRIA, Sophia-Antipolis, France (1996), and AT&T Labs-Research, Florham Park, NJ (1997). His research interests include networks, multimedia systems, and performance evaluation. He currently serves as an area editor of the *Journal of the ACM* and on the editorial board of *Performance Evaluation*. He was a Program Co-chair of the joint ACM SIGMETRICS and PERFORMANCE '92 conference.

Dr. Towsley has received the 1998 IEEE Communications Society William Bennett Paper Award and several conference best paper awards. He has previously served on several editorial boards including those of the IEEE TRANSACTIONS ON COMMUNICATIONS and IEEE/ACM TRANSACTIONS ON NETWORKING. He is a member of ORSA and the IFIP Working Groups 6.3 and 7.3, and a Fellow of the ACM.