Inferring Congestion Sharing and Path Characteristics from Packet Interarrival Times

Dina Katabi
LCS
MIT
dina@lcs.mit.edu

Charles Blake
LCS
MIT
cb@mit.edu

ABSTRACT

This paper presents new non-intrusive measurement techniques to detect sharing of upstream congestion and discover bottleneck router link speeds. Our techniques are completely passive and require only arrival times of packets and flow identifiers. Our technique for detecting shared congestion is based upon the observation that an aggregated arrival trace from flows that share a bottleneck has very different statistics from those that do not share a bottleneck. In particular the entropy of the inter-arrival times is much lower for aggregated traffic sharing a bottleneck. Additionally this paper identifies mode structure in the inter-arrival distribution that enables discovery of the link bandwidths of multiple upstream routers.

We validate these ideas with extensive experiments on a wide-scale Internet testbed and with multiple rate controlling routers. We find that the method can detect any bottleneck sharing among hundreds of flows. The classification errors decrease exponentially in the number of traced packets. Further, the method copes well with heavy cross-traffic and the errors decrease exponentially as the fraction of cross traffic at the bottleneck decreases. Unlike prior proposals, our technique does not inject any new probe traffic, does not require any sender cooperation, and works with any type of traffic (UDP, TCP, or multicast), and a wide variety of queuing disciplines. The method is simple and fast enough to be real-time for rates beyond 10,000 packets per second.

1. INTRODUCTION

In this report,[1] we show that the passive collection of packet inter-arrival times can reveal substantial information about the congestion state along upstream paths. We address two particular problems: single-flow bottleneck capacities and multi-flow bottleneck sharing. The necessary measurements can be collected completely at endpoints. The appeal of endpoint measurements is that they require no additional infrastructure and are accessible to a large population of users.

End-to-end measurements can be active or passive. Active methods inject new traffic (e.g., probes) into the network to induce a certain response, which is then used to infer a performance metric while passive methods observe traffic already present. Despite their usefulness, active methods have some drawbacks. Probes increase the load on the network by some additional traffic which could be on the order of hundreds of kilobytes per experiment [4, 10, 30, 23]. Moreover, the active traffic may perturb the network, bias the ensuing results, and complicate the analysis [26]. Our work focuses on deducing as much as possible from passive measurements alone.

First, we devise methods that enable an end receiver to discover the capacities of potentially *multiple* bottlenecks traversed by a flow and their traversal order from the arrival times of the packets in the flow. In particular, we show that the distribution of the packet inter-arrival times in a flow shows a few common patterns, which we analyze and relate to the bottlenecks along the path. Our results confirm that the common practices for estimating the bottleneck bandwidth using the minimum inter-arrivals of two consecutive packets in a flow [4, 10, 30] or the global mode in the distribution of its packet inter-arrivals [23] can make significant errors. Nonetheless, we show how to adjust the use of the inter-arrival PDF so that the minimum capacity along the path still can be extracted. Since this method relies solely on processing of network-level traces which are easily producible at any receiver, it provides a general, non-intrusive, and resource efficient approach to learning Internet path characteristics.

Second, we develop a novel passive technique that exploits the information embedded in packet inter-arrival distributions to detect flows that share the same bottleneck.

Detecting shared bottlenecks using end-to-end measurements is useful for sharing congestion information [12, 18], constructing the topology [28], and monitoring and debugging the network. Performing this detection using a passive approach is highly desirable because it is resource efficient (i.e., it does not generate probe traffic) and is extremely general (i.e., it makes no assumptions about the transport protocols or the queuing discipline).

Our approach relies on the observation that by clocking (i.e., pacing) the packets, a bottleneck imposes some structure on the probability distribution of the inter-arrival times of packets that traverse it. This structure is lost when packets that do not share a bottleneck get mixed together. The loss of structure shows up as more randomness in the inter-arrivals of the aggregate. Using entropy as our measure of randomness (the lack of structure), we develop a passive technique that enables an end receiver or a passive observer to detect flows that share bottlenecks by minimizing the Rènyi entropy of the packet inter-arrivals.[2]

The paper shows that the developed passive technique can detect any bottleneck sharing among hundreds of flows and is efficient and practical for use over the Internet. In particular, using the RON

[1]D. Katabi was supported by ARPA Agreement J958100, under contract F30602-00-20553. C. Blake was supported under DARPA contract N66001-00-1-8933. The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policy or endorsements, either express or implied of DARPA or the US government.

[2]Rènyi entropy is a generalized form of Shannon entropy. The exact definition is in Section 3.2

testbed [5], we show that our bottleneck detection method gives correct results in extensive Internet experiments run between 17 different Internet sites.

The method requires a relatively small numbers of packets per flow. In all cases, we find that errors decrease exponentially in the number of packets. The exact number of per-flow packets varies between 10 and 100 packets depending on the number of bottlenecks, classified flows, and the type of errors that matter. TCP connections in the Internet are often short-lived. However, depending on the application, the source for a "flow" may be defined as an aggregate. For example, if the focus is wide-area congestion analysis, it may be acceptable to define a source to be the entire LAN of the sender.

Further, the technique is robust in the presence of heavy cross-traffic, though more packets may be required. The method can be applied in real-time. On a commodity PC our implementation can classify samples with thousands of packets in less than a second.

The structure of this paper is as follows. In Section 2 we describe the properties of inter-arrival distributions for single flows and discuss the congestion and bandwidth implications. In Section 3 we exhibit the properties of multi-flow inter-arrival distributions and describe our bottleneck detection algorithm. In Section 4 we evaluate this algorithm in realistic experimental environments. Section 5 discusses possible future avenues and Section 6 concludes.

2. INTERARRIVAL TIME STATISTICS

In this section, we study the time between arrivals of consecutive packets in a TCP flow and plot its probability distribution function (PDF). Our objective is to relate the characteristics of the inter-arrival PDF to the congestion characteristics of the path traversed by the flow. In particular, we show how to interpret the PDF to discover the capacities of potentially two traversed bottlenecks, to discern their relative location, to assess their degree of congestion, and to probe the distribution of traffic burst sizes.

Before proceeding to analyze the PDF of the packet inter-arrivals, we clarify three terms. We use *"Minimum capacity link"* to refer to the link that has the minimum absolute capacity along a path. We use *"Bottleneck"* for a link/router where a flow experiences significant queuing. A bottleneck is a congested link; it is not necessarily the minimum capacity link along a path. Finally, the *"Nominal Transmission Time (NTT)"* of a link is the time it takes to transmit a 1500 byte packet over the link. For example, the nominal transmission time of a T1 is around 8 msec, while the nominal transmission time of a 10 Mbps Ethernet is 1.2 msec. (See Table 1 for a reference on the NTT of various link technologies.)

2.1 Measurement Methodology

We conducted our measurements over the RON testbed [5]. Table 1 provides a complete list of the RON nodes their locations and their access links. Note the heterogeneity in the measurement environment, which was chosen to reflect the heterogeneity of Internet paths. Five machines are located at US universities, three are at European or Asian Universities, three are broadband home Internet hosts connected by Cable or DSL, one is located at a US ISP and five are at various US corporations. The length of the measured paths is between 11 and 30 hops and the minimum capacity along a path varies between 0.384 Mbps and 100 Mbps.

Each experiment involved a 5 minute TCP download from one RON node to another.[3] The RON machines run FreeBSD 4.4 and the TCP stack uses an MTU of 1500 bytes. The receiver ran `tcpdump` [2] to log microsecond precision arrival times of the packets at the Ethernet card. We computed the time difference between successive arrivals and histogrammed them to plot the PDF of the packet inter-arrival in the flow. We repeated these experiments to cover periods of congestion (e.g., peak hours on weekdays) and periods of low traffic (e.g., weekends). In all, we conducted over a hundred experiments over several months.

Name	Description	Access Link	BW	NTT
MS	Residence, CA	DSL	0.384	31
Sightpath	.COM in MA	T1	1.544	8
Mazu	.COM in MA	T1	1.544	8
NC	Residence, NC	Cable Modem	2	31
M1MA	Residence, MA	Cable Modem	10	1.2
Aros	ISP in UT	Fractional T3	12	1.0
CCI	.COM in UT	Ethernet	100	.12
PDI	.COM in CA	Ethernet	3..100	N/A
CMU	Pittsburg, PA	Ethernet	10	1.2
Cornell	Ithaca, NY	Ethernet	100	.12
MIT	Cambridge, MA	Ethernet	100	.12
NYU	Manhattan, NY	Ethernet	100	.12
ACIRI	ACIRI, CA	Ethernet	10	1.2
Utah	U. of Utah,SLC	Ethernet	100	.12
NL	Vrije U,Holland	Ethernet	100	.12
Lulea	Sweden	Ethernet	100	.12
Korea	Korea	Ethernet	100	.12

Table 1: The RON testbed. Bandwidths are in Mbps. NTTs are in msec. The top block are ordinary Internet hosts. The bottom block have additional Internet2 connectivity.

Below we present a summary result of our findings. The appendix presents more graphs that show the persistence of our findings over various Internet paths that differ in their link technologies, path length, and whether the end nodes are at universities or corporations.

2.2 PDF of Packet Inter-Arrival in a TCP Flow

A few common patterns appear in the inter-arrival PDFs for TCP flows. These patterns are illustrated in Figure 1. In particular, note the multiple spikes of various heights, widths, locations, and spacings. The PDF might show a single *spike* such as Figure 1a, a *spike bump* such as Figure 1b, a *spike train* such as Figure 1c, or a *train of spike bumps* such as Figure 1d. The roughly equal spacing between the spikes in a spike train and or a spike bump is the *spike gap*. The roughly equal spacing between the bumps in a train of spike bumps and is the *bump gap*. In the following subsections, we show how to interpret these PDF patterns in terms of the congestion characteristics along the path the packets took.

Below, we interpret the patterns in Figure 1 and show how a proper understanding of the PDF allows one to discover bottleneck link bandwidths for multiple congested routers.

Single Spike: In this case, the flow traverses a bottleneck with no substantial cross traffic. As such, most of the packets arrive back-to-back at the receiver. The spike in the PDF corresponds to the

[3] While our experiments use TCP, these methods only rely upon large, relatively constant size packet transmissions.

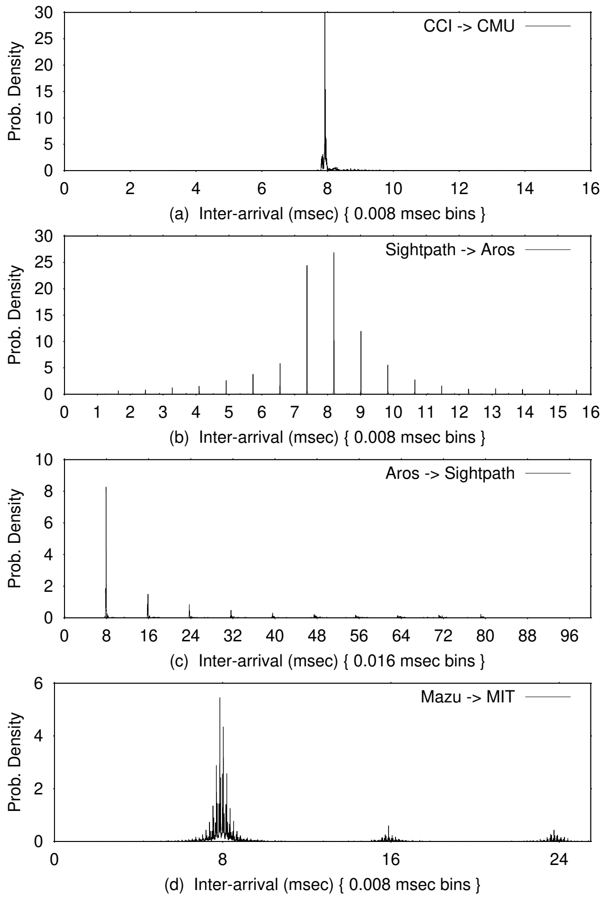

Figure 1: Common patterns in the PDF of inter-arrival times in a single TCP flow. (a): a single spike; (b): a spike bump; (c): a spike train; (d): a train of spike bumps.

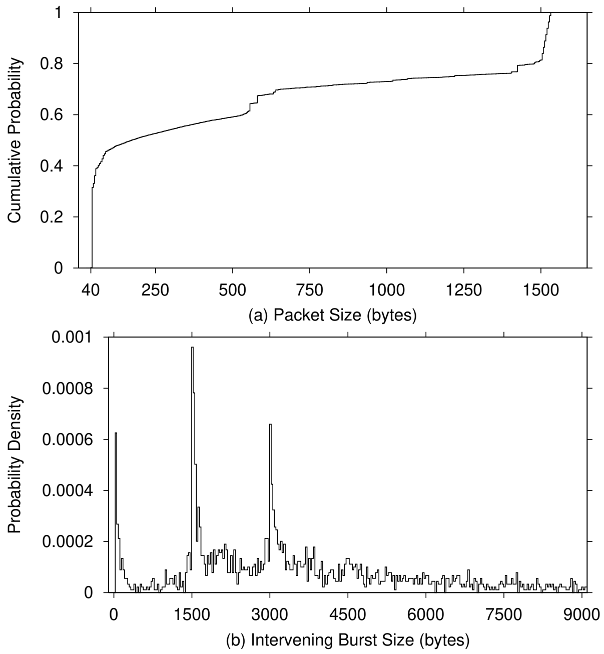

Figure 2: (a): The cumulative distribution of packet sizes for cross traffic at a congested OC3 link. (b): The probability distribution of cross traffic burst sizes for the same trace above. Notice the spikes at multiples of 1500 bytes.

NTT of the bottleneck. This situation is depicted in Figure 1a where the bottleneck is a T1.

Spike Bump: In this case, the flow traverses a low bandwidth bottleneck followed by a high bandwidth bottleneck. The two bottlenecks might be separated by a number of uncongested hops. We will show that the spike bump is centered at the NTT of the low bandwidth upstream bottleneck. Further, the gap between the spikes is the NTT of the high bandwidth downstream bottleneck. Thus, a spike bump carries information about two traversed bottlenecks.

We explain the spike bump using the example in Figure 1b. In this experiment the flow traverses a a T1 bottleneck (the access link at Sightpath), then a lightly congested 12 Mbps fractional T3 (the access link at Aros). The packets leave the upstream bottleneck spaced by its NTT (or some integer multiple of the NTT). In the experiment in Figure 1b, most of the packets left the upstream T1 with an inter-arrival of 8 msec. When any of these packets hits the congested downstream high bandwidth bottleneck, the packet is queued.

There are then 8 msec until our next packet arrives at the downstream bottleneck. During this interval a number of cross traffic packets arrives and is queued before our packet. After 8 msec, our second packet arrives at the higher bandwidth queue. Thus, the burst of cross traffic between any pair of packets in the traced flow depends on the number of cross traffic packets arriving in 8 msec at the downstream bottleneck. When all these bytes are transmitted on the output link, our packets have now been re-spaced. The time between the arrival of a pair of packets from the traced flow at the receiver is the time taken to transmit the first packet in the traced pair and the potential cross traffic burst at the downstream bottleneck. Depending on the size of the cross traffic burst, this time is sometimes larger than 8 msec and sometimes smaller. That is why the PDF shows a bump centered at 8 msec.

Next, we consider why the bump is composed of equally spaced spikes. Close inspection of many collected PDFs (see the appendix for more) reveals that the spikes are always separated by the NTT of the high bandwidth bottleneck. Thus, the most common case was always for a cross traffic burst at the downstream bottleneck to be a multiple of 1500 bytes. This is somewhat surprising. Though the traced TCP download used a 1500 byte MTU, the cross traffic packets have various sizes and reflect the variability of packet sizes in the Internet. (The appendix shows similar graphs in which the downstream bottlenecks are the access links at big universities where the cross traffic is fairly representative of cross traffic in the Internet.) It therefore seems possible that though cross traffic has various packet sizes, the most common cross traffic bursts are multiple of 1500 bytes.

To confirm that this is not a peculiarity of the RON sites, we studied the distribution of the cross traffic burst size from traces collected by NLANR [1] at various monitored links.[4] Since we are interested in bursts of cross traffic at a bottleneck, we chose traces in which the average traffic rate exceeds two thirds of the capacity of

[4]Trace file is from October 2001 and contains over 60,000 flows. It is at http://pma.nlanr.net/Traces/Traces/daily/20011005/COS-1002219707-1.tsh.gz

the monitored link. Figure 2a, shows the packet size accumulative distribution for a typical trace. The distribution looks similar to the one reported by CAIDA [11]. In particular, it shows that over 50% of the packets are around 40 bytes; 10% of the packets are about ≈ 560 bytes; and 20% of the packets are around 1500 bytes.

Figure 2b shows the cross traffic burst distribution for the same trace. To compute the burst size, we randomly picked a TCP flow and recorded the size of all traffic separating each pair of its packets. This is therefore precisely the traffic which, if subsequently sent through a bottleneck link, would be clocked and converted to inter-arrival times. We repeated the procedure over a large number of active TCP flows and plotted the PDF of the resulting cross traffic bursts. The PDF reveals the existence of a strong mode at 40 bytes and strong modes at integer multiples of 1500 bytes. The first mode at 40 bytes would make the traced packets look as if they arrived back-to-back. The other modes would create inter-arrivals spaced by one and 2 NTTs.[5]

Spike Train: This case is similar to the single spike case except that the traversed bottleneck is shared with a substantial amount of cross traffic. Consequently, it becomes more likely that a burst of cross traffic intervenes between any pair of the traced packets. Similarly to the spike bump case, the gap between the spikes is the NTT of the bottleneck, as illustrated in Figure 2c. Note though that a spike train need not always have a decreasing spike length. In a few of our experiments it was more common for the traced packets to be separated by a packet of cross traffic than to be back-to-back.

A Train of Bumps: In this case the flow first traverses a low bandwidth upstream bottleneck shared with a substantial amount of cross traffic. As a result the packet inter-arrival at the output of this bottleneck is a decreasing spike train as in Figure 1b. Later, the flow traverses a lightly congested high bandwidth bottleneck. The queuing at this latter bottleneck transforms every spike in the spike train into a spike bump creating a train of bumps. The gap between the spikes in a single bump is a the NTT of the high bandwidth downstream bottleneck, while the gap between the bumps is the NTT of the low bandwidth upstream bottleneck. For example, in Figure 3d, the upstream congested link is a T1 and the down stream link is a 12 Mbps fractional T3. Packets leave the congested T1 spaced by multiples of 8 msec, (i.e., the cross traffic burst size is either 0 bytes or 1500 bytes or 3000 bytes). However, when they reach the downstream link each spike is transformed into a spike bump with a gap of 1 msec (the NTT of a 12 Mbps link).

2.3 Capacity Inference

The Internet literature proposes a few approaches to discovering the minimum capacity along a path. The most common approach is to use the minimum inter-arrival of back-to-back packets [10, 21, 4, 27]. Other proposals suggest the most common inter-arrival (i.e., the global mode in the distribution of packet inter-arrivals). [23]. Below, we show that both approaches may give wrong results even in situations where the bottleneck bandwidth can be easily determined from a simple examination of the inter-arrival PDF.

Figure 3 (Mazu → Aros) shows the inter-arrival PDF of a flow where the sender is behind a T1 and the receiver is behind a 12 Mbps fractional T3. The minimum capacity along the path is the T1 link with an NTT of 8 msec. However the minimum inter-arrival is 1.7 msec. As such, a minimum capacity estimator based on using the minimum inter-arrival would mistakenly conclude that the bottleneck bandwidth in 7 Mbps, much more than a T1 bandwidth.

[5] A Spike bump need not be symmetric; Figure 11 in the appendix shows a non-symmetric spike bump. The non symmetry there is caused by severe congestion and high multiplexing at the downstream high bandwidth bottleneck. Hence, it was more likely to spread a pair of traced packets than to squeeze them.

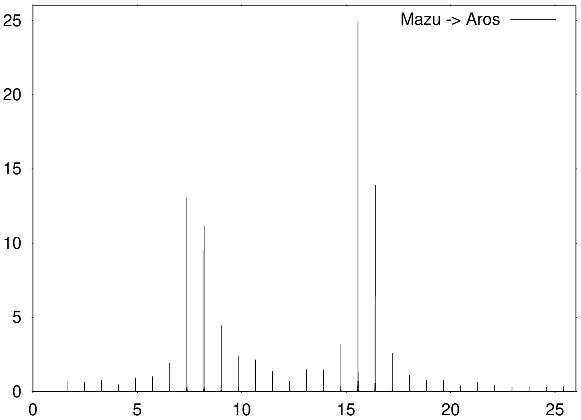

Figure 3: The NTT of the minimum capacity link is the gap between the bumps. It shows the link is a T1. Inferring the minimum capacity link from the minimum inter-arrival time or the global mode of the PDF would have yielded wrong results.

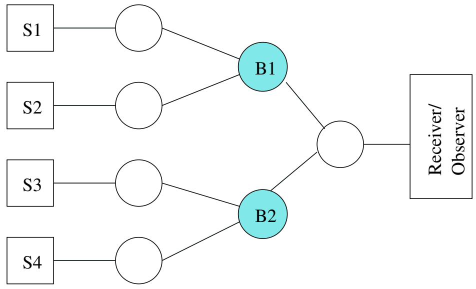

Figure 4: A simple clustering example; Sources $S1$ and $S2$ share the bottleneck $B1$, Sources $S3$ and $S4$ and share the bottleneck $B2$. The observer is co-located with the receiver. It receives all flows over the same link yet wants to cluster $S1$ and $S2$ together and $S3$ and $S4$ together.

The same figure shows the global mode of the inter-arrival PDF does not lead a good estimator of the minimum capacity along the path. In particular, the global mode in this trace happens at 16 msec, which would yield a 0.77 Mbps minimum capacity link. However, one can see from the PDF that the minimum capacity link is a T1 with an NTT of 8 msec. The 16 msec is the result of many of the traced packets being separated by exactly one 1500 byte cross traffic packet.

Thus, our analysis of the past few sections shows how to strengthen previously proposed techniques by computing the bump and spike gaps and relating them to the traversed bottlenecks.

3. DETECTING SHARED BOTTLENECKS

In the previous section, we have developed an understanding of the statistics of packet inter-arrivals in the Internet. In this section, we look at applying this understanding to multiple flows with the goal

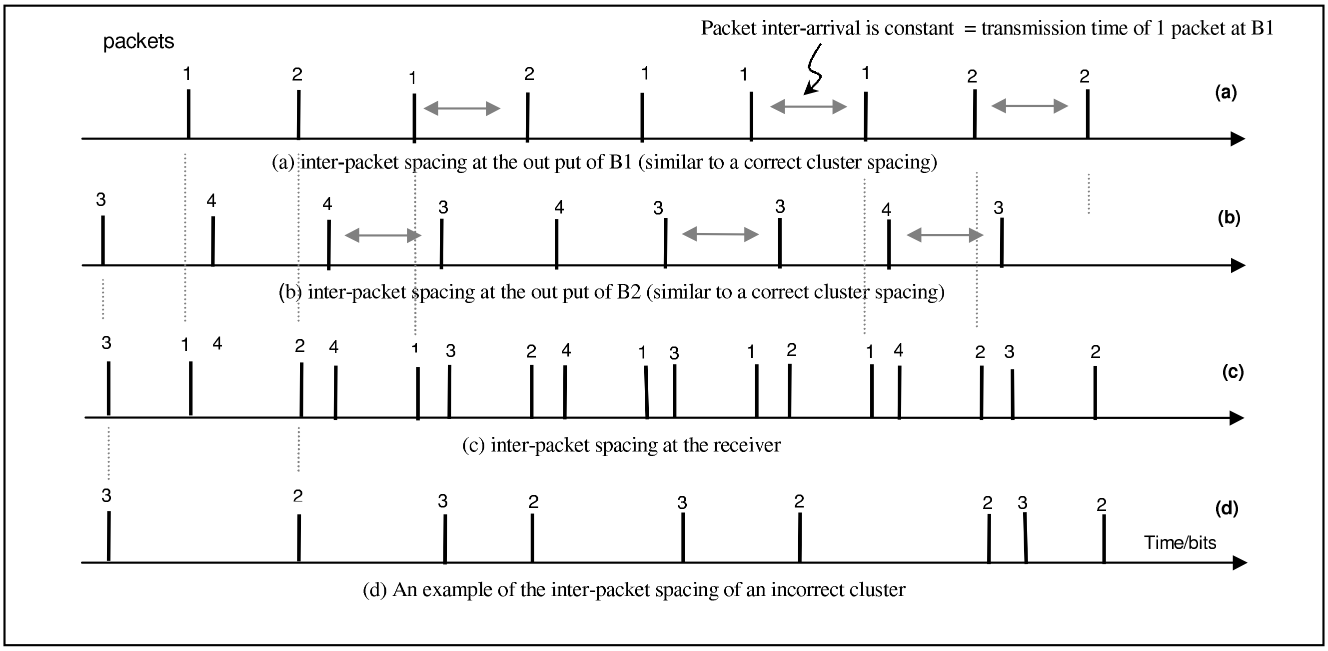

Figure 5: Packets inter-arrivals in various clusters of flows in Figure 4. The thick lines represent packets. They are numbered according to the sender. The dotted lines emphasize the alignment in time. The x-axis is time. (a) and (b) are the outputs of $B1$ and $B2$ respectively, and the correct clusters. (c) is the packet inter-arrivals as seen by the observer, which corresponds to putting all sources in the same cluster. (d) is an example of an incorrect cluster, namely $\{S2,S3\}$.

of detecting bottleneck sharing. Particularly, we demonstrate that a passive observer watching the arrivals of packets at some link can use the information embedded in the packet inter-arrivals to cluster the flows into groups such that all flows in one group share a common bottleneck.

Before describing our approach to passive bottleneck detection, we note that detecting shared bottlenecks is a clustering problem, where the clustered objects are flows. A correct clustering groups flows that share a bottleneck into the same cluster and produces one cluster per bottleneck. An incorrect clustering fails to group flows that share the bottleneck or groups flows that do not share a bottleneck into the same cluster. We also note that for the purpose of detecting bottleneck sharing, a *"flow"* is a stream of traced IP packets with the same source identifier. The source identifier is defined by the user to fit the application of interest. It is usually defined as the source IP-address in the packets, because traced packets with the same sender share the upstream part of their path. However, when NAT boxes [16] are suspected, the user may define the source identifier to be the source IP-address and port pair.

Finally, we note that when a flow traverses more than one bottleneck, bottleneck sharing is resolved based on the most dominant bottleneck along the path. For example, consider two flows that have the same receiver. Each of these flows experiences severe queuing at its sender access link. However, occasionally, both flows share a transient queue at the receiver access link. In this case, the flows do not share the same point of congestion and the clustering technique should not group them together.

3.1 Basic Idea

We use the simple topology in Figure 4 to describe the intuition underlying our approach to discriminating the sharing of a bottleneck. In this scenario, four sources send to the same receiver. $S1$ and $S2$ are behind the same bottleneck $B1$, and their total sending rate is larger than the capacity of $B1$. $S3$ and $S4$ share the bottleneck $B2$ and their total rate exceeds its capacity. The passive observer is colocated with the receiver. It receives packets from all four sources on the same link yet wants to group together the sources that share the same bottleneck.

Figure 5 shows the packets' inter-arrivals at different points in our simple topology. Figures 5a and 5b show the inter-arrival of packets at the output of $B1$ and $B2$ respectively. Furthermore, they represent the inter-arrivals of packets in the correct clusters ($\{S1,S2\}$ and $\{S3,S4\}$). Figure 5c shows the packet inter-arrivals at the receiver. It is the overlay of the output of $B1$ and $B2$. Note that 5c does not show the constant inter-arrival observed in 5a and 5b. If the receiver succeeds in clustering the flows that share the bottleneck, it ends up with two clusters in which the packet inter-arrival is constant. If the receiver mistakenly groups the flows $S2$ and $S3$ together, the resulting incorrect cluster $\{S2,S3\}$ exhibits more random packet inter-arrivals as illustrated in 5d.

Thus, the inter-arrival of interleaved packets from flows that do not share a bottleneck is more random than the inter-arrival of interleaved packets from flows that do share a bottleneck. We can further confirm this intuition via the following experiment. We use an MIT machine to download simultaneously a file from both MS and Sightpath. The resulting two TCP flows experience bottlenecks at the source access links, namely a T1 and a 0.38 Mbps DSL (very little bandwidth compared to the 100 Mbps Ethernet to which the

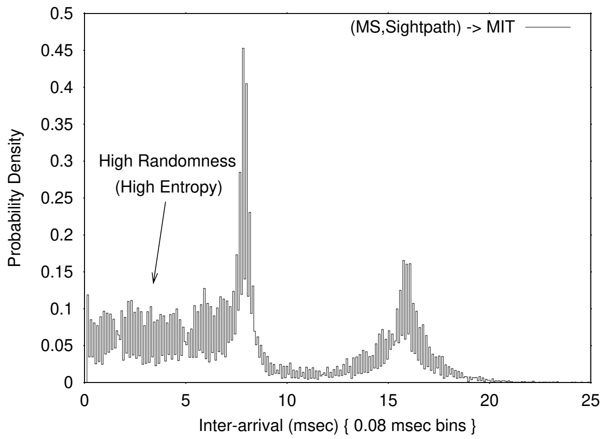

Figure 6: The PDF of the inter-arrivals over the aggregated trace of unassociated flows. The heavy near-uniform distribution before the first peak and between the peaks is exactly the sort of smooth value-diversity measured by entropy.

MIT machine is connected). Thus, they do not share a common bottleneck. We log the arrival of the packets at MIT and plot the inter-arrival PDF of the aggregate trace. Figure 6 shows that the PDF of this incorrect cluster exhibits an area of almost uniform distribution before the first mode. Consider that since all of the inter-arrival PDF's of Section 2 were single flows, they *de facto* shared whatever bottlenecks they passed through. Comparing this new aggregate trace PDF against the PDFs in Figure 1 we see that the inter-arrival PDF's for incorrectly clustered flows has substantially more randomness. A quantitative measure of this randomness should therefore discriminate between combinations of flows sharing bottlenecks and combinations not sharing bottlenecks.

3.2 Generalized Entropy

We start with the definition of Shannon entropy, a traditional measure of the uncertainty (i.e., randomness) in a random variable. The Shannon entropy $H(x)$ of a discrete random variable x that takes on the value v_i with probability p_i is defined as:

$$H(x) = \sum_i p_i \, \log_2 \, p_i \qquad (1)$$

In [20], the authors propose minimizing the Shannon entropy as a means for discriminating between bottleneck sharing and non-sharing flows. They provide simulation results that show the validity of the approach in environments with low to moderate cross-traffic. We found this measure to do a reasonably good job of discriminating shared from non-shared flow aggregations. However, the spiky nature of the inter-arrival distributions causes problems. Even for correct flow combinations, many new small probability spikes can arise in the PDF as it simply fills out with more data points from the larger, combined trace. The Shannon entropy can increase in this circumstance, even though the small spikes are at a place that makes them a continuation of existing PDF structure.

To overcome this difficulty we propose the use of Rènyi entropy [29], a generalization of the Shannon entropy, defined as:

$$K_q(x) = \frac{1}{1-q} \log_2 \sum_i p_i^q \qquad (2)$$

The parameter q specifies the order of the Rènyi entropy. In the limit as $q \to 1$ the Rènyi entropy converges to the Shannon entropy (i.e., $\lim_{q\to 1} K_q(x) = H(x)$). Rènyi entropy shares many properties with Shannon entropy. Both entropies achieve their maximum for uniform distributions. Neither depends upon the value where the probability occurs. Also, for both entropies, the entropy of two independent subsets of a data set is the sum of the individual entropies.

The effect of the Rènyi entropy is to weight high probability values more than the problematic low probability incidental noise caused by small sample effects. This is because raising probabilities on $(0, 1)$ to high powers (i.e., large q) spreads them out, lifting peaks and depressing tiny values. On the other hand, one should not choose very large q since then only the peaks would matter. We chose q by assessing the end-to-end classification performance for a few experiments. We found of $q = 4$ and $q = 5$ to yield good results.

3.3 Practical Issues

The simple scenarios in Figure 4 and Figure 5 are useful for explaining the intuition underlying passive detection of shared bottlenecks using entropy minimization, but they do not reveal the full complexity of the problem. In this section, we discuss the various complications that arise in practice. Nonetheless we show that the main idea still holds; namely, that a bottleneck imposes *detectable* structure on the inter-arrivals of packets that traverse it. This structure is lost when the packets get mixed with other packets that have not crossed the same bottleneck.

A number of issues could potentially confound the passive detection of shared bottlenecks with entropy metrics. First, many effects add randomness to the PDF of the inter-arrivals in a correct cluster, e.g. the dynamics of TCP congestion control. For example, when a relatively small number of TCPs share a Drop-Tail bottleneck, the bottleneck link might cycle between periods of severe congestion with large number of drops followed by periods of underutilization. During the periods of underutilization, packets do not leave the bottleneck equally spaced. However, these periods of underutilization are short or absent when the number of competing flows is large. More importantly, the duration of such periods at a bottleneck is relatively short compared to the duration of the periods during which the bottleneck clocks the packets. Consequently, the structure imposed on the packet arrival times by the bottleneck clocking should dominate any randomness introduced by TCP dynamics. This is supported by our empirical findings.

A second reason for randomness in the inter-arrival of packets in a correct cluster is the fact that routers downstream from a bottleneck might build transient queues without being congested. For lower capacity routers with very occasional queues the number of packets and inter-arrivals affected is small (since these routers are by definition not the bottleneck). For higher capacity routers, single spike structure may be transformed into a spike bump (or a spike train may be transformed into a train of bumps), but the overall entropy remains quite low compared to aggregations of unclocked flows (see Figure 6).

Another issue that complicates passive detection of shared bottlenecks is that most of the traffic at the output of a bottleneck may end up being unobserved by the receiver. For example, in Figure 4, if the packets sent by $S1$ do not cross the link monitored by the observer then the correct clustering is $\{\{S2\}, \{S3\ S4\}\}$. In this case, though the cluster $\{S2\}$ does not exhibit a constant inter-arrival, the observer is likely to discover the correct clustering. In particular, although the cluster $\{S2\}$ has high entropy, any attempt to put $S2$ in the same cluster with $S3$ or $S4$ (or to put $S3$ and $S4$ in different clusters) is likely to further increase the entropy of the clustering. In general, cross-traffic plays the role of noise on the signal of interest. As more of the output traffic at bottlenecks becomes cross-traffic, the information embedded in the inter-arrival PDF becomes more immersed in noise. In Section 4.2, we investigate the robustness of the algorithm against heavy cross-traffic.

Another potential obstacle comes from the fact that packets do not have the same length; consequently, the time to transmit one packet over the bottleneck is not constant. In practice, this is not an issue. To see why, recall that the distribution of packets inter-arrivals in the single TCP flows of Section 2 showed a considerable amount of structure despite the fact that cross-traffic packets have various sizes.

3.4 Iterative Passive Technique for Detecting Shared Bottlenecks

To develop a clustering technique based on entropy-minimization, two design issues must be resolved.

The first issue is choosing the function that should be minimized. Equation 2 shows how to compute the Rènyi entropy of the inter-arrivals of packets in a cluster. However, it does not indicate how to combine the entropies of the various clusters into a quantity that we can minimize. We call the quantity we want to minimize the 'cost function', which we define as follows:

$$Cost = \sum_{c=1}^{N} n_c \, K_q(p_c) \tag{3}$$

where n_c is the number of packets in cluster c, K_q is the Rènyi entropy of p_c of the inter-arrivals of the aggregate flows in c, and N is the number of clusters.

Weighting the entropy by the number of packets in the cluster is important because it prevents the clustering technique from reducing the cost by collapsing all of the flows into the same cluster. For example, there might be two correct clusters each having an entropy of 2 bits. The entropy resulting from combing all flows together could be 3 bits. Although, this latter entropy is larger than the entropy of any of the correct clusters, without the weighting factor the algorithm can reduce the entropy by putting all the flows in the same cluster, which would produce an incorrect outcome. In general, a statistical understanding of the packet-weighting of entropy in a global cost derives from the subsample additivity of both Shannon and Rènyi entropy. That is, the entropy of two independent subsets of a data set is the sum of the individual entropies. Thus the entropy of a whole aggregated sample of packets is simply the entropy of the parent distribution multiplied by the number of packets. This notion also makes it meaningful to sum the entropies of each cluster to define the total entropy of the entire arrangement.

The second issue is the computational complexity of the optimization problem. The search space is exponential in the number of flows. In particular, there are $C^F/C!$ ways to group F flows into C clusters [15]. When the number of bottlenecks is unknown the search space is even larger. A brute force search is infeasible for all but a small number of flows and simple candidate topologies. The optimization surface is also quite rough. E.g, changing the cluster of a flow of n packets can change up to $2n$ inter-arrival times in both its old and new clusters. Simpler distance-based clustering problems are already NP-complete complexity [17, 7].

To reduce the computational complexity, we use an iterative procedure which starts with an initial random clustering and iterates by moving a source from one cluster to another to obtain an incremental reduction in the Rènyi entropy. Despite that this technique is not guaranteed to find the global minimum, our empirical results show that it almost always yields the correct clustering, which is after all the end goal.

The optimization strategy is as follows:

1. Start with each flow in a cluster by itself.
2. Pick a source S_i in round-robin fashion.
3. Try moving S_i from its cluster to every other cluster.
4. Accept the move that most reduces the total cost.
5. Repeat from step 2 as long as progress can be made.

Finally, a few important points are worth noting. First, our clustering technique is designed so that the errors decrease as the number of flows increases. In particular, it is conceptually possible to cluster the flows that share the same bottleneck based on some similarity metric defined over a pair of sources' inter-arrival PDFs. However, clustering based on similarity would cause the errors to accumulate as the number of flows increases. In contrast, since our algorithm computes the entropy of entire clusters (rather than flows), the more sources there are the more packets we get and the easier it is to identify the structure resulting from bottleneck clocking. Having the error decreases with the number of flows is an important feature given that the complexity of the problem increases with the number of flows. Furthermore, clustering based on similarity may not distinguish between two different bottlenecks that have the same bandwidth. For example, It may not differentiate between two flows that share the same T1 link and two flows that cross different T1 links.

A second advantage of the entropy-based technique is its generality. In particular, the approach does not make any assumptions about the bandwidth of the bottlenecks nor about their queuing disciplines. It works when the different bottlenecks have exactly the same capacities. It also works with Drop-Tail, RED, and other work-conserving queue disciplines.

4. CLUSTERING EVALUATION

We used extensive Internet measurements to evaluate the effectiveness of the passive techniques in detecting flows that crossed the same bottleneck. Although simulation-based evaluation is an option it does not reflect the variability encountered in the Internet. By evaluating the technique in the environment it is meant to work in, we ensure that it works with the different link technologies, real cross traffic patterns, existing router policies, and various TCP implementations.

4.1 Measurement Methodology

The basic problem in evaluating any bottleneck sharing detection technique on real Internet traces is to verify that the output of the algorithm matches bottleneck sharing in the network. In particular, we must design experiments in which we are confident about which flows share bottlenecks. We address this problem with two different approaches that create three classes of sharing topologies.

In the first approach, we exploit our knowledge of the topology of the RON network to ensure that the flows share congestion at specific bottlenecks. In particular, we know the capacities of access links connecting certain RON nodes to the Internet. Thus, we can create experiments in which the senders are connected to 100 Mbps Ethernets and the receiver is behind a T1 link. By inspecting aggregate throughput achieved by senders we can verify that the flows all faced congestion at the T1 link connecting the receiver site to the broader Internet.

Similarly, we can create experiments in which each of the senders is behind a low bandwidth link such as a T1, a DSL, or a cable modem, while the receiver is connected to a 100 Mbps Ethernet and located at a big university with good connectivity. By checking the throughput of each sender against the capacity of its access link, we can ensure that each sender has faced congestion locally. We can further confirm local outbound congestion by checking that the aggregate throughput of the senders is significantly less than the typical bandwidth share available on the receiver access links. Thus, our knowledge of the topology and connectivity of the RON testbed provide us with a non-intrusive way to construct experiments that have reasonably unambiguous outcomes.

Our second approach for creating experiments with controlled outcomes relies on the use of the Click router [22]. Click was designed to allow flexible reconfiguration, packet re-writing, and traffic shaping. In particular, we use IP masquerading and bandwidth throttling to very closely emulate the behavior of a pair of real routers with diminished capacity. The IP masquerading re-writes packets so that TCP connections can be transparently established between arbitrary RON hosts even though the routes of packets are pinned to go through the Click routers under our control. This arrangement ensures unambiguous bottleneck sharing.

Using the methodology described above, we conducted hundreds different Internet experiments. Each one involves a number of TCP senders streaming data to the same receiver. Using tcpdump, we record arrival times at the receiver and feed the log files to our clustering program.

4.2 Clustering Accuracy

There is no standard method for evaluating the accuracy of clustering algorithms[15]. To evaluate our technique, we use three error metrics that we judge useful to the specific application of the bottleneck detection technique.

The first metric is *the probability of any error*, which provides the most conservative view of the accuracy. For any particular bottleneck sharing scenario, the probability of any error is computed by clustering multiple different data sets and taking the percentage of outcomes that do not completely match the correct answer. The difficulty with using this metric alone arises from the fact that not all clustering errors are equivalent. For example, assume that we have 50 flows that share the same bottleneck. A clustering technique that puts 49 flows in the same cluster and one flow in a different cluster is definitely better than a technique that puts each of the fifty flows in its own cluster. Yet, both outputs would be treated the same if we use the probability of any error as our metric of accuracy.

The second metric is the probability of creating incorrect clusters where some of the flows do not share the same bottleneck. We call this metric the probability of *false grouping*. This metric measures the *correctness* of the algorithm. For example, if the user wants to identify the flows that traverse the same bottleneck to share their congestion information, then the probability of false grouping would tell the user how likely the technique is to produce incorrect results that would lead to the wrong sharing of congestion information. For any particular bottleneck sharing scenario, the probability of false grouping is computed by clustering multiple different data sets and taking the fraction of clusters that contain flows that do not share a bottleneck.

The third metric is the probability that the algorithm might fail in grouping some flows that share the bottleneck, which we call the probability of *false separation*. This metric measures the *efficiency* of the algorithm. For example, consider a user who is interested in sharing congestion information between flows that cross the same bottleneck. Then the better the technique is in collapsing the flows that share the bottleneck into the same cluster, the more the user can share their congestion state and the less the total number of states maintained by the system. To find the probability of false separation for a particular bottleneck sharing scenario, we run the clustering technique over multiple different data sets. The probability of false separation is the difference between the number of generated clusters and the correct number of clusters divided by the number of generated clusters.

Senders	Bottlenecks	Configuration	P[Any Error]	Pkts/ flow
7	1	Shared cong. at M1MA	2%	90
10	1	Shared cong. at MS	0%	90
10	1	Shared cong. at Sightpath	1%	65
10	1	Shared cong. at Mazu	1%	60
11	1	Shared cong. at Aros	0%	50
11	1	Shared cong. at CMU	5%	90
6	6	Separate cong.; Recv at MIT	7%	25
6	6	Separate cong.; Recv at CCI	1%	30
6	6	Separate cong.; Recv at Cornell	2%	35
6	6	Separate cong.; Recv at NYU	0%	10
12	2	Click Bottlenecks	0%	50
24	2	Click Bottlenecks	0%	60
48	2	Click Bottlenecks	1%	≈100
88	2	Click Bottlenecks	0%	≈100
102	2	Click Bottlenecks	2%	≈100
170	2	Click Bottlenecks	2%	≈100
88	2	Click; 50% cross traffic	3%	≈200
40	2	Click; 75% cross traffic	1%	≈800
25	2	Click; 85% cross traffic	8%	≈2000

Table 2: Efficiency of the iterative technique. Summary results showing that the technique eventually converges to almost perfect accuracy even for scenarios with large number of senders and fairly complex bottleneck sharing.

Table 2 shows the efficiency of the iterative clustering technique in dealing with large numbers of sources and fairly complex bottleneck sharing. The table has three blocks. Experiments reported in the first and second blocks do not use the Click router. The cross traffic in these experiments is uncontrolled. Experiments in the third and fourth blocks are controlled using two Click routers. The probability of any error is computed over 100 different samples. The table shows that although the iterative technique does not try all possible combinations of sources and bottlenecks, it always converges to almost perfect result. This convergence happens even when the number of sources is 170 and the search space is on the

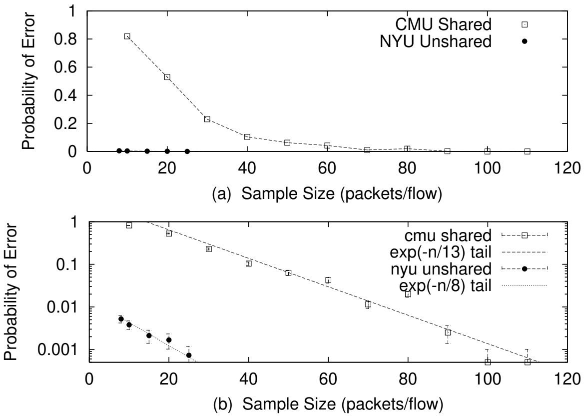

Figure 7: Probability of any clustering error at all vs. sample size for two simple topologies: every flow sharing and no flows sharing. Note that preventing false grouping errors requires very little data. Preventing false separation is harder, but not onerous. Trend lines in the bottom graph show the exponential convergence of error probabilities.

order of $2^{170}/2!$. A key observation in Table 2 is that as cross traffic increases or the clustering experiment becomes more complex (i.e., more flows or more cross traffic) more packets per flow are needed for correct clustering. Below, we examine these aspects in more detail.

First, we address the number of packets per flow necessary for correct clustering. Figure 7a illustrates the probability of any error as a function of the average number of packets from each flow. The figure shows two representative graphs: The first graph, labeled "CMU Shared", is for the case where all senders share the same bottleneck; the second graph, "labeled NYU Unshared", is for the case where each sender has a separate bottleneck. The probability is computed over 500 different samples. The figure shows that a few dozen packets are enough for correct clustering. Figure 7b shows trend line on the log scale. It indicates that although the absolute number of packets required for correct clustering differs from one type of experiment to the next, the error probability dies off exponentially.

Note that though the data plotted in Figure 7 is the probability of any error, the nature of the two types of "natural" experiments makes the them representative of our two other types of error metric. The upper curve is the probability of any error for the case where all flows share a bottleneck. In that case the only type of error is false separation. The lower curve, barely visible on the same scale, is the probability of any error for the case where no flows share bottlenecks. In that case the only type of possible error is false grouping. The extremely fast convergence of false grouping errors is a highly desirable property of our technique. This is because grouping senders that do not share the bottleneck together is a more severe error than failing to recognize senders that share a bottleneck.

Next, we consider the robustness of the technique against heavy cross-traffic. The experiments in Table 2 were run during midday. As such, they experienced natural cross traffic along their path. Given that many of the sites involved in these experiments are large universities with continuous Internet activity, we argue that the results in Table 2 are representative of the technique's behavior under common cross traffic situations.

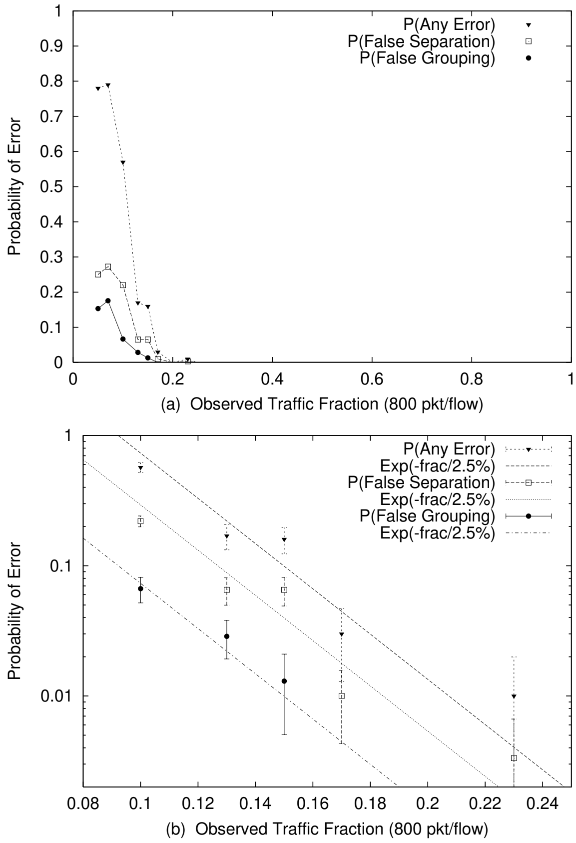

Figure 8: Error probabilities vs. observed traffic fraction. The first graph shows the error rate rapidly vanishes when more than 15% of the bottleneck traffic is observed. The second is a log-scale graph which shows the trend is consistent with exponential improvement in traffic fraction. (The scale on the x-axis is reduced since the error reaches zero for larger fractions of observed traffic)

To discover the behavior of both false grouping and false separation under heavy cross traffic, we funnel a large number of TCP flows from many senders through a pair of Click routers and back out to a receiver across the Internet. We considered various cross-traffic fractions by censoring various subsets of flows from our data set. This effectively gives the algorithm exactly the data it would have had if the censored flows had been diverted before reaching the receiver. We ran the algorithm on many random censorings to get reasonable failure rate estimates.

Figure 8a shows the clustering error as a function of the fraction of the bottleneck link traffic seen at the observer. The probabilities are computed by taking the average of 1000 different measurements for sample sizes of on average 800 packets/flow. The graph shows that the clustering technique provides perfect clustering as long as at least 20% of traffic crossing the bottleneck can be observed. As observed traffic drops below 20% of the total bottleneck traffic, the technique begins to make minor false separation errors (i.e., occasionally separating flows that share the same bottleneck but never grouping flows that do not share the bottleneck). False grouping errors do not become an issue until over 95% of the traffic goes unobserved. The straight trend line on the semi-log plot in Figure 8b

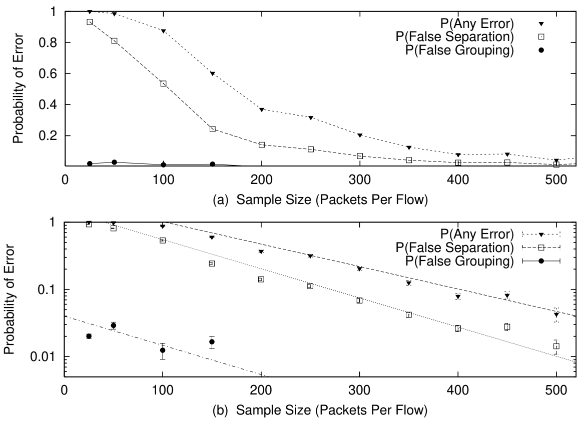

Figure 9: Sample size convergence for 25% cross-traffic. The first graph shows how false grouping rates are only marginally worse than when 100% of traffic can be observed. The second graph shows that substantial amounts of hidden traffic does not destroy the exponential convergence.

shows that the probability of error decreases exponentially as the fraction of observed traffic increases.

Figure 9 shows that the number of packets per flow required for correct clustering when 75% of the bottlenecked traffic is cross traffic. Note that this number decreases as the fraction of observed bottleneck traffic increases. Only a few hundred packets per flow are required for correct classifications even when 75% of the bottleneck traffic is unobserved cross traffic.

A final note is that the simplicity of the algorithm lends itself to efficient software or hardware implementations. All the program must do is iterate over the aggregated arrival time trace of a potential flow combinations and bin successive differences. While one might imagine an $O(N_{packet} \ \log \ N_{packet})$ algorithm based on sorting the arrival times of potential combinations, it is actually possible to merge the arrivals in $O(N_{packet} \ \log \ N_{flow})$ time since the individual arrival lists can be pre-sorted just once. The histograms can be kept compact and in fast memory and the entropies can be computed almost entirely with lookup tables for logarithms since the range of bin counts is relatively small for reasonable sample sizes. Our implementation can cluster samples with 1,000 packets in under 10 msec on commodity PC hardware. This translates to over 10,000 packets/sec.

The principal scaling issue for large numbers of flows is the larger number of total packets involved and the much larger number of combinations that must be tried. Even so, our algorithm successfully classifies traces with tens of thousands of packets and 170 flows in under a second of CPU time.

5. FUTURE WORK

This work lends itself to extension in several directions. One open issue is determining congestion sharing in a multiple bottleneck scenario. Namely, sharing or not sharing is more than simply a binary variable. Consider two flows that share congestion at the access link of their common receiver; yet, one of them crosses a separate upstream bottleneck. In such a scenario, some kind of hierarchical congestion classification is desirable.

Another direction for future work is a more detailed investigation of the shape of the inter-arrival distributions. In particular, the envelopes formed by the tips of the spikes in Figure 1 trace out very regular curves. It would be informative to fit the spike train and the spike bump to well-known distributions and analyse the shape of their tails. This may lead to a better understanding of the distribution of the cross traffic burst. Furthermore, finding good models for the inter-arrival distribution in a flow would improve the ability to cluster flows that share the bottlenecks. Particularly, if a catalog of common shapes is developed then it might be possible to embed this in a clustering algorithm to improve recognition of correct clusterings. In principle it should also be possible to improve recognition of incorrect clusterings. As Figure 6 shows, the noise in the inter-arrival PDF due to unsynchronized packets does not occur just anywhere.

6. RELATED WORK

Much prior work has studied learning Internet path characteristics from endpoint measurements[10, 20, 4, 27, 28, 30, 12, 18, 13, 25, 8, 19, 24]. The objective of these measurements could be bottleneck bandwidth detection[4, 28, 14, 13, 23], topology discovery[28, 12, 18], detecting the state of congestion and the available bandwidth[9, 12, 18, 25, 8, 19, 24], or simply understanding the network and the traffic patterns[6].

For example, pathchar and cprobe are useful tools for discovering the bandwidth available along a path. However, they consume a large amount of network resources. In particular, pathchar generates at least 10 Kbytes of probe traffic per hop and cprobe generates 5 Kbytes of probe traffic per hop [30]. The accuracy of these tools is acceptable for low bandwidth links (less than 10 Mb/s), yet they become significantly inaccurate for high bandwidth links [14].

The Packet Bunch Mode (PBM) estimates the raw bottleneck bandwidth of a connection by looking for modalities in the timing structures of groups of back-to-back packets. Although more robust than pathchar, it requires information from both the sender and receiver sides [27].

Traceroute[3] is a widely used tool for learning the intermediate routers and the latency along a path. It requires that intermediate routers reply to ICMP echo messages, a feature that might be disabled due to security concerns.

The authors in [9] propose the use of multicast loss-correlation to infer the loss rates over individual links along a path. Their simulation shows that the estimator tracks the changes in the loss rate. However, the proposed approach sends probe packets into the network and requires the existence of a multicast service.

The authors of [28] use loss correlation among the receivers in a multicast group to infer the logical shape of a multicast tree. Their approach does not inject probe traffic in the network; however, its reliance on loss information limits its use to significantly long multicast sessions. The authors in [25] use loss pairs to infer some characteristics of input buffering behavior such as RED parameters. While this work used active probes they note that their approach might be used in a passive context.

Packet pair dispersion and bandwidth histograms have been examined in [13] toward the end of bandwidth estimation. The focus of the analysis there was fixed bin-width bandwidth histograms. We found however that there is also much significant information to be gleaned from the equal spacings in inter-arrival time distributions.

Recently, there were two proposals for detecting whether *pairs* of flows share the same bottleneck [12, 18]. Despite the usefulness of these proposals in simple circumstances, they have a number of practical disadvantages that limit applicability. Since they generate probe traffic, both proposals are non-passive and require sender cooperation. Additionally they make stronger queuing discipline assumptions. Also, these proposals do not generalize their techniques to more than two flows while ours handles many. Thus, the clustering problem that we address in this paper is intrinsically harder than the problem addressed by these proposals.

7. CONCLUSION

This paper demonstrates effective, efficient, and robust techniques for inferring interesting properties of networks seen by packet flows. The only input data required is a completely passive collection of time stamps of packet arrivals at end nodes or at intermediate monitors.

We demonstrated that correct interpretation of inter-arrival PDFs allows inference about the bandwidth and degree of multiplexing at potentially multiple bottleneck links. In the spike bump and spike train cases we relate inter-arrival distributions to the distributions of cross traffic burst sizes. Finally we show how to correctly infer bottleneck capacity from the locations and gaps of spikes and bumps in the inter-arrival PDF.

Higher order statistics defined on the arrival times of combinations of flows allow sensitive detection of bottleneck sharing. We demonstrate that this detection can be both fast and reliable given small to moderate amounts of data even in the face of substantial fractions of unobserved cross traffic at the bottleneck routers in question.

We validated these techniques with extensive experiments on the RON testbed and with controlled experiments using a pair of Click routers. We found that the method can detect any bottleneck sharing among hundreds of flows. Near perfect sharing detection efficiency required on the order of 100 packets per flow. The classification errors decrease exponentially in the number of traced packets. Further, the method copes well with heavy cross-traffic and the errors decrease exponentially as the fraction of cross traffic at the bottleneck decreases.

8. REFERENCES

[1] National laboratory for applied network research (nlanr). http://www.nlanr.net/.

[2] tcpdump - the protocol packet capture and dumper program. http://ee.lbl.gov/tcpdump.tar.Z.

[3] traceroute - a tool for printing the route packets take to a network host. http://ee.lbl.gov/traceroute.tar.Z.

[4] pathchar - a tool to infer characteristics of internet paths, 1997. ftp://ee.lbl.gov/pathchar.tar.Z.

[5] D. Andersen, H. Balakrishnan, M. F. Kaashoek, and R. Morris. Resilient overlay networks. In *Proceedings of the 18th ACM Symposium on Operating Systems Principles (SOSP '01)*, Chateau Lake Louise, Banff, Canada, Oct. 2001.

[6] J. C. Bolot. Characterizing end-to-end packet delay and loss in the internet. *Journal of High Speed Networks*, 2(3):289–297, 1993.

[7] P. Brucker. On the complexity of clustering problems. In R. Henn, B. Korte, and W. Oletti, editors, *Optimizing and Operations Research*. Springer-Verlag, Berlin, West Germany, 1977.

[8] R. Caceres, N. Duffield, A. Feldmann, J. Friedmann, A. Greenberg, R. Greer, T. Johnson, C. Kalmanek, B. Krishnamurthy, D. Lavelle, P. Mishra, K. Ramakrishnan, J. Rexford, F. True, and J. van der Merwe. Measurement and analysis of ip network usage and behaviour, May 2000.

[9] R. Caceres, N. Duffield, J. Horowitz, D. Towsley, and T. Bu. Multicast-based inference of network-internal characteristics: Accuracy of packet loss estimation. In *IEEE Infocom '99, New York*, Mar. 1999.

[10] R. Cater and M. Crovella. Measuring bottleneck link speed in packet-switched network. Technical Report TR-96-006, Boston University, Mar. 1996.

[11] K. Claffy, G. Miller, and K. Thompson. The nature of the beast: Recent traffic measurements from an internet backbone, Apr. 1998. http://www.caida.org/outreach/resources/learn/packetsizes/.

[12] J. K. D. Rubenstein and D. Towsley. Detecting shared congestion of flows via end-to-end measurements. In *The Proc. of ACM SIGMETRICS '00*, June 2000.

[13] C. Dovrolis, P. Ramanathan, and D. Moore. What do packet dispersion techniques measure? In *In Proceedings of IEEE INFOCOM '01*, Apr. 2001.

[14] A. B. Downey. Using pathchar to estimate internet link characteristics. In *The Proc. of ACM SIGCOMM '99*, Aug. 1999.

[15] R. O. Duda, P. E. Hart, and D. G. Stork. *Pattern Classification and Scene Analysis Part 1: Pattern Classification*. John Wiley, 2 edition, 2001.

[16] K. Egevang and P. Francis. The ip network address translator (nat), May 1994.

[17] M. Gary and D. Johnson. *Computers and Intractability – a Guide to the Theory of NP-Completeness*. W.H. Freeman, 1979.

[18] K. Harfoush, A. Bestavros, and J. Byers. Robust identification of shared losses using end-to-end unicast probe. Technical Report BUCS-2000-013, Boston University, March 2000.

[19] P. Huang, A. Feldmann, and W. Willinger. A non-intrusive, wavelet-based approach to detecting network performance problems. In *Proc. of ACM SIGCOMM Internet Measurement Workshop 2001, San Francisco Bay Area*, Nov. 2001.

[20] D. Katabi, I. Bazzi, and X. Yang. A passive approach for detecting shared bottlenecks. In *The 11th IEEE International Conference on Computer Communications and Networks (ICCCN '01)*, Scottsdale, Arizona, Oct. 2001.

[21] S. Keshav. *Congestion Control in Computer Networks*. PhD thesis, University of California, Berkeley, Aug. 1991.

[22] E. Kohler, R. Morris, B. Chen, J. Jannotti, and M. F. Kaashoek. The click modular router. *ACM Transactions on Computer Systems*, 18(3):263–297, Aug. 2000.

[23] K. Lai and M. Baker. Nettimer: A tool for measuring bottleneck link bandwidth. In *Proc. of the USENIX Symposium on Internet Technologies and Systems*, Mar. 2001.

[24] K. Li and S. Jamin. A measurement-based admission-controlled web server. In *INFOCOM '00, Tel Aviv, Israel*, pages 651–659, Mar. 2000.

[25] J. Liu and M. Crovella. Using loss pairs to discover network properties. In *Proc. ACM SIGCOMM Internet Measurement Workshop 2001*, Nov. 2001.

[26] V. Paxson. Toward a framework for defining internet performance metrics. In *INET'96*, June 1996.

[27] V. E. Paxson. *Measurements and Analysis of End-to-End Internet Dynamics*. PhD thesis, University of California, Berkeley, Apr. 1997.

[28] S. Ratnasamy and S. McCanne. Inference of multicast routing trees and bottleneck bandwidths using end-to-end measurements. In *IEEE INFOCOM 99 (New York, NY, March 1999)*, number CSD-98-1019, Mar. 1999.

[29] A. Renyi and L. Vekerdi. *Probability Theory*. North-Holland, Amsterdam, 1970.

[30] S. Seshan, M. Stemm, and R. H. Katz. SPAND: Shared passive network performance discovery. In *USENIX Symposium on Internet Technologies and Systems*, 1997.

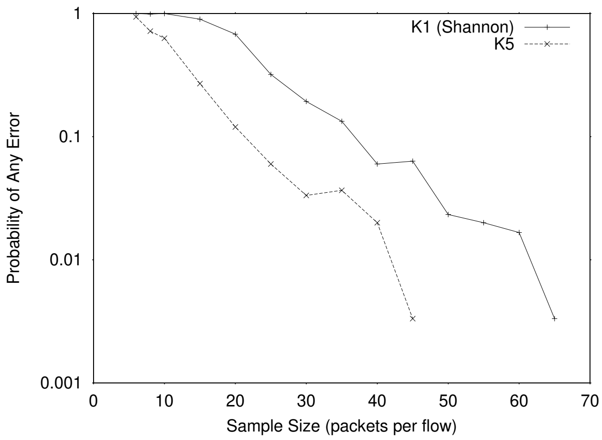

Figure 12: Here we compare our classification performance using Shannon and fifth order Rènyi entropy for 25% cross traffic with a pair of Click routers. The y-axis is a log scale. Note the improved statistical efficiency at small sample sizes.

APPENDIX

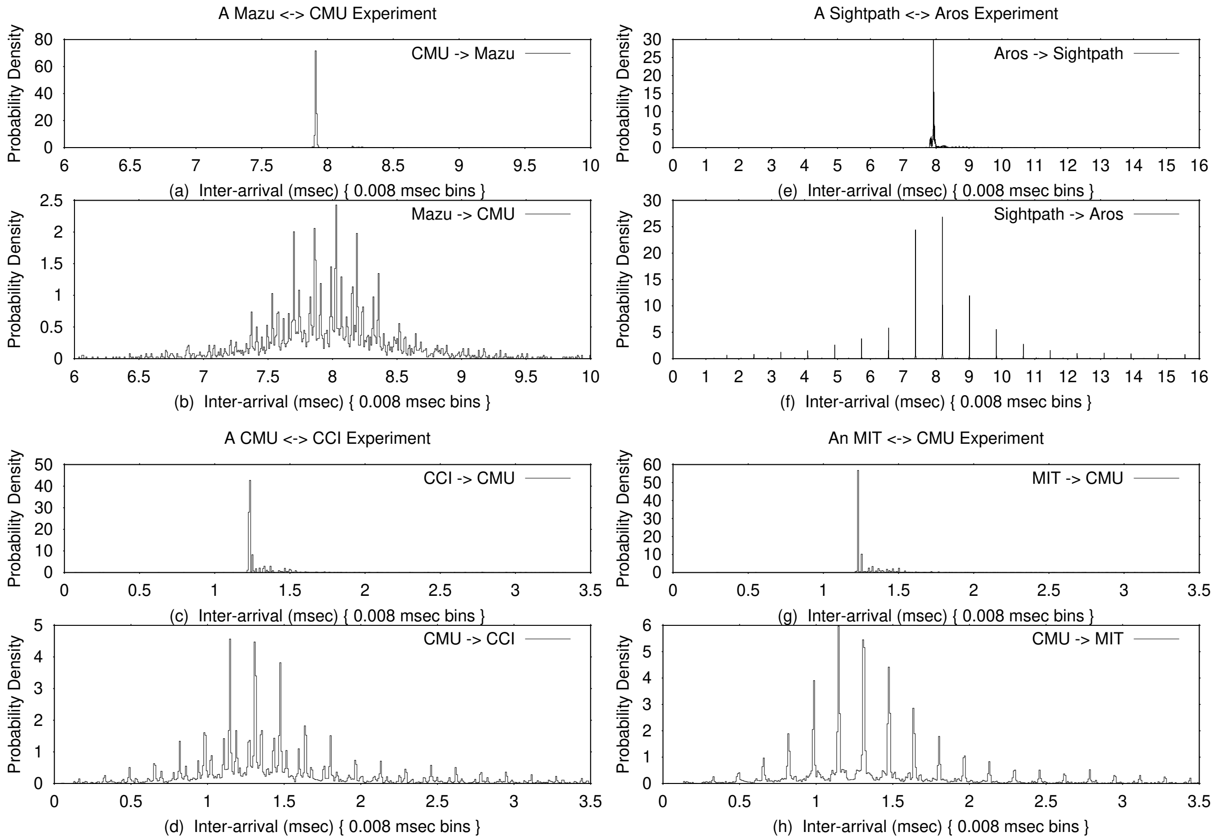

Figure 10: Several additional experiments. In each experiment, we choose a pair of RON nodes and send a TCP flow from the first node to the second, record the arrival times and construct the inter-arrival PDF of the forward path. Then, we start a second TCP flow from the second node to the first one, log the arrival times and construct the inter-arrival PDF of the reverse path. Using the reverse path is a device to construct a comparison case where it is likely that a bottleneck whose bandwidth is the same as the access link of the forward path. In all experiments the inter-arrival PDF of flows traversing a high bandwidth access link then a low bandwidth access link shows a single spike at the NTT of the low bandwidth link. On the other hand, the inter-arrival PDF of flows that first traverse a low bandwidth access link then a high bandwidth access link shows a bump of spikes whose local mode (tallest spike in the bump) coincides with the NTT of the low bandwidth link and with gap that coincide with the NTT of the high bandwidth link.

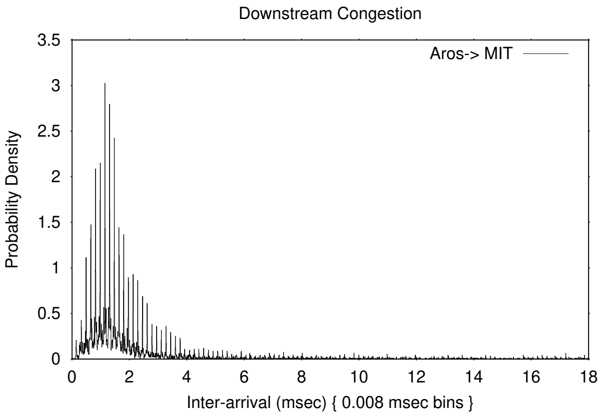

Figure 11: Here we exhibit the effect of congestion at the downstream high bandwidth bottleneck.