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

Dan Rubenstein, Jim Kurose, and Don Towsley

Technical Report 99-66
Department of Computer Science
November, 1999

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. We validate these techniques via queueing analysis, simulation, and experimentation within the Internet.

1 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.

*This material was supported in part by the National Science Foundation under Grant No. ANI-9805185, CDA-9502639, by DARPA under Grant No. 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.

- 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 rate 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] 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* (or **POC** for short) 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*. 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 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.

The key idea underlying the approaches investigated in this paper is the fact that adjacent packets in the same flow experience some amount of 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 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 can vary 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 almost 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. Last, we demonstrate the capabilities of the tests in practice using actual network traces over simple topology configurations.

To our knowledge, there is no published work that presents techniques for detecting flows that are congested at the same point. In [5], the authors identify potential benefits of having separate end-systems share locally observed statistics, such as available bandwidth and loss rate. 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 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 [6]. A recent project report by Katabi et al [7] 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 techniques require more packet transmissions and as of yet do not easily scale to large receiver sets. However, our techniques remain robust under heavier background traffic 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 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 [8] and that of the MINC project [9] 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.

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 2 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 3 presents queuing analyses that demonstrate the effectiveness of the tests using theoretical models of the POCs. Section 4 presents simulation results that demonstrate the performance of the techniques behave under more realistic traffic conditions. Section 5 addresses practical issues, such as computing confidence levels of results when a small set of samples is taken, and also addresses extensions to the techniques for quantifying the "level" to which a pair of flows share POCs. Section 6 presents results of experiments conducted over the Internet. Section 7 briefly discusses some open issues. Last, Section 8 concludes the paper.

2 Technique Description

In this section, we present two techniques, the *loss-corr* technique and the *delay-corr* technique, that respectively use loss and delay measurements at receivers to determine whether or not a pair of sessions (a.k.a. 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.

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 timestamping of packets. We have noticed in our experimental results that performing the timestamping 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).

(a) Inverted-Y topology

(b) Y topology

Figure 1: Virtual topologies

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 of co-located hosts i) are more easily identified from network endpoints (i.e., they can easily be identified at the point of co-location), ii) are more likely to share congestion, since a portion of their paths are guaranteed to overlap, and iii) require less communication overhead (i.e., they can communicate over a LAN) to perform aggregation.

Figure 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 little black balls are routers at the intermediate hops. In the *Inverted-Y* topology (Figure 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 (Figure 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 participating hosts (single sender and two receivers, or two senders and a single receiver). 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_2 represent the two flows that we are testing. We refer to each of these flows as a *foreground flow*, and 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 $p_j = p_{2,i}$.

Last, 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 $adj(p_a, p_b) = 1$ if $b = a + 1$, and 0 otherwise. $adj(p_a, p_b)$ indicates whether or not two foreground packets are adjacent with respect to the other foreground packets. In other words, $adj(p_{1,i}, p_{2,j}) = 1$ implies that there is some k for which $p_{1,i} = p_k$ and $p_{2,j} = p_{k+1}$.

2.1 Comparison Tests

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 a POC.
Step 4:	Else, the flows do not share a POC.

Figure 2: A comparison test.

Our techniques for detecting whether or not a pair of flows share 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 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, the degree of correlation decreases as the time between the packets' transmissions is increased.
- The losses or delays experienced by two packets that do not experience the same POC will exhibit little or no correlation.

Our idea is to collect samples in such a way that we ensure that, on average, the time between arrivals at the POC for a sample pair of packets within a *single* flow is *larger* than that for a sample that contains two packets, one from *each* flow. We then simply compare the levels of correlation between packets within each pair. If we find that on average, the correlation is higher in the pair of packets drawn from the two flows (that are closer together in time than packets drawn from separate flows), we conclude that this is due to the packets in the two-flow pairs passing through a common POC in shorter time intervals than do the pairs of a single flow, and hence the POC is shared. Otherwise, we conclude that the lower correlation between pairs of packets sent closer together in time is due to the packets being congested at different, independent POCs, hence the flows do not share. We refer to this simple method of making this determination as a *comparison test*. The basic steps are reiterated in Figure 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 probe traffic, making it difficult to determine the right value for α.

2.2 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 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, the techniques are clearly not robust for all possible distributions of traffic. One such 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 using a probing bandwidth as little as half a kilobyte per second each.

2.3 The 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 is high, and increases 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 $j = 1, 2$.

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

$$M_x = \Pr(L_{2,i} = 0 \mid L_{1,j} = 0, adj(p_{1,j}, p_{2,i}) = 1) \quad (1)$$

$$M_a = \Pr(L_{2,i} = 0 \mid L_{2,i-1} = 0) \quad (2)$$

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, barring reordering within the network, the order in which packets are transmitted from the co-located senders is the order in which they arrive at the POC. In the Y-topology, on the other hand, it would be difficult in practice to sequence the order of arrivals of packets over both foreground flows based on the departure times from the senders, since propagation times to the POC from the two senders can differ and need not be known. Receivers can determine the sequence of arrivals of foreground packets that were not lost at the POC with respect to one another, but they cannot precisely place the order in which packets that were lost at a POC initially arrived at the POC. 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

[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 $p_{2,j+1}$.

$p_{2,i+1}$ (or whether $p_{1,j+1}$ preceded $p_{2,i}$, etc.)[2] It follows that co-located receiving hosts cannot determine whether or not $adj(p_{1,j}, p_{2,i}) = 1$ when both $p_{1,j}$ and $p_{2,i}$ are lost. As a consequence, we cannot compute the cross- and auto-measures defined by (1) and (2).

Instead, we define separate cross- and auto-measures for use in the Y-topology. We define $adj_R(p_i, p_j) = 1$ if and only if $i < j, L_i = 1, L_j = 1$, and $L_k = 0$ for all $i < k < j$, and let $adj_R(p_i, p_j) = 0$ otherwise. In other words, $adj_R(p_i, p_j)$ 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:

$$M_x = \Pr(L_{2,i-1} = 0 \mid L_{2,i} = 1, L_{1,j-1} = 0, L_{1,j} = 1, adj_R(p_{1,j}, p_{2,i}) = 1) \tag{3}$$

$$M_a = \Pr(L_{2,i} = 0) \tag{4}$$

M_x is the conditional probability that for any i, a packet, $p_{2,i-1}$, from f_2 is lost, given that i) the subsequent packet from $f_2, p_{2,i}$, is received, ii) the nearest foreground packet that is subsequently received after $p_{2,i}$ is from f_1 ($p_{1,j}$ for some j), and iii) that the preceding packet from f_1, $p_{1,j-1}$, is lost. The reader should note that the sequence of events used in equation (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 larger than that of (3) only when the POCs are shared.

2.4 The 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:

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

where $E[a_i] \equiv \sum_{s \in S} a_i/|S|$. Define D_i to be the *observed delay* incurred by packet i. the observed delay of a packet, p_i, is measured by timestamping the packet with d_i, the sender's current clock time at the time of its departure from the sending host, timestamping the packet with a_i, the receiver's clock time at the time of its arrival at the receiving host, and taking the difference, i.e., $D_i = a_i - d_i$. Note that because of unsynchronized clocks and/or clock drift, the observed delay we compute need not equal the true time elapsed between departure from the sender and arrival at the receiver. The lack of time synchronization between clocks will have little impact on 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 [10, 11]. 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, we use the following for M_x and M_a:

$$M_x = C(\{(D_{1,i}, D_{2,j}) : adj(p_{1,i}, p_{2,j}) = 1\}) \tag{6}$$

$$M_a = C(\{(D_{2,i}, D_{2,i+1})\}) \tag{7}$$

[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.

M_x is the correlation coefficient computed from the delays of pairs of packets that are adjacent with respect to the foreground flow. 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 adjacent arrivals (transmissions) within f_2.

3 Queuing Analysis

In this section, we demonstrate the correctness of the comparison tests described in Section 2 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.

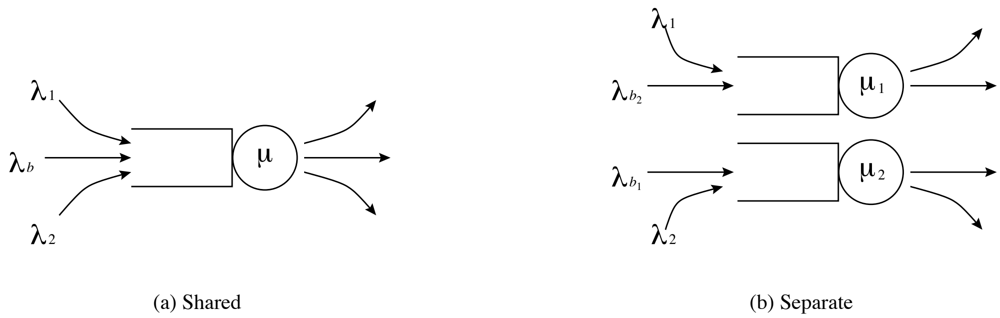


Figure 3: Queuing models for shared and separate POCs.

Figure 3 depicts our models of (a) a shared POC for flows f_1 and f_2, and (b) separate POCs for the flows. A POC is represented by a queue. A shared POC (Figure 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 (Note that unless otherwise stated, the background traffic need not be Poisson). The queue services packets at a rate of μ. Separate POCs (Figure 3(b)) are represented by two queues. Packets from f_i enter one queue whose background traffic arrival rate is λ_{b_i} and whose service rate is $\mu_i, i = 1, 2$. Unless otherwise stated, there are no restrictions on these rates, or the distributions between arrivals / service completions. 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.

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 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 Equations (3) and (4) as functions of $\lambda_1, \lambda_2, \lambda_b$, 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 $\lambda_1, \lambda_2, \lambda_b$, and K, and found equation (3) to always be larger than (4) (the desired result). These results are presented in Appendix B.

In the subsequent subsection, we demonstrate that, given all queues are M+G/G/1/∞ queues (foreground traffic remains Poisson, background traffic and service times are i.i.d. using any 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.

3.1 The 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, $\omega = (e_1, e_2, \cdots, e_m)$, and let $Q_i(\omega, j)$ be the number of packets in q_i after the jth event in ω is applied to the queue. We write $Q_i(\omega, 0)$ to be the number of packets in the queue prior to 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., $\omega' = (f_1, f_2, \cdots, f_{m'})$ where for some $i \geq 1, m' = m - i + 1$ and $f_j = e_{j+i-1}$ where $1 \leq j \leq m'$ for some $i \geq 1$. Then $Q_1(\omega', j) \geq Q_1(\omega, j + i)$.*

Proof: Consider the application of ω to the queue. After applying the (possibly empty) prefix $(e_1, \cdots, e_{i-1})$ 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 $0 \leq j \leq m - i + 1$. ∎

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

Proof: Let $\omega = (e_1, \cdots, e_{m_\omega})$ 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. Let g_p map any ω to its longest prefix whose final event is a 1. i.e., $g_p(\omega) = (e_1, \cdots, e_n)$ where $e_n = 1$ and $e_i \neq 1$ for $n < i \leq m_\omega$. 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., $g_s(\omega) = (e_{n+1}, \cdots, e_{m_\omega})$ 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.

Define P to be the probability measure over S.[3] 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 $P(\omega) = P(g_p(\omega))P(g_s(\omega))$.

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 contains no event $e_i = 2$, and 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 $\omega_2 = g_s(\omega)$.

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

[3] We emphasize that P is a probability *measure* [12] 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 $\sum_{\omega \in S'} P(\omega)$.

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 $L_K(\omega_p \cdot \omega_s) \leq L_K(\omega_p \cdot \omega_s)$. Defining π_i to be the steady-state probability that the queue length is i, we have

$$\Pr(L_{2,j+1} = 0, L_{2,j} = 0) = \sum_{\omega \in S} \pi_k P(\omega) X(\omega) L_K(\omega) \quad = \quad \pi_k \sum_{\omega \in S} P(\omega) X(\omega) L_K(\omega) \tag{8}$$

$$\Pr(L_{2,j} = 0) = \sum_{\omega \in S} \pi_k P(\omega) X(\omega) \quad = \quad \pi_k \sum_{\omega \in S} P(\omega) X(\omega) \tag{9}$$

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

$$\begin{aligned}
\Pr(L_{2,j+1} = 0 \mid L_{2,j} = 0) = \frac{\sum_{\omega \in S} P(\omega) X(\omega) L_K(\omega)}{\sum_{\omega \in S} P(\omega) X(\omega)} &= \frac{\sum_{\omega_p \in S} \sum_{\omega_s \in S} P(\omega_p) P(\omega_s) X_p(\omega_p) X_s(\omega_s) L_K(\omega_p \cdot \omega_s)}{\sum_{\omega_p \in S} \sum_{\omega_s \in S} P(\omega_p) P(\omega_s) X_p(\omega_p) X_s(\omega_s)} \\
\leq \quad \frac{\sum_{\omega_p \in S} \sum_{\omega_s \in S} P(\omega_p) P(\omega_s) X_p(\omega_p) X_s(\omega_s) L_K(\omega_s)}{\sum_{\omega_p \in S} \sum_{\omega_s \in S} P(\omega_p) P(\omega_s) X_p(\omega_p) X_s(\omega_s)} & \qquad (10) \\
= \quad \frac{\left(\sum_{\omega_p \in S} P(\omega_p) X_p(\omega_p)\right) \left(\sum_{\omega_s \in S} P(\omega_s) X_s(\omega_s) L_K(\omega_s)\right)}{\left(\sum_{\omega_p \in S} P(\omega_p) X_p(\omega_p)\right) \left(\sum_{\omega_s \in S} P(\omega_s) X_s(\omega_s)\right)} &= \frac{\sum_{\omega_s \in S} L_K(\omega_s) X_s(\omega_s) P(\omega_s)}{\sum_{\omega_s \in S} P(\omega_s) X_s(\omega_s)} \\
= \quad \frac{\sum_{\omega_s \in S} \pi_k P(1 \cdot \omega_s) L_K(\omega_s) X_s(\omega_s)}{\sum_{\omega_s \in S} \pi_k P(1 \cdot \omega_s) X_s(\omega_s)} &= \Pr(L_{2,j} = 0 \mid (L_{1,i} = 0), adj(p_{1,i}, p_{2,j}) = 1)
\end{aligned}$$

where we use $L_K(\omega_p \cdot \omega_s) \leq L_K(\omega_s)$ to establish the inequality in (10). This inequality is strict since there exists at least one $\omega = \omega_p \cdot \omega_s$ where $X(\omega) L_K(\omega) < X_p(\omega_p) X_s(\omega_s) L_K(\omega_s)$. ∎

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

Proof: Because all arrivals and departures from the queues are Poisson, arrivals and departures into the first queue have no impact on the second queue, and can be ignored when considering the status of the second queue. Hence, by PASTA [13], $\Pr(L_{2,j} = 0 \mid (L_{1,i} = 0), adj(p_{1,i}, p_{2,j})) = \Pr(L_{2,j} = 0)$ for any packet in f_2. Thus, we need only prove that $\Pr(L_{2,j} = 0) < \Pr(L_{2,j+1} = 0 \mid L_{2,j} = 0)$.

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 $\omega = (e_1, e_2, \cdots, e_{m_\omega})$ 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 the queue length is $i \leq K$ when the capacity is K, then applying the sequence $\omega = (e_1, \cdots, e_{m_\omega})$ causes the last event, e_{m_ω} to result in a packet drop, and 0 otherwise.

We can rewrite our probabilities for which we need to prove the inequality as follows:

$$\Pr(L_{2,j} = 0) \quad = \quad \sum_{\omega \in S} \sum_{i=0}^{K} \pi_i P(\omega) X(\omega) Y_i(\omega) \tag{11}$$

$$\Pr(L_{2,j+1} = 0 \mid L_{2,j} = 0) \quad = \quad \frac{\sum_{\omega \in S} \pi_k P(\omega) X(\omega) Y_K(\omega)}{\sum_{\omega \in S} \pi_k P(\omega)} = \sum_{\omega \in S} P(\omega) X(\omega) Y_K(\omega) \tag{12}$$

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:

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

which completes the proof. ∎

3.2 The 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 queuing 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 Appendix A:

Lemma 3 *Let G be a non-decreasing 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 non-increasing with $0 < \lim_{x \to \infty} G(x) < G(0)$, then $\int_{x=0}^{\infty} G(x) f(x)\, dx > \int_{x=0}^{\infty} G(x) g(x)\, dx$*

We define A_i to be the time of arrival of p_i at the queue. The following Lemma implies that correlations between two foreground packets decreases as the time between their arrivals increases. Its proof appears in Appendix A.

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 $E[D_i D_{i+1}] > E[D_i D_{i+n}]$ 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, $\forall i, j, k, m = 1, 2m E[(D_{1,i})^2] = E[(D_{1,j})^2] = E[(D_{2,k})^2] = E[(D_{2,m})^2]$. Hence, to prove the theorem, we need only show that $E[D_{1,i} D_{2,j} | (adj(p_{1,i}, p_{2,j}) = 1)] > E[D_{2,i} D_{2,i+1}]$.

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, $E[D_{1,i} D_{2,j} | (adj(p_{1,i}, p_{2,j}) = 1)] = E[D_i D_{i+1}]$. Alternatively, as defined in (7), M_a is the correlation coefficient between packets from f_2 that are adjacent with respect to f_1 (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 $i < j < i+n$ 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. Using the fact that packet delays are independent of their marking ($E[D_i D_{i+n} | \Lambda_1(i, i+n), \Lambda_2(i, i+n)] = E[D_i D_{i+n}]$), then

$$
\begin{aligned}
E[D_{2,j}D_{2,j+1}] &= \sum_{n=1}^{\infty} E[D_i D_{i+n} \mid \Lambda_1(i,i+n)=1, \Lambda_2(i,i+n)=1] \Pr(\Lambda_1(i,i+n)=1 \mid \Lambda_2(i,i+n)=1) \\
&< \sum_{n=1}^{\infty} E[D_i D_{i+1}] \Pr(\Lambda_1(i,i+n)=1 \mid \Lambda_2(i,i+n)=1) = E[D_i D_{i+1}] \qquad (13)
\end{aligned}
$$

where Lemma 4 yields the above inequality. ∎

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

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 $p_{2,i}$. delay increases, we would expect to see a higher delay for $p_{2,i+1}$ as well. A detailed proof is given in Appendix A.

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 across packets in the two 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:

$$
\begin{aligned}
&E[D_{2,i+1}D_{2,i}] - E[D_{2,i+1}]E[D_{2,i}] \\
&= \int_{x=0}^{\infty} x \Pr(D_{2,i}=x) E[D_{2,i+1}|D_{2,i}=x]\, dx - \int_{x=0}^{\infty} x \Pr(D_{2,i}=x) E[D_{2,i+1}]\, dx \qquad (14)
\end{aligned}
$$

By Lemma 5, $E[D_{2,i+1}|D_{2,i} = x]$ is an increasing function of x. Applying 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 (14) is larger than 0, which completes the proof. ∎

4 Performance in Simulation

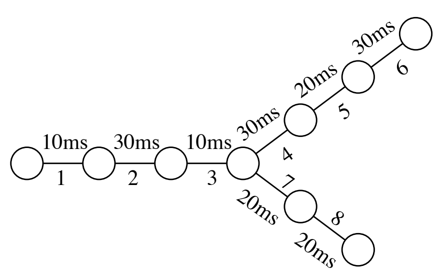
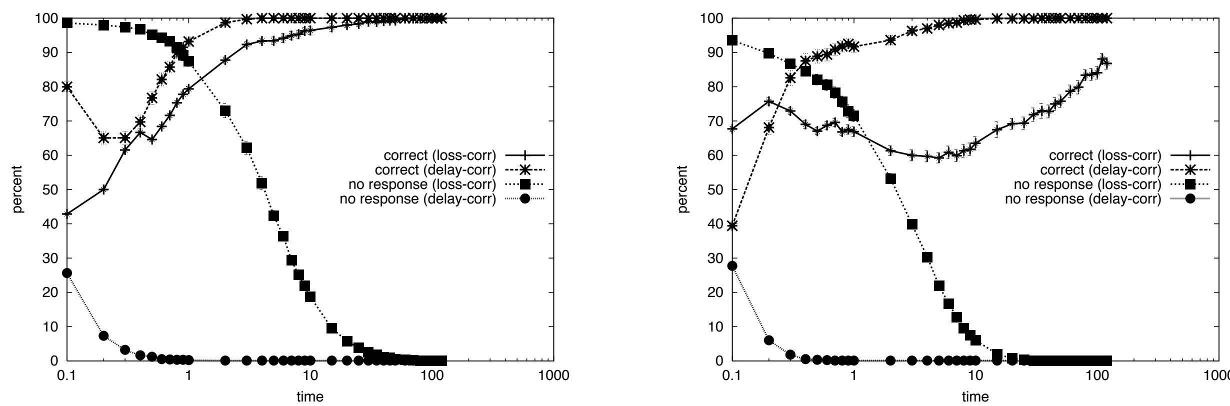


Figure 4: Topology used in simulation experiments.

In this section, we use simulation to examine scenarios where POCs are either shared by both flows, or are not shared by both flows. Figure 4 demonstrates the topology on which we run our simulations using the ns-2 simulator [14]. For the Y topology, probe receivers are connected to the left-most 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 limit the potential POCs by assigning links that we want congested to process at a rate of 1.5Mbs, and links that we do not want congested process at a rate of 1000Mbs. The links that are assigned the 1.5Mbs capacity are either the set of links numbered 1 through 3 or else are the set of links numbered 4 through 8. 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 1.5Mbs capacity (i.e., there is no background traffic on the high bandwidth links). 10 through 20 background flows are placed on the path of each probe, each background flow uses the TCP protocol with probability of .75. Otherwise, it is a CBR flow with on-off service times. The CBR rate is uniformly chosen between 10 and 20 Kbs, and the average on time and off time is chosen independently between 0.2 and 3.0 seconds.

For each of the four configurations (Y topology or Inverted-Y topology, shared or independent POCs), 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 $t = 120$.



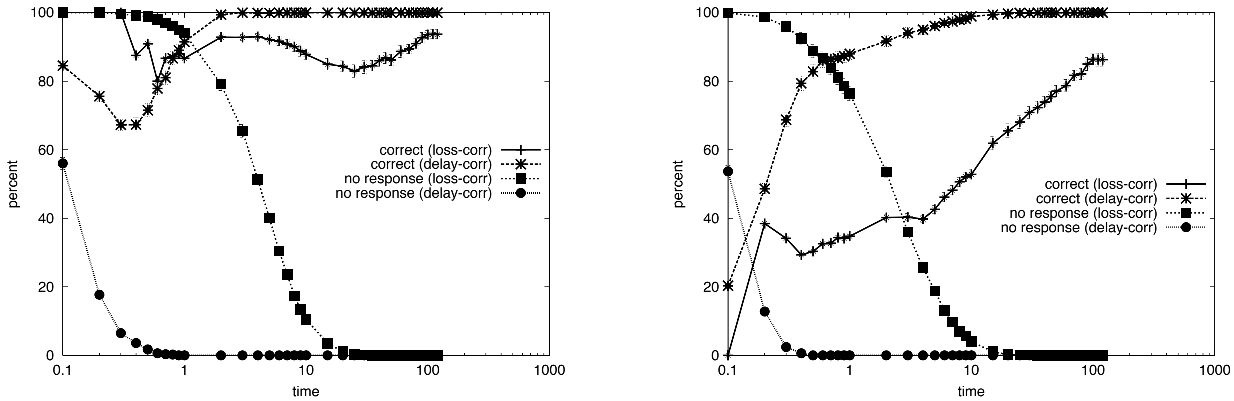
(a) Independent congestion

(b) Shared congestion

Figure 5: Inverted-Y topology

Figure 5 plots the percentage of experiments run over the Inverted-Y topology that correctly infer whether or not the flows share as a function of time using the loss-corr and delay-corr techniques. In Figure 5(a), the flows do not share the POC in any of the 1000 experiments, whereas in Figure 5(b), the flows share the same POC in all 1000 experiments. In each experiment, both probe flows send 20 byte packets at an average rate of 25 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 one sample before it forms a hypothesis). Curves labeled "correct" plot the percentage of tests whose hypothesis is correct at the time indicated on the x-axis. 95% level confidence intervals are generated by averaging over twenty samples at a time, such that the distribution of the average of the samples is approximately normal.

We can make several observations from these graphs. First, the delay-corr technique is able to correctly assess whether or not a POC is shared by an order of magnitude faster than 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 seconds 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 case loss-corr technique, the percentage of hypotheses that are correct initially decreases with time. This is likely due to the fact that with low loss rates, both measures are initially estimated to be 0, and in a tie, the hypothesis is that the POCs are independent.



(a) Independent congestion

(b) Shared congestion

Figure 6: Y topology

Figure 6 plots similar results for a Y-topology as those in Figure 5. 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 slightly 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.

We also examined the applicability of the comparison tests when the routers initiated Random Early Detection, and found no significant impact on our reported results for the delay-corr technique. However, we observed that the loss-corr technique failed to identify shared POCs in more than half the cases. This is 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 that probe loss will become bursty.

5 Detecting sharing in a practical setting

This paper focuses on demonstrating that from end-hosts, we can distinguish between two flows that are congested by the same set of resources from two flows that are congested by a mutually exclusive set of resources. However, the true nature of congestion in the Internet is still not well understood. If we assume that a flow can experience congestion at several POCs within a short period of time, then the POCs for two flows need not be identical or mutually exclusive, but may partially overlap. If such is the case, then a simple yes/no answer is not always sufficient.

We now introduce one possible way of measuring the *level of sharing*, $\mathcal{L}_{1,2}$, a metric that quantifies the degree to which the POCs of f_1 overlap. To compute this metric, we require that the two probe rates be equal (i.e., $\lambda_1 = \lambda_2$). We define a new adjacency function, $adj_2(p_{1,j}, p_{2,i})$ which equals 1 iff any foreground packets arriving at the POC between packets $p_{1,j}$ and $p_{2,i}$ are from f_1.[4]

We define the *share-measure*, M_{share} similarly to the definitions of the cross-measure, M_x, except we replace $adj(p_{1,j}, p_{2,i})$ with $adj_2(p_{1,j}, p_{2,i})$. For instance, in the case of delay, $M_{share} = C(\{(D_{1,i}, D_{2,j}) : adj_2(p_{1,i}, p_{2,j}) = 1\})$. Because we are using Poisson probes, the conditioning on $adj_2(p_{1,i}, p_{2,j}) = 1$, combined with the memoryless nature of the Poisson distribution ensures that the distribution of times between pairs used for samples in both M_a and M_{share} are the same. We also define an *independence measure*, M_{ind}, which estimates M_{share} if the flows share no bottleneck. For instance, for the loss-corr technique, $M_{ind} = \Pr(L_{2,i} = 0)$, and for the delay-corr technique, it is 0. The level of sharing is then $\mathcal{L}_{1,2} = (M_{share} - M_{ind})/(M_a - M_{ind})$.

$\mathcal{L}_{1,2}$ measures the "level of information" that a loss (delay) observation on a packet from f_1 yields about the loss (delay) experienced by the next packet from f_2. Assuming flows cannot be negatively correlated, a value of zero indicates that observations of packets from f_1 give no useful information pertaining to f_2. A positive value less than one indicates that it yields some information, but less than what can be obtained by using loss and delay measurements from previous transmissions in f_2. A value equal to one means that the levels of information are the same, and a value larger than one means that observations on packets from f_1 yield more information than observing the previous transmissions from f_2. This last case can only occur due to statistical error in estimating the measures used to construct $\mathcal{L}_{1,2}$, or when f_2's POC covers multiple routers, and f_1 utilizes those routers that are most heavily congested. The measure increases as the POCs for f_1 increase in the severity to which they cause congestion for f_2. They do not, however, indicate the extent to which f_1 causes the congestion at the POC.

6 Actual Traces

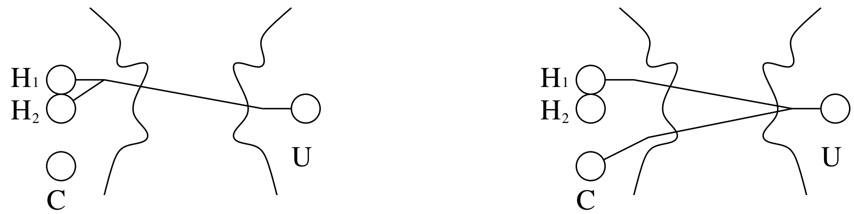
C	Columbia (New York)	M_1	UMass-1	M_2	UMass-2	U	UCL (London)
S	AT&T-San Jose (California)	M_3	UMass-3	A	ACIRI (California)		

Table 1: Site name abbreviations

We have demonstrated the robustness of our comparison tests through queueing analysis and simulation. Now, we give evidence that these tests work in practice. We apply the tests over the Internet, choosing end-system locations such that we can be reasonably sure as to whether or not the flows share. 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, UK), Columbia (New York City), AT&T-San Jose (California), and three of our own machines, labeled "UMass-1" through "UMass-3". Table 1 presents a shorthand notation for these sites that is used in the subsequent figures and tables.

Our first set of experiments involve four of these 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 (Figure 7(a)), 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, the two flows will share. 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 disjoint along the trans-Atlantic links, as well as all links in the U.S. (Figure 7(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.

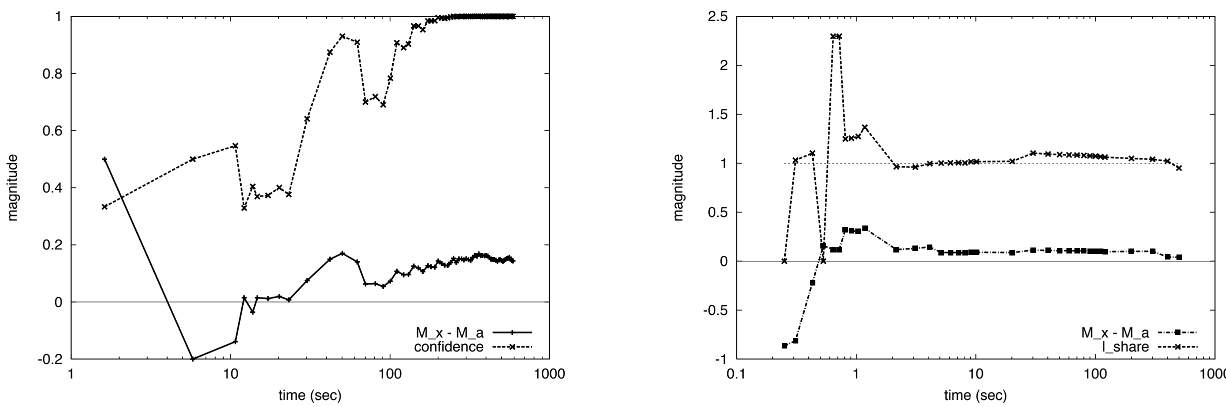
[4]Note that for a given j, there can be several i that satisfy $adj_2(p_{1,j}, p_{2,i}) = 1$, whereas there is a unique i that satisfies $adj(p_{1,j}, p_{2,i}) = 1$. We emphasize this to point out that the lack of uniqueness is not a problem, but does increase the correlation between successive samples.



(a) Shared topology

(b) Independent topology

Figure 7: Experimental topologies



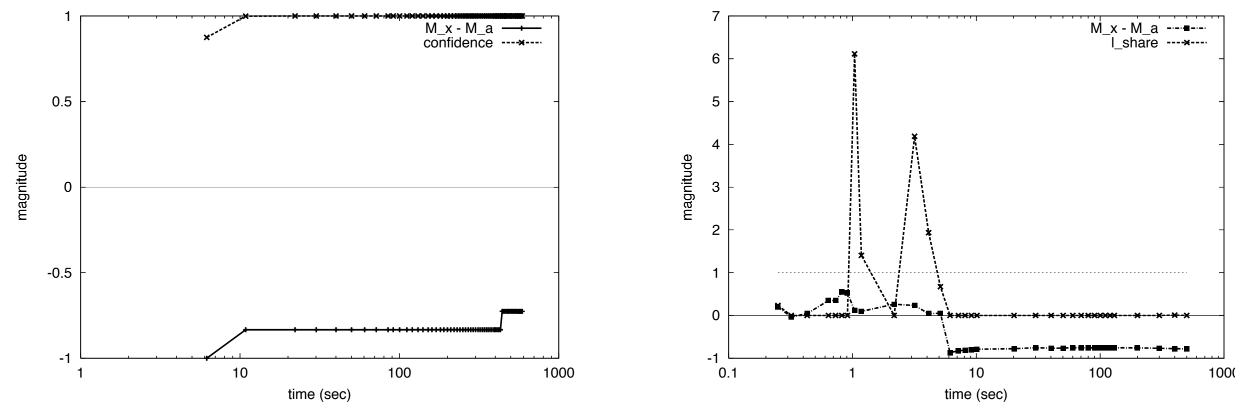
(a) loss-corr

(b) delay-corr

Figure 8: Europe to US, Inverted-Y, correlated

Figure 8 presents the results for the Inverted-Y topology where the U.S. end-systems reside in the same LAN. Figure 8(a) plots the results of the loss-corr test with time, measured in seconds, plotted via a log-scale along the x-axis. The bottom curve is the difference, $M_x - M_a$, between the two measures of the comparison test. After the 15th second, the value remains positive, which means that the test hypothesizes that the POCs for the flows are shared. The top curve is the level of confidence, computed using a permutation test [15], that does not require that the distribution is normal. A value of 1 means that we have 100% confidence that the hypothesis is correct. Figure 8(b) plots the results of the delay-corr test, with time plotted via a log-scale along the x-axis. The bottom curve again indicates the difference, $M_x - M_a$, between the measures. Since the value is positive, the test also hypothesizes that the flows share a POC. The top curve is a computation of $\mathcal{L}_{1,2}$. The fact that $\mathcal{L}_{1,2}$ converges to one indicates that the level of correlation between packets across flows f_1 and f_2 is similar to the correlation seen between packets in f_2, which matches our assumption that these two flows observe almost identical POCs.

Figure 9 presents results for the Inverted-Y topology where different paths are taken across trans-Atlantic links to reach the two end-systems in the U.S. In Figure 9(a), the bottom curve, which plots, $M_x - M_a$ for the loss-corr technique, is negative. Hence, the test hypothesizes that the POCs for the two flows are not shared. The top curve, which shows the level of confidence as computed via the permutation test, shows that the confidence in this hypothesis is close to 100% after only 10 seconds. The bottom curve in Figure 9(b) plots $M_x - M_a$ for the delay-corr technique. Because the value is negative, the test hypothesizes that the flows do not share. The top curve in the Figure plots $\mathcal{L}_{1,2}$ as it varies with time. Its value remains close to zero, which indicates that observing the delays of packets from f_1



(a) loss-corr

(b) delay-corr

Figure 9: Europe to US, Inverted-Y, uncorrelated

will give no information about the delays observed by packets in f_2. These results are consistent with our evaluation of the topologies: the POCs for these two flows are separate.

6.1 Results Summary

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

Table 2: Trace results

Table 2 summarizes the results of the other experiments performed during the middle of the day on November 1 and November 3, 1999 using the hosts listed in Table 1. Each experiment ran for 600 seconds, with each 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 timestamp 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.

The first column in the table 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 1. For the Y topology, the labeling, (A,B $\rightarrow$ C), indicating that senders at host A and host B are transmit 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 transmit 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 portions of the paths that are not shared. These values were obtained through two calls to `traceroute` that were executed during the experiment, one for each source-destination pair (the call to `traceroute` was always initiated by the probe sender(s)). We consider the shared links to be 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 considered 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 value is returned that is less than zero, we assume that the delay on this sequence of links is negligible, and write the delay as ~ 0.

For the Y topology, the entry, (x,y $\rightarrow$ z), $x, y, z \in \Re$ that is associated with the labelling, (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 labelling, (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 non-shared 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" that we are able to make given our limited access to routing information.

The next column presents the loss rates. An entry, a, b : c, associated with the labelling, (A,B $\rightarrow$ C), or the labelling, (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 percents, so values less than one indicate that fewer than one out of every one hundred packets were 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 seconds; to its right is the time of the experiment when the hypothesis was last changed. 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 sixteen 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 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

[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.

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.

7 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 [7] 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.

8 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.

A Proofs of Delay Lemmas

Proof of Lemma 3:

Proof: If $G(x)$ is non-decreasing, we have that $0 < \int_{x=0}^{\gamma} G(x)(f(x)-g(x))\ dx < G(\gamma)\int_{x=0}^{\gamma}(f(x)-g(x))\ dx$. Also, $0 < G(\gamma)\int_{x=\gamma}^{\infty}(g(x)-f(x))\ dx < \int_{x=\gamma}^{\infty} G(x)(g(x)-f(x))\ dx$, or equivalently (multiplying by -1), $\int_{x=\gamma}^{\infty} G(x)(f(x)-g(x))\ dx < G(\gamma)\int_{x=\gamma}^{\infty}(f(x)-g(x))\ dx < 0$. Hence, $\int_{x=0}^{\infty} G(x)(f(x)-g(x))\ dx = \int_{x=0}^{\gamma} G(x)(f(x)-g(x))\ dx + \int_{x=\gamma}^{\infty} G(x)(f(x)-g(x))\ dx < G(\gamma)\int_{x=0}^{\gamma}(f(x)-g(x))\ dx + G(\gamma)\int_{x=\gamma}^{\infty}(f(x)-g(x))\ dx = G(\gamma)\int_{x=0}^{\infty}(f(x)-g(x))\ dx = 0$. Thus, $\int_{x=0}^{\infty} G(x)(f(x)-g(x))\ dx < 0$, which gives $\int_{x=0}^{\infty} G(x)f(x)\ dx < \int_{x=0}^{\infty} G(x)g(x)\ dx$. The case where $G(x)$ is non-increasing is proven similarly. ∎

Proof of Lemma 4:

Proof: $D_{i+n} = E_{i+n} - A_{i+n}$, 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 i) p_i's service is completed, and then ii) all background packets that arrive between p_i and p_{i+n} are and all foreground packets $p_{i+1}, \cdots, p_{i+n}$ 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 time the processor is idle (no jobs in queue) during the time interval $[x,y)$.

Substituting in for E_{i+1}, we obtain $D_{i+n} = D_i - t_n + \sum_{j=1}^{N(A_i,A_{i+n})} s_j + \sum_{j=1}^{n} S_j + \gamma(A_i, A_{i+n})$, 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}, 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 $E[\sum_{j=1}^{N(x,y)} s_j] = E[N(x,y)]E[s_j]$, where j can be arbitrary (because service times are i.i.d.). The rate at which packets can be processed at the queue is $\mu = 1/E[s_j] = 1/E[S_j]$.[6] Last, 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 1st and nth arrivals of Poisson process with rate λ_f), we have that $E[t_n] = 1/\lambda_f$.

We now prove the result by showing that for $n > 1, E[D_i D_{i+n}] - E[D_i D_{i+1}] = E[D_i(D_{i+n} - D_{i+1})] < 0$. 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

$$\begin{aligned} E[D_i(D_{i+n} - D_{i+1})] &= E[D_i](-E[t_n] + E[N(A_{i+1}, A_{i+n})]E[s] + (n-1)E[s]) + E[D_i \gamma(A_{i+1}, A_{i+n})] \\ &= E[D_i](E[t_n](-1 + \lambda_b/\mu) + (n-1)/\mu) + E[D_i \gamma(A_{i+1}, A_{i+n})] \end{aligned} \tag{15}$$

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 non-negative service time can only reduce the expected idle time.

Applying this result into Equation (15), and substituting $E[t_n] = (n-1)/\lambda_f$, we get

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

■

Proof of Lemma 5:

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

Proof: The proof is via sample-path analysis. We construct a sample path , ω, that starts at the arrival of $p_{2,i}$, and ends with the completion of service of $p_{2,i+1}$ at the server. We represent $\omega = (N, \{t_1, \cdots, t_n\}, \{s_1, \cdots, s_N\}, S)$, where N is the number of packets that arrive during the interval covered by ω, t_j is the arrival time of the j arrival in the time interval covered by ω, s_j is the processing time of this packet within the queue, and S is the processing time for $p_{2,i+1}$. Let X be a random variable on $W = \{\omega\}$ where $X(\omega) = 1$ if and only if the events at the queue in the range of time covered by ω are correctly described by ω. Note that X is independent of $D_{2,i}$. Hence, $\forall x_1, x_2 > 0, \Pr(X(\omega) = 1 | D_{2,i} = x_1) = \Pr(X(\omega) = 1 | D_{2,i} = x_2) = \Pr(X(\omega) = 1)$. Note that given the delay of $p_{2,i}$ and ω, one can compute the delay experienced by $p_{2,i+1}$. Let $D(D_{2,i}, \omega)$ be this delay.

For a given ω, as $D_{2,i}$ is increased, then the packet that arrived at time t_1 cannot begin being processed until a later time, and hence its time to finish processing increases. Inductively, it can be shown that the time by which the packet that arrived at time t_N completes being processed can only increase due to an increase in $D_{2,i}$. Thus, the same holds true for the completion time of the processing for $p_{2,i+1}$. This makes $D(D_{2,i}, \omega)$ an increasing function of $D_{2,i}$. Given $x_1 < x_2$,

$$\begin{aligned} E[D_{2,i+1} | D_{2,i} = x_1] &= \frac{\sum_{\omega \in W} D(x_1, \omega) \Pr(X(\omega) = 1 | D_{2,i} = x_1)}{\sum_{\omega \in W} \Pr(X(\omega) = 1 | D_{2,i} = x_1)} \\ = \frac{\sum_{\omega \in W} D(x_1, \omega) \Pr(X(\omega) = 1 | D_{2,i} = x_2)}{\sum_{\omega \in W} \Pr(X(\omega) = 1 | D_{2,i} = x_2)} &< \frac{\sum_{\omega \in W} D(x_2, \omega) \Pr(X(\omega) = 1 | D_{2,i} = x_2)}{\sum_{\omega \in W} \Pr(X(\omega) = 1 | D_{2,i} = x_2)} \\ &= E[D_{2,i+1} | D_{2,i} = x_2]. \end{aligned}$$

The strict inequality above is due to the fact that there is a set $W' \subset W$ with non-zero measure where $\omega \in W' \rightarrow D(x_1, \omega) < D(x_2, \omega)$. ∎

B Analysis of the loss-corr technique, Y-topology

In this section, we derive closed-form recursive solutions that give separate solutions for M_x and M_a when the bottleneck is shared, and when the bottlenecks are separate, all for the Y topology. We assume that the POC for each flow is an M/M/1/K queue (the same queue when the POC is shared).

B.1 Y Topology, Shared Bottleneck: M_x

Let f_b represent the aggregate flow consisting of all background traffic that passes through the shared POC. Define $p_1(j)$ to be the probability that the next event in the system is the arrival of a packet from f_1, conditioned on the event that the queue length is j. Define $p_2(j)$ and $p_b(j)$ similarly for the events corresponding to packet arrivals from f_2 and f_b, and $p_s(j)$ to be the probability that the next event is a service completion conditioned on the event that the queue length is j. Since the flow arrivals and the completion time are all exponentially distributed (i.e., memoryless), we have:

$$p_1(j) = \lambda_1/\gamma_j, \quad p_2(j) = \lambda_2/\gamma_j, \quad p_b(j) = \lambda_b/\gamma_j \tag{16}$$

$$p_s(j) = \left\{ \begin{array}{ll} 0 & j = 0 \\ \mu/\gamma_j & j > 0 \end{array} \right\} \tag{17}$$

where

$$\gamma_j = \left\{ \begin{array}{ll} \lambda_1 + \lambda_2 + \lambda_b & j = 0 \\ \lambda_1 + \lambda_2 + \lambda_b + \mu & j > 0 \end{array} \right\} \quad (18)$$

γ_j differs at $j = 0$ since there are no service completions when $j = 0$. To simplify notation, we write p_1 in place of $p_1(j)$ when it is implied that $j > 0$. We do this similarly for p_2, p_b, and p_s.

We now compute $M_x = \Pr(L_{2,i-1} = 0 \mid L_{2,i} = 1, L_{1,j-1} = 0, L_{1,j} = 1, adj_R(p_{1,j}, p_{2,i}) = 1)$. The value of M_x depends on the success or failure of receiving and order of receipt of four packets in particular: $p_{1,j-1}, p_{1,j}, p_{2,i-1}$, and $p_{2,i}$. To do this, we derive four regular expressions that represent the sequence of foreground arrival events at the bottlneck queue. Three of these regular expressions are mutually disjoint (i.e., no two regular expressions contains an identical sequence of arrival events). Furthermore, the union of the set of sequences generated by all three regular expressions is the set of sequences that satisfy both of the following:

$$L_{2,i-1} = 0 \quad (19)$$

$$L_{2,i} = 1, L_{1,j-1} = 0, L_{1,j} = 1, adj_R(p_{1,j}, p_{2,i}) = 1 \quad (20)$$

The fourth regular expression is a superset of this union, and the set of sequences that it generates is the set of sequences that satisfy (20) (and need not satisfy (19)).

Let Γ_1^i be the event that $p_{1,j}$ is admitted into the bottleneck queue. Let Λ_1^{i-1} be the event that $p_{1,j-1}$ is dropped at the bottleneck queue. Similarly, define Γ_2^j and Λ_2^{j-1} to be the respective events that $p_{2,i}$ is accepted into the queue, and that packet $p_{2,i-1}$ is dropped from the POC. Based on our packet ordering assumptions, since the co-located receivers receive $p_{1,j}$ prior to packet $p_{2,i}$, the two packets arrived in this same order at the POC. Furthermore, since packet ordering is maintained within a flow, $p_{1,j-1}$ must have arrived at the POC prior to $p_{1,j}$, and packet $p_{2,i-1}$ must have arrived prior to packet $p_{2,i}$. Hence, the following three orderings of arrivals at the receiver are the only orderings that satisfy both (19) and (20):

- $\Lambda_1^{i-1}\Lambda_2^{j-1}\Gamma_1^i\Gamma_2^j$
- $\Lambda_1^{i-1}\Gamma_1^i\Lambda_2^{j-1}\Gamma_2^j$
- $\Lambda_2^{j-1}\Lambda_1^{i-1}\Gamma_1^i\Gamma_2^j$

The set of sequences that contain the following ordering of $\Lambda_1^{i-1}, \Gamma_1^i$, and Γ_2^j is the set of sequences that satisfy (20) (and may or may not satisfy (19):

- $\Lambda_1^{i-1}\Gamma_1^i\Gamma_2^j$

We now produce the regular expressions, each one represents the set of all sequences of foreground events that produce one of the subsequences of events given above. Each regular expression starts with the first event in this subsequence, and ends with the last event in the subsequence. We write l_1 for a loss in f_1 of a packet other than $p_{1,j}$ or $p_{1,j-1}$, and write l_2 as a loss of a packet in f_2 other than the $p_{2,i}$ or $p_{2,i-1}$. Write g_1 and g_2 for receipt of these packets in f_1 and f_2, respectively. The above subsequences belong to the following regular expressions:

- $\Lambda_1^{i-1}\Lambda_2^{j-1}\Gamma_1^i\Gamma_2^j \rightarrow \Lambda_1^{i-1}(g_2 + l_2)^*\Lambda_2^{j-1}\Gamma_1^i l_1{}^*\Gamma_2^j$
- $\Lambda_1^{i-1}\Gamma_1^i\Lambda_2^{j-1}\Gamma_2^j \rightarrow \Lambda_1^{i-1}(g_2 + l_2)^*\Gamma_1^i(l_1 + l_2)^*\Lambda_2^{j-1}l_1{}^*\Gamma_2^j$
- $\Lambda_2^{j-1}\Lambda_1^{i-1}\Gamma_1^i\Gamma_2^j \rightarrow \Lambda_2^{j-1}(g_1 + l_1)^*\Lambda_1^{i-1}\Gamma_1^i l_1{}^*\Gamma_2^j$
- $\Lambda_1^{i-1}\Gamma_1^i\Gamma_2^j \rightarrow \Lambda_1^{i-1}(g_2 + l_2)^*\Gamma_1^i(l_1 + l_2)^*\Gamma_2^j$

Because each regular expression's first event is a packet loss, and because we have poisson arrivals which satisfy the PASTA property, the probability of a given sequence occuring in any of the expressions is simply the probability that the queue is full when the first packet in the sequence arrives at the queue, times the probability that the foreground events occur in the order specified by that sequence. We will only be interested in values of i and j after the system has been running for a significant period of time, so that the probability of the system's queue being full is simply the steady-state probability that the system's queue is full (contains k entries), π_k.

We must also account for arrivals of background traffic and service completions. Note that the arrival of a packet from f_b is possible at any point in time (i.e., between any two events in the regular expressions given above). A similar criterion holds for service completions, though we must ensure that service completions do not occur when the queue is empty.

We define several recursive functions that we use to compute useful conditional probabilities:

- $\phi_0(j)$ is the conditional probability that, given the queue has length j, a service completion occurs, and the queue eventually returns to length j, in between there are no arrivals from f_1 or f_2.
- $\phi_{0,2}(j)$ is the conditional probability that, given the queue has length j, a service completion occurs, and the queue eventually returns to length j, in between there are no arrivals from f_1 (but we allow arrivals from f_2).
- $\phi_{0,1}(j)$ is the conditional probability that, given the queue has length j, a service completion occurs, and the queue eventually returns to length j, in between there are no arrivals from f_2 (but we allow arrivals from f_1).
- $\phi_1(j)$ is the conditional probability that, given the queue has length j, a service completion occurs, and the queue eventually returns to length j, in between there are no arrivals from f_2, and there is a single arrival from f_1.
- $\psi_2(j)$ is the conditional probability that, given the queue has length j, a service completion occurs, and prior to returning to a length of j, a packet arrives from f_2 prior to any packets arriving from f_1 (this packet from f_2 can cause the queue to fill back to j). Any arbitrary ordering of packets is permitted once the packet from f_2 arrives.
- $\psi_{1,2}(j)$ is the conditional probability that, given the queue has length j, a service completion occurs, and prior to or returning to a length of j, one packet arrives from f_1, followed by a packet arrival from f_2 (this packet from f_2 can cause the queue to fill back to length j). Any arbitrary ordering of packets is permitted once the packet from f_2 arrives (but prior to the arrival of the first packet from f_2, only a single packet from f_1 arrives).

The solutions for these recursive functions follow:

$$\phi_0(j) = \left\{ \begin{array}{ll} p_s p_b(0) & j = 1 \\ \dfrac{p_s p_b}{1 - \phi_0(j-1)} & j > 1 \end{array} \right\} \tag{21}$$

$$\phi_{0,2}(j) = \left\{ \begin{array}{ll} p_s(p_b(0) + p_2(0)) & j = 1 \\ \dfrac{p_s(p_b + p_2)}{1 - \phi_{0,2}(j-1)} & j > 1 \end{array} \right\} \tag{22}$$

$$\phi_{0,1}(j) = \left\{ \begin{array}{ll} p_s(p_b(0) + p_1(0)) & j = 1 \\ \dfrac{p_s(p_b + p_1)}{1 - \phi_{0,1}(j-1)} & j > 1 \end{array} \right\} \tag{23}$$

For $j > 1, \phi_1(j) = p_s \left[\sum_{n=0}^{\infty}(\phi_0(j-1))^n\right] \left(p_1 + \phi_1(j-1)\left[\sum_{n=0}^{\infty}(\phi_0(j-1))^n\right] p_b\right)$. Hence,

$$\phi_1(j) = \left\{ \begin{array}{ll} p_s p_1(0) & j = 1 \\ \dfrac{p_s}{1-\phi_0(j-1)}\left(p_1 + \dfrac{\phi_1(j-1)p_b}{1-\phi_0(j-1)}\right) & j > 1 \end{array} \right\} \tag{24}$$

For $j > 1, \psi_2(j) = p_s \left[\sum_{n=0}^{\infty}(\phi_0(j-1))^n\right] (p_2 + \psi_2(j-1))$. Hence,

$$\psi_2(j) = \left\{ \begin{array}{ll} p_s p_2(0) & j = 1 \\ \dfrac{p_s}{1-\phi_0(j-1)}(p_2 + \psi_2(j-1)) & j > 1 \end{array} \right\} \tag{25}$$

For $j > 2, \psi_{1,2}(j) = p_s \left[\sum_{n=0}^{\infty}(\phi_0(j-1))^n\right] \cdot (\psi_{1,2}(j-1) + \phi_1(j-1)\left[\sum_{n=0}^{\infty}(\phi_0(j-1))^n\right] \psi_2(j-1))$. Hence,

$$\psi_{1,2}(j) = \left\{ \begin{array}{ll} \dfrac{p_s^2 p_1(0)}{1-p_b(0)p_s}(p_2 + \psi_2(1)) & j = 2 \\ \dfrac{p_s}{1-\phi_0(j-1)}\left(\psi_{1,2}(j-1) + \dfrac{\phi_1(j-1)\psi_2(j-1)}{1-\phi_0(j-1)}\right) & j > 2 \end{array} \right\} \tag{26}$$

Define $\Pr(\Lambda_1^{i-1}\Lambda_2^{j-1}\Gamma_1^i\Gamma_2^j)$ to be the probability that a sequence of events occurs that produces the above subsequence of events, and similarly define $\Pr(\Lambda_1^{i-1}\Gamma_1^i\Lambda_2^{j-1}\Gamma_2^j), \Pr(\Lambda_2^{j-1}\Lambda_1^{i-1}\Gamma_1^i\Gamma_2^j)$, and $\Pr(\Lambda_1^{i-1}\Gamma_1^i\Gamma_2^j)$. We can now compute these probabilities using our recursive function definitions, and our regular expressions that indicate the permitted sequences of foreground events that complete the above subsequences:

$$\Pr(\Lambda_1^{i-1}\Lambda_2^{j-1}\Gamma_1^i\Gamma_2^j) = \pi_k \frac{p_1}{1-p_b-p_2-\phi_{0,2}(k)} \frac{p_2}{1-p_b-\phi_0(k)} \left(\psi_{1,2}(k) + \frac{\phi_1(k)\psi_2(k)}{1-p_1-p_b-\phi_0(k)}\right) \tag{27}$$

$$\Pr(\Lambda_1^{i-1}\Gamma_1^i\Lambda_2^{j-1}\Gamma_2^j) = \pi_k \frac{p_1}{1-p_2-p_b-\phi_{0,2}(k)} \frac{\phi_1(k)}{1-p_1-p_2-p_b-\phi_0(k)} \frac{p_2}{1-p_1-p_b-\phi_0(k)}\psi_2(k) \tag{28}$$

$$\Pr(\Lambda_2^{j-1}\Lambda_1^{i-1}\Gamma_1^i\Gamma_2^j) = \pi_k \frac{p_2}{1-p_1-p_b-\phi_{0,1}(k)} \frac{p_1}{1-p_b-\phi_0(k)} \left(\psi_{1,2}(k) + \frac{\phi_1(k)\psi_2(k)}{1-p_1-p_b-\phi_0(k)}\right) \tag{29}$$

$$\Pr(\Lambda_1^{i-1}\Gamma_1^i\Gamma_2^j) = \pi_k \frac{p_1}{1-p_2-p_b-\phi_{0,2}(k)} \left(\psi_{1,2}(k) + \frac{\phi_1(k)\psi_2(k)}{1-p_1-p_2-p_b-\phi_0(k)}\right) \tag{30}$$

We can then compute M_x:

$$M_x = \frac{\Pr(\Lambda_2^{j-1}\Lambda_1^{i-1}\Gamma_1^i\Gamma_2^j) + \Pr(\Lambda_1^{i-1}\Gamma_1^i\Lambda_2^{j-1}\Gamma_2^j) + \Pr(\Lambda_2^{j-1}\Lambda_1^{i-1}\Gamma_1^i\Gamma_2^j)}{\Pr(\Lambda_1^{i-1}\Gamma_1^i\Gamma_2^j)} \tag{31}$$

B.2 Y Topology, Shared Bottleneck: M_a

M_a is simply the loss probability that, in the case of a shared bottleneck with the system in steady state, is

$$
\begin{aligned}
M_a = \pi_k &= \Pr(L_1^i = 0) = \Pr(L_2^j = 0) \\
&= \frac{1-\rho}{1-\rho^{k+1}}\rho^k
\end{aligned} \tag{32}
$$

where $\rho = \frac{\lambda_1+\lambda_2+\lambda_b}{\mu}$.

B.3 Y Topology, Separate Bottlenecks: M_x

We begin by constructing variables similar to those used in the previous subsections, except that they account for the fact that f_1 does not pass through the POC of interest. Let $\widehat{\pi}_n$ equal the steady state probability that the queue through which f_2 passes is filled to height n, and let k_2 be the capacity of the queue.

Here, the two foreground flows are bottlenecked by separate queues, so that the random variable $X_1 = ((L_2^{j-1} = 0), (L_2^j = 1))$ is independent of the random variable $Y = ((L_1^{i-1} = 0), (L_1^i = 1), (R_1^i < R_2^j < R_1^{N_1^i}))$. For similar reasons, the random variable $X_2 = (L_2^j = 1)$ is also independent of Y. Hence:

$$
\begin{aligned}
M_x &= \frac{\Pr(X_1, Y)}{\Pr(X_2, Y)} = \frac{\Pr(X_1)\Pr(Y)}{\Pr(X_2)\Pr(Y)} \\
&= \Pr(L_2^{j-1} = 0 \mid L_2^j = 1)
\end{aligned} \tag{33}
$$

We now proceed to solve for equation (33):

$$
\begin{aligned}
\Pr(L_2^{j-1} = 0 \mid L_2^j = 1) &= \Pr(L_2^j = 1 \mid L_2^{j-1} = 0)\frac{\Pr(L_2^{j-1} = 0)}{\Pr(L_2^j = 1)} \\
&= \left[1 - \Pr(L_2^j = 0 \mid L_2^{j-1} = 0)\right]\frac{\Pr(L_2^{j-1} = 0)}{\Pr(L_2^j = 1)} \\
&= \left[1 - \Pr(L_2^j = 0 \mid L_2^{j-1} = 0)\right]\frac{\widehat{\pi}_{k_2}}{1-\widehat{\pi}_{k_2}}
\end{aligned} \tag{34}
$$

Above, we have applied PASTA to get $\Pr(L_2^{j-1} = 0) = \Pr(L_2^j = 0) = \widehat{\pi}_{k_2}$. We have $\widehat{\pi}_n = \frac{(1-\rho_2)\rho_2{}^n}{1-\rho_2{}^{k_2+1}}$, where $\rho_2 = \frac{\lambda_{b_2}+\lambda_2}{\mu_2}$, (and note $\widehat{\pi}_n = (k_2+1)^{-1}$ when $\rho_2 = 1$). Recall that λ_{b_2} is the rate of the background flow into the queue, and μ_2 is the processing rate of the queue. Hence, $\frac{\widehat{\pi}_{k_2}}{1-\widehat{\pi}_{k_2}} = 1/k_2$ when $\rho_2 = 1$, and $\frac{(1-\rho_2)\rho_2{}^{k_2}}{1-\rho_2{}^{k_2}}$ otherwise.

Let $p_{s_2}(j)$ be the probability that the next event at the POC is an arrival from f_2, conditioned on the fact that the queue is filled to height j at the time of the arrival. This probability is different than $p_2(j)$ (in the case when the flows share a POC), since here, packets from f_1 no longer proceed through the POC. Similarly, define $p_{b_2}(j)$ to be the probability that the next event at the POC is an arrival from f_b, and $p_{s_2}(j)$ to be the probability that the next event at the POC is a service completion. Then $p_{2_2} = \frac{\lambda_1}{\lambda_{b_2}+\lambda_1+\mu_2}$, $p_{2_2}(0) = \frac{\lambda_1}{\lambda_{b_2}+\lambda_1}$, $p_{b_2} = \frac{\lambda_{b_2}}{\lambda_{b_2}+\lambda_1+\mu_2}$, $p_{b_2}(0) = \frac{\lambda_{b_2}}{\lambda_{b_2}+\lambda_1}$, and $p_{s_2} = \frac{\mu_2}{\lambda_{b_2}+\lambda_1+\mu_2}$ (as before we omit the parameter j when it is clear tha $j > 0$).

$\Pr(L_2^j = 0 \mid L_2^{j-1} = 0)$ is identical to $\Pr(L_2^{j+1} = 0 \mid L_2^j = 0)$, and its solution (where f_1 does not enter the POC) is

$$M_a = \Pr(L_2^{j+1} = 0 \mid L_2^j = 0) \quad = \quad \frac{p_{2_2}}{1 - p_{b_2} - \phi_3(k)} \tag{35}$$

$$\phi_3(n) \quad = \quad \left\{ \begin{array}{ll} p_{s_2} p_{b_2}(0) & n = 1 \\ \dfrac{p_{s_2} p_{b_2}}{1 - \phi_3(n-1)} & n > 1 \end{array} \right\} \tag{36}$$

B.4 Y Topology, Separate Bottleneck: M_a

The loss rate of probability of flow f_2 for the separate bottleneck case is:

$$M_a \quad = \quad \Pr(L_2^j = 0) = \pi_{k_2} = \frac{1 - \rho_2}{1 - \rho_2{}^{k_2+1}} \rho_2{}^{k_2}.$$

B.5 Experiments

We now discuss how we demonstrate that the values of M_x and M_a presented in equations (3) and (4) are such that if the POC(s) are M/M/1/K queues, then $M_x > M_a$ if and only if the POC(s) for f_1 and f_2 are shared. The recursive equations formulated in this section compute the values of M_x and M_a as functions of $\lambda_1, \lambda_2, \lambda_b, \mu$, and k in a system in which the POC(s) are M/M/1/K queues. Thus, for any set of values, $\{\lambda_1, \lambda_2, \lambda_b, \mu, k\}$, we compute the values for M_x and M_a for the case where the POC is shared, and verify that $M_x > M_a$, as well as the case where the POCs are separate, and verify that $M_x < M_a$. The recursive equations were coded in Mathematica v3.0, which allowed us to easily sample a large suite of test values.

Suite #	λ_1, λ_2	λ_b	μ	k
1	$\{0.1, 5.0\}$	10.0	$\{5, \lambda_1 + \lambda_2 + \lambda_b, \lambda_1 + \lambda_2 + \lambda_b \pm 0.1, 20\}$	$\{5, 10, 30\}$
2	10.0	1.0	$\{5, 20.9, 21, 21.1, 40\}$	$\{5, 10, 30\}$
3	$E_r(0.01, 10.0)$	$E_r(0.01, 25.0)$	$E_r(0.01, 30)$	10

Table 3: Suites of test sets used to verify the loss-corr technnique on the Y-topology where POCs are M/M/1/K queues.

Table 3 summarizes the sets chosen in the three suites of experiments. Each row gives the values used within that particular suite. Each column gives the values used for a particular parameter. The first suite consists of 30 experiments, consisting of all possible combinations where $\lambda_1 \in \{0.1, 5.0\}, \lambda_2 \in \{0.1, 5.0\}, \lambda_b = 10.0, k \in \{5, 10, 30\}$, and μ is chosen as 5, 20, $\lambda_1 + \lambda_2 + \lambda_b, \lambda_1 + \lambda_2 + \lambda_b + .1$, or $\lambda_1 + \lambda_2 + \lambda_b - .1$. The last three values cover the cases where the service rate of a shared POC is equal to, slightly more than, and slightly less than the aggregate rate of traffic into the POC. This first suite considers cases where the foreground flow rates are less than the background flow rate, where the aggregate rate into the queue is significantly less, slightly less, equal, slightly more, and significantly more than the aggregate service rate of the queue. The second suite consists of 15 additional sets of values, in which values are chosen with similar characteristics, except that the rates of the foreground flows are higher than the rate of the background traffic.

The final suite of traffic consists of 1,000 experiments with values for $\lambda_1, \lambda_2, \lambda_b$, and μ chosen randomly. $E_r(m, M)$ is a random variable whose values range between m and M, and is heavily weighted toward the minimum. $E_r(m, M)$ is computed by choosing a random value, x, uniformly distributed within the interval, $(0, 1)$, and returning $\exp(x(\ln(M) - \ln(m)) + \ln(m)) = m(M/m)^x$.

In all 1,045 experiments performed, the comparison test using the loss-corr technique for the Y-topology returned the correct hypothesis.

References

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

[2] V. Padmanabhan. Coordinated Congestion Management and Bandwidth Sharing for Heterogeneous Data Streams. In *Proceedings of NOSSDAV'99*, Basking Ridge, NJ, June 1999.

[3] L. Gautier, C. Diot, and J. Kurose. End-to-end Transmission Control Mechanisms for Multiparty Interactive Applications in the Internet. In *Proceedings of IEEE INFOCOM'99*, New York, NY, March 1999.

[4] J. Byers, M. Luby, and M. Mitzenmacher. Accessing Multiple Mirror Sites in Parallel: Using Tornado Codes to Speed Up Downloads. In *Proceedings of IEEE INFOCOM'99*, New York, NY, March 1999.

[5] S. Seshan, M. Stemm, and R. Katz. SPAND: Shared Passive Network Performance Discovery. In *Proceedings of the USITS'97*, Monterey, CA, December 1997.

[6] V. Padmanabhan. Optimizing Data Dissemination and Transport in the Internet, September 1999. slides presented at the BU/NSF Workshop on Internet Measurement, Instrumentation and Characterization.

[7] D. Katabi, I. Bazzi, and X. Yang. An Information Theoretic Approach for Shared Bottleneck Inference Based on End-to-end Measurements. Class project, MIT Laboratory for Computer Science, contact: dina@ai.mit.edu, 1999.

[8] S. Ratnasamy and S. McCanne. Inference of Multicast Routing Trees and Bottleneck Bandwidths using End-to-end Measurements. In *Proceedings of IEEE INFOCOM'99*, New York, NY, March 1999.

[9] R. Caceres, N. Duffield, J. Horowitz, and D. Towsley. Multicast-Based Inference of Network-Internal Characteristics: Accuracy of Packet Loss Estimation. *Transactions on Information Theory*, November 1999.

[10] S. Moon, P. Skelly, and D. Towsley. Estimation and Removal of Clock Skew from Network Delay Measurements. In *Proceedings of IEEE INFOCOM'99*, New York, NY, March 1999.

[11] V. Paxson. On Calibrating Measurements of Packet Transit Times. In *Proceedings of ACM SIGMETRICS'98*, Madison, WI, June 1998.

[12] G. Folland. *Real Analysis: Modern Techniques and Their Applications*. John Wiley and Sons, New York, NY, 1984.

[13] S. Ross. *Stochastic Processes*. John Wiley and Sons, New York, NY, 1983.

[14] S. McCanne and S. Floyd. ns-LBL Network Simulator, 1997. Obtain via http://www-nrg.ee.lbnl.gov/ns/.

[15] B.W. Lindgren. *Statistical Theory, 4th Ed.* Chapman and Hall, New York, NY, 1993.