

# Why Don't Delay-based Congestion Estimators Work in the Real-world?\*

Sushant Rewaskar    Jasleen Kaur    Don Smith

Department of Computer Science  
University of North Carolina at Chapel Hill

*Technical report No. TR06-001*

January 20, 2006

## Abstract

A number of designs have been proposed for complementing TCP's treatment of packet loss as an implicit signal of congestion, with a signal derived from measurements of round-trip times (RTT). The premise of such delay-based congestion estimators (DBCEs) is that congestion is reflected in queueing delays that can be detected by measuring changes in RTT. We conduct a large-scale empirical analysis of real-world TCP connections to evaluate the effectiveness and limitations of five prominent DBCEs. Our findings are that none of the five perform well (correctly indicate congestion before a loss is experienced) for a large percentage of real-world TCP connections. They also often perform poorly by having high rates of false-positive and false-negative estimates of congestion. Further, we find that the connection characteristics that most influence the performance of these DBCEs are so diverse that designing an effective DBCE for all types of connections is still an open research problem.<sup>1</sup>

## 1 Introduction

TCP is the dominant transport protocol used in the Internet; it provides several services including reliable trans-

fer, flow-control, and congestion-control. For congestion-control, most deployed versions of TCP rely on *packet losses* in order to detect network congestion and respond to it by drastically reducing their sending rate. Consequently, packet losses significantly degrade the connection performance. An alternate strategy is that of *delay-based congestion-control* (DBCC), that attempts to avoid packet losses by: (i) detecting congestion early through increase in packet round-trip times (RTTs), and (ii) reducing the connection sending rate in order to alleviate congestion before packet losses can occur.

Two key issues determine the effectiveness of DBCC mechanisms: *can RTTs be used to reliably predict impending packet losses*; and *can the subsequent reduction in sending rate prevent packet losses from occurring*. In this paper, we conduct a large-scale empirical analysis of real-world TCP connections in order to understand the first of these issues. Specifically, we: (i) evaluate well-known delay-based congestion estimators (DBCEs) for their effectiveness in signaling congestion before packet loss occurs, and (ii) investigate which connection characteristics are likely to influence the performance of such estimators.

We analyze more than 280,000 real-world TCP connection traces, captured at five different locations around the world. We first extract reliable estimates of per-segment RTTs and packet losses for each connection. We then run the sequence of extracted RTTs for each connection through each DBCE in order to evaluate its efficacy in predicting losses. We then study the characteristics of connections for which the estimators fare well, as well as

---

\*This research was supported in part by NSF CAREER grant CNS-0347814, a UNC Junior Faculty Development Award, and NSF RI grant EIA-0303590.

<sup>1</sup>This research was supported in part by NSF CAREER grant CNS-0347814, a UNC Junior Faculty Development Award, and NSF RI grant EIA-0303590.

of those for which they do not fare well. Our high-level conclusions are:

- CIM is overall the best estimator, but even it has better than 80% loss-prediction ability for less than 20-40% of the connections. Further, it fails completely (predicts none of the losses) in 20-30% of the connections.
- Vegas has the poorest performance. It almost never makes a correct prediction of congestion before a loss occurs (on the other hand it is almost never wrong because it is quite conservative).
- The connection characteristics for which the DBCEs have excellent (or poor) performance are highly diverse and often have opposing effects for different estimators or performance metrics. For example, the accuracy for one estimator (CARD) increases with increasing loss rate while for another (Vegas) it decreases.

The rest of this paper is organized as follows. In Section 2, we discuss the state of the art in the design and evaluation of DBCEs. We outline our approach in Section 3 and present our analysis results in Sections 4 and 5. We conclude in Section 6.

## 2 State of the Art

In order to avoid packet losses and the associated performance costs of TCP loss-recovery, Delay-based Congestion Control (DBCC) works using two kinds of mechanisms. The first is a *delay-based congestion estimator* (DBCE), which maintains running estimates of the network delays experienced by packets of the connection, and uses these to predict the onset as well as the end of network congestion. The second is a congestion-avoidance (CA) mechanisms, which reacts by reducing the connection sending rate, whenever the DBCE indicates the onset of congestion. The performance of any DBCC scheme depends on at least two factors: *how well and early can the DBCE predict impending packet loss, and how efficiently can the CA avoid packet losses?* In this paper, we address the first of these issues. Below, we briefly describe the state of the art in DBCE design and evaluation.

### 2.1 Delay-based Congestion Estimators (DBCEs)

DBCEs rely on the assumption that during periods of congestion, a connection's packets would experience higher queuing delays at the congested link — this should translate to an increase in packet round-trip times (RTTs). So by sampling per-packet RTTs, and comparing them to a base RTT value (measured in the absence of congestion), a DBCE infers the onset as well as alleviation of congestion. The hope is that DBCEs can detect the onset of congestion much earlier than the occurrence of packet loss and the corresponding CA mechanisms can potentially avoid the loss. Existing DBCEs differ primarily in the RTT-derived metric and the base metric used for estimating congestion. Table 1 lists the choice of these metrics for some of the prominent DBCEs. We briefly discuss each here—we refer the reader to the respective publications for more details. *CARD* [13] is based on the argument that when the network is lightly loaded, the connection throughput increases with increase in sending rate. But as network load reaches saturation, connection throughput does not increase further and RTTs start to increase. *CARD* uses this change in delay as an indicator of congestion. The *Tri-s* scheme [23] compares the current throughput gradient to the initial throughput gradient. Depending on whether the former is greater or small, *Tri-s* infers the absence or presence of congestion, respectively. *Dual* [24] is based on the assumption that the minimum RTT of a connection corresponds to the absence of queue build-ups along the network path, and the maximum RTT is the sum of this minimum RTT and the maximum queuing delay that can be experienced before packet loss. When the current RTT exceeds the average of the min and max RTTs, *Dual* estimates congestion and its CA algorithm modifies the sending rate. *TCP Vegas* [9] attempts to maintain enough data in the network such that it exceeds the delay-bandwidth product by a small amount. Its congestion-estimator relies on the fact that if the sending rate is much larger than that corresponding to this amount, the connection RTT would increase (and its throughput would decrease). The decrease in connection throughput is taken as an indicator of congestion. *TCP FAST* [15] is a variant of Vegas designed for high-speed networks. Its congestion estimator is similar to that of Vegas; although it uses increase in RTT as an

Estimator	Year proposed	Metric Used	Compared to	Equation Used by DBCE
CARD	1989	Delay Gradient	Previous RTT	$\frac{RTT - prevRTT}{RTT + prevRTT} * \frac{Window + prevWindow}{Window - prevWindow} > 0$
Tri-s	1991	Throughput Gradient	Initial RTT	$\frac{firstRTT}{Window - prevWindow} * (\frac{Window}{RTT} - \frac{prevWindow}{prevRTT}) < 0.5$
Dual	1992	Delay	Min and Max RTT	$\frac{minRTT + maxRTT}{2} < RTT$
Vegas	1994	Throughput	Min RTT	$window * (1 - \frac{minRTT}{RTT}) > 2$
CIM	2003	Delay	Previous 20 RTTs	$avg(RTT_2) > avg(RTT_{20}) + RTT_{std-dev}$
FAST	2003	Delay	Min RTT	$1 - \frac{100}{window} \leq \frac{minRTT}{avgRTT}$

Table 1: Estimator Descriptions

indicator of congestion, its DBCE can be shown to be a simple derivative of the Vegas throughput-based DBCE. The *Congestion Indication Metric* (CIM) was proposed in [18] as a DBCE metric. This metric compares the most recent RTT samples (typically 1-2 samples) to the average RTT of several immediately preceding samples (typically, 20). If the most recent sample is greater than the average RTT, it concludes congestion has occurred. Of all of the DBCEs listed in Table 1, CIM is the only one that relies on a *history* of recent RTT samples—all of the rest use only the most recent RTT in their decision-making process.

## 2.2 Evaluation of DBCEs

The efficacy of a DBCE depends on its ability to predict the onset of congestion (or predict an impending loss) accurately and in a timely manner. Several past studies have attempted to evaluate the efficacy of existing DBCE designs (or DBCCs, in general). We briefly mention these below.

- [16] argues that loss-based congestion control is inherently unstable in high-bandwidth networks. It contends that the binary signal of loss/no-loss is too coarse to allow fine adjustment of send-rates, which is needed for stability at high speeds. A multi-bit signal can be obtained by using queuing delays as an additional indicator of network conditions. The authors, however, do not present any experimental results to illustrate/prove their point.
- In [21], the authors briefly discuss a set of conditions under which DBCC will fail. They argue that DBCC is bound to fail if: (i) the max queuing delay at the

bottleneck link is too small compared to the connection RTT, or (ii) the RTT sampling rate is less than the required Nyquist rate, or (iii) there is high degree of aggregation along a path and a connection’s contribution to the total load is too small, or (iv) packet loss is not handled effectively. The paper, however, merely presents arguments and does not conduct a detailed investigation to validate the conclusions.

- [8] investigates the correlation between the number of packets in flight and RTT. The idea is that DBCC is likely to be effective only if its CA can alleviate congestion by reducing the sending-rate (that is, the connection is *self-congesting*). This is not likely to happen if the correlation between sending rate (or packets in flight) and observed RTTs is low. The authors passively analyze 14,218 connections instantiated over 737 different paths. They found that, in general, the coefficient of correlation between RTT and packets in flight is weak [8]. The paper, however, does not investigate the collective impact of *all* other connections, that share such a bottleneck link, also responding to congestion.
- [18] presents a case against use of DBCC. It defines several metrics for quantifying correlation between packet losses and high RTTs. The authors conduct passive analysis of a large number of connections instantiated over 7 paths using the CIM DBCE. They find that using CIM as a congestion estimator can reduce connection loss rates. However, it also results in a large number of false positives resulting in a 37% reduction in the aggregate throughput. This paper does not analyze false negatives as well as the

influence of connection characteristics on the performance of CIM.

- In [6, 7] the authors have used network simulations to evaluate the ability of Vegas, CARD, and Tri-s to predict loss. They found that all three methods are rarely better than a “random coin-tossing” estimator. However, the Vegas method was found to be slightly better than the other two methods. Unfortunately, their use of simulations prohibits evaluation under a wide range of connection characteristics, as well as prohibits sampling the wide variety of real-world network conditions which may affect the performance of an estimator.

In this paper, we make two key contributions. First, we conduct a comprehensive evaluation of all DBCEs listed in Table 1 using more than 280,000 real-world TCP connections, captured at a diverse set of locations around the world. We believe that this is the *first* DBCE evaluation of such a large scale and diversity. Second, we study the characteristics of these connections in order to analyze their influence on the efficacy of delay-based congestion estimation. To the best of our knowledge, this issue has not been addressed before. Note that it is not our objective to design an optimal DBCE, but rather to evaluate prominent estimators and investigate why they succeed or fail for certain connections.

### 3 Methodology

**Basic Approach:** Our basic approach for evaluating DBCEs is to analyze these against passively-collected traces containing large numbers of real-world TCP connection. For each connection, we first compute reliable estimates of its per-packet RTTs and packet losses. The packet loss information is computed as a series of *loss episodes*, where we group all losses that occur within the same flight of packets into a single loss-episode. We then run the RTT estimates against prominent DBCEs and divide the connection into alternating phases of congestion and no-congestion periods, interspersed with the above-defined loss episodes. In this section, we describe each of the above steps in some detail.

#### 3.1 Extracting Valid RTT Samples

The *round-trip time* of a TCP segment is defined as the time it takes for the segment to reach the receiver and for a segment carrying the generated acknowledgment to return to the sender. We use the methodology described in our previous work in [5] for extracting reliable estimates of per-segment RTTs from connection traces collected near TCP sources. The basic idea is fairly simple: *measure the time difference between the observed transmission of a data segment from the source and the observed receipt of an ACK containing an acknowledgment number that exactly corresponds to (is one greater than) the highest sequence number contained in an observed data segment.*<sup>2</sup> Since most of the traces we use are collected using DAG cards [1], the relevant timestamps are accurate to sub-microsecond precision.

The above simple idea, however, is complicated by several factors. In choosing how to deal with these, our guiding principle was to be conservative and include in our data only those RTT values where there is an unambiguous correspondence between an acknowledgment and the data segment that triggered its generation. We start with Karn’s algorithm [17] and add additional tests to ensure such valid RTT estimates. Some of these are mentioned below.

The most serious complications arise from lost and re-ordered segments. If a SYN or data segment is retransmitted and an ACK matching it is received, it is ambiguous whether the RTT should be calculated from the transmission time of the initial segment or that of the retransmitted segment. Further, in a flight of data segments, the last segment may have a matching ACK but it may be generated only after the retransmission and receipt of a lost segment earlier in the flight. To eliminate the possibility of invalid (and large) RTT measures in such cases, we ignored all RTT estimates yielded by retransmitted data segments and by those transmitted between an original segment and its retransmitted copy. Another subtle complication arises because segments may occasionally be lost in the network between the sender and the tracing monitor. In this case, the retransmission of the segment will be detected as an out-of-order transmission of a sequence number, not as

---

<sup>2</sup>We also obtain an RTT sample value for the connection by measuring the elapsed time between an out-bound SYN segment and the corresponding SYN+ACK segment.

a duplicate transmission. We tackled such cases by ignoring all RTT estimates for data segments that were in-flight (not yet acknowledged) when an out-of-order segment was seen. Table 2 lists the total number of valid RTT samples yielded by our traces.

Sometimes a TCP endpoint may delay sending the ACK for an incoming segment for up to  $500ms$  in order to piggyback the ACK on the next outgoing data segment (common implementations delay the ACK only up to  $200ms$ ). This means that some RTT values may have additional time added because the ACK is delayed. However, TCP implementations may not have more than one delayed ACK pending at any time—commonly called “ACK every other segment”. In this case, half of the data segments will yield valid, non-delayed RTTs and the others will not yield any RTT sample at all.

### 3.2 Extracting Packet Losses

**Basic Idea:** TCP senders infer packet losses using one of several mechanisms: triple-duplicate ACKs (TDAs), retransmission timeouts (RTOs), partial ACKs and selective ACKs received during fast retransmit [11, 19]. We derive reliable estimates of packet losses in each connection using the approach described in [22] (under submission). The basic idea is to: *implement and execute a partial state-machine representing the TCP sender, that uses the ACK stream of each connection observed at the trace-collection point, in order to track the triggering of loss detection/recovery mechanisms at the corresponding sender.*

**Challenges:** Several challenges complicate the validity and implementation of this basic idea:

- Simply replicating TCP-sender state and logic while processing a connection trace is not sufficient for reliably inferring packet losses. This is because:
  - *Some packet losses do not trigger TCP’s loss detection/recovery phases.* For implementation efficiency, TCP senders maintain only a limited history/memory about unsuccessful transmissions. In particular, if multiple packet losses are followed by a timeout, the sender explicitly discovers and recovers only from the first of those losses. As a result, the remaining

packet losses may not get discovered by simply tracking the invocation of sender-side loss detection/recovery mechanisms.

- *A TCP sender may incorrectly infer packet losses.*

TCP may retransmit a packet too early if its RTO computation is not conservative. Furthermore, some packet re-ordering events may result in the receipt of triple-duplicate ACKs, triggering a loss detection/recovery phase in TCP. In [22] we present the example of a real connection in which a *single* packet reordering event resulted in the triggering of 64 subsequent phases of fast retransmit/recovery, which lasted for more than 5 seconds!

- TCP implementations differ (sometimes significantly) across different operating systems (OSes) in either their interpretations or their conformance to TCP specification/standards. Furthermore, a few aspects of TCP—such as how a sender responds to SACK blocks—are not standardized. As a result, the sender-side state machines are specific to the OS they run on. This results in two main challenges:
  - The difference in implementations on different OSes necessitates that we implement different programs to analyze connections originating from different sender-side OSes. More significantly, given the trace of a TCP connection, it is non-trivial to identify the corresponding sender-side OS and decide which OS-specific analysis program to use for analyzing the connection.
  - Most OSes either have proprietary code or have insufficient documentation on their TCP implementations. Without detailed knowledge of the loss detection/recovery implementations, it is not possible to replicate these mechanisms in our OS-specific analysis programs.
- Packet traces used in passive analysis are typically collected at links that aggregate traffic from a large and diverse population. As a result, there may be several network links on the path between a TCP sender and the trace monitoring point. Thus, the

data packets transmitted by the sender may experience delays, losses, or reordering before the monitor observes them; the same is true for ACK packets that traverse between the monitor and the sender. Consequently, the data and ACK streams observed at the monitor may differ from those seen at the TCP sender. Thus, the trace-analysis programs may not be able to accurately track the sender-side state machine.

**Our Methodology:** Our methodology for addressing the above challenges and reliably inferring TCP losses can be summarized as follows.

1. We first extract the implementation details of four prominent TCP stacks (Windows XP, Linux 2.4.2, FreeBSD, Solaris) by using an approach similar in spirit to the *t-bit* approach described in [20]. Specifically, we run the Apache web-server on each of the above OSes in an experimental lab, and implement an application-level TCP receiver that initiates TCP connections to each of the server machines and requests HTTP objects. By artificially generating pre-specified ACK sequences, the receiver triggers loss detection/recovery mechanisms on the sender-side stacks (including TDAs, timeouts, partial ACKs, etc). We then infer the sender-stack details based on the manner in which the server responds to the ACK stream. Details of the extracted characteristics can be found in [22].
2. We then replicate the loss detection/recovery related mechanisms in four OS-specific trace-analysis state machines—these state machines use the data and ACK streams as input. Loss indications in the ACK stream are used to only tentatively trigger state transitions, which are confirmed only by subsequent segment retransmission behavior observed in the data stream.
3. We then augment these machines with extra logic and state about all previously-transmitted packets, in order to infer packet losses with accuracy greater than TCP.
4. We then run each connection trace against all four machines and use the results from the one that can

explain most of the observed retransmissions. In case more than one machine matches this criteria, we select one randomly (both give same results).

We have implemented the above machines in the C programming language. Our implementations can run all four state machines on more than a million connections in a few minutes.

Several details of our loss-extraction methodology and implementation, as well as comparison to existing methodologies, have not been included in this section due to space constraints. These details can be found in [22].

### 3.3 Data Sources

We analyze TCP connection traces collected from 5 different global locations. Table 2 describes the traces used in our analysis. These traces are collected from links with transmission capacity ranging from 155 Mbps to OC-48. The *abi* traces [3] are collected from a backbone link of the Internet-2 network (Abilene); the *jap* trace [4] is collected off a trans-Pacific link connecting Japan to the US; the *unc* and *lei* [2] traces are collected at the campus-to-Internet links of the University of North Carolina and University of Leipzig, respectively; the *ibi* trace captures traffic served by a cluster of high-traffic web-servers hosted at UNC (*ibiblio.org*). All traces except the one from the link to Japan were collected using Endace DAG cards. All traces represent a fairly diverse and large population.

For our analysis in the rest of this paper, we use only those connections that transmit at least 10 segments. Furthermore, since our objective is to study the ability of DBCEs in predicting packet losses, we select only those connections that experience the loss of at least one packet. Table 3 shows the impact of applying the latter filter. While less than 13 – 41% of the set of connections that transmit at least 10 segments also experience at least one packet loss, these connections carry most of the bytes among this set. We also found that the traces vary significantly in the distribution of bytes transmitted per connection—this adds to the diversity of our results.

Figure 1 plots the distribution of loss rates observed for each connection among the lossy connections. We find that per-connection loss rates observed in our traces vary from less than 0.1% to more than 80% with most in the range of 1-10%. Furthermore, the distribution of loss rates

Trace	Duration	Average TCP Load	# Destination	# Connections	# Bytes	# Packets	# RTT
Abilene-OC48-2002 (abi)	2h	211.41 Mbps	3452.5 K	7.1 M	190.3 G	160.1 M	81.3 M
Liepzig-1Gbps-2003 (lei)	2h 45m	9.53 Mbps	1430.2 K	2.4 M	11.8 G	17.3 M	11.8 M
Japan-155Mbps-2004 (jap)	4h	1.93 Mbps	28.2 K	0.3 M	3.5 G	3.7 M	1.9 M
UNC-1Gbps-2005 (unc)	4h	74 Mbps	1082.9 K	14.5 M	133.3 G	151.0 M	85.0 M
Ibiblio-1Gbps-2005 (ibi)	4h	90.64 Mbps	88.7 K	0.9 M	163.2 G	158.9 M	75.6 M

Table 2: General Characteristics of Packet Traces

Trace	Aggregate Loss Rate	All Connections			Lossy Connections		
		# Connections	# Bytes	# Packets	% Connections	% Bytes	% Packets
abi	0.7 %	388.9 K	180.1 G	148.5 M	13.41 %	31.2 %	46.8 %
lei	2.2 %	75.4 K	10.5 G	12.6 M	13.37 %	34.0 %	21.74 %
jap	6.7 %	18.5 K	3.3 G	3.1 M	41.4 %	52.42 %	54.84 %
unc	1.8 %	774.8 K	121.3 G	129.6 M	18.39 %	61.55 %	57.0 %
ibi	1.2 %	287.5 K	161.8 G	157.2 M	24.84 %	67.7 %	73.3 %

Table 3: Characteristics of Connections That Transmit More Than 10 Segments

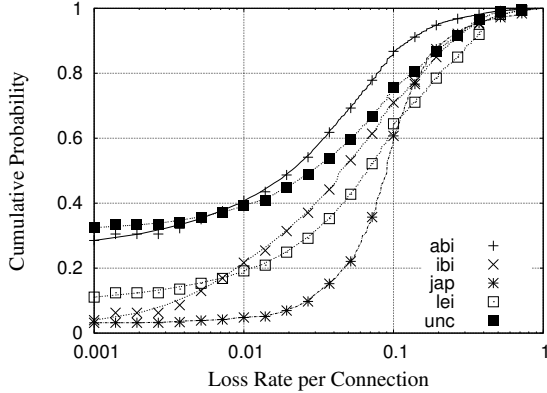


Figure 1: Loss Rate Distribution Among Lossy Connections

among these connections varies significantly across the traces. This illustrates the diversity of the data sets we are using for our analysis.

### 3.4 Analysis Methodology

**Defining Connection States:** For each of the connections analyzed, we first extract all reliable estimates of RTTs and packet losses using the approaches described in Section 3.1 and 3.2. Further, we group losses into *loss-episodes*, in which we group all losses that occur within the same *flight* of packets into a single loss-episode; each loss episode starts with the transmission of the first packet in a flight that later is lost, and ends with the transmission of the last packet that gets lost. Thus, the analysis of each connection yields a series of loss-episodes with well-defined beginning and ending packet transmissions.

We next run the extracted RTT samples of each connection through each of the DBCEs we are evaluating—for each RTT sample, a DBCE either signals congestion (CN) or no-congestion (NCN). Thus, for each connection-estimator pair, we obtain a corresponding series of CN and NCN indications. Note that reliable RTT estimates can not be obtained during a loss-episode, as explained in Section 3.1. Thus, the time-series of CN and NCN indications does not overlap with the time-series of loss-episodes. We use this fact to divide each connection into a series of three *mutually-exclusive* phases (on a per-DBCE

basis): congestion-phase, no-congestion phase, and loss-episode. Loss-episodes are defined as mentioned above. A congestion phase begins with the transmission time of the data segment that is the first to signal a CN either after a loss-episode or after a previous NCN indication. A congestion phase ends with the transmission time of the last contiguous data segment that signals a CN. We define the no-congestion phase to begin with the transmission time of the first data segment of the connection, or the end of a loss-episode or congestion-phase. A no-congestion phase ends with the start of either a congestion-phase or a loss-episode.

At the end of the above classification for each connection-estimator pair, the connection is divided into a sequence of these three phases. Below we describe our metrics for evaluating a DBCE based on this sequence of phases.

**Evaluation Metrics:** For each connection, we define a set of three evaluation metrics for each DBCE:

- *Loss Prediction Ability (LPA):* The first metric captures the ability of the DBCE to predict an impending loss. It is defined as the fraction of loss-episodes that are immediately preceded by a congestion-phase. The higher the value of this metric, the better is the corresponding DBCE (referred to as an HB, higher-is-better, metric) [14].
- *False Positives:* Note that an aggressive DBCE that signals a congestion-phase most of the time is likely to lead to a high value of the loss-prediction ability. However, the corresponding CA algorithm would keep reducing the connection send rate and degrade the connection throughput. We capture this undesirable factor by defining the rate of false positives to be the fraction of congestion-phases that are not succeeded by a loss-episode (but by a no-congestion phase, or the end of connection). The lower the value of this metric, the better is the corresponding DBCE (referred to as an LB, lower-is-better, metric).
- *False Negatives:* Our third metric evaluates the conservative nature of a DBCE. We define the rate of false negatives to be the fraction of no-congestion phases that are succeeded by a loss-episode. The lower the value of this metric, the better is the corresponding DBCE (an LB metric).

Note that all of the above metrics take values that lie between 0 and 1. We define a connection to have a “excellent” value of an HB (LB) metric if it is 0.8 or more (0.2 or less); a connection is defined to have a “poor” value of an HB (LB) metric if it is 0.2 or less (0.8 or more).

**Influence of Connection Characteristics:** Several connection characteristics are likely to influence the ability of prominent DBCEs in predicting impending loss (as also argued in [21]). For instance, if the min RTT of a connection is much larger than the max queuing delays experienced at the bottleneck link, a DBCE that use the min RTT as a base value is unlikely to accurately detect connection. Furthermore, if the sending rate (and thus, the RTT-sampling rate) of a connection is too small, a DBCE is unlikely to sample enough RTTs before a packet loss occurs. We study the influence of these and other connection characteristics on the performance of a DBCE. For this, we rely on regression analysis — our methodology is described in detail in Section 5.

## 4 Evaluation of Congestion Estimators

### 4.1 Loss Prediction Ability

Each of the five algorithms (CARD, CIM, Dual, Tri-S, and Vegas) was simulated on each of the five traces (Abilene, Japan, Leipzig, Ibiblio servers, UNC-Campus). For each connection in a given trace, we computed the loss-prediction ability (LPA) metric. The results are shown in Figures 2, 3, 4, 5, 6. These figures give the cumulative distribution over all connections of their per-estimator LPAs.

While there are a few differences in the performance of the algorithms for different traces, the results are still remarkably similar across traces. In general, the most effective algorithms are CARD and CIM with Dual being somewhat less effective but still comparable. But even these better-performing algorithms do not appear to have very strong capability of predicting impending loss, and in fact may entirely miss all losses in a significant number of connections. Most striking is how ineffective Tri-S and (especially) Vegas are at indicating congestion before a loss occurs. The most notable differences among the results for various traces are that Vegas performed some-



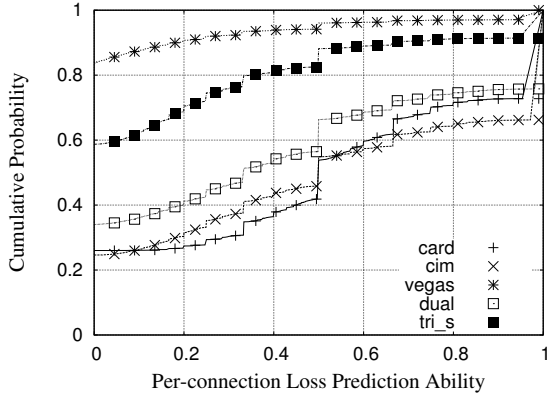


Figure 2: Abilene: Loss Prediction Ability

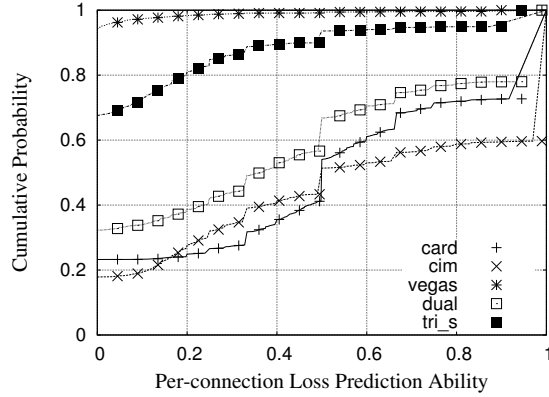


Figure 4: Japan: Loss Prediction Ability

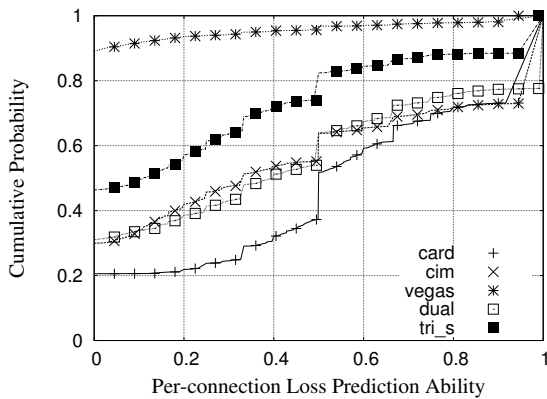


Figure 3: Leipzig: Loss Prediction Ability

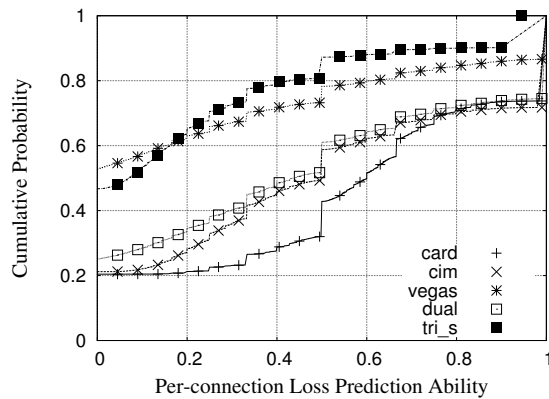


Figure 5: Ibiblio: Loss Prediction Ability

what better than Tri-S on the Ibiblio trace while Tri-S performed much better on the Leipzig trace.

Consider the results from the UNC-Campus trace shown in Figure 6 which are typical of the results for the other traces. The best estimator of congestion overall is the CIM algorithm but for almost 30% of the connections in this trace, the LPA was 0%. On the other hand, for almost 40% of the connections, CIM had 100% LPA. For the remaining 30% of connections, CIM had an LPA of 50% or more in less than one-third of them. Note that the results for CARD are comparable—this seems to indicate there might be certain characteristics of the connection dynamics or of the network path followed by the connection that result in either very good or very poor

performance by these algorithms. We will explore this issue further in Section 5. The Vegas algorithm is clearly ineffective in its loss-prediction ability. For example, in about 85% of connections, Vegas has 0% LPA. On the other hand, Vegas had an LPA of 100% for only about 5% of the connections.

For the remainder of this section we will show plots only for the UNC-Campus trace for space concerns. The results from the other four traces are very similar and this trace is the most recent (May 2005) among those used for the analysis.

Another issue to consider in evaluating these algorithms is whether a correct indication of congestion (one that precedes a loss event) is sufficiently timely to allow a

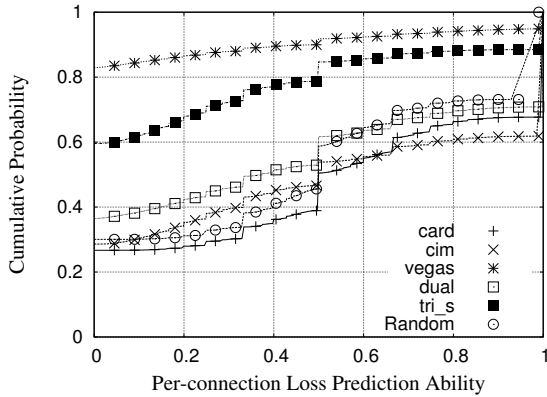


Figure 6: UNC-Campus: Loss Prediction Ability

congestion control mechanism to take action based on the indicator. For many congestion-control designs an indicator of congestion that precedes a loss by at least one RTT should be timely enough to allow effective adjustment of the connection sending rate. Figure 7 shows the distribution of the durations of all congestion-phases that precede a loss-episode in the UNC-Campus trace. The durations are expressed normalized to the current exponentially-weighted average of the RTT for the corresponding connection (using a weight of 1/8 for the most recent observation). The results are quite consistent for all five algorithms. They show that 50-60% of congestion-phases that are actually followed by a loss-episode begin at least one RTT before the first loss. This indicates that a congestion control mechanism reacting to the congestion indicator should potentially be able to react in time in order to reduce congestion.

Considering that even the best DBCEs, CARD and CIM, have an LPA greater than 50% in only about 50% of connections, we compared the performance of all the algorithms with an essentially *random* congestion indicator. To do this, we re-processed the UNC-Campus trace replacing the five algorithms in our analysis with a simple simulation of a random estimator. For each valid RTT obtained that could be used by one of the algorithms, we generated a random number between 0 and 1. If the number was 0.5 or greater, the connection state was set to indicate congestion and if less than 0.5 it was set to indicate no congestion. This result is also plotted in Figure 6. Sur-

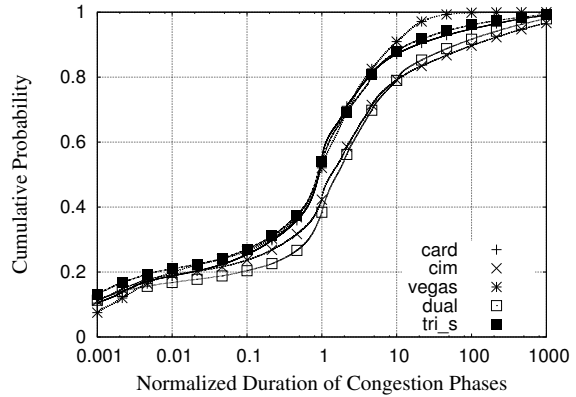


Figure 7: UNC-Campus: Congestion Phase Durations Before Loss

prisingly, the random estimator performed almost as well as CIM and CARD and significantly better than Tri-S and Vegas. For example, the random estimator had a 0% LPA for about 30% of connections as did CARD and CIM. It had greater than 80% LPA in almost 30% of connections compared with about 40% of connections for CIM!

## 4.2 False Positives

Another metric we consider in evaluating these congestion-estimator algorithms is the rate of false-positive indications. This false-positive rate is important because a high rate indicates that a congestion control mechanism, reacting to this signal, may take action too often and reduce the connection's sending rate when such action is not needed. Figure 8 shows the cumulative distribution of false-positive rate over all connections in the UNC-Campus trace for the five algorithms and the random estimator. Notice that Vegas exhibits excellent performance (low rate) for this metric. For about 75% of connections its false-positive rate was 0% and it had a false-positive rate of 100% in less than 15% of connections. Considering, however, that the overall performance of Vegas in detecting congestion (LPA) is so poor, this seems more of an indication that the algorithm is too conservative (in the sense that it is rarely wrong when it indicates congestion but it misses many instances of congestion severe enough to lead to losses). The best LPA algorithm, CIM, is also the best among the

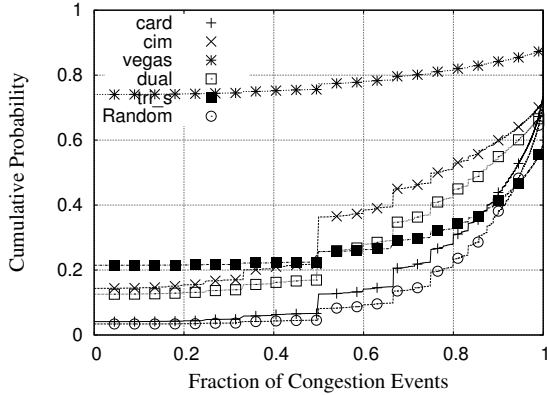


Figure 8: UNC-Campus: False Positive Rate

other algorithms for the false-positive rate but even its performance is relatively poor (high rate) compared to that of Vegas. CIM had a false-positive rate of 50% or more in about 65% of connections and a false-positive rate of more than 80% in 50% of connections. This result indicates that CIM is too aggressive at indicating congestion and that a congestion control mechanism, reacting to its signals, would take action too often and reduce the connection's transfer rate needlessly. Similarly, all the other algorithms are even somewhat more aggressive than CIM. By comparison, note that the false-positive rate for the random-indicator algorithm is the worst but only marginally poorer than CARD.

Additional insights on the issue of false positive rates can be gained by examining the duration of the congestion-phases that are false-positives. Figure 9 shows for the UNC-Campus trace, the distribution of all congestion-phases that end in either a no-congestion phase or the end of the connection without seeing a packet loss. The durations are expressed normalized to the current exponentially-weighted average of the RTT for the connection. For all algorithms, the duration of almost 80% of false-positive intervals last less than 1 RTT and over 50% last less than 0.10 RTT. This illustrates that the aggressive tendency to indicate congestion is often rescinded by a transition back to a no-congestion indication in substantially less than one RTT. Put another way, the algorithms often indicate congestion based on sampling the RTT from one segment in a flight only to change the

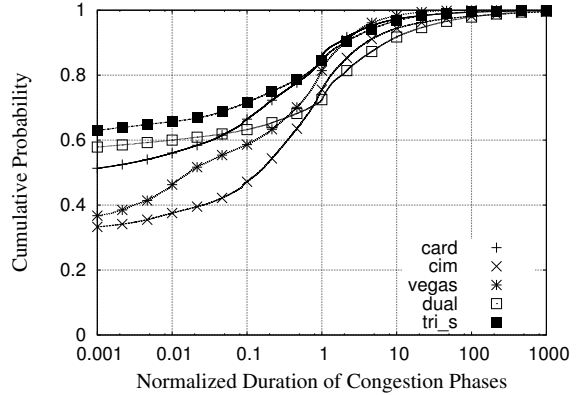


Figure 9: UNC-Campus: Duration of False Positives Phases

indication to no congestion based on the RTT from another segment in the same flight. This suggests that the algorithms could perhaps achieve a lower false-positive rate by introducing a form of hysteresis using a delay of 0.5 to 1 RTT before a transition to indicating congestion. The effects of introducing such a delay on the LPA and on the timeliness of the indicator will require further investigation.

### 4.3 False Negatives

We also evaluated the false-negative rates for the algorithms. A false-negative indication occurs when a no-congestion phase is terminated by a loss-episode. In other words, the algorithm fails to indicate congestion when, in fact, congestion is present as indicated by the subsequent loss. The false-negative rate is important because a low false-negative rate indicates that the algorithm is efficient in providing a correct signal. Figure 10 shows the cumulative distribution over all connections of the fraction of false-negative indications in the UNC-Campus trace for the five algorithms and the random estimator. CARD achieves the lowest false-negative rate (80% of connections had a false negative rate of 20% or less) but note that CARD also had the worst false-positive rate. Thus, its effectiveness is achieved at the potential cost of signaling congestion too often to a congestion control mechanism. Even though CARD had a low false-negative rate, it was no better than the random indicator. As expected, Vegas

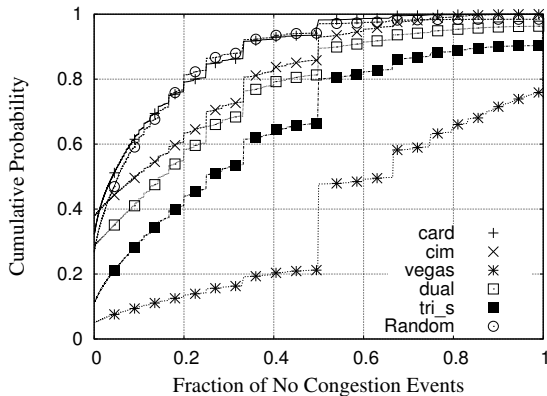


Figure 10: UNC: False Negative Rate

had the poorest (highest) false-negative rate with 50% or more false-negatives in 80% of connections.

High false-negative rates might be associated with conditions in which there is little opportunity for the algorithm to receive data and make a determination before congestion becomes severe enough to cause packet loss. One measure of this opportunity is the duration of the false-negative phases. Figure 11 plots the distribution of the durations of all no-congestion phases that are terminated by a loss event. The durations are expressed normalized to the current exponentially-weighted average of the RTT for the connection. For all algorithms the duration of more than 80% of false-negative intervals last more than 1 RTT and over 50% last more than 3-5 RTT. This indicates that high false-negative rates are *not* associated with short durations of observation intervals.

#### 4.4 Overlap of Connections

The results presented above indicate that for some connections, algorithms like CIM and CARD had excellent performance (80% or more LPA) while for other connections their performance was poor (20% or less LPA). Similarly, while the overall performance of Vegas was poor, there were some connections for which it yielded 100% LPA. False-positive rates (and to some extent false-negative rates) also tended to indicate that some connections led to poor algorithm performance while others had excellent performance. We next examine more closely

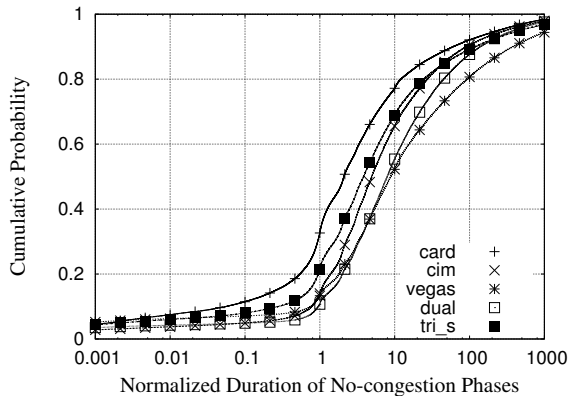


Figure 11: UNC-Campus: Duration of False Negative Phases

the relationship between characteristics of individual TCP connections (including those of the network path they traverse) and the ability of DBCEs to identify congestion and no-congestion conditions.

First, we investigate whether the subset of connections for which one algorithm has excellent (or poor) performance is the same subset for which another algorithm also has excellent (or poor) performance—refer Section 3.4 for the definition of “excellent” and “poor”. To highlight the influence of different sets of connections on excellent or poor performance, we consider only CIM and CARD (which have excellent performance for some connections and poor for others) and Vegas (that generally has poor performance for most connections). The results are expressed by Venn diagrams in Figure 12, for the UNC-Campus trace. The numbers shown in each region of a Venn diagram are the percentages of all connections in the UNC-Campus trace that appear in the subset represented by the region.

**Overlap Across DBCEs:** Figure 12(a) shows that CARD had excellent LPA for 33.7% (13.6 + 17.3 + 1.7 + 1.1) of connections while CIM had excellent LPA for 39.1% (18.8 + 17.3 + 1.7 + 1.3). Only about 19% (17.3 + 1.7) of the connections in the trace had excellent LPAs for both CARD and CIM, while similar fractions produced excellent performance only for CARD (13.6%) or only for CIM (18.8%). Thus it is clear that each algorithm is also influenced by some connection characteristics that the other is not. Interestingly, the small subsets of connections for

which Vegas has excellent LPA are common to CARD or CIM except for a small subset (1.8%) for which only Vegas had excellent performance. Figure 12(b) shows the same type of analysis for those connections with poor LPA. In this case there is an even smaller subset of connections that give poor performance for both CARD and CIM and larger subsets that influence only one or the other (especially CIM, where 22% are not common with CARD). Since Vegas had poor performance for such a large set of connections it is not surprising that there is significant overlap with both CARD and CIM.

A somewhat different result is shown in Figure 12(c) for the UNC-Campus trace for poor performance on the metric of false positives. Most of the connections that produce poor performance on this metric for CIM also produce poor performance for CARD but the converse is not true, about 23% of all connections produce a poor false positive rate on CARD but do not for CIM while only about 4% of connections are poor for CIM but not for CARD. Clearly there is a significant subset of connections with characteristics that cause a large false-positive rate for CARD that do not for CIM. A similar (but not as strong) result for excellent performance on false-negative rates is shown in Figure 12(d). While a large subset of connections (over 47%) give excellent false-negative rates for both CARD and CIM, over twice as many (27.1% vs 12.7%) are good for only CARD than are good for only CIM on this metric.

**Overlap Across Metrics:** Another way to examine the influence of connection characteristics is to study connections that yield excellent (or poor) performance for all three metrics for a given algorithm. Consider the results shown in Figure 12(e) for the UNC-Campus connections with excellent performance on each of the metrics by the CARD algorithm. Strong overlap (33.5% of all connections) is found only between connections with low false-negative rates and high LPA. An even larger subset of connections (42.5%), however, had a low false-negative rate but did not also yield a high LPA. This shows that for this large fraction of connections, when CARD indicated no congestion there was usually no loss experienced (low false-negative rate), but for those same connections a high fraction of the losses were not preceded by a congestion indication (low success rate). For these connections, thus, CARD was able to reliably indicate the *absence* of congestion but *not its presence*.

The connections for which CARD had poor performance on the three metrics are shown in the Venn diagram of Figure 12(f). Here the 27.3% of all connections with a high rate of false positives have an almost complete overlap with those having a low LPA. These connections may represent the worst-case set of connection characteristics for the algorithm; the algorithm is usually wrong when it indicates congestion (no loss occurs) and usually wrong when it indicates no congestion (a loss occurs). Similar observations about CIM (but not as strong differences) were found for both excellent and poor performance on these metrics but are not shown to save space.

Since Vegas had uniformly poor performance we show only those results for all three metrics in Figure 12(g). The only strong overlap is for those connections with high false negative rates and a low LPA. This shows that the Vegas algorithm is usually wrong when it indicates no congestion (a loss occurs) for a large fraction of connections.

## 5 Influence of Connection Characteristics

The Venn diagrams of Figure 12 highlight two facts: (i) there are certain connection characteristics which allow all DBCEs to work well (or poorly); while (ii) there are other characteristics which allow only one of the estimators to perform well (or poorly). In this section, we study the influence of characteristics of significance for the three estimators—CIM, CARD, and Vegas. We first identify the connection characteristics which are of interest for each estimator—we do this by selecting characteristics that have significantly different distributions for the “excellent” and “poor” subsets of connections for each estimator-metric pair. We then run multivariate linear regression to identify which of the selected characteristics affect a given estimator’s performance the most. Since most of the estimators perform well for the false negatives metric, we perform the regression analysis only on the false positives and the LPA metric.

### 5.1 Connection Characteristics of Interest

A DBCE uses RTT samples to predict congestion. If network congestion evolves slowly and the estimator is able

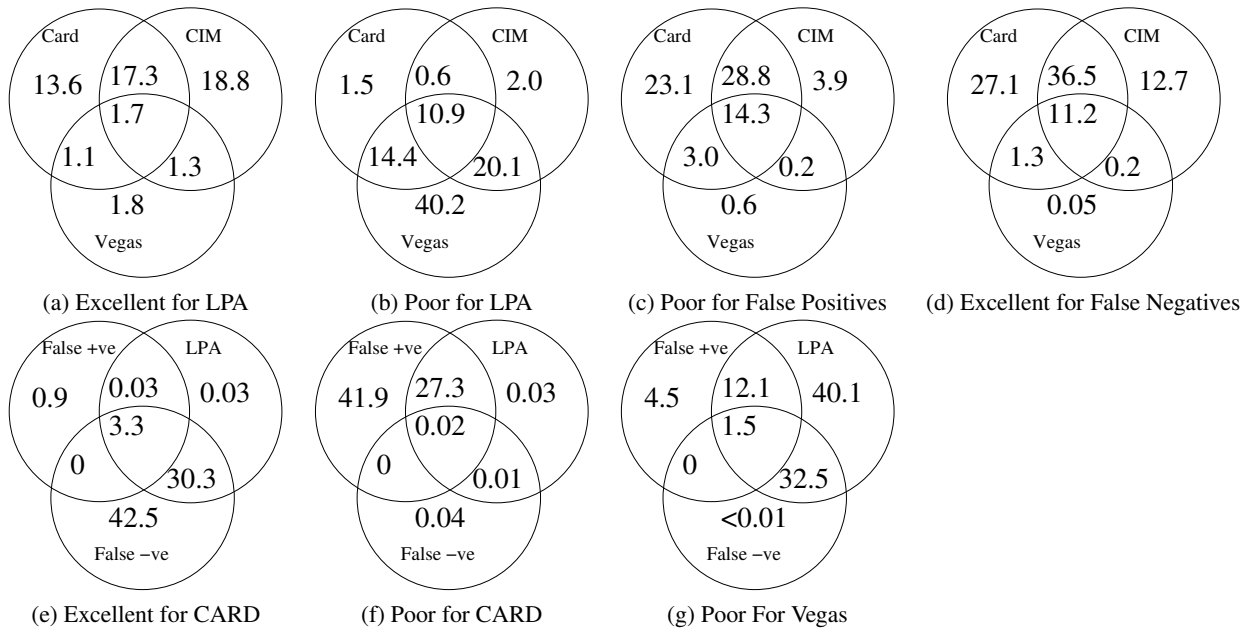


Figure 12: Venn Diagrams

to collect enough RTT samples during this evolution, it should be able to better predict congestion. On the other hand, an estimator that does not get enough RTT samples before a packet loss it may not be able to predict the loss. We capture this effect by considering the *frequency of sampling RTTs*, averaged over up to 3 round-trip times before a loss.

Most estimators use the *relative* increase in RTTs as a congestion-predictor. On high-bandwidth paths with long propagation delays, even the maximum queuing delay may be significantly smaller than the minimum path RTT—this limits the magnitude of the relative increase in RTT. We capture this effect by considering the *ratio of the 75<sup>th</sup> percentile of connection RTT to the minimum RTT*.

The pattern of packet losses in a connection affects a DBCE in several ways. First, for a connection with a high loss rate (high frequency of loss episodes), an aggressive DBCE that often predicts congestion would perform well on the LPA and false positives metrics; while a conservative estimator would perform poorly on LPA. At low loss-rates, on the other hand, the performance of these estimators would reverse. Hence, we include the *packet loss rate* of a connection as a characteristic of interest. Also,

if the gap between loss-episodes is small, a DBCE may not obtain enough RTT samples to reliably infer congestion. We capture this effect by considering the *average loss-per-episode* and the *average interval* between loss-episodes for each connection.

For self-congested connections (that contribute significantly to the total load on the congested network link), an increase in the number of packets in flight would result in queue-buildup, which in turn will lead to an increase in observed RTTs. We use the *correlation between the number of packets in flight and the RTT* to identify such connections. We also consider the *size* and *duration* of each connection.

**Characteristic Selection:** If the distribution of a particular characteristic does not differ significantly for the excellent and poor connections corresponding to a given estimator-metric pair, then it implies that the estimator’s performance does not depend on this particular characteristic. On the other hand, if the distributions differ, the characteristic *may* influence the performance of the estimator. Before we use regression to identify the most influential characteristics for an estimator-metric pair, we filter out the characteristics which clearly have no influence on

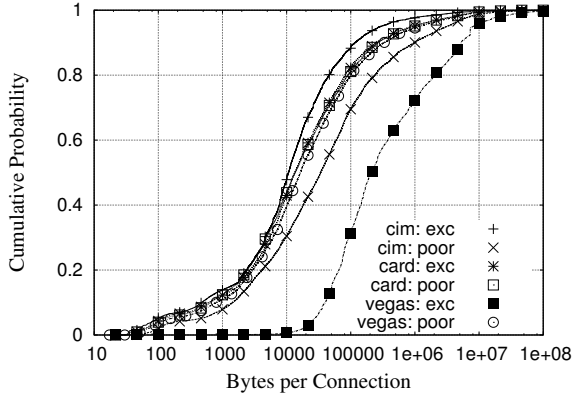


Figure 13: Connection-size Distribution: LPA metric

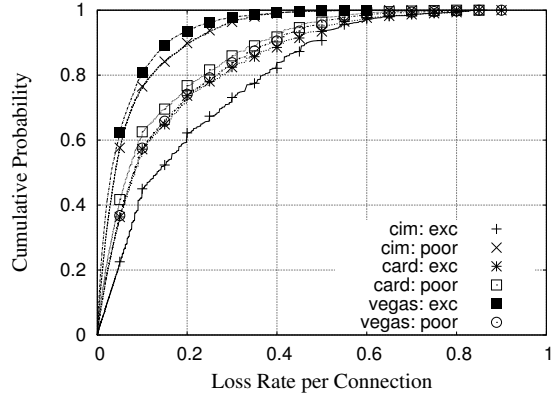


Figure 14: Loss-rate Distribution: LPA metric

the estimator using the above criteria. We illustrate this strategy below.

Figure 13 plots the distribution of connection size for the excellent and poor connections for the LPA metric, for CIM, CARD, and Vegas. The distributions differ significantly for CIM and Vegas, while they are similar for CARD. Hence we include *connection-size* for regression analysis of CIM and Vegas, but eliminate it for CARD. Figure 14 plots the loss rate distribution for excellent and poor connections for the LPA metric, for CIM, CARD, and Vegas. The distributions differ for all three estimators (though by different amounts). Hence this characteristic is selected for all three estimators for the LPA metric. Table 4 lists characteristics selected in the above manner, by studying the respective CDFs for all estimators, for the LPA and false-positive (FP) metrics.

## 5.2 Regression Methodology

Multiple-regression helps understand the relationship between several independent (or predictor) variables and a dependent variable [10]. We use *multiple linear regression* to identify the relationship between each estimator’s performance metric and the connection characteristics selected in Table 4. We use the Matlab [12] functions “regress” and “regstat” to solve the equation  $Y = X\beta + \epsilon$ , where  $Y$ , the dependent variable, is the estimator’s performance metric (either LPA or FP), and  $X$ , the independent variables, represent the selected connection characteris-

Characteristic	CARD		CIM		Vegas	
	LPA	FP	LPA	FP	LPA	FP
Sampling Frequency	✓	✓	✓	✓	✓	✓
RTT variability	✓	✓	×	✓	×	×
Loss Rate	✓	✓	✓	✓	✓	✓
Size	×	✓	✓	✓	✓	✓
Duration	×	×	✓	×	✓	×
Correlation	×	×	×	×	×	×
Loss Distance	×	×	×	×	✓	×
Loss per Episode	×	✓	×	✓	✓	×

Table 4: Selection of Characteristics for All Estimators

tics.  $\beta$  are the coefficients of regression.

Both LPA and FP metrics vary from 0 to 1 and are directly used as  $Y$  in the above equation. Each of the connection characteristics, however, span a diverse range of values; further, their distribution across connections can be quite skewed (see Figure 13 for example). In order to make the distribution of  $X$ ’s uniform and similar in range, we first take a log of all the characteristics,  $C_i$ , and then normalize the logs to the range  $[0, 1]$  by using the following equation:

$$X_i = \frac{\log(C_i) - \min\{\log(C_i)\}}{\max\{\log(C_i)\} - \min\{\log(C_i)\}}$$

All  $X_i$ s vary from 0 to 1; hence, after regression analysis, the coefficients  $\beta_i$  of the various characteristics can

be directly compared to study the relative impact of the corresponding characteristic on the dependent variable.

It is important to note that the above method assumes that it is reasonable to fit a linear model to the logarithms of the various characteristics in order to assess their influence on the estimator metrics. While this assumption may not be well-informed, the above model does accomplish our limited goal here—that of assessing the *relative* impact of each characteristic.

### 5.3 Regression Results

Tables 5 and 6 list the regression coefficients for the LPA and false-positives metrics, respectively. The tables also list the 95<sup>th</sup> percentile confidence interval (CI) for all coefficients. We see that in all cases the CI passes the zero test. We also tested the significance-levels by verifying that the *Pvalue* reported for all significant coefficients was less than or equal to 0.001.

We first consider the LPA regression analysis for the three estimators. We find that none of the connection characteristics considered have a significant positive influence on CARD performance. However, both RTT-variability and sampling-frequency seem to have a slightly negative effect on the performance (recall that LPA is an HB metric; hence a negative coefficient indicates that with increase in this characteristics the performance degrades). For CIM performance, both sampling-frequency and connection-duration have a strong negative effect. This is counter-intuitive, since a higher RTT-sampling frequency just before loss should let an estimator predict losses more efficiently. This result is further illustrated in Figure 15 that plots the distribution of average sampling-frequency for the excellent and poor connections with respect to CIM’s LPA performance. We find that the distribution of sampling frequency for the excellent connection is almost an order of magnitude lower than that for the poor connections. The connection loss-rate has a very strong positive influence on CIM performance, which is also illustrated in figure 14. For Vegas performance, we find that the loss-per-episode has a strong positive influence. The connection-size has a mild positive influence while the loss-rate has slight negative influence. Figure 14 also illustrates that the influence of loss rate on Vegas is exactly opposite to that on CARD.

We next consider the false-positive regression results

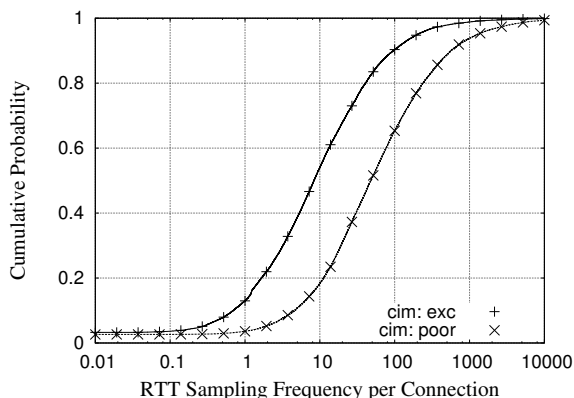


Figure 15: Sampling Frequency Distribution for CIM: LPA metric

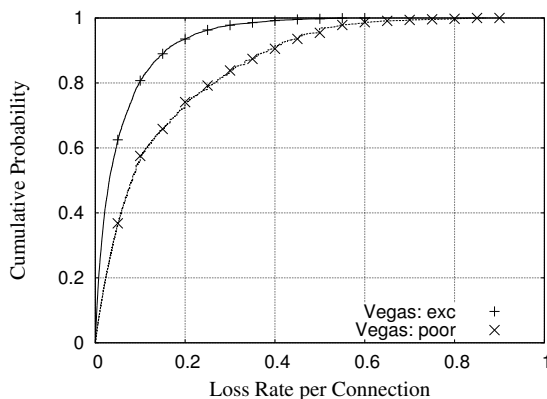


Figure 16: Loss Rate Distribution for Vegas: FP metric

for the three estimators. The false positive is an LB metric—a negative coefficient implies a positive influence of a characteristic on the estimator performance. For CARD, the loss-rate has a very strong positive influence (negative coefficient), indicating that with increase in loss rate the performance of this estimator-metric pair improves. For CIM, again the loss-rate has a very strong positive influence while the sampling-frequency and RTT-variability has a mild negative influence. Connection-size exhibits a very strong negative influence on Vegas’s false-positive rate.

It is natural to question the utility of using regression for the above analysis—it can be argued that comparison



Characteristics	CARD			CIM			Vegas					
	$\beta$	CI	Pvalue	$\beta$	CI	Pvalue	$\beta$	CI	Pvalue			
Constant	0.64	0.62	0.66	<0.001	1.09	1.06	1.11	<0.001	-0.07	-0.08	-0.06	<0.001
Sampling Frequency	-0.24	-0.28	-0.2	<0.001	-1.0	-1.03	-0.97	<0.001	-0.05	-0.07	-0.02	<0.001
RTT variability	-0.31	-0.37	-0.24	<0.001	-	-	-	-	-	-	-	-
Loss Rate	0.08	0.05	0.11	<0.001	0.7	0.67	0.72	<0.001	-0.27	-0.28	-0.26	<0.001
Size	-	-	-	-	-0.22	-0.26	-0.19	<0.001	0.29	0.27	0.31	<0.001
Duration	-	-	-	-	-0.62	-0.64	-0.61	<0.001	-0.07	-0.1	-0.08	<0.001
Loss Distance	-	-	-	-	-	-	-	-	-0.15	-0.17	-0.13	<0.001
Loss per Episode	-	-	-	-	-	-	-	-	0.9	0.88	0.92	<0.001

Table 5: Regression coefficients for LPA

Characteristics	CARD			CIM			Vegas					
	$\beta$	CI	Pvalue	$\beta$	CI	Pvalue	$\beta$	CI	Pvalue			
Constant	0.99	0.98	1.0	<0.001	0.83	0.82	0.85	<0.001	-0.53	-0.54	-0.52	<0.001
Sampling Frequency	0.14	0.12	0.16	<0.001	0.38	0.36	0.4	<0.001	0.03	0.006	0.05	0.012
RTT variability	0.24	0.22	0.26	<0.001	0.37	0.33	0.41	<0.001	-	-	-	-
Loss Rate	-0.69	-0.7	-0.68	<0.001	-1.34	-1.36	-1.32	<0.001	-0.17	-0.19	-0.16	<0.001
Size	-0.16	-0.18	-0.14	<0.001	-0.13	-0.15	-0.1	<0.001	1.5	1.47	1.52	<0.001
Loss per Episode	0.15	0.13	0.17	<0.001	0.3	0.27	0.34	<0.001	-	-	-	-

Table 6: Regression coefficients for False Positives

of the excellent and poor distributions for each characteristic (as is done in Figures 13 and 14) should suffice in identifying the strongly-influencing characteristics. Figure 16 illustrates that this is *not* the case. It plots the loss-rate distribution for the excellent and poor connections corresponding to the Vegas false-positives performance. We observe that the distributions differ significantly; however, regression indicates that this particular characteristic has a very small influence on the Vegas false-positives rate. This confirms that simple inspection of the distributions is not sufficient and regression is needed for identifying influential connection characteristics.

## 6 Concluding Remarks

We have developed a state-machine-based approach to accurately estimate the RTTs and losses for TCP con-

nections. Using this tool, we conducted a large scale study with a diverse set of traces where we extracted the RTTs seen by each TCP connection and used them to evaluate five prominent delay-based congestion estimators (DBCEs). We tested each estimator’s performance in terms of (i) the loss prediction ability (LPA), (ii) fraction of erroneous congestion prediction (false positives), and (ii) fraction of erroneous no-congestion indication (false negatives). Finally, we used multivariate linear regression to identify the characteristics of the connections that effect the performance of these estimators.

For LPA, CIM was the best but it is 100% accurate for at most 40% of connections. Further, it is 0% accurate for over 20% of connections. CARD had the best performance for false negatives with 80% of connections having less than 20% false negatives. Both CIM and CARD, however had poor rates of false positives which has significant performance implications. Vegas had the clearly

worst performance for both LPA and false negatives. We found that a purely random congestion predictor’s performance was comparable to the best predictor for LPA and false negatives.

Overall, we find that the connection characteristics for which the DBCEs have excellent (or poor) performance are highly diverse and often have opposing effects for different estimators or metrics. For example, the regression analysis surprisingly shows that for CIM (one of the best estimators) the LPA decreases with increasing sampling frequency. We also note that the effect of loss rate on CARD and Vegas is opposite of each other. While LPA for CARD improves with loss rate, LPA of Vegas decreases with loss rate. Our high-level conclusion is that the state-of-the-art in delay-based congestion estimation is, at best, rudimentary and that designing a DBCE algorithm that will work with the diversity of connection characteristics found “in the wild” is likely to remain an open research problem for some time to come.

## References

- [1] The DAG Project, Univ. of Waikato, <http://dag.cs.waikato.ac.nz/>.
- [2] <http://pma.nlanr.net/Special/leip1.html>.
- [3] <http://pma.nlanr.net/Traces/long/ipls1.html>.
- [4] <http://tracer.csl.sony.co.jp/mawi/>.
- [5] J. Aikat, J. Kaur, D. Smith, and K. Jeffay. Variability in TCP Round-trip Times. In *Proceedings of the ACM SIGCOMM Internet Measurement Conference*, October 2003.
- [6] Saad Biaz and Nitin H. Vaidya. Performance of TCP congestion predictors as loss predictors. Technical report, College Station, TX, USA, 1998.
- [7] Saad Biaz and Nitin H. Vaidya. Sender-Based Heuristics for Distinguishing Congestion Losses from Wireless Transmission Losses. Technical report, College Station, TX, USA, 1998.
- [8] Saad Biaz and Nitin H. Vaidya. Is the round-trip time correlated with the number of packets in flight? In *IMC '03: Proceedings of the 3rd ACM SIGCOMM conference on Internet measurement*, pages 273–278, New York, NY, USA, 2003. ACM Press.
- [9] Lawrence S. Brakmo, Sean W. O’Malley, and Larry L. Peterson. TCP Vegas: new techniques for congestion detection and avoidance. *SIGCOMM Comput. Commun. Rev.*, 24(4):24–35, October 1994.
- [10] Samprit Chatterjee, Ali S. Hadi, and Bertram Price. *Regression Analysis by Example, 3rd Edition*. Wiley, November 1999.
- [11] S. Floyd, T. Henderson, and A. Gurtov. The NewReno Modification to TCP’s Fast Recovery Algorithm, 2004.
- [12] The Math Works Inc. *MATLAB, High-performance Numeric Computation and Visualization Software. User’s Guide*. 1992.
- [13] R. Jain. A delay-based approach for congestion avoidance in interconnected heterogeneous computer networks. *SIGCOMM Comput. Commun. Rev.*, 19(5):56–71, October 1989.
- [14] R. Jain. *The Art of Computer Systems Performance Analysis: Techniques for Experimental Design, Measurement, Simulation, and Modeling*. Wiley- Interscience, New York, NY, April 1991.
- [15] Cheng Jin, David X. Wei, and Steven H. Low. FAST TCP: motivation, architecture, algorithms, performance. In *IEEE Infocom*, March, 2004.
- [16] Cheng Jin, David X. Wei, and Steven H. Low. The case for delay based congestion control. In *IEEE computer communication workshop*, October, 2003.
- [17] P. Karn and C. Patridge. Improving Round-Trip Time Estimates in Reliable Transport Protocols. *ACM SIGCOMM Computer Communication Review*, 17(5):2–7, Oct-Nov 1987.
- [18] Jim Martin, Arne Nilsson, and Injong Rhee. Delay-based congestion avoidance for TCP. *IEEE/ACM Trans. Netw.*, 11(3):356–369, June 2003.
- [19] M. Mathis, J. Mahdavi, S. Floyd, and A. Romanow. TCP Selective Acknowledgement Options, 1996.
- [20] J. Padhye and S. Floyd. On Inferring TCP Behavior. In *Proceedings of ACM SIGCOMM*, 2001.
- [21] Ravi S. Prasad, Manish Jain, and Constantinos Dovrolis. On the Effectiveness of Delay-Based Congestion Avoidance. In *PFLDnet*, 2004.
- [22] S. Rewaskar, J. Kaur, and D. Smith. Passive Inference of TCP Losses Using a State-Machine Based Approach. *Technical Report, Department of Computer Science, University of North Carolina at Chapel Hill*, October 2005.
- [23] Zheng Wang and Jon Crowcroft. A new congestion control scheme: slow start and search (Tri-S). *SIGCOMM Comput. Commun. Rev.*, 21(1):32–43, January 1991.
- [24] Zheng Wang and Jon Crowcroft. Eliminating periodic packet losses in the 4.3-Tahoe BSD TCP congestion control algorithm. *SIGCOMM Comput. Commun. Rev.*, 22(2):9–16, April 1992.