Using Loss Pairs to Discover Network Properties

Jun Liu and Mark Crovella
Department of Computer Science, Boston University.
Email:{junliu,crovella}@cs.bu.edu

Abstract— **Accurate estimation of network characteristics based on endpoint measurements is a challenging and important problem. In this paper we propose and evaluate a new tool for the discovery of network characteristics called *loss pairs.* A loss pair is a pair of packets travelling the same path and close together in time, such that exactly one of the two packets is dropped in the network. Loss pairs provide insight into network conditions close to the time when loss is occurring. Using loss pairs, we show that it is possible to characterize the packet dropping behavior of drop-tail and AQM routers internal to the network, from measurements made at end systems. Furthermore, we show that loss pairs can be used to estimate bottleneck buffer sizes of droptail routers. An attractive feature of loss pairs is that they lend themselves naturally to use in a passive-measurement setting. Using such an approach, we show that loss pairs provide estimates that are generally accurate and robust over a wide range of network conditions.**

I. Introduction

Methods for discovering network-internal characteristics using measurements taken at endpoints are increasingly valuable as applications and services seek to adapt to network properties. Previous work on estimation of network properties has focused on estimating bottleneck bandwidth [3], [5], [22], [17] and estimating per-link bandwidth, latency, and loss [15], [19], [7], [23]. While each of these techniques is useful, an important network resource that has *not* previously been addressed is the configuration of routers in the network — for example, the amount of buffering present and the packet dropping scheme in use.

This work was partially supported by NSF research grant ANI-9986397.

Knowing such properties of a router can be useful in a number of settings. First, from the simple standpoint of network element discovery, knowing how much storage a router uses to buffer a link allows one to configure simulations intended to mimic the routing element. In a more sophisticated setting, knowing the buffering and dropping scheme in use at the bottleneck link of an Internet path may be useful for adapting congestion control behavior of the sources using the path; for aggregate flow control of multiple concurrent streams using the path [2]; and for predicting delay and jitter for multimedia applications using the path.

In some settings it may be relatively easy to estimate the amount of buffering on a link. When packets flow over only a single hop (or in the absence of any cross traffic on a multi-hop path) then one can estimate buffering based on minimum and maximum observed delays along the path. However methods based on min- and max-filtering can be error prone in the presence of cross traffic since the final estimate is determined by measurements of (typically) only the two packets that experienced the extreme cases (similar problems are noted in [7]). To obtain a robust estimate, it is necessary to develop an estimator that allows the identification and elimination of noise due to cross traffic along the network path.

To this end we have developed a method called *loss pairs.* A loss pair is a pair of packets traveling the same network path and close together in time, such that exactly one of the two packets is dropped in the network. The utility of the loss pair derives from the fact that the packet that is not lost can be used to estimate network conditions. If the two packets traverse the network sufficiently close to each other in time, the conditions seen by each packet will be roughly similar. Thus in the case where one packet is lost, the other packet provides information about network conditions close to periods of loss.

Clearly, for arbitrary pairs of closely-spaced packets, either loss of both packets, or successful transmission of both packets, is more likely than the drop of exactly one packet. However, it is not necessary for loss pairs to occur frequently for them to be useful. Furthermore, we find that in the course of observing data flowing over a heavily used link, loss pairs occur sufficiently often in practice to yield good estimates.

The basic idea of loss pairs can be cast into either an active or a passive measurement style. Active measurement (injection of measurement or "probe" packets into the network) has been the more common approach historically. More recently, the drawbacks of active measurement (the additional network load due to injected traffic, and the potential for probes to perturb network conditions) have led researchers to look for ways to discover network conditions passively, *i. e.,* though observation of the traffic that is already flowing through the network [1], [24].

While it is clear that loss pairs could form the basis for an active measurement tool, in this paper we focus in the use of loss pairs in a passive mode. Thus we are interested in discovery of characteristics of the bottleneck router in a busy network path, simply by observing traffic flowing over that path. In this case we assume that most (but not all) losses occur in the bottleneck router. Furthermore, in all our experiments we make the assumption that traffic is primarily using TCP as its transport; this allows us to gather necessary loss and delay information from snooping on TCP's acknowledgment process. As a result we are able to derive all of our estimates from observations made at only a single point in the network.

In this setting we show that the loss pairs approach is quite versatile. When the bottleneck router employs a drop-tail queueing policy, the measured round trip time obtained from a loss pair includes the time spent to drain the (approximately) full bottleneck queue. Comparing round-trip measurements taken from loss pairs with the round trip times of arbitrary (non-loss-pair) packets allows us to distinguish the portion of delay that is due to bottleneck buffer draining. Combining the knowledge of this delay with the estimated bottleneck bandwidth (*e.g.,* as obtained from one of the bottleneck-bandwidth tools mentioned above) yields the buffer size in bytes.

We also demonstrate the use of the loss pairs approach for characterizing AQM routers. For example, when the bottleneck router employs the RED active queue management scheme [10, see also [6]] there are three parameters that determine the router's behavior. Since loss pairs provide information about network conditions during loss, they can inform us about the state of a RED queue when it chooses to drop packets. Using this insight, we show how to estimate the three parameters of a bottleneck RED router using the loss pairs approach — again, from measurements made at just one end system along the network path. We also show that the loss-pairs approach can distinguish RED dropping patterns from those used by BLUE [9], [8].

The remainder of the paper covers related work in Section II; detailed specification of the loss pairs method in Section III; application of loss pairs to buffer size estimation in Section IV; application of loss pairs to RED router characterization in Section V; and our conclusions in Section VI.

II. Related Work

Network measurement tools have been in existence since the earliest deployment of TCP/IP; `ping` (for discovering reachability) and `traceroute` (for discovering IP routing) are standard tools on most Internet hosts. As the Internet has grown, more sophisticated tools have become worth developing.

Link bandwidth was first estimated using *packet pairs* by Keshav in [16]. A packet pair consists of two packets that queue back-to-back in front of the link being measured. This results in an effect (originally illustrated in [14]) in which the inter-packet gap of the two packets leaving the bottleneck is inversely proportional to the link bandwidth. This phenomenon was explored further in [3] and developed into a tool in [5]. A number of further fruitful directions have followed, including the use of groups of packets and single packets [22], [17], [18], [12]. While some of these methods use pairs of packets, just as the loss pair method does, the loss pair method focuses on relating drop behavior to network state, rather than on bandwidth estimation.

A large effort has gone into the development of tools to measure dynamic network properties: utilization, latency, and loss rate. The `TReno` tool [20] measures the throughput achievable by a TCP connection over a given path, and `Sting` [23] uses TCP as a probe to distinguish between loss rates in each direction along the network path. `Pathchar` [15] and its improvements `pchar` [19] and `clink` [7] focus on link-by-link analysis of network conditions; unfortunately these tools rely on injecting a large amount of probe traffic into the network.

Recent work has used correlations among measurements as a basis for inferring network conditions. In [4] the authors rely on the correlation in measurements induced by multicast splitting of packets to develop an efficient estimator for link-by-link loss rates. In [11] the authors use the correlation in measurements experienced by pairs of packets that travel through the network close together in time to measure link-by-link loss rates without the need for multicast probes. The loss pair method bears some similarity to that method in the sense that it relies on temporally adjacent packets experiencing similar, and thus correlated, conditions.

Finally, the loss pair method can be applied passively, a direction also taken in some other recent measurement efforts. In [1], the authors show how to estimate bottleneck link bandwidth from interpacket spacing in a TCP connection; and in [24] the authors use passive measurements to estimate network properties to a large set of hosts.

III. Using Loss Pairs

We define a loss pair as two packets p_1 and p_2 injected into the network such that:

1. p_2 initially follows p_1 with time Δ between the trailing edge of p_1 and the leading edge of p_2;
2. exactly one of p_1 and p_2 happens to be dropped in the network; and
3. p_1 and p_2 traverse the same sequence of links and routers up to the point where one of them is dropped.

In all of the estimations used in this paper we take $\Delta = 0$. In an active measurement setting, this is the natural choice. In a passive measurement setting, such as we consider here, increasing Δ would serve to increase the number of loss pairs that can be extracted from any fixed traffic trace. However increasing Δ would lessen the likelihood that the two packets experience similar network conditions. We have not explored this tradeoff since we find that even in short traces, a set of loss pairs sufficient for accurate estimation can generally be extracted even when $\Delta = 0$.

A loss pair is formed when either packet p_1 or p_2 is dropped. In this paper we chose to consider those pairs in which p_1 is dropped, but we tested other cases (p_2 is dropped, either is dropped) and found no difference in results based on this selection.

The use of loss pairs in this paper relies on the following assumptions, some of which could be relaxed under more sophisticated approaches. The most important assumption is is that most packet losses and delays happen at the bottleneck (*i.e.,* the highest-loss link in the path, not necessarily the lowest-bandwidth link). This makes it possible to ascribe losses and delays seen at the endpoint to the internal bottleneck. A second assumption is that the round-trip path and the location of the bottleneck do not change during measurement. This ensures that the non-dropped packet in a loss pair is likely to see similar queue state as the dropped one. In order to relate delay in the router to queue occupancy, we assume that the packet scheduling at routers along the path is FCFS. Finally, in order to convert delay at the bottleneck into queue occupancy in bytes, we assume that the bandwidth of the bottleneck is known to us.

In this paper we focus on the round-trip time measurement obtained from packet p_2. In an active measurement setting, this round-trip time could be obtained if p_2 were an ICMP ECHO packet, or some other packet designed to generate an immediate response from the destination end system. In the passive measurement setting that we study here, round-trip time is obtained from those TCP data packets that are immediately acknowledged by the destination. Furthermore, we identify the fact that p_1 was lost by observing TCP's eventual retransmission of p_1 at a later time (our simulation uses TCP Reno).

Figure 1 shows a set of graphs that illustrate how the loss pairs approach works in practice. This figure is based on a simulation (described in detail in the next section) in which TCP traffic flows through a multi-hop network with considerable cross traffic on the bottleneck link, as well as on links upstream and downstream of the bottleneck. The minimum round trip time (due only to propagation delay) in this network is 32 ms, which is seen when a packet encounters no queueing at any router. The drain time of a completely full bottleneck queue is 67 ms.

In this simulation there were 883,912 packets that traversed the bottleneck link; of these, there were 246 packets that yielded estimates as part of a loss pair. Figure 1(a) shows a histogram of the set of all 883,912 packets traversing the bottleneck link. From this histogram it's clear that the bottleneck queue is almost always nearly empty. Almost all round trip times are close to the empty-queue value of 32 ms, and the tail of the histogram (corresponding to the queue's occupancy distribution) drops off roughly exponentially.

Figure 1(b) shows a histogram of the subset of the round trip times corresponding to loss pairs. This figure shows how dramatic the effect is of filtering using the loss pair method. The principal peak in this histogram is around 99ms, which is the network propagation delay (32ms) plus the full queue drain time (67ms).

Figure 1(c) provides more insight by showing both histograms in the region around 99ms. This figure shows the remarkable selection ability of the loss pairs method, since clearly the overall round trip time distribution is quite noisy in this range. In fact, we can see a sharp decline in the overall distribution around 99ms, since round trip time values greater than 99ms are only possible due to queueing at non-bottlenecks, and noise introduced upstream or downstream of the bottleneck. Despite this sharp decline in the overall distribution, the loss pairs method is able to extract a sharply peaked histogram tightly centered on the critical value of 99ms.

Figure 1 illustrates the basic idea of loss pairs and shows that they can be remarkably precise at identifying network conditions at the time when packet loss is occurring. We can make use of this basic idea in solving a number of problems related to discovery of network properties. In the next section we discuss the use of loss pairs for drop-tail buffer size estimation, while in the following section we apply this idea to characterizing AQM routers.

IV. Estimating Buffer Size

In this section we explore the utility of the loss pairs method for bottleneck buffer size estimation. To do so we employ a simulation using the `ns-2` simulator [13] that is intended to capture the essential characteristics of a typical network measurement setting.

Our principal simulation topology is shown in Figure 2. We simulate a generalized network environment in which a set of TCP sources are connected to a corresponding set of TCP sinks. The path used by the TCP flows traverses a single bottleneck link. Three sets of cross-traffic sources and sinks are present: one each upstream, concurrent, and downstream of the bottleneck link.

In this network, the link between router B and C is configured to be the bottleneck. There are four groups of TCP sources/sinks of which groups 1, 2, and 3 generate cross traffic. We set the observation point at the leftmost group of TCP sources and assume that all the traffic generated by those TCP sources, as well as the ACKs from TCP sinks, are visible to the estimation agent at the observation point, while all other (cross) traffic is invisible to the agent.

We generate network traffic intended to incorpo-

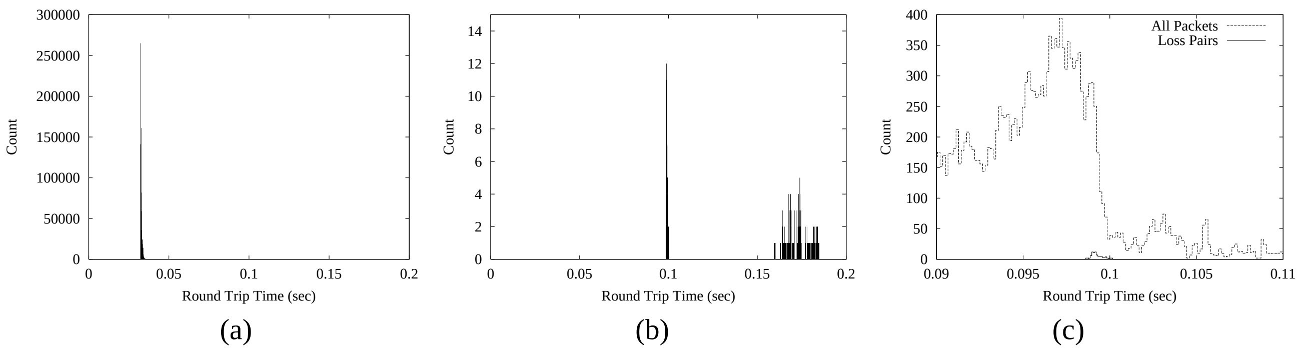
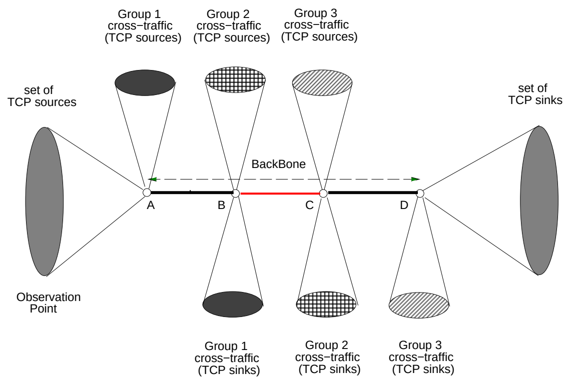


Fig. 1. Histograms of round trip times of (a) all packets (b) only loss pair packets (c) both sets, zooming in on time near 99ms.

rate realistic burstiness and self-similar properties by using heavy-tailed ON-OFF TCP sources. Sources used TCP Reno without delayed ACKs. All sources alternate between transmitting using TCP, and lying idle. Transmission sizes are drawn from a Pareto distribution with shape parameter $\alpha = 1.1$. Idle times are drawn from a Pareto distribution with shape parameter $\alpha = 1.5$. This method has been shown to be effective at generating self-similar traffic when the sources use the TCP transport protocol [25], [21]. In the baseline case there are 200 TCP sources and sinks generating observable traffic, and 90 sources and sinks generating cross traffic. The 90 cross traffic sources are divided evenly among groups 1, 2, and 3.


Group 1 cross−traffic (TCP sources)
Group 2 cross−traffic (TCP sources)
Group 3 cross−traffic (TCP sources)
set of TCP sources
set of TCP sinks
BackBone
A
B
C
D
Observation Point
Group 1 cross−traffic (TCP sinks)
Group 2 cross−traffic (TCP sinks)
Group 3 cross−traffic (TCP sinks)
Fig. 2. Principal network simulation topology.

This network setting can be considered to be an abstraction of the typical end-to-end environment in which the set of TCP sources can be considered as the subset of traffic that is observable from a given vantage point. The observable traffic does not constitute all of the traffic flowing through the buffer of interest, nor is the traffic observed at a point that is necessarily "close" to the buffer of interest.

The observer has access to a single clock and is able to observe traffic traces, matching acknowledgments with the data packets that trigger them. This assumes that significant delays are not introduced between data packet arrival and acknowledgment generation on the TCP sink, which is not necessarily always the case in practice. However such delays would generally take the form of noise that would be filtered out in the estimation process.

Link	Queue Length (bytes)	Bandwidth in Mbps	Link Delay in ms
$A \to B$	5000	3	4
$B \to C$	5000	1.5	4
$C \to D$	5000	3	4
$B \to A$	∞	10	4
$C \to B$	∞	10	4
$D \to C$	∞	10	4
$endnode \leftrightarrow router$	∞	10	2

TABLE I
BASELINE CONFIGURATION OF THE NETWORK ILLUSTRATED IN FIGURE 2

We varied system settings over a wide range to explore sensitivity and accuracy of the estimation process. The set of baseline simulation settings is shown in Table I.

As the table shows, the central link from $B \to C$ is the bottleneck, and the outgoing buffer at router B leading to C is the buffer whose size we are concerned with estimating. The majority of losses occur at this router, as shown in Table II. Table II shows that even when varying the size of the critical buffer over a wide range, the constriction of bandwidth on the $B \to C$ link results in the majority of losses occuring due to overflow of this buffer.

To estimate buffer size from round trip time requires knowledge of bottleneck link capacity (bandwidth) as well as minimum (*i.e.*, propagation) delay along the path. In the estimations we do here, we assume that bottleneck link bandwidth (C) is known, perhaps by the use of tools such as [5], [22], [17]. We also assume that the minimum delay along the path (t_p) has been estimated, perhaps by observing round trip times over some period and min-filtering (as in [15], [19], [7]). Then if the estimator returns a value t_q to characterize the round trip time when the bottleneck buffer is full, the final estimate becomes:

$$E[B] = C(t_q - t_p). \tag{1}$$

The question then arises how to estimate t_q accurately in the presence of noise. To illustrate our method we show a set of plots in Figure 3. The plots in this figure are histograms of the round trip times of loss pairs taken from our simulation. Simulation parameters followed the baseline case except that the size of the bottleneck buffer B was varied over a range from 5000 bytes to 12,500 bytes. This figure shows that over this wide range of buffer sizes, the histogram of loss pair round trip times is extremely sharply peaked, with tight clustering in each case around the correct value for the given buffer size. The histograms show that there is considerable noise in the measurements (spread in the histogram), but this noise is nowhere near serious enough to complicate the estimation problem. These plots show that the straightforward method for estimating the mode of this distribution is sufficient: we use small bin size, e.g., 1 ms, in the histogram, and simply choose the bin with the largest count. This is the method we adopt in all of the results we report here.

To test our estimation procedure, we varied a number of parameters in our simulation setting. First, we varied the link propagation delay, setting it to 40ms for each link to simulate delays closer to wide-area networks. These longer delays had no observable impact on the quality of the estimates (shown next).

In our main set of experiments, we varied the bottleneck buffer size B over a range of values from 1000 to 20000 bytes, and the link capacity C from 62500 Bytes/sec to 250000 Bytes/sec. In addition, we varied the amount of cross traffic (groups 1, 2, and 3 traffic) in the simulation over three levels: low, medium, and high. These levels are characterized in Table III. In this Table, the loss rate at each router is presented, and in addition the loss rate due to the traffic from the TCP (Observable) sources is presented.

The Table shows that the loss rates due to observable traffic are generally low compared to the total loss rates. Thus all of our results shown here are based on observing only a fraction of all losses at each link. The Table also shows that the loss rates at Router C are much higher than Router B under high load. That is, under high load the bottleneck has shifted to Router C and so our principal assumption (most losses are at the bottleneck) is being violated.

The accuracy of our buffer size estimation procedure under the various settings is presented in Figure 4. The Figure shows that when cross traffic is low (majority of losses are at Router B) the loss pairs method is extremely accurate over all tested values of B and C. Relative error of the estimate (*i.e.*, $(\hat{B} - B)/B$) is less than 5% in all cases. (Note that for the smaller link capacities, relative error is often so small as to be invisible in these plots.) When cross traffic is at a medium level (losses at Router B are comparable to those at Router C) the relative error is still within 50% for all cases, and is considerably better in many cases. Finally, when cross traffic is high (losses at Router C dominate) the relative error is unacceptably high in some cases, but still generally less than 100% in most cases.

	B=5000 bytes		B=7500 bytes		B=10000 bytes		B=12500 bytes	
	All	Fixed	All	fixed	All	Fixed	All	Fixed
Node A	0.103	0.086	0.108	0.075	0.159	0.148	0.064	0.053
Node B	8.568	8.140	7.992	7.517	7.203	6.692	6.155	5.671
Node C	0.030	0.011	0.042	0.019	0.025	0.010	0.021	0.007

TABLE II

LOSS RATE IN THE BASELINE SIMULATIONS, IN PERCENT. "ALL" MEANS THE TOTAL LOSS RATE EXPERIENCED BY THE ROUTER AND "FIXED" MEANS THE LOSS RATE OF THE OBSERVABLE TRAFFIC.

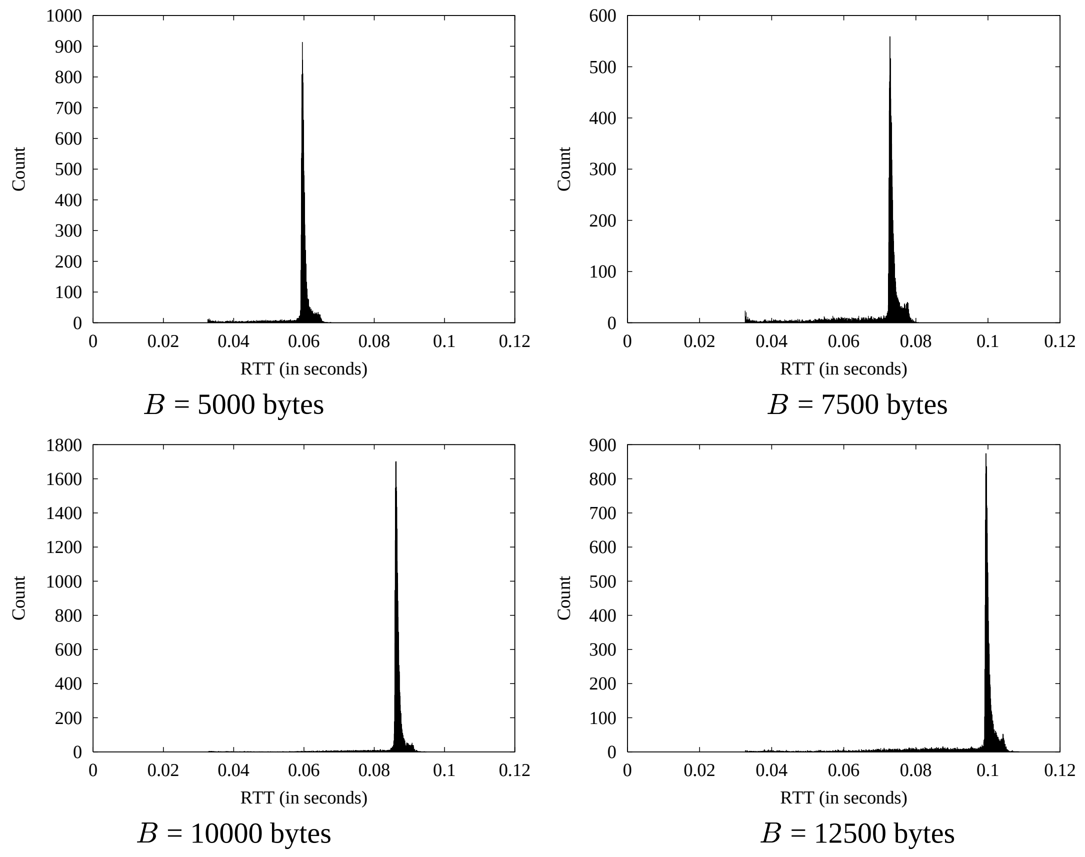


Fig. 3. Loss pairs histograms under different settings of the buffer size at the bottleneck link in Figure 2.

The Figure shows that the accuracy of the estimations improve as the buffer size increases, and as the link capacity decreases. These conditions correspond to buffers with long drain times. The better performance of the loss pairs method on buffers with long drain times can be understood by thinking about the most common way in which loss pairs become inaccurate: when packets p_1 and p_2 become separated in time, they experience the queue in different states. As cross traffic increases, it becomes more likely that the two packets will be separated in time on downstream links. As buffer drain time shrinks, the relative error introduced by the arrival of packets at different times grows larger.

	Router A		Router B		Router C	
	Total	Observable	Total	Observable	Total	Observable
Low Cross Traffic	0.0015	0.00094	0.072	0.063	0.00043	0.00009
Medium Cross Traffic	0.011	0.0074	0.054	0.020	0.0050	0.0002
High Cross Traffic	0.019	0.0081	0.029	0.010	0.050	0.017

TABLE III
LOSS RATES AT EACH ROUTER FOR LOW, MEDIUM, AND HIGH CROSS TRAFFIC LEVELS.

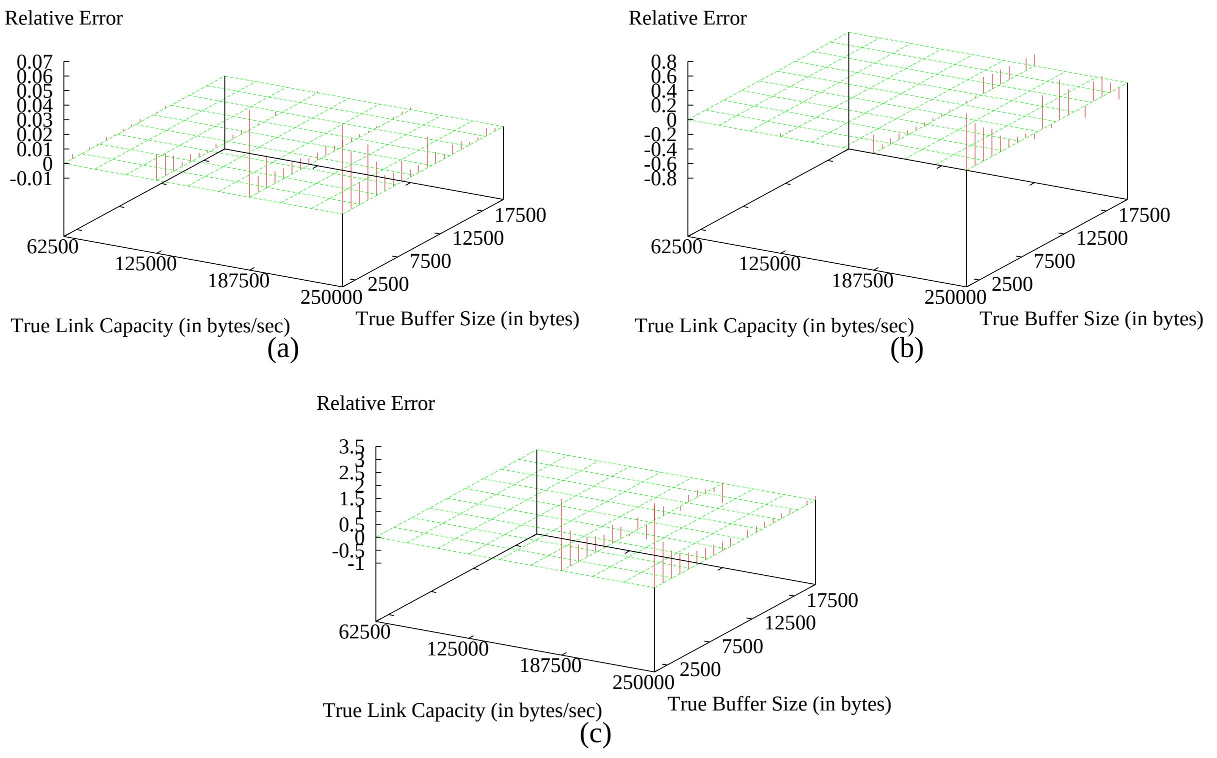


Fig. 4. Performance (relative error) of the loss pairs based buffer size estimation for varying B, C, and (a) Low cross traffic (b) Medium cross traffic, and (c) High cross traffic.

V. CHARACTERING PACKET DROPPING BEHAVIOR IN AQM ROUTERS

One way to characterize the behavior of an AQM scheme is to plot a curve of its drop rate as a function of queue occupancy. Although many such schemes are not fully captured by this curve alone, for any given workload and network configuration this curve is generally fixed. In this section we show how to use loss pairs to discover such curves empirically.

To do so we make certain assumptions. First, we assume that packet drops for a queue occupancy are independent. This is in fact often an explicit goal of AQM schemes. Second we make the assumption that the formation of loss pairs (whether in active or passive style) is independent of the state of the queue. Finally, we make the assumption that the queue dropping probability does not change significantly between the arrival of the first and second packets of a loss pair. These latter two assumptions may not be precisely true in practice, but are reasonable in most cases.

Under these assumptions, we proceed as follows. Let L be the set of packets that are lost at the router during a particular experiment. Let X be the set of packets that arrive at the router and see queue occupancy equal to x. Finally, let us call each pair of packets (p_1, p_2) in the traffic trace that have $\Delta = 0$ a *trial pair.* (Note that trial pairs include all of the loss pairs as well as pairs in which a packet is not lost). Let F be the set of all of the first packets in trial pairs. Each packet $p \in F$ has a successor packet *succ(p)* that completes the trial pair.

To form the curve of drop rate vs. queue occupancy, we would like to estimate

$$d_x = P[\, p \in L \mid p \in X\,] \tag{2}$$

for an arbitrary packet p, over a range of queue occupancies x. Our independence assumptions imply that

$$\{p \in F\} \text{ is independent of } \{p \in X\} \tag{3}$$

and

$$\{p \in F\} \text{ is independent of } \{(p \in X) \;\&\; (p \in L)\} \tag{4}$$

Combining (2), (3), and (4) yields

$$d_x = P[p \in L \mid (p \in X) \;\&\; (p \in F)] \tag{5}$$

Finally we note that

$$\{(p \in X) \;\&\; (p \in F)\} \Leftrightarrow \{\textit{succ(p)} \in X\} \tag{6}$$

so combining (5) and (6) yields

$$d_x = P[\, p \in L \mid \textit{succ(p)} \in X\,] \tag{7}$$

which suggests a feasible estimation algorithm as follows:

$$\hat{d}_x = \frac{\text{number of loss pairs with rtt} = r}{\text{number of trial pairs with rtt} = r}$$

When the fluctuation of round trip time due to other reasons than the AQM queue fluctuations is ignored, we can directly associate a particular queue occupancy x with a particular rtt r via the same method used in the previous section for calculating buffer sizes based on round trip times (Equation 1).

A. Characterizing packet dropping behavior in RED

The original RED active queue management algorithm [10] uses a drop function involving three parameters: th_{min}, th_{max}, and M_p. Roughly speaking, when average queue occupancy is below th_{min} the queue does not drop packets; when it is between th_{min} and th_{max}, the queue drops packets at a rate between 0 and and a function of M_p; and when it is above th_{max}, the queue drops all incoming packets. Improved versions of RED and many variants have been proposed; it is not our intention to evaluate RED or other active queue management algorithms but simply to use RED as an example of the utility of the loss pairs approach on a router that shows relatively sophisticated behavior.

Average queue occupancy is estimated using a exponentially weighted moving average of instantaneous occupancy, with weight parameter w. When $w = 1$, average queue occupancy is considered to be equal to current queue occupancy. As w approaches zero, memory in the system increases and average queue size changes more slowly. In this presentation we will assume $w \approx 1$; in future work we intend to explore estimation when $w \neq 1$.

The dropping rate of a RED queue is defined as zero at th_{min} rising to a maximum at th_{max}. If we can use the procedure implied by (7) to empirically characterize the relationship between dropping rate and queue occupancy between the endpoints th_{min} and th_{max}, then we will succeed.

We demonstrate this estimation procedure under the same the multi-hop network configuration shown in Figure 2. In this case however, the outgoing queue of node B is a RED queue. The parameters of this RED router are shown in Table IV.

The characterization results are shown in Figure 5. Figure 5(a) shows the true (internally measured) dropping rate, while Figure 5(b) shows the dropping

Link Capacity	Buffer Limit	min threshold	max threshold	M_p
1.5Mbps	18 KB	9 KB	18 KB	0.1

TABLE IV
CONFIGURATION OF THE RED QUEUE USED IN OUR EXPERIMENT

rate as measured using the loss pairs method. For comparison we also plot the vertical lines at the values of th_{min} and th_{max} in these two plots. The two curves look quite similar in the range of queue occupancy where they both have clear shapes, *i.e.,* between 9 KB and 13 KB. The reason that both plots do not have clear shapes outside this range is because the queue rarely takes on these values, so sampled values are erratic. This is shown in the histogram of trail pairs in Figure 5(c), which can be used as a guide to indicate the regions where sampling is high enough to be accurate. This property (variable sampling rates for different occupancies) occurs because of our use of passive measurements; presumably, via an active monitoring scheme, the queue could be driven into different regimes.

B. *Charactering Packet Dropping Behavior in BLUE*

In this section we show that our estimation approach can characterize other AQM methods as well, by showing its use on BLUE [9], [8]. The basic setting in this experiment still follows the one in 2, except that the bottleneck queue at router B is now a BLUE queue with its configuration shown in table V which are the same as are used in [9].

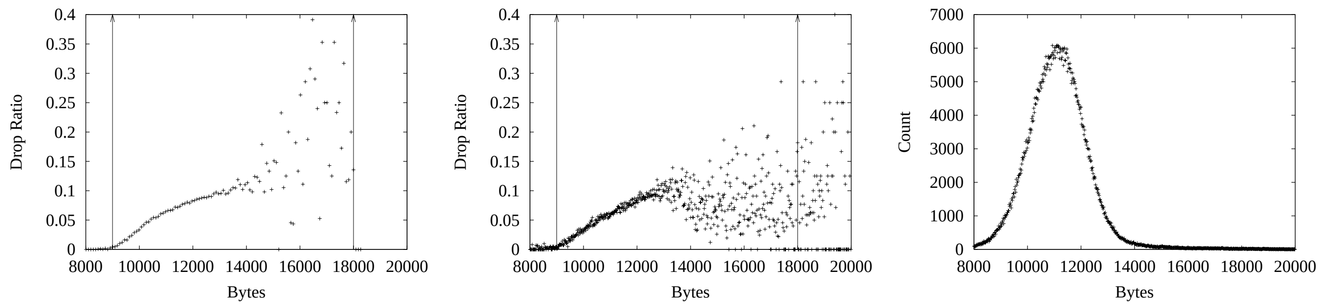
Figure 6 shows the results of our estimation. Fig. 6(a) shows the internal dropping function inside the BLUE queue we monitored in the experiment; it looks roughly flat, consistent with the description in [9], [8]. Fig. 6(b) shows the measured BLUE queue dropping function using loss pairs. It is clear that the measured function is close to the true values up to about 250 packets, after which both are quite noisy. The reason for the noise about 250 packets is again shown in Fig. 6(c) which is a histogram of the actual BLUE queue occupancy. There are very few samples for queue occupancy greater than 250 packets, which makes the sampled values erratic just as in the case of the RED queue. We also note in Fig. 6(b), that there are quite a few sample points lying on the x-axis. This is an artifact of the bin size, which is smaller than the increments of delay possible in the queue; as a result certain bins have no contributions.

Thus we conclude that the loss pairs method can characterize the dropping function of a AQM router quite well; and when used in a passive manner the method is restricted to the range of occupancies that is well sampled.

VI. CONCLUSION

As the Internet grows in size and complexity the need for automated methods for discovery of network properties becomes more acute. The design philosophy of the Internet has focused on keeping network-internal elements simple, while pushing complexity to the end systems; this has affected the kind of network information that is available and the style and methods best used for gathering network information. As a result tools that can make use of measurements taken in end systems to discover network-internal properties are of special interest.

In this paper, we've proposed and evaluated a new such tool for the discovery of network properties: loss pairs. We have shown that loss pairs allow the discovery of router properties which have previously been not directly measurable. In particular, we have shown how to use loss pairs to measure the amount of buffering present at bottleneck routers in the network. We've shown that this method is reasonably accurate and robust over a wide range of network configurations as well as under noisy network conditions. These results have been developed while using loss pairs in a completely passive way, injecting no additional traffic into the network. In addition we've shown that loss pairs are versatile; they can

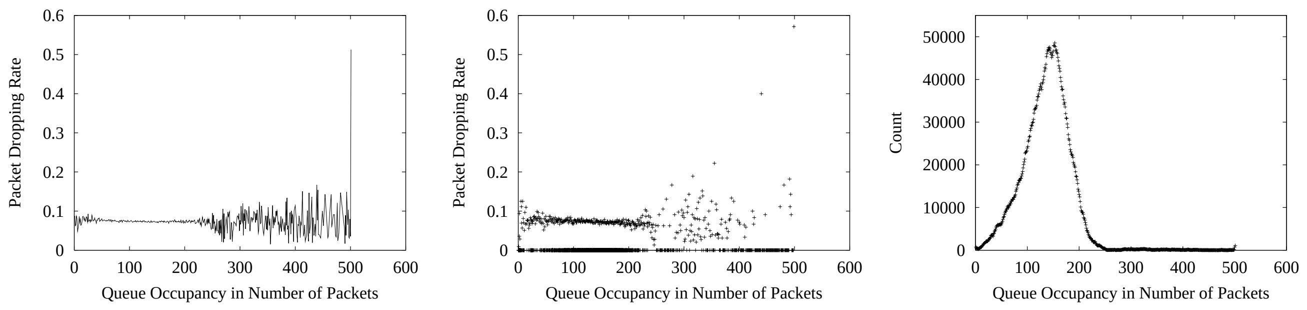


(a) Internal Dropping Function (b) Measured Dropping Function (c) Histogram of Queue Occupancy

Fig. 5. Characterization of RED queue dropping behavior using loss pairs.

Link Capacity	Buffer Limit	Increment	Decrement	Increment Hold Time	Decrement Hold Time
1.5Mbps	500 packets	0.0025	0.00025	0.01 sec	0.01 sec

TABLE V
CONFIGURATION OF THE BLUE QUEUE USED IN OUR EXPERIMENT



(a) Internal Dropping Function (b) Measured Dropping Function (c) Histogram of Queue Occupancy

Fig. 6. Characterization of BLUE queue dropping behavior using loss pairs.

give insight into particular network conditions such as correlations in queue occupancy. Finally, in another example of their versatility, we've shown how to successfully apply loss pairs to the discovery of the configurations of routers employing active queue management.

REFERENCES

[1] Mark Allman and Vern Paxson. On estimating end-to-end network path properties. In *Proceedings of ACM SIGCOMM 1999*, 1999.

[2] Hari Balakrishnan, Hariharan S. Rahul, and Srinivasan Seshan. An integrated congestion management architecture for internet hosts. In *Proceedings of ACM SIGCOMM 1999*, 1999.

[3] Jean-Chrysostome Bolot. Characterizing end-to-end packet delay and loss in the Internet. *Journal of High Speed Networks*, 2(3):305–323, 1993.

[4] R. Caceres, N.G. Duffield, J. Horowitz, and D. Towsley. Multicast-based inference of network-internal loss characteristics. *IEEE Transactions on Information Theory*, November 1999.

[5] Robert L. Carter and Mark E. Crovella. Measuring bottleneck link speed in packet switched networks. *Performance Evaluation*, 27&28:297–318, 1996.

[6] D. Clark, J. Crowcroft, B. Davie, S. Deering, D. Estrin, S. Floyd, V. Jacobson, G. Minshall, C. Partridge, L. Peterson, K. Ramakrishnan, S. Shenker, J. Wroclawski, and L. Zhang. Recommendations on queue management and congestion avoidance in the internet. RFC 2309, April 1998.

[7] A. B. Downey. Using pathchar to estimate internet link

characteristics. In *Proceedings of ACM SIGCOMM 1999*, August 1999.

[8] W. Feng, D. Kandlur, D. Saha, and K. Shin. Blue: An alternative approach to active queue management. In *Proc. of NOSSDAV 2001*, June 2001.

[9] W. Feng, D. Kandlur, D. Saha, and K. Shin. Stochastic fair blue: A queue management algorithm for enforcing fairness. In *Proc. of INFOCOM 2001*, April 2001.

[10] S. Floyd and V. Jacobson. Random early detection gateways for congestion avoidance. *IEEE/ACM Transactions on Networking*, 1(4):397–413, August 1993.

[11] Khaled Harfoush, Azer Bestavros, and John Byers. Robust identification of shared losses using end-to-end unicast probes. In *Proceedings of ICNP'98: The 6th IEEE International Conference on Network Protocols*, Osaka, Japan, October 2000.

[12] Khaled Harfoush, Azer Bestavros, and John Byers. Measuring bottleneck bandwidth of targeted path segments. Technical Report BUCS-2001-017, Boston University Computer Science, July 31 2001.

[13] Information Sciences Institute. The `ns-2` simulator. Available at `http://www.isi.edu/nsnam/ns/`.

[14] Van Jacobson. Congestion avoidance and control. In *Proceedings of SIGCOMM '88*, pages 314–329, 1988.

[15] Van Jacobson. pathchar. ftp://ftp.ee.lbl.gov/pathchar/, 1997.

[16] Srinivasan Keshav. A control-theoretic approach to flow control. In *Proceedings of SIGCOMM 1991*. ACM SIGCOMM, 1991.

[17] K. Lai and M. Baker. Measuring bandwidth. In *Proceedings of IEEE INFOCOM '99*, March 1999.

[18] K. Lai and M. Baker. Measuring link bandwidths using a deterministic model of packet delay. In *Proceedings of ACM SIGCOMM 2000*, August 2000.

[19] Bruce Mah. pchar. http://www.employees.org/˜bmah/Software/pchar, 1999.

[20] M. Mathis. Treno bulk transfer capacity. Internet-Draft: Work in progress; draft-ietf-ippm-treno-btc-03.txt, Feb 1999.

[21] Kihong Park, Gi Tae Kim, and Mark E. Crovella. On the relationship between file sizes, transport protocols, and self-similar network traffic. In *Proceedings of the Fourth International Conference on Network Protocols (ICNP'96)*, pages 171–180, October 1996.

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

[23] Stefan Savage. Sting: a tcp-based network measurement tool. In *Proceedings of the USENIX Symposium on Internet Technologies and Systems*, 1999.

[24] Srini Seshan, Mark Stemm, and Randy Katz. SPAND: shared passive network performance discovery. In *Proceedings of the USENIX Symposium on Internet Technologies and Systems*, 1997.

[25] Walter Willinger, Murad S. Taqqu, Robert Sherman, and Daniel V. Wilson. Self-similarity through high-variability: Statistical analysis of Ethernet LAN traffic at the source level. In *Proceedings of SIGCOMM '95*, pages 100–113, Boston, MA, 1995.