# Delay-Based Congestion Avoidance for TCP

Jim Martin, *Member, IEEE*, Arne Nilsson, *Member, IEEE*, and Injong Rhee, *Senior Member, IEEE*

***Abstract*—The set of TCP congestion control algorithms associated with TCP/Reno (e.g., slow-start and congestion avoidance) have been crucial to ensuring the stability of the Internet. Algorithms such as TCP/NewReno (which has been deployed) and TCP/Vegas (which has not been deployed) represent incrementally deployable enhancements to TCP as they have been shown to improve a TCP connection's throughput without degrading performance to competing flows. Our research focuses on delay-based congestion avoidance algorithms (DCA), like TCP/Vegas, which attempt to utilize the congestion information contained in packet round-trip time (RTT) samples. Through measurement and simulation, we show evidence suggesting that a single deployment of DCA (i.e., a TCP connection enhanced with a DCA algorithm) is not a viable enhancement to TCP over high-speed paths. We define several performance metrics that quantify the level of correlation between packet loss and RTT. Based on our measurement analysis we find that although there is useful congestion information contained within RTT samples, the level of correlation between an increase in RTT and packet loss is not strong enough to allow a TCP/Sender to reliably improve throughput. While DCA is able to reduce the packet loss rate experienced by a connection, in its attempts to avoid packet loss, the algorithm will react unnecessarily to RTT variation that is not associated with packet loss. The result is degraded throughput as compared to a similar flow that does not support DCA.**

***Index Terms*—TCP congestion control, delay-based congestion avoidance (DCA), TCP/Vegas, loss and round-trip time (RTT) correlation patterns.**

## I. Introduction

THE foundation of TCP's congestion control is the principle of conservation of packets [1]. New packets are admitted into the network as packets are confirmed to be removed from the network via the arrival of acknowledgments (ACKs). Other aspects of TCP's congestion control include the slow-start and the congestion avoidance algorithms which utilize packet loss as an indicator of network congestion [2], [3]. As TCP was designed for a best-effort packet switched network, even moderate levels of packet loss are acceptable. However, paths over the Internet can experience very high packet loss rates, well beyond the optimal operating range of TCP. Improvements such as TCP/NewReno and TCP/SACK have been deployed to enhance the efficiency of TCP's loss recovery [4], [5]. The limited transmit enhancement is a further proposed incremental enhancement for TCP [6]. While these algorithms can improve TCP throughput by reducing the frequency of TCP timeouts, they still rely on packet loss as an implicit congestion signal.

An alternative congestion control technique for TCP, one which is preventive rather than reactive, is end-to-end delay-based congestion avoidance algorithms (DCA). Originally described by Jain [7], DCA is best represented by TCP/Vegas and Dual [8], [9]. DCA algorithms monitor packet round-trip times (RTTs) and react to increases in RTT in an attempt to avoid network congestion before it becomes significant. Previous studies of TCP/Vegas have shown that the algorithm increases TCP throughput by reducing the frequency of packet loss and timeouts. Furthermore, the improvement does not come at the expense of competing TCP flows. Because Vegas provides a benefit with a single deployment and because the algorithm operates at the TCP/Sender, one can argue that Vegas is a viable incrementally deployable improvement to TCP.

Previous studies of DCA have concentrated either on low-speed networks or on networks where DCA flows consume a significant percentage of the total traffic [10]–[12]. In this paper, we focus on the performance of DCA algorithms in a high-speed network where DCA flows constitute only a fraction of the total traffic. Further, our goal is to utilize a methodology such that the conclusions we derive can be generalized to any DCA algorithm.

Any change proposed for a mature and widely deployed protocol such as TCP will be met with much resistance. Alternative congestion control algorithms must be TCP-compatible, which means that they must result in the same throughput as achieved by a similarly situated host (i.e., over the same path with the same TCP parameters) [13]. An incremental enhancement to TCP that meets the following requirements will have the best chance for deployment.

- It must improve the throughput of the TCP connection that employs the enhancement.
- It must not reduce the performance of other competing TCP flows on the same path where the "enhanced" TCP flow travels. The objective is to make better use of available bandwidth without penalizing other TCP flows.
- Ideally, it requires changes only to a TCP sender.

Furthermore, the above properties must hold regardless of the number of "enhanced" TCP flows on the same end-to-end path. This implies that the properties must hold even if there is only one "enhanced" flow in the path. In today's Internet, to support a wide deployment of TCP enhancements, there must exist sufficient economic incentives for adopting them. When these incentives are weak or even if they result in temporary sacrifice of resources for users at least for a while before its wider deployment, the eventual wide deployment of such protocols

Manuscript received February 6, 2000; revised October 15, 2002; approved by IEEE/ACM TRANSACTIONS ON NETWORKING Editor C. Diot.

J. Martin is with the Department of Computer Science, Clemson University, Clemson, SC 29634-0974 USA (e-mail: jim.martin@cs.clemson.edu).

A. A. Nilsson is with the Department of Electrical and Computer Engineering, North Carolina State University, Raleigh, NC 27695-7914 USA (e-mail: nilsson@eos.ncsu.edu).

I. Rhee is with the Department of Computer Science, North Carolina State University, Raleigh, NC 27695-7534 USA (e-mail: rhee@eos.ncsu.edu).

Digital Object Identifier 10.1109/TNET.2003.813038

becomes unlikely. Therefore, our research assesses the benefits associated with the incremental deployment of DCA where the "enhanced" flows constitute only a fraction of the total traffic in the bottleneck link.

The following describes the attributes of DCA.

- DCA augments the TCP/Reno protocol.
- DCA monitors TCP packet RTTs, allowing the algorithm to reside entirely within the TCP sender.
- RTT variations are assumed to be caused by changes in queueing delays experienced by packets being timed.
- Based on RTT variations, DCA makes congestion decisions that reduce the transmission rate by some percentage by adjusting the TCP congestion window (cwnd).

We limit the scope of our study of DCA to Internet environments where the lowest link capacity along the path is 10 Mb/s. Measurement studies have shown that Internet backbone switches, many of which now support multigigabit link speeds, are subject to tens or even hundreds of thousands of low-bandwidth ON/OFF TCP flows [14]–[16]. One recent measurement study of a tier one provider's backbone network found that in the worst case, there are on the order of 30 000 flows per 100 Mb/s of traffic [17]. Thus, we study the performance of DCA under realistic Internet environments where thousands of TCP flows may come and go over short time periods.

In this paper, we present evidence suggesting that RTT-based congestion avoidance may not be reliably incrementally deployed over high-speed Internet paths. Based on a measurement study conducted in 1999 over seven high-speed Internet paths, we find that congestion information contained in TCP RTT samples cannot be reliably used to predict packet loss.[1] The success of DCA highly depends on a strong correlation between packet loss events and increases in RTT prior to the loss. By tracing TCP connections along each path, we are able to extract a time series of packet RTTs along with an indication of packet loss events. Depending on the path, we find that only 7%–18% (on average) of observed loss events were preceded by a significant increase in RTT. With the data, we are able to evaluate the ability of a DCA algorithm to "predict" a loss event based on an observed increase in packet RTT. Even if we assume that every loss predicted by DCA is avoided, our analysis indicates that noise in RTT samples leads to degraded throughput by guiding TCP to reduce at wrong times (when there is no loss). We modify an analytic TCP throughput model given by Padhye *et al.* [18] to assess the impact of DCA's congestion reactions (both right and wrong ones) [19]. Based on measured data, the throughput model predicts that DCA would degrade TCP throughput in the range of 7%–58%.

We conjecture that the level of correlation between packet loss and TCP RTT samples is weak due to the following reasons.

1) A TCP constrained RTT congestion probe is too coarse to accurately track the bursty congestion associated with packet loss over high-speed paths.
2) A DCA algorithm cannot reliably assess the congestion level at the router. Short-term queue fluctuations which are not associated with loss at a router, persistent queueing, and congestion at multiple bottlenecks (especially when the congested routers are provisioned differently) make it difficult for a DCA algorithm to make accurate congestion decisions.

To verify this conjecture, we must be able to observe the queue levels at the bottleneck links over the path. We resort to simulation using the ns simulation tool [20]. We construct two ns models that emulate the end-to-end traffic characteristics of two of the seven Internet paths being measured. These models closely match the loss behaviors and burstiness of RTT variations of the two paths. Under these models, we confirm the result we found from the Internet measurement data: a DCA flow suffers from throughput degradation.

This paper is organized as follows. First, we overview related work in Section II. Then we present the measurement analysis and throughput analysis followed by the simulation analysis in Sections III and IV. We end the paper with conclusions and a discussion of future work in Sections V and VI.

## II. Related Work

Jain first coined the term *delay-based congestion avoidance* in [7]. While Jain admits that his proposed algorithm was not sufficient for a practical network, his work provided the foundation for future research of DCA. The work described in [8] and [21] shows that TCP/Vegas can improve TCP throughput over the Internet by avoiding packet loss. However, these studies were based on Internet paths that existed in the early 1990s which generally involved at least one T1 speed link and consequently allows any given flow to consume a significant fraction of available bandwidth. These studies also did not isolate the impact of the congestion avoidance algorithm (i.e., CAM) from the non-DCA aspects of Vegas (i.e., the loss recovery enhancement). More recent studies, however, have recognized that Vegas has several very different algorithms and that the congestion avoidance algorithm must be studied independently [11], [10]. The work in [10] correctly points out that the benefit associated with the original Vegas algorithm is in fact due to the enhanced loss recovery algorithms rather than the Vegas DCA algorithm. However, [11] also points out that the Vegas enhanced recovery algorithm was designed to be more aggressive than Reno and that the congestion avoidance algorithm was designed to compensate to reduce loss (by being less aggressive in the sending rate increase) to provide a balance. While the work of [11] is an interesting analysis of the Vegas DCA algorithm from a "global network" perspective, it is very different from our "incremental enhancement" analysis of DCA.

A key aspect of our research focuses on assessing the ability of a TCP constrained RTT-based congestion sampling algorithm in predicting future packet loss events. Several previous studies are relevant [22]–[25]. End-to-end packet delay and loss behavior over the Internet was studied in [22] and [24]. A common result was that packet loss events were observed to be correlated over timescales of up to 200 ms. The work in [25] found loss correlation up to timescales of 1 s. However, the authors did not focus on the correlation between an increase in RTT with packet loss. Moon *et al.* did look at the correlation that exists between a loss conditioned delay (i.e., a packet delay immediately preceding a loss event) and packet loss. Their motivations were similar to ours in that they wanted

[1]Even though the measurement data was collected in 1999, we believe that the observed dynamics are still representative of current public Internet behavior.

to see if an endpoint could predict loss. They found a higher level of correlation than we did. There are several differences. First, the method used by [23] utilized a finer-grained one-way delay-based UDP-based probe technique as opposed to a TCP constrained probe process. Second, and more significantly, we were interested in the "usable" level of correlation. In other words, we measured the relative increase in RTT that occurs prior to the transmission of the segment that is eventually dropped. If we were to consider the increase in RTT associated with segments that surround the segment that gets dropped, we would see a moderate increase in the observed correlation. However, this implicit congestion signal would not arrive in time to prevent the TCP sender from transmitting the soon-to-be dropped segment. Paxson [26] also looked at the correlation between one-way packet delay variation and loss. He concluded that loss is weakly correlated to rises in packet delay and conjectured that the linkage between the two is weakened by routers with large buffer space and because the end-to-end delay reflects the accruing of a number of smaller variations into a single, considerably larger, variation. Our measurement results support this conjecture, but we believe that another factor is that the delay variation associated with loss is bursty (in the milliseconds to several hundreds of milliseconds range) which makes it difficult to accurately assess the fine-grained correlation that exists between latency and loss events.

More recent work based on measurements from the NIMI infrastructure focuses on assessing the "constancy" of loss and delay over a variety of paths [27]. The authors extend the findings of [22] and [25] by observing that much of the correlation in the loss process comes from back-to-back loss episodes as opposed to a series of nearby losses. The authors conclude that congestion epochs associated with loss are actually spikes with of timescales of roughly 200 ms or shorter. A recent measurement study of low-speed streaming flows over the Internet led the authors to conjecture that the loss process associated with congested Internet routers might be less than 3 ms [28]. This was based on a relatively large number of observed single loss events (as opposed to bursts) for traffic with a peak burst rate metered by a T1 speed link. Our data shows that the congestion epochs are complex processes that incorporate timescales spanning very long (hours), medium (minutes), and very brief (milliseconds) amounts of time. Further, we saw that more than one congestion process might be active at any given time. This behavior is nicely described in [29], postulating that traffic arrival processes have "spikes (that) ride on ripples that ride on still longer term swells. . .." Based on our measurements, we observed that the magnitude of congestion epochs varies tremendously and, further, that loss is almost equally as likely during any level of congestion epoch. This can be explained if we assume that over the observation period (one week), a path experienced congestion at more than one location and, further, if we assume that a path might periodically experience multiple bottlenecks at a given time. However, if we examine specific loss events, we find that the RTT increase surrounding loss is usually short lived, which confirms the findings of [27] which state that "loss processes are better thought of as spikes during which there is a short-term outage, rather than epochs over which congested router's buffer remains perilously full." This result is in fact fundamental to our analysis and subsequent conclusion that DCA is not able to improve TCP performance and is therefore not incrementally deployable.

In addition to TCP congestion control issues, the Internet research community has also been vigorously exploring the development and deployment of TCP-friendly algorithms for real-time streaming applications. Such algorithms can be TCP equivalent if they use additive-increase/multiplicative-decrease (AIMD) behavior with the same parameters as TCP [13]. TCP-friendly algorithms can be further classified based on steady-state or dynamic behavior. For example, the TCP-friendly rate control (TFRC) algorithm is an example of a *slowly responsive* algorithm that reacts to a single packet loss with a send rate reduction smaller than TCP [30]. TCP-friendly algorithms strive to be *TCP compatible* by emulating TCP's behavior (i.e., slow start, congestion avoidance, exponential timeout backoff, and even self-clocking). Rate-based protocols that adjust their rates based on a computation will generally include a measured RTT primarily to add an element of delay-based congestion avoidance to the algorithm [30]–[32]. Although different from our notion of DCA (i.e., as an incremental enhancement to TCP) and while primarily an artifact of being TCP compatible, these algorithms assume that implicit congestion feedback based on delay rather than packet loss (or rather, supplemental to packet loss) can offer global network improvements as long as the majority of traffic performs the same algorithm. The benefit is the same as what drives the alternative best effort (ABE) initiative, namely a network that provides a service that strives for lower delay possibly at the cost of reduced throughput [33]. This service would be useful for multimedia applications with strict delay requirements.

## III. Measurement Analysis

The objective of the measurement analysis is to show that an increase in a per packet RTT sample (i.e., the *tcpRTT* samples) is not a reliable indicator of future packet loss events and cannot be used to improve TCP throughput. To show this, we trace many TCP connections over different paths. We then post-process the trace files to extract the *tcpRTT* time series (along with the loss indication) associated with each traced connection. This data is the basis of our analysis presented below.

### A. Data Collection Methodology

We selected seven high-speed Internet paths. Each path consists of many hops (at least 11) with a minimum link capacity of 10 Mb/s. The sender of each TCP connection is a host located on the campus of North Carolina State University; each of the seven receivers is located over the Internet. We run a bulk mode TCP application between the host and each destination. The TCP sender in our experiments as well as the trace point (we use *tcpdump* [2] to trace the TCP connection) is a 333-MHz PC running freeBSD. The machine is equipped with a 3COM PCI Ethernet adapter and is attached to a campus network via a 10-Mb/s ethernet connection.

Table I describes each of the seven paths. Of the seven paths, five are located outside of North America. We did not modify the window size associated with each receiver as we want to evaluate the feasibility of an incremental deployment of DCA

TABLE I
SUMMARY OF TRACED PATHS

| # | Destination Host | # Hops | Max Wnd | MSS |
|---|---|---|---|---|
| 1 | Dilbert.emory.mathcs.edu | 12 | 17376 | 1448 |
| 2 | comeng.ce.kyungpk.ac.kr | 19 | 17376 | 1460 |
| 3 | ccg.ee.ntust.edu.tw | 17 | 8760 | 1460 |
| 4 | www.nikhef.nl | 17 | 4096 | 512 |
| 5 | www.snafu.de | 13 | 17520 | 1460 |
| 6 | icawww1.epfl.ch | 18 | 8760 | 1460 |
| 7 | www.eas.asu.edu | 14 | 8760 | 1460 |

TABLE II
SUMMARY OF MEASURED PERFORMANCE. VALUES IN PARENTHESES REPRESENT THE STANDARD DEVIATION OF THE STATISTIC

| # | Avg RTT (seconds) | Avg throughput (Kbytes/sec) | Avg loss rate | Avg % of loss that ended in Time-out |
|---|---|---|---|---|
| 1 | .066(.012) | 185.1(77.5) | .8(1.1) | 13.1(8.0) |
| 2 | .249(.056) | 46.4(19.2) | 1.7(3.2) | 14.5(11.6) |
| 3 | .32(.037) | 12.8(7.6) | 6.5(7.1) | 48.3(9.8) |
| 4 | .117(.02) | 23.8(8.5) | .89(1.1) | 56.1(14.9) |
| 5 | .174(.017) | 71.7(22.4) | .65(.78) | 15.2(9.2) |
| 6 | .171(.01) | 46.2(6.6) | .55(1.1) | 11.2(7.5) |
| 7 | .179(.06) | 10.8(7.3) | 10.5(4.1) | 47.9(13.4) |

on standard TCP/IP host configurations. For three of the paths, we used the *ttcp* application [34]. For the others, we used the *echoping* application [35] which sends data to standard TCP discard servers. The five discard servers were located on Web servers connected to the Internet. Over the course of five days, we traced TCP connections at regular intervals (five runs each day beginning at 9:00 am followed by a run every two hours). Each run transferred between 6 and 20 MB (depending on the path) and lasted anywhere from 3 to 45 minutes depending on the level of congestion over the path.

Table II summarizes the average performance of each path. Paths 1 and 7 represent the best and worst performing path, respectively. We specifically selected paths that crossed several provider's networks, as this is more representative of Internet connections. Subsequent analysis of time-of-day patterns reveal that that all paths (except for path 2) experienced their worst performance during the afternoon runs.

We post-processed a *tcpdump* trace to obtain the round-trip delay associated with each data packet with a unique acknowledgment. We refer to these RTT samples as the *tcpRTT* time series. The *tcpRTT* samples contain more congestion information than does the standard TCP RTT algorithm (i.e., that is used for the retransmit timeout calculation as described in [36]) because the *tcpRTT* samples are more frequent and they are based on a more precise time measurement.

The algorithm used to generate the *tcpRTT* time series is summarized as follows. We describe the algorithm as it would be implemented by a TCP/Sender even though we actually run the algorithm on a *tcpdump* trace. The sender records the departure time of every packet. To filter out error in RTT samples caused by the TCP delayed acknowledgment, *tcpRTT* samples are generated only for the highest segment acknowledged by an ACK. Furthermore, only ACKs that acknowledge more than one segment of data will generate a *tcpRTT* sample. During periods of recovery, the algorithm does not take any *tcpRTT* samples to avoid errors.

The *tcpRTT* time series consists of the following tuple:

$$i : (\mathrm{time}_i,\ tcpRTT_i,\ l_i) \quad \text{where } l_i = \begin{cases} 1 \\ 0 \end{cases}$$

When $l_i = 1$, this implies that the next segment sent after $\mathrm{time}_i$ is dropped and when $l_i = 0$, the segment is not dropped. In the event that multiple loss events are associated with the same *tcpRTT* sample, rather than having a duplicate entry, we keep only one. Therefore, our analysis treats a burst of loss as a single loss event. Also, packets that are retransmitted more than one time will be considered as separate loss events as long as they have a unique *tcpRTT* sample.

### B. Analyzing the Loss Conditioned Delay

We are interested in learning if the *tcpRTT* sample prior to loss (or perhaps the average of some small number of *tcpRTT* samples prior to loss) is greater than a smoothed average of preceding *tcpRTT* samples. In other words, we want to know how frequently loss events in a connection might have been predicted by monitoring RTT. To help assess this, we have developed three metrics which we apply to the *tcpRTT* time series data. The first and the third metrics provide an indication of how well a DCA algorithm might predict future loss events based on measured RTT samples. The second metric assesses the correlation between loss and increases in RTT and provides insight into the observed queueing delay that surrounds loss events.

*1) Correlation Indication Metric:* The correlation indication metric counts the number of times that the *tcpRTT* samples prior to packet loss are greater than the average of some previous *tcpRTT* samples. The metric is based on occurrences of the following delay event:

$$sampledRTT(x) > (windowAVG(w) + \mathrm{std}).$$

The $sampledRTT(x)$ is the average of the $x$ number of *tcpRTT* samples prior to the transmission of a dropped segment. We refer to this as the *loss conditioned delay*. For a given *tcpRTT* time series that contains $K$ loss events, each loss event leads to a $sampledRTT_k(x)$ value defined as follows:

$$sampledRTT_k(x) = \frac{\sum_{j=(i-x+1)}^{i} tcpRTT_j}{x}$$

for all *tcpRTT* samples where $l_i = 1$.

$x$ controls the size of the moving window associated with the *sampledRTT* and determines the responsiveness of the algorithm. Similarly, $windowAVG(w)$ is a moving window average of the previous $w$ *tcpRTT* values prior to the transmission of a segment that is dropped. The $\mathrm{std}$ is the standard deviation associated with $windowAVG(w)$. The $sampledRTT(x)$ represents an "instantaneous" RTT measurement while the

$windowAVG(w)$ is a longer term average. The difference, $sampledRTT(x) - windowAVG(w)$, is reflective of an increase or decrease in queue delays relative to the previous $w$ RTT samples.

For a given run, we calculate the $sampledRTT(x)$ and $windowAVG(w)$ values that are associated with each loss event and count the number of times that the delay event is true. Dividing this count by the total number of packet loss occurrences estimates the probability that the *tcpRTT* samples prior to a loss are higher than some average and consequently is an indicator of how well loss events might be predicted in time to avoid loss. We refer to this as the *correlation indication metric* (CIM) and define it as follows:

$$P[sampledRTT(x) > windowAVG(w) + \text{std}].$$

The objective of the CIM metric can be fine tuned with the selection of the $(x, w)$ pair. When both parameters are large, the metric is an indicator of long-term congestion. As an example, assume the metric is applied to a dataset using an $(x, w)$ pair of (10 000, 500 000). For this $(x, w)$ pair, once the metric value exceeds 0.5, it can be assumed that the network is experiencing long-term congestion (i.e., the *sampledRTT* is based on the most recent several minutes, and the threshold is based on the most recent several hours). An $(x, w)$ pair of (2, 500 000) makes the CIM more sensitive to any increases in RTT (which will cause a DCA algorithm to incorrectly react) but the decision is not relative to the most recent network dynamics. Smaller $(x, w)$ values focus the metric to be sensitive to bursty congestion. For example, an $(x, w)$ pair of (1, 5) makes the algorithm sensitive to packet jitter. As the value of $w$ moves toward $x$, the metric value should approach 0 because the $windowAVG(w)$ values will be identical to the $sampledRTT(x)$ values.

The CIM metric assesses how effectively packet loss over a network might be predicted. The prediction might be incorrect causing a hypothetical DCA algorithm to react to an increase in RTT that is not associated with loss. The CIM uses a standard deviation of the $windowAVG(w)$ statistic to filter incorrect loss predictions. Based on our analysis, we found that the $(x, w)$ pair of (2, 20) along with a threshold of a standard deviation is most effective in accurately predicting loss events across a range of path dynamics. Given that DCA must differentiate longer term congestion swells from short-term queue increases that accompany loss events, an $(x, w)$ pair of (2, 20) achieves this goal (at least better than other combinations of $x$ and $w$).

Table III illustrates the results of the metric applied to the traced data grouped by paths and by time of day (i.e., the time at which the trace was obtained). The parameters of the analysis were (2, 20) with a threshold of one standard deviation. Each data point is the average of five samples of the metric (i.e., the average of the Monday through Friday samples over a particular path at a specific time of day) and the standard deviation associated with the mean. As an example, on average, at 9:00 am, 18.6% (with a standard deviation of 0.1) of all loss events on path 1 were preceded by a detectable increase in RTT. Paths 1, 2, and 5 exhibit the highest level of correlation between delay and loss, while paths 6 and 7 exhibit the lowest level. Taking the average of all values for each path shows that between 7%–18% of loss events that were observed could potentially be predicted and avoided.

TABLE III
CORRELATION INDICATION METRIC RESULTS OVER THE SEVEN PATHS

| # | 9:00AM | 1:00PM | 3:00PM | 5:00PM |
|---|---|---|---|---|
| 1 | .186(.1) | .137(.068) | .162(.034) | .149(.022) |
| 2 | .19(.044) | .135(.028) | .15(.022) | .107(.039) |
| 3 | .065(.047) | .14(.058) | .14(.14) | .08(.1) |
| 4 | .34(.06) | .1(.017) | .132(.165) | .099(.07) |
| 5 | .21(.08) | .21(.16) | .128(.022) | .19(.17) |
| 6 | .34(.29) | .047(.036) | .054(.035) | .097(.077) |
| 7 | .081(.011) | .073(.022) | .063(2.0) | .068(.023) |

The loss conditioned delay indication metric is an indicator of how successful a DCA algorithm could be in avoiding the loss events found in the measured data. While the results indicate that some level of correlation exists between loss and increases in RTT, the results also indicate that loss is typically *not* preceded by an (observed) increase in RTT. Further, the results do not convey the accuracy of the loss prediction decision. In Section III-C, we apply a DCA algorithm on the traced data and, using the loss prediction algorithm described above, show that the frequent incorrect decisions lead to poor performance.

*2) Loss Conditioned Delay Correlation Metric:* The *loss conditioned delay correlation metric* (LCDC) provides a quantification of the magnitude and time scale associated with the correlation between increases in *tcpRTT* samples and loss events. Our algorithm is essentially identical to that used in [23], although a significant difference is that the LCDC metric is constrained by TCP's congestion control algorithms while the approach in [23] utilizes more frequent periodic UDP probes. The LCDC defines the lag $-1$ to be the first *tcpRTT* sample prior to the transmission of a segment that is lost (and lag $-2$ is the second *tcpRTT* sample before the transmission of the lost segment). Likewise, lag $+1$ is the first *tcpRTT* sample that is associated with the segment transmitted after a dropped segment is initially transmitted. The *average packet delay conditioned on a loss at a lag $j$* is defined to be the average of *tcpRTT* samples of packets whose $j$th prior packet is lost. This value is then normalized with the average RTT of all samples. The timescale associated with a lag depends on the connection's window size (or rather how many segments are in flight when a packet is dropped by a router) and the path RTT. Consequently, the time between lags will typically range from several milliseconds to an RTT. DCA has a much better chance of being successful if this metric shows a distinct peak in the loss conditioned delay in the lags immediately prior to packet loss. Such a peak implies that the delay that is associated with packet loss is uniquely detectable (i.e., its magnitude is above the noise of nonloss related RTT variation) and is of duration of at least one RTT which gives the TCP/Sender time to react.

The LCDC results illustrated in Figs. 1–7 confirm that increases in RTT and loss are weakly correlated. There are interesting details that can be observed in the results.

1) All paths exhibit some level of correlation between RTT and loss. The previous CIM metric found paths 1, 2, and

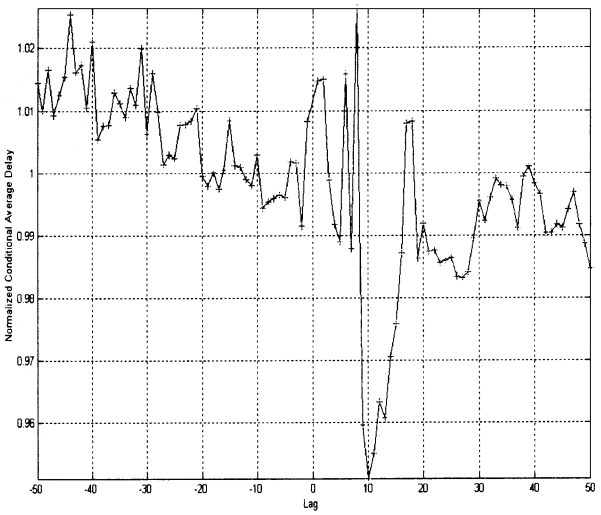


Fig. 1. LCDC metric for path 1.

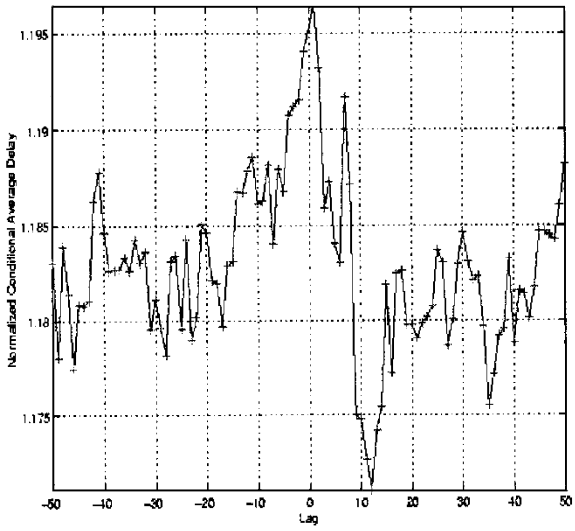


Fig. 2. LCDC metric for path 2.

5 to be the most correlated, but the LCDC finds that paths 2, 3, 4, 5, and 7 are the most correlated. The difference is that the CIM applies a filter to its assessment, thereby ignoring a certain level of correlation that might exist.

2) Each path exhibits a distinct drop in the level of correlation in the lags immediately following the loss event (i.e., lag 1 through lag 10). There are two possible explanations. First, the loss episode (i.e., we cannot tell if the loss episode involves more than one packet being dropped at the router) has no impact on the aggregate traffic pattern. Consequently, the drop in RTT that follows loss indicates that the traffic arrival burst that caused the loss episode has diminished. Second, the loss episode involves multiple packets from different flows. The reduction in RTT is caused by the drop in traffic load as it responds to the congestion indication. This implies that some level of global synchronization between competing flows exists. Global synchronization would also explain the oscillations in the LCDC results (especially path 3, 4, and 5). The authors of

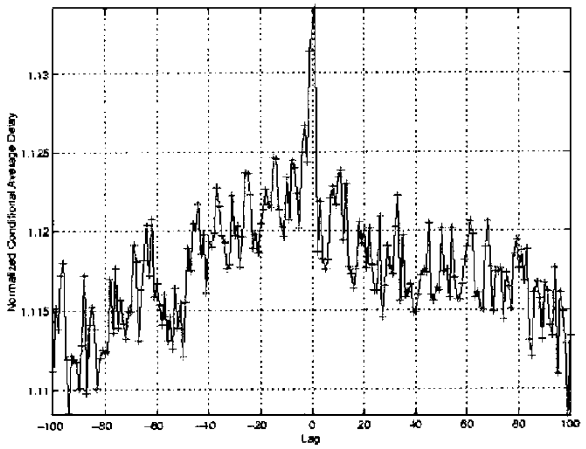


Fig. 3. LCDC metric for path 3.

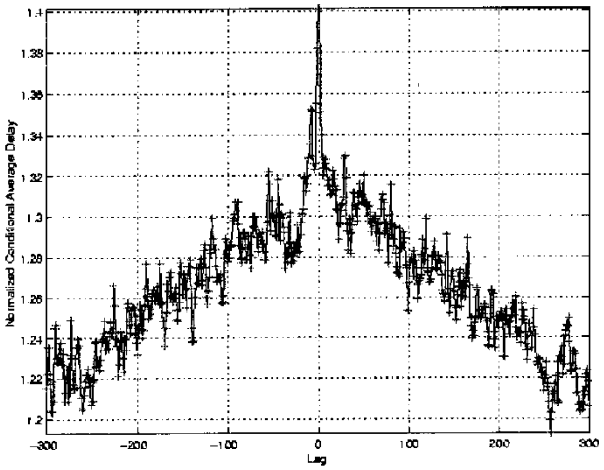


Fig. 4. LCDC metric for path 4.

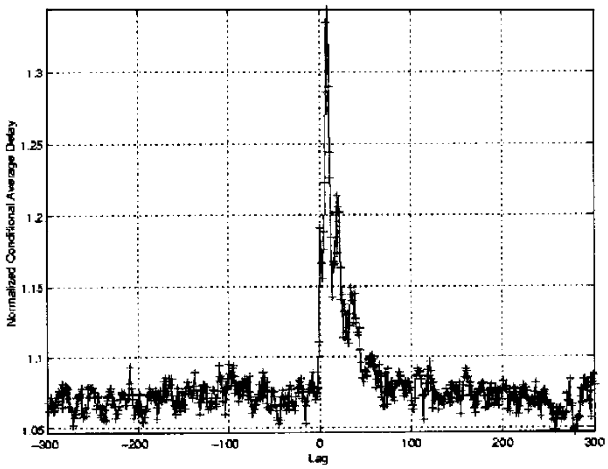


Fig. 5. LCDC metric for path 5.

[23] saw similar signs of global synchronization and conjectured that multiple TCP connections become synchronized by the loss process (as identified in [37]). Through additional simulation analysis, we have determined that if global synchronization were coming into play, all three of our metrics would indicate a much stronger level of correlation. Although further study is necessary, we conjec-

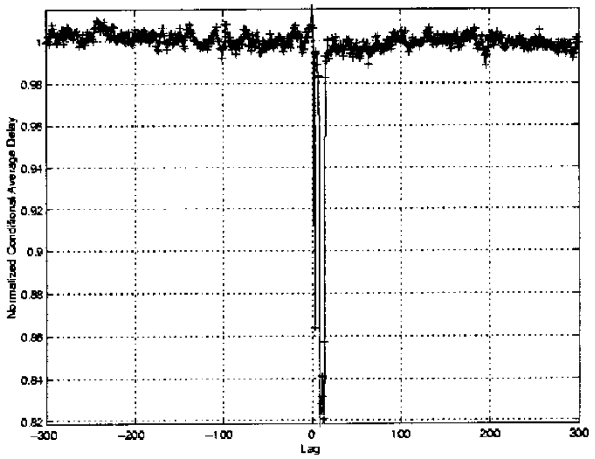


Fig. 6. LCDC metric for path 6.

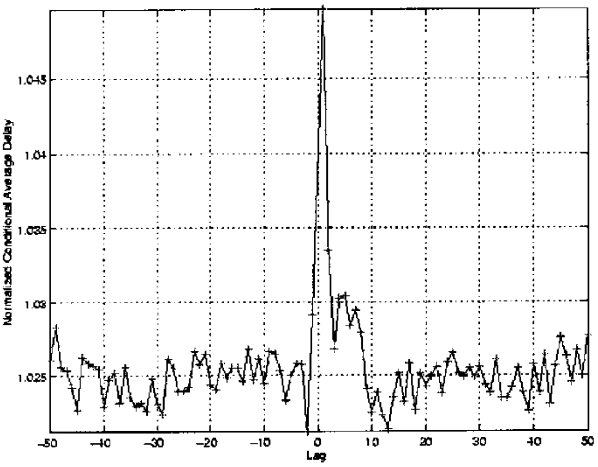


Fig. 7. LCDC metric for path 7.

ture that the drop in RTT following lag 0 simply reflects the traffic arrival process, and the apparent periodic behavior of several of the LCDC results is caused by random increases and decreases in traffic arrival intensity rather than by a global synchronization phenomenon.

3) A further observation is that for all paths, lag 1 exhibits a higher level of correlation than lag $-1$. This differs from the data found in [23] where the authors found a greater level of correlation in the negative lag region (i.e., lag $-1$ through lag $-5$). The LCDC metric defines lag $-1$ as the *tcpRTT* sample that was measured prior to the transmission of the packet that gets dropped. Depending on TCP dynamics (i.e., the number of packets in flight), the time between lag 0 (i.e., the time the dropped packet is first transmitted) and lag $-1$ might be as large as one RTT. Because the congestion associated with loss is of duration less than one RTT, the LCDC is less likely to detect the queue buildup that occurs prior to loss. The method used in [23], on the other hand, produces periodic probes such that the time difference between lags will be 20 ms. Our method measures the correlation observed by a TCP constrained sender-based probe algorithm rather than by a more frequent UDP-based probe which is based on a one-way packet latency measurement rather than an RTT measurement. The result is that the LCDC will lead to a less accurate picture of the congestion process when loss occurs simply because there are fewer probes surrounding the loss event. However, the key point is that the LCDC metric is designed to assesses the "usable" level of congestion information available to a TCP sender.
4) The timescale associated with the delay surrounding the actual loss process is very brief (1 to 5 lags) which supports the findings of [27], where the authors find that the RTT increase surrounding loss is short lived.
5) Several paths (namely, 3 and 4) exhibit correlation over multiple timescales (large and small). For example, path 3 shows that loss tends to occur when the RTT is elevated over a time scale of 200 lags. Intuitively, this makes sense as loss is more likely during periods of long-term congestion. This again suggests that loss is caused by "spikes that ride on ripples that ride on still longer term swells" [29].
6) We find that our LCDC values were slightly lower than that found in [23], but we believe that the difference is due to the dynamics of each path.

*3) Loss Conditioned Delay Cumulative Distribution Function Metric:* The *loss conditioned delay cumulative distribution function* (CDF) metric is a visual metric in that it superimposes a plot of the *tcpRTT* distribution CDF on a plot of the CDF of the *sampledRTT* distribution. If the two distributions appear identical, this suggests that there is nothing statistically unique about the *tcpRTT* samples prior to loss events, making it difficult if not impossible for a DCA algorithm to avoid loss. We apply the CDF metric to the concatenated data for each path. For brevity, we show the results only for paths 1 and 7 (Figs. 8 and 9). In each figure, the histogram with the black bars represents the *tcpRTT* CDF and the histogram with the white bars represents the *sampledRTT* CDF. The results from the other five paths were very similar. We summarize the results as follows.

All paths exhibit a threshold RTT value (i.e., *thresholdRTT*) below which the loss rate is very low. Unfortunately, the *thresholdRTT* value is very close to the minimum *tcpRTT* value for most paths, which makes it difficult for an algorithm to reliably predict future loss events. A DCA algorithm requires the *thresholdRTT* value to be well beyond the mean *tcpRTT* value, otherwise, the algorithm will have frequent incorrect prediction decisions. For the paths with *sampledRTT* distributions that were almost identical to the *tcpRTT* distributions (i.e., paths 1, 5, 6, and 7), possible explanations are: 1) the loss occurs at a very high-speed links such that the queueing delay that surrounds loss is undetectable by the endpoint or 2) there were multiple bottlenecks active at any given time such that one bottleneck produced large queue delays with small loss rates while another bottleneck contributed higher loss rates with minimal delay.

*4) Summary of Metric Results:* By applying the three metrics on traced TCP connections, we further understand the relationship between a TCP constrained RTT measurement and loss events and, more importantly, the viability of DCA. The correlation indication metric suggests that an algorithm that

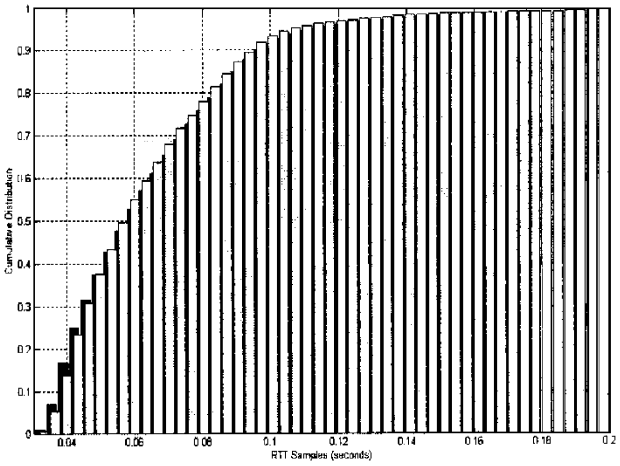


Fig. 8. CDF metric for path 1.

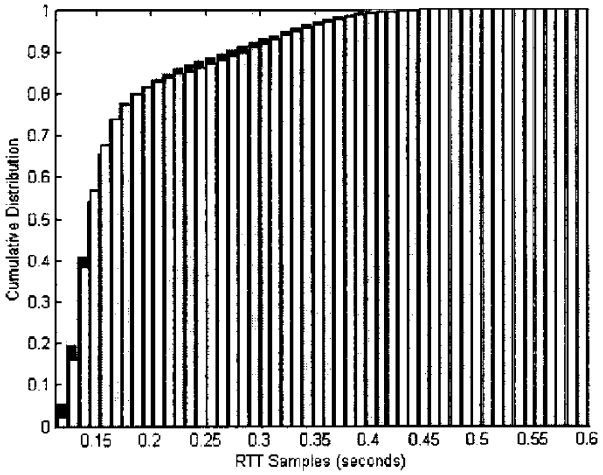


Fig. 9. CDF metric for path 7.

predicts loss based on TCP RTT samples will succeed 7%–18% of the time.[2] If a TCP/DCA algorithm is able to reduce its loss rate by 7%–18%, this would improve performance given that 13%–56% of the loss events results in a TCP timeout (i.e., as indicated in Table III, line 2). The loss conditioned delay CDF metric shows that the "instantaneous" RTT samples are not significantly statistically different from the *tcpRTT* samples. In other words, loss is almost as likely to occur for lower values of *tcpRTT* as for higher values of *tcpRTT* samples. While a TCP/DCA algorithm might reduce the packet loss rate, it will also be reacting frequently to increases in *tcpRTT* that are not associated with loss. We quantify the impact of this on TCP performance in the next section. The second metric (the LCDC metric) again confirms that loss and delay are correlated. However, it indicates that the time scale associated with the queue buildup leading to loss is usually less than one RTT, which supports our method of incorporating the "usable" level of correlation into our analysis of DCA.

[2]This result is based on data from the seven paths which we measured. We believe that these paths are representative of high-speed Internet paths. Consequently, we generalize the result to apply over high-speed Internet paths.

TABLE IV
AGGREGATE RESULTS OF TRACE THROUGHPUT ANALYSIS. THROUGHPUT IS DESIGNATED AS T (IN KB/s)

| # | Avg LR (%) | Avg RTT (secs) | Avg T | Model T | % change in T |
|---|---|---|---|---|---|
| 1 | .8 | .066 | 185.1 | 137.6 | -50.5 |
| 2 | 1.7 | .249 | 46.4 | 31.1 | -45 |
| 3 | 6.5 | .32 | 12.8 | 9.3 | -16 |
| 4 | .89 | .117 | 23.8 | 22.7 | -42.7 |
| 5 | .65 | .174 | 71.7 | 69.2 | -58.7 |
| 6 | .55 | .171 | 46.2 | 41.9 | -34.8 |
| 7 | 10.5 | .179 | 10.8 | 7.9 | -7.5 |

### C. Throughput Analysis

For each traced TCP connection, we run a simple DCA algorithm over the *tcpRTT* time series. In brief, the algorithm assesses the following congestion decision based on the delay event defined earlier:

$$sampledRTT(2) > windowAVG(20) + \text{std}.$$

When the delay event is true, TCP/DCA reduces the congestion window (cwnd) by 50%, which is equivalent to the TCP reaction of a loss event that is recovered using the base TCP loss recovery algorithm (i.e., via a triple duplicate acknowledgment which we refer to as a TD event). A 50% send rate reduction is justified for two reasons. First, the RED/ECN algorithm mandates a 50% cwnd reduction in response to a congestion indication [38]. Second, a 50% reduction (rather than a smaller reduction) improves the chances that a loss event will be avoided. We explore the use of smaller cwnd reduction values in the simulation analysis in the next section.

We adapt a TCP throughput model [18] to help quantify the tradeoff involving DCA reactions that are successful and the DCA reactions which are unnecessary. For a detailed discussion of the throughput model, refer to [19]. We limit the discussion to Table IV, which summarizes the results. The second, third, and fourth columns of the table show the measured results for each path. The fifth column indicates the predicted throughput based on the original TCP model (i.e., as defined in [18]). The model's prediction is consistently lower than the observed throughput. One possible explanation for this (as noted in [39]) is that the loss rate observed by a TCP flow might be higher than the actual loss rate associated with a path resulting in lower predicted throughput. The final column of the table shows the predicted throughput degradation caused by DCA based on our modified TCP throughput model. In summary, our analysis suggests that a TCP/DCA application might experience a reduction in throughput in the range of 7%–58% compared to a similar TCP application.

The amount of degradation tends to decrease as the loss rate becomes large (i.e., paths 3 and 7). This is because the penalty associated with a DCA rate reduction on a low bandwidth connection (i.e., a TCP connection that consumes a small amount of bandwidth because of a high loss rate) is less severe than the penalty imposed on a TCP connection consuming a higher bandwidth. In other words, a low-bandwidth connection can recover quickly from a DCA send rate reduction (e.g., possibly within

several RTTs) while it might take the higher bandwidth connection many RTTs to recover.

## IV. Simulation Analysis

The objectives of the simulation analysis are to validate our measurement analysis and to extend it in a manner that was not possible with measurement. In particular, the objectives are:

- to obtain additional insight into why our measurement analysis suggests that a DCA congestion probe can predict at the most (on average) 7%–18% of loss events. We can only do this by looking at the actual bottleneck link queue levels along with the *tcpRTT* time series;
- to validate our claim that a TCP/DCA algorithm will indeed degrade TCP throughput as compared to TCP/Reno;
- to provide additional validation of our result by showing that TCP/Vegas and TCP/Dual also degrade throughput.

### A. Developing the Models

Using the ns simulation package [20], we developed simulation models based on the two U.S. paths from the set of high-speed Internet paths that we analyzed in the previous section. These two paths, the Emory and ASU paths, were the best and the worst performing paths, respectively, and consequently represent the two most interesting network scenarios for our DCA analysis. We used *pathchar* and *traceroute* to obtain an estimate of the static attributes of the path. For each path, we calibrated the simulation parameters so that the end-to-end characteristics of the simulation models, such as RTT variations, throughput, and loss dynamics are statistically similar to the measured results. By using *pathchar* and correlating the RTT samples obtained from concurrent pings to different routers along the measured paths, we saw evidence that both of the Internet paths are subject to congestion at multiple hops. We used this information to help us design the background traffic levels necessary to emulate the end-to-end dynamics observed over the measured path.

Fig. 10 illustrates the network diagram associated with the Emory model. From *traceroute*, we learned that there are 13 hops along a path that traverses four domains and two ISPs. Each small circle in the figure represents a router. Within an ISP, router links are 135 Mb/s (i.e., simulating 155-Mb/s ATM hops). The interconnection points between ISPs are 45 Mb/s. The link propagation delays range from 1 to 4 ms. The uncongested RTT is roughly 27 ms (which reflects the minimum RTT of the actual Emory path). The 10-Mb/s LANs at both endpoints are the lowest speed links over the path, however, they were not the bottlenecks. Based on the measured data (with some conjecture), we assume that the main congestion point over the path is located at the peering point between ISP 1 and ISP 2. Secondary congestion points are located at emory.net's Internet access point and within ISP 2. The shaded flow arrows in Fig. 10 illustrate the congestion points.

Fig. 11 illustrates the network diagram associated with the ASU model. The measurements indicate that the path is highly congested (i.e., Table II shows that the average loss rate was 10.8% and the average RTT was 0.179 s). The majority of congestion occurs between ISP 2 and ISP 3 (which represents a peering point located at the MAE-EAST exchange). However, as in the Emory path, we saw evidence of multiple points of congestion. Therefore, we designed the traffic loads such that secondary congestion occurs between ISP 1 and ISP2 as well as within ISP 2 and ISP 3's backbone networks.

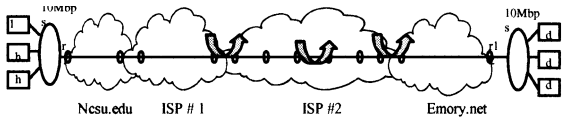


Fig. 10. Simulation model of the Emory path.

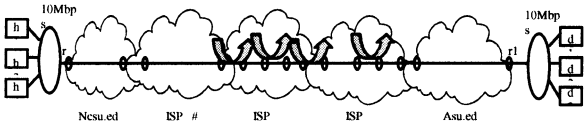


Fig. 11. Simulation model of the ASU path.

We duplicated the measurement experiments and analysis methods using the two simulation models. In other words, we wanted to run a bulk-mode TCP application between a host located at the ncsu.edu network (i.e., the sender) and a destination located at the Emory or Asu networks (i.e., the receiver). We derived *tcpRTT* samples from the simulated connection using a method similar to that used in our measurement work. From this data, in addition to the average loss rate, percentage of loss that leads to timeouts, and average RTT, we were also able to obtain the three correlation metric results that were defined in the previous section.

We designed a set of TCP and UDP flows to create bidirectional background traffic for each model. We adjusted the details of the background traffic such that the end-to-end path dynamics resembled the measured results. We found that even when using thousands of low-bandwidth ON/OFF TCP flows (using a pareto traffic generator configured to emulate the traffic generated by a "web" user as described in [40]), we could not duplicate the burstiness associated with RTT variations observed over the paths. Therefore, we used a combination of TCP flows (several hundred) along with several high-bandwidth ON/OFF UDP flows.

Figs. 12 and 13 illustrate a portion of the observed behavior over the real and simulated Emory paths, respectively. Fig. 12 is based on data that was a part of our measurement analysis (i.e., one *ttcp* transfer between NCSU and Emory). The top curve in both figures plots the *tcpRTT* time series (the "+" marks forming the curve are the samples) of an end-to-end TCP/Reno connection between h1 and d1. The "+" marks along the top border of the graph identify the *tcpRTT* samples just prior to the transmission of a segment that is dropped somewhere along the path. The lower curve plots the TCP "goodput" (i.e., the rate that data is acknowledged) averaged over intervals of 0.5 s (the dark line) and 2 s (the dashed line). The end-to-end TCP/Reno connection under observation is configured similarly to the TCP stack used in the measurements. The maximum window size is limited to 12 packets with the maximum segment size set to 1448 bytes. An ftp application sources the TCP connection (the sender is located at node h1). The sink (node d1) is configured to perform delayed acknowledgment. The loss rate at the link

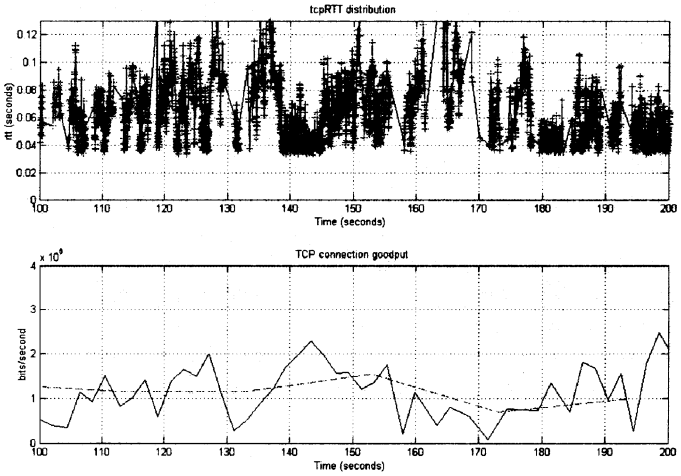


Fig. 12. Measured results over the Emory path.

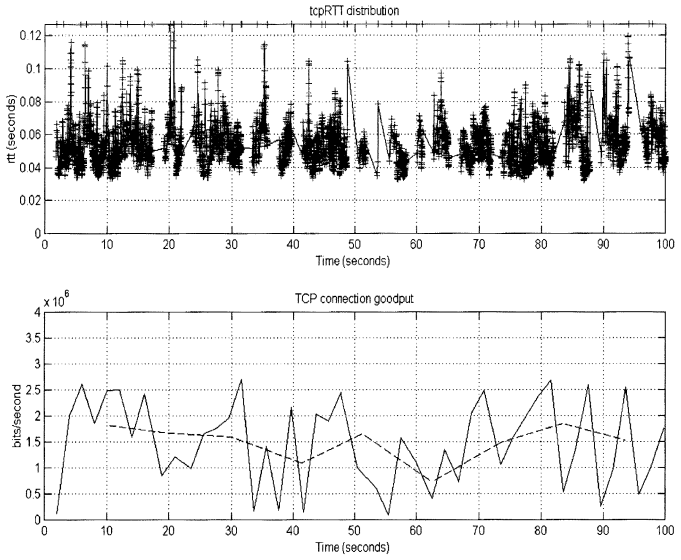


Fig. 13. Simulated results over the Emory path.

between ISP 1 and ISP 2 is 1.5%, 0.86% at the congestion point within ISP 2, and 0.6% at the link between ISP 2 and Emory's network.

For the measured connection visualized in Fig. 12, the loss rate was 1% while the loss rate experienced by the simulated TCP/Reno connection was 0.55%, which explains the throughput achieved by the simulation model. In spite of the difference in loss rates, the more important dynamic the simulation model must reproduce is the relationship between RTT and loss. The top curves in Figs. 12 and 13 show that the *tcpRTT* variation of the measured path and of the simulated path are similar. We applied the three metrics that we have developed (i.e., the CIM, the LCDC and the CDF metrics) on sets of data from simulation experiments based on the Emory and the ASU models and confirmed that the correlation dynamics between RTT and loss is representative of the measured results. Refer to [19] for further validation of the simulation models.

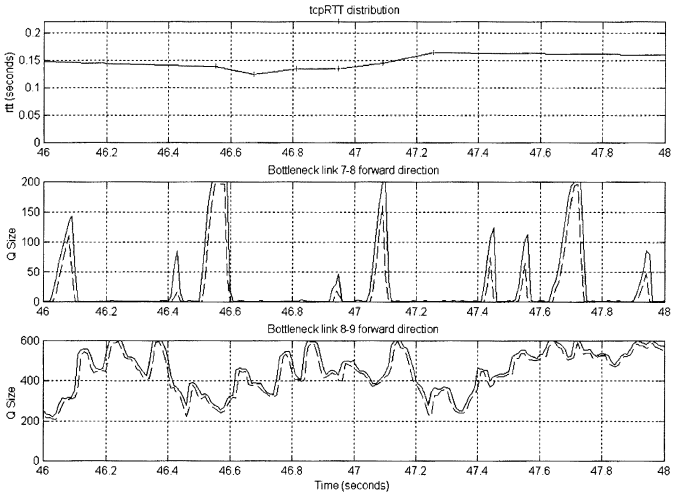


Fig. 14. ASU simulation results.

### *B. Fundamental Problems*

Our measurement experiments suggest that DCA might only be able to correctly predict, on average, 7%–18% of loss events over high-speed Internet paths. Through simulation, we want to confirm our intuitive understanding of why a DCA congestion probe has difficulties predicting future packet loss events. The fundamental problem is that a TCP constrained RTT sampling process prevents endpoints from accurately tracking bursty congestion associated with packet loss. Factors such as bursty traffic arrival processes at high-speed switches along with the dynamics of TCP's congestion control algorithms make it difficult for TCP endpoints to reliably avoid packet loss. For example, TCP tends to send packets "clumped" together, rendering the probe more granular especially over long paths. In the extreme, the TCP RTT probe granularity is limited to once per RTT which is not able to track bursty congestion over high-speed paths. We verify this through simulation.

Fig. 14 illustrates a portion of a run over the ASU model. The top curve plots the *tcpRTT* time series of the end-to-end TCP/Reno connection under observation. The second curve plots the queue level at the congested link in ISP 2 and the lower curve plots the queue level at the congested link connecting ISP 2 and ISP 3. In the queue plots, the solid line plots the maximum queue level observed every 0.1 s and the dashed curve plots the minimum queue level observed. A single loss event occurs at time 47.15 s and is caused by the bursty traffic arrival process located at the congested link within ISP 2. Due to the congestion level, the granularity of the RTT samples is extremely coarse (i.e., the connection experienced a 7% loss rate, which reduced the TCP throughput to less than 64 Kb/s. The *tcpRTT* sample at 46.95 s is the RTT sample preceding the transmission of the packet that is dropped and completely misses the queue buildup that is associated with loss. Although this scenario depicts the worst case network conditions for DCA, it illustrates the fundamental problems a DCA algorithm might experience over high-speed Internet paths: 1) the time scale associated with queue buildup that precedes loss is less than one RTT time; 2) congestion at multiple links reduces the

TABLE V
THROUGHPUT DEGRADATION FOR VARYING LEVELS OF THE *CONGESTIONREDUCTION* (% DROP)

| % drop | % change in throughput using the Emory model mean, (95% confidence interval) | % change in throughput using the ASU model mean, (95% confidence interval) |
|---|---|---|
| 50% | -37, (-42,-31) | -13.2, (-19.8, -6.6) |
| 25% | -12, (-21,-4) | -9.8, (-24, 4) |
| 12.5% | -6.8, (-13, -.8) | -7.4, (-18.1, 3.1) |

chances of being able to accurately predict future loss events; and 3) TCP's congestion control makes DCA more difficult by limiting the number of probes in flight when the algorithm needs accurate congestion information the most.

### C. Simulating the TCP/DCA Protocol

In this section, we present and analyze a TCP/DCA algorithm. We show that because of the fundamental problems facing DCA (presented in the previous section), a TCP/DCA connection will experience reduced throughput compared to TCP/Reno. To further validate the result, we modify the background traffic generators to increase the level of correlation between delay and loss. We find that while TCP/DCA is able to reduce the loss rate experienced by the connection, the connection performs worse. The analysis supports our claim that TCP/DCA is not a viable incremental enhancement for TCP over high-speed Internet paths.

The TCP/DCA protocol augments a TCP/Reno sender with additional DCA congestion decisions. One of the parameters associated with the TCP/DCA algorithm is the send rate reduction level when DCA reacts to an increase in RTT (i.e., the *CongestionReductionLevel*). While the normal reaction to a non-loss-based congestion indication is to reduce the cwnd by 50%; this is not a requirement [38]. A DCA enhanced TCP sender will compute the *tcpRTT* sample using the algorithm we have defined and used in our measurement analysis. After the normal TCP *cwnd* adjustment (either slow start or congestion avoidance), the following additional code is performed.

```
If (tcpRTT is updated) && (if a DCA reduction has not been
performed for at least one RTT time period) {
   If (sampledRTT(2) > windowAVG(20) + std)
      cwnd = cwnd − (cwnd * CongestionReductionLevel)
}
```

We ran multiple simulation experiments and compared the throughput achieved by two TCP connections (one Reno and one DCA) using the Emory and ASU simulation models. Of interest is the impact that different *CongestionReductionLevel* values have on the assessment. Each experiment is based on ten runs, each lasting 500 s of simulation time. Table V shows the results for three values of *CongestionReductionLevel*. The second column indicates the change in throughput of TCP/DCA as compared the competing TCP/Reno connection using the Emory model. The value is the mean of the results from the ten runs along with the 95th percentile associated with the statistic. The third column indicates the change in throughput (the mean and the 95th percentile statistic) when using the ASU model. When the *CongestionReductionLevel* is 50%, TCP/DCA experienced a throughput loss of 37% over the Emory model and a loss of 13% over the ASU model. At the 50% *CongestionReductionLevel*, even though the algorithm is able to reduce the packet loss rate, the algorithm is frequently reacting to increases in RTT unnecessarily (i.e., where the increase in RTT is not associated with packet loss). As the *CongestionReductionLevel* decreases, the penalty caused by incorrect congestion decisions is reduced, which is why the throughput loss is not as significant. However, the ability for DCA to successfully avoid a loss event is reduced as well.

TABLE VI
THROUGHPUT DEGRADATION WITH THE MODIFIED EMORY MODEL

| Average queue level in packets at ISP #2's internal congestion point (link capacity is 200 packets) | % change in throughput mean, (95% confidence interval) |
|---|---|
| 143.7 | -42.4, (-47.3, -37.5) |
| 38.3 | -50, (-53.5, -4.7) |

In order to match the measured end-to-end dynamics, the Emory model relied on a link level traffic model with high-bandwidth UDP flows to define the loss dynamics over the path. Although the ASU model relied on a larger percentage of TCP traffic, it still consists of a fairly significant level of UDP traffic (8%). To be complete, we also evaluate DCA when the loss was driven entirely by TCP flows (even though this leads to dynamics that differ from the dynamics of the traced connections). We modified the congestion dynamics associated with the Emory model such that loss was isolated to congested link within ISP 2 and changed the background traffic to consist of many low-bandwidth ON/OFF TCP flows (rather than by a combination of TCP and UDP flows). We designed two cases: a heavily congested scenario and a lightly congested scenario. In both cases, the background traffic consisted of 2200 TCP flows along with a small amount of low-bandwidth UDP traffic (less than 5% of the traffic is UDP). The idle time associated with the pareto traffic generator attached to the TCP flows was 1–4 s in the highly congested case and 3–5 s in the lightly congested case. The loss rate in the heavily congested case was in the range of 0.8%–2% and the link experienced sustained queue delay. In the lightly congested case, the loss rate was low (less than 0.5%) and the link experienced periodic epochs of queue delay.

Table VI illustrates the results. Based on a DCA *CongestionReductionLevel* of 50%, the top row shows the average queue level of the bottleneck link and the throughput degradation for the heavily congested scenario (the bottom row shows the results for the lightly congested case). Running the correlation indication metric on the *tcpRTT* time series generated by the TCP/DCA connection confirmed that the path exhibits a significantly higher level of correlation between loss and increased RTT (as compared to the measured data). We verified that the queue levels increase at a slower rate,

giving DCA a better chance at avoiding loss. In both cases, the throughput degrades significantly. In the less congested case, it turns out that DCA is somewhat successful at avoiding loss (up to 30% of loss events were avoided). However, for both cases, the "unnecessary" reactions by DCA to RTT increases not associated with loss drove the throughput down (by 42% in the moderately congested case and by 53% in the more congested case).

### *D. Simulating the TCP/Vegas and TCP/DCA Protocols*

In this section, we support our claim that DCA will result in throughput degradation over high-speed Internet paths by examining two additional DCA algorithms. In particular, we analyze the performance of TCP/Vegas and TCP/Dual using the simulation models and methods that we have developed. The model of TCP/Dual is a straightforward extension to TCP/Reno and is described in [9]. We created a TCP/Dual model for the ns simulator and use the existing TCP/Vegas ns model. As described in [8], TCP/Vegas represents two enhancements: a DCA algorithm referred to as *congestion avoidance mechanism* (i.e., CAM) and an enhanced loss recovery algorithm. To evaluate the DCA algorithm of Vegas, we had to isolate the Vegas enhanced loss recovery improvement from the CAM algorithm. We created two versions of Vegas. TCP/Reno-Vegas includes the enhanced loss recovery algorithm without CAM. TCP/Vegas-CAM includes the CAM algorithm but without the enhanced loss recovery.

We compared TCP/Reno-Vegas to TCP/Reno and to TCP/NewReno. We ran three connections (a Reno, Reno-Vegas, and NewReno connection) over the simulated Emory and ASU paths (using the background traffic described in Section IV-A that was based on a mix of TCP and UDP traffic). The experiment consists of five runs, each run being 500 s. For each run, we found the relative change in TCP throughput between the two enhanced protocols with respect to the TCP/Reno connection. We found that the TCP/Reno-Vegas flow is able to improve TCP throughput by about 70% over the Emory path. The NewReno algorithm is able to improve throughput by 76% over the path. Both enhanced loss recovery algorithms improve throughput by significantly reducing the frequency of timeouts (on the order of 65% in our example).

It has been observed that the Vegas loss recovery mechanism makes a greater contribution to performance than CAM [10]. The more general conclusion was articulated in [11] where the authors stated that the impact of each Vegas enhancement varies depending on buffer sizes. We extend this conclusion by suggesting that the impact of the different components of Vegas depends on network conditions (i.e., static network parameters as well as network dynamics). The objective of this section is to examine the TCP/Vegas-CAM and the TCP/Dual algorithms and show that these algorithms also lead to TCP throughput degradation over the two simulated high-speed Internet paths.

The simulation analysis is similar to previous experiments. Using the Emory and ASU models (as illustrated in Figs. 10 and 11), we run three ftp connections between hosts h1 through h3 and destination d1 through d3. Each connection was configured for TCP/Reno, TCP/Vegas, and TCP/Dual, respectively. Table VII shows the throughput degradation experienced by a Vegas-CAM flow and TCP/Dual over each path. As before, the table values represent the mean and 95% confidence interval of the observed change in throughput based on ten simulation runs. Comparing the results with the equivalent TCP/DCA experiments (Table V), Vegas and Dual experience less of a throughput reduction than TCP/DCA. This is primarily because both Vegas and Dual react less aggressively to the delay-based congestion indication. The throughput reduction caused by Vegas is less over the congested ASU path. This can be explained by looking at the details of the Vegas algorithm. In brief, Vegas reacts indirectly to RTT (i.e., changes in throughput) such that the level of control by the CAM algorithm decreases as the TCP sending rate decreases. Another factor is that the path suffers from sustained congestion. Vegas is more likely to underestimate the state of congestion over a path that exhibits sustained congestion than DCA or Dual.

TABLE VII
THROUGHPUT CHANGE FOR TCP/VEGAS-CAM AND TCP/DUAL

| Model | TCP/Vegas-CAM Mean, 95% confidence interval | TCP/Dual Mean, 95% confidence interval |
|---|---|---|
| Emory | -21, (-27,-15) | -14.2, (-22.9, -5.43) |
| ASU | -7.3, (-20.3,5.8) | -18, (-23.3, -12.9) |

## V. CONCLUSIONS AND FUTURE WORK

We have studied the performance of a class of TCP end-to-end congestion avoidance algorithms which use an increase in packet RTT as an indicator of congestion and future packet loss. We have provided evidence suggesting that DCA cannot be incrementally deployed over high-speed Internet paths. The measurement results we have presented in this paper suggest that the correlation between loss events and increases in TCP RTT samples is weak. Using simulation, we have shown that the fundamental problems involve the bursty nature of aggregate TCP traffic arriving at high-speed switches and the coarseness of a TCP constrained RTT probe mechanism. We have shown that a consequence of this result to DCA is that it is rare (only 7%–18% of the time on average) for a TCP/DCA endpoint to detect the queue buildup that precedes packet loss and react in time to avoid the loss. We have also shown that DCA will frequently make incorrect congestion decisions as it reacts to the many increases in RTT that are not associated with packet loss.

Our work does have several limitations. First, the measurements represent a small sample of Internet dynamics, two of which exhibited very high loss rates. We believe that the set of paths chosen reflect the wide variety of path characteristics an end user might experience over the commercial Internet. We purposely added two high-loss paths to the set to fully assess the viability of DCA. Second, the throughput analysis assumes that the analytic throughput model that we use is accurate (at least to some degree). Finally, the simulation models, especially the design of the background traffic, relies in part on conjecture. Some of our results, especially the conjectures we make that explain the fundamental problems behind DCA, are dependent on accurately modeled dynamics at the bottleneck links.

We plan to evaluate DCA over low-speed paths that utilize broadband access technology such as xDSL or cable. Wireless networks (i.e., 3G or 802.11 wireless LANs) represent a possible application for DCA, as it is useful to know if a loss event was caused by congestion or channel errors. However, in these environments that are potentially more suitable for DCA, explicit congestion notification must also be considered.

A problem that we struggled with was how to accurately model the dynamics of the Internet. The difficulties surrounding large-scale Internet simulation are well known [41]. We obtained satisfactory results using a combination of many TCP flows along with several high-bandwidth UDP flows. To accurately assess the impact of a large deployment of DCA, the design of the background traffic becomes vital. In our analysis, when we used primarily TCP flows (i.e., thousands of flows), we found that we were unable to accurately reproduce the characteristics of the measured paths. However, using too much UDP traffic (i.e., aggregate link-level traffic model) might interfere with the evaluation. We plan on focusing our measurement analysis techniques (which has provided us with interesting clues and insight into congestion dynamics over the Internet) to further develop simulation models that better represent Internet behavior.

## ACKNOWLEDGMENT

The authors would like to thank the anonymous reviewers and the editor for their comments and input, which have greatly contributed to the improvement of the presentation of this paper.

## REFERENCES

[1] V. Jacobson, "Congestion avoidance and control," in *Proc. ACM SIGCOMM*, 1988, pp. 314–329.

[2] V. Jacobson, C. Leres, and S. McCanne. (1989, June) tcpdump. [Online]. Available: ftp://ftp.ee.lbl.gov

[3] M. Allman, V. Paxson, and W. Stevens, "TCP congestion control," IETF, RFC 2581, Apr. 1999.

[4] S. Floyd, "A report on some recent developments in TCP congestion control," *IEEE Commun. Mag.*, vol. 39, pp. 84–90, Apr. 2001.

[5] M. Mathis, J. Mahdavi, S. Floyd, and A. Romanow, "TCP selective acknowledgment options," Network Working Group, RFC 2018, Apr. 1996.

[6] M. Allman and S. Floyd. (2000, Aug.) Enhancing TCP's loss recovery using limited transmit. IETF. Internet Draft. [Online]. Available: draft-ietf-tsvwg-limited-xmit-00.txt

[7] R. Jain, "A delay-based approach for congestion avoidance in interconnected heterogeneous computer networks," *Comput. Commun. Rev.*, vol. 19, no. 5, pp. 56–71, Oct. 1989.

[8] L. S. Brakmo, S. W. O'Malley, and L. L. Peterson, "TCP Vegas: New techniques for congestion detection and avoidance," in *Proc. ACM SIGCOMM*, Aug. 1994, pp. 24–35.

[9] Z. Wang and J. Crowcroft, "Eliminating periodic packet losses in the 4.3-Tahoe BSD TCP congestion control algorithm," *Comput. Commun. Rev.*, vol. 22, no. 2, pp. 9–16, Apr. 1992.

[10] U. Hengartner, J. Bolliger, and T. Gross, "TCP Vegas revisited," in *Proc. IEEE INFOCOM*, Mar. 2000, pp. 1546–1555.

[11] S. Low, L. Peterson, and L. Wang, "Understanding TCP Vegas: A duality model," *J. ACM*, vol. 49, no. 2, pp. 207–235, Mar. 2002.

[12] J. Mo *et al.*, "Analysis and comparison of TCP/Reno and Vegas," in *Proc. IEEE INFOCOM*, 1999, pp. 1556–1563.

[13] D. Bansal, H. Balakrishnan, S. Floyd, and S. Shenker, "Dynamic behavior of slowly-responsive congestion control algorithms," in *Proc. ACM SIGCOMM*, Aug. 2001, pp. 263–274.

[14] K. Claffy, G. Miller, and K. Thompson, "The nature of the beast: Recent traffic measurements from an internet backbone," in *Proc. INET Conf.*, 1998, [Online.] Available: http://www.isoc.org/inet98/proceedings/6g/6g_3.htm.

[15] W. Fang and L. Peterson, "Inter-AS traffic patterns and their implications," in *Proc. IEEE GLOBECOM*, 1999, pp. 1859–1868.

[16] K. Thompson, G. Miller, and R. Wilder, "Wide area internet traffic patterns and characteristics," *IEEE Network*, vol. 11, pp. 10–23, Nov. 1997.

[17] C. Fraleigh, S. Moon, B. Lyles, C. Cotton, M. Khan, D. Moll, R. Rockwell, T. Seely, and C. Diot, "Packet-level traffic measurements from the Sprint IP backbone," *IEEE Network*, to be published.

[18] J. Padhye *et al.*, "Modeling TCP throughput: A simple model and its empirical validation," in *Proc. ACM SIGCOMM*, 1998, pp. 303–314.

[19] J. Martin, "RTT-based congestion avoidance for high speed TCP Internet connections," Ph.D. dissertation, North Carolina State Univ., Raleigh, Dec. 1999.

[20] The Network Simulator. Univ. California, Berkeley, CA. [Online]. Available: http://www-mash.cs.Berkeley.EDU/ns/

[21] J. Ahn, P. Danzig, Z. Liu, and L. Yan, "Evaluation of TCP Vegas: Emulation and experiment," in *Proc. ACM SIGCOMM*, 1995, pp. 185–195.

[22] J. Bolot, "End-to-end packet delay and loss behavior in the Internet," in *Proc. ACM SIGCOMM*, 1993, pp. 289–298.

[23] S. Moon, J. Kurose, and D. Towsley, "Correlation of packet delay and loss in the Internet," Dept. Comput. Sci., Univ. Massachusetts, Amherst, Tech. Rep. 98–11, 1998.

[24] V. Paxson, "End-to-end internet packet dynamics," *IEEE/ACM Trans. Networking*, vol. 7, pp. 277–292, June 1999.

[25] M. Yajnik, S. Moon, J. Kurose, and D. Towsley, "Measurement and modeling of the temporal dependence in packet loss," in *Proc. IEEE INFOCOM*, Mar. 1999, pp. 345–352.

[26] V. Paxson, "Measurements and analysis of end-to-end Internet dynamics," Ph.D. dissertation, Univ. California, Berkeley, CA, 1997.

[27] Y. Zhang, N. Duffield, V. Paxson, and S. Shenker, "On the constancy of internet path properties," in *Proc. ACM SIGCOMM Internet Measurement Workshop (IMW2001)*, Nov. 2001, pp. 197–211.

[28] D. Loguinov and H. Radha, "Large-scale experimental study of internet performance using video traffic," *Comput. Commun. Rev.*, vol. 32, no. 1, pp. 7–19, Jan. 2002.

[29] W. Leland *et al.*, "On the self-similar nature of Ethernet traffic," *IEEE Trans. Networking*, vol. 2, pp. 1–15, Feb. 1994.

[30] S. Floyd, M. Handley, J. Padhye, and J. Widmer, "Equation-based congestion control for unicast applications," in *Proc. ACM SIGCOMM*, Aug. 2000, pp. 43–56.

[31] S. Cen, C. Pu, and J. Walpole, "Flow and congestion control for internet streaming applications," in *Proc. Multimedia Computing and Networking*, Jan. 1998, pp. 250–264.

[32] R. Rejaie, M. Handley, and D. Estrin, "RAP: An end-to-end rate-based congestion control mechanism for realtime streams in the Internet," in *Proc. IEEE INFOCOM*, 1999, pp. 1337–1345.

[33] P. Hurley *et al.*, "ABE: Providing a low-delay service within best effort," *IEEE Network*, vol. 15, pp. 60–69, May/June 2001.

[34] R. Miles. *ttcp* measurement tool. The FreeBSD Project. [Online]. Available: http://www.freebsd.org/ports

[35] S. Bortzmeyer. (2002, Oct.) Echoping measurement tool. [Online]. Available: http://echoping.sourceforge.net

[36] V. Paxson and M. Allman, "Computing TCP's retransmission timer," Network Working Group, RFC 2988, Nov. 2000.

[37] S. Shenker, L. Zhang, and D. Clark, "Some observations on the dynamics of a congestion control algorithm," in *Proc. ACM SIGCOMM*, 1990, pp. 30–39.

[38] S. Floyd and K. Ramakrishnan. (1999, Jan.) A Proposal to add explicit congestion notification to IP. Experimental RFC 2481. Info. Sci. Inst., Los Angeles, CA. [Online]. Available: ftp://ftp.isi.edu/in-notes/rfc2481.txt

[39] M. Goyal, R. Guerin, and R. Rajan, "Predicting TCP throughput from noninvasive network sampling," in *Proc. IEEE INFOCOM*, June 2002, pp. 180–189.

[40] P. Barford and M. Crovella, "Generating representative web workloads for network and server performance evaluation," in *ACM SIGMETRICS*, July 1998, pp. 151–160.

[41] S. Floyd and V. Paxson, "Difficulties in simulating the internet," *IEEE/ACM Trans. Networking*, vol. 9, pp. 392–403, Aug. 2001.

**Jim Martin** (M'88) received the B.S degree in electrical engineering from the University of Illinois, Urbana-Champaign, in 1983, the M.S. degree from Arizona State University, Tempe, in 1989, and the Ph.D. degree from North Carolina State University, Raleigh, in 1999.

He is currently an Assistant Professor with the Department of Computer Science, Clemson University, Clemson, SC. He was previously a Senior Consultant for the Gartner Group, where he provided consulting in the area of network design and performance management to service providers and companies. Prior to joining Gartner, he spent ten years at IBM Network Systems Division where he worked on the research and development of network and network security products. His research interests are in communication networks, network performance management, and Internet transport issues.

**Arne A. Nilsson** received the Master of Electrical Engineering degree and the Ph.D. degree in telecommunication systems from the Lund University of Technology, Lund, Sweden, in 1968 and 1976, respectively.

He is currently a Professor with the Department of Electrical and Computer Engineering, North Carolina State University (NCSU), Raleigh. He has more than 30 years of experience with telecommunication networks. He was a Principal Investigator for the Swedish Telecommunication Administration in the effort to build a Nordic packet switched network. He joined the faculty at NCSU in 1978. In 1986, he was a Professor at the Aeronautics Institute of Technology (ITA), Sao Jose dos Campos, Brazil. He was on the committee responsible for the design and architecture of the MCNC data and video network in North Carolina, and was a Principal Investigator on the VISTAnet, one of 5-Gb test beds. He has been an active participant in the industry/university cooperative research center on communications and signal processing at NCSU, Center for Communications and Signal Processing (CCSP), and the Center for Advanced Computing and Communication (CACC). He has recently become an External Evaluator for some of the European efforts in high-speed networking. He has also worked for the U.S. Army Research Office in the wireless network engineering area.

**Injong Rhee** (SM'89) received the Ph.D. degree from the University of North Carolina, Chapel Hill.

He is currently an Associate Professor of computer science with North Carolina State University, Raleigh. In 2000, he founded Togabi Technologies, Inc., a company that develops and markets mobile wireless multimedia applications for next-generation wireless networks. He served as CTO and CEO of the company until 2002. His research interests are in computer networks, congestion control, multimedia streaming, video compression, distributed systems, and operation systems.

Dr. Rhee is a Member of the Association for Computing Machinery.