5 Measurement Data and Analysis

In this chapter we apply the three metrics described in the previous chapter to the *tcpRTT* time series data that is extracted from the TCP traces. The results support our claim that an increase in RTT is not a reliable indicator of packet loss. While the results indicate that loss events are usually accompanied by an increase in RTT, only 7-18% of loss events are preceded by an increase that exceeds a moving window average of previous RTT samples by a standard deviation. The results also indicate that increases in *tcpRTT* samples are common and are usually not associated with packet loss.

To provide evidence that the level of correlation that does exist between increases in RTT and packet loss is not sufficient to allow a DCA algorithm to improve TCP throughput, we present a technique that assesses how well a DCA algorithm might perform if it were to encounter the *tcpRTT* time series that we extract from the TCP traces. We run a hypothetical DCA algorithm (which we will refer to as TCP/DCA) on the *tcpRTT* data to obtain an adjusted packet loss rate (adjusted for the number of loss events successfully avoided by the TCP/DCA algorithm) and to obtain the frequency of DCA reactions to increases in RTT (with respect to the total number of segments transmitted during the run). Therefore, we generate two probabilities: a modified packet loss rate and a DCA reaction rate. We modify an analytic TCP throughput model to account for the reduction in TCP throughput caused by packet loss (due to both timeouts and to the TCP fast recovery algorithm) and also to account for the reduction in TCP throughput caused by the DCA congestion reactions. Using this approach, we show that even if DCA is able to reduce the packet loss rate, the frequent DCA decisions will degrade TCP throughput.

The DCA throughput analysis described above is based on two assumptions:

- The analysis assumes that the reactions of a DCA flow will have a minimal impact on the congestion process that is active over the path. By making this assumption, we are assured that the decisions that the algorithm makes will not significantly change the *tcpRTT* data points. We validate this assumption using simulation in Part 2 of the dissertation.
- The TCP/DCA algorithm is designed to react aggressively to congestion by reducing the window by 50%. The metric results that we describe shortly indicates that the queue increase that precedes loss

occurs quickly (on the order of 1-2 RTTs or less) which suggests that an aggressive reaction is necessary to avoid loss. Clearly other DCA algorithms are possible. In Part 2 of the dissertation, we show that our results are valid for other TCP/DCA algorithms that react less aggressively to increases in RTT.

5.1 Introduction

Table 5-1 summarizes information associated with the aggregated results from all runs. Over the course of 5 days, we initiated 5 runs periodically each day over each path (beginning at 9:00AM, repeated every 2 hours with the last run taking place at 5:00PM). The amount of data sent during a single run ranged from 20 Mbytes (for paths 1,2,4,5,6) to 6.5 Mbytes (for paths 3 and 7). The duration of the transfer ranged from several minutes to 45 minutes depending on the path and the congestion level.

Table 5-1. Summary of measured paths

Path #	Destination Host	# Hops	MAX Window	MSS	service
1	dilbert.emory.mathcs.edu	12	17376	1448	ttcp
2	comeng.ce.kyungpook.ac.kr	19	17376	1460	ttcp
3	ccg.ee.ntust.edu.tw	17	8760	1460	ttcp
4	www.nikhef.nl	17	4096	512	discard
5	www.snafu.de	13	17520	1460	discard
6	icawww1.epfl.ch	18	8760	1460	discard
7	www.eas.asu.edu	14	8760	1460	discard

Table 5-2 summarizes the average performance observed over all runs for each path[1]. The range of performance across the different paths was tremendous. The range of dynamics associated with each path was almost as great. For a given path, performance tended to be tied to particular time of day patterns (although there were exceptions to this). The www.eas.asu.edu path was the most predictable as the loss

[1] Our analysis examines the relationship between the *tcpRTT* time series and loss events which we consider to be a dynamic associated with a path. Therefore, the data analysis is performed on a path basis.

rate experienced by the flow was typically 7% at the first run of the day and deteriorated with each subsequent run. As expected, performance was driven by maximum window size, loss rates and the frequency of time-outs.

Table 5-2. Summary of measured performance

Path #	Avg RTT (seconds) (standard deviation)	Avg throughput (Kbytes/sec) (standard deviation)	Loss rate(%) mean (standard deviation)	% of loss that ended in Time-out mean (standard deviation)
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)

Tables 5-3, 5-4 and 5-5 illustrate the performance of each path based on the time of day. We observe that over the US paths, there is a tendency for the level of congestion to increase in the afternoon runs. Other than this, there are no clear patterns of behavior based on the time of day.

Table 5.3. Average RTT in seconds with standard deviation based on time of day (EST)

Path #	9:00AM	11:00AM	1:00PM	3:00PM	5:00PM
1	.066(.017)	.065(.07)	.062(.014)	.073(.015)	.062(.009)
2	.248(.031)	.239(.025)	.247(.033)	.231(.022)	.279(.118)
3	.332(.039)	.31(.024)	.302(.03)	.304(.042)	.355(.03)
4	.132(.032)	.114(.020)	.117(.007)	.112(.019)	.110(.012)
5	.181(.02)	.177(.007)	.167(.005)	.186(.026)	.162(.010)
6	.185(.017)	.171(.003)	.169(.001)	.164(.005)	.168(.001)
7	.144(.003)	.152(.005)	.197(.08)	.239(.09)	.162(.026)

Table 5.4. Average throughput in Kbytes/sec with standard deviation based on time of day (EST)

Path #	9:00AM	11:00AM	1:00PM	3:00PM	5:00PM
1	228.4(42.6)	192.6(57.4)	241.9(63.7)	113.8(65.3)	148.6(90.4)
2	48.7(18.9)	41.5(20.5)	42.4(18.7)	52.8(15.4)	46.7(27.5)
3	11.2(7.4)	14.3(4.8)	16.5(7.2)	15.7(8.7)	5.9(7.4)
4	22.2(10.3)	27.4(8.4)	17.1(5.2)	22.4(7.1)	28.9(8.3)
5	84.2(13.7)	73.4(22.4)	77(15.1)	47.8(24.6)	76.1(22.5)
6	45.2(6.2)	48.7(2.2)	47.5(2.7)	40.1(11.7)	49.7(1.1)
7	17.7(8.7)	11.5(5.7)	7.7(5.3)	5.8(4.2)	11.2(8)

Table 5.5. Average TCP loss rates (%) with % standard deviation based on time of day (EST)

Path #	9:00AM	11:00AM	1:00PM	3:00PM	5:00PM
1	.2(.05)	.5(.45)	.27(.08)	1.65(1.57)	1.5(1.4)
2	.9(.6)	1.6(1.1)	1.4(1.1)	.77(.4)	3.7(7.1)
3	5.6(3.8)	3.1(2.3)	2.8(3.5)	4.9(7.5)	15.8(8.4)
4	1.2(1.9)	.5(.5)	1.4(.8)	.9(1)	.4(.7)
5	.19(.16)	.5(.7)	.4(.2)	1.6(1.1)	.5(.47)
6	.2(.3)	.18(.2)	.4(.38)	1.7(2.1)	.2(.15)
7	7.4(2.7)	9.7(3.1)	12(3.9)	12.7(4.3)	10.4(5.2)

Before we present the metric results based on the aggregated data, we introduce the analysis approach by looking at several individual runs. We begin with one of the more typical runs over the Emory path (i.e., path 1). The observed packet loss probability was 1%. Figure 5-1 illustrates 20 seconds of the run and Figure 5-2 shows more detail by focusing on the interval between time 80 and 82 seconds. The top curve plots the *tcpRTT* time series (the "+" marks forming the curve are the samples). The "+" marks along the top border of the graph identifies the *tcpRTT* samples that occur just prior to the transmission of a segment that will be dropped somewhere along the path. Figure 5-2 illustrates a loss that occurs sometime after time 82.65 seconds. The lower curve plots the TCP "goodput" (i.e., the rate that data is acknowledged) using an interval of .5 seconds (the dark line) and 2 seconds (the dashed line).

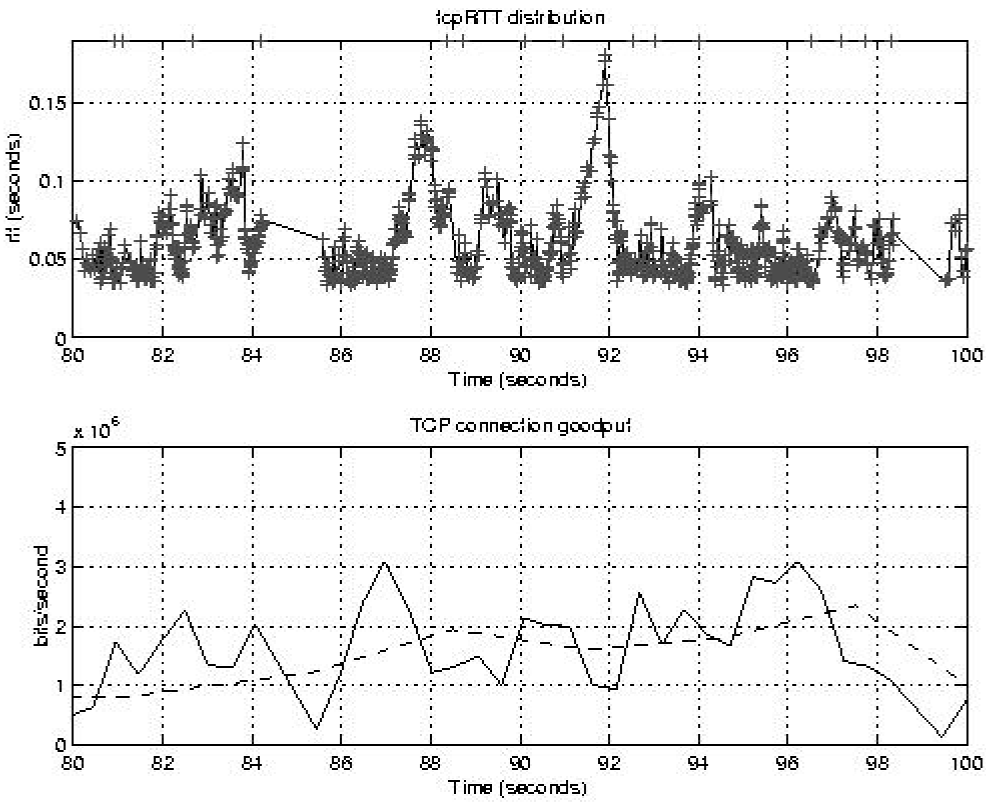

Figure 5-1. Path 1 single run dynamics

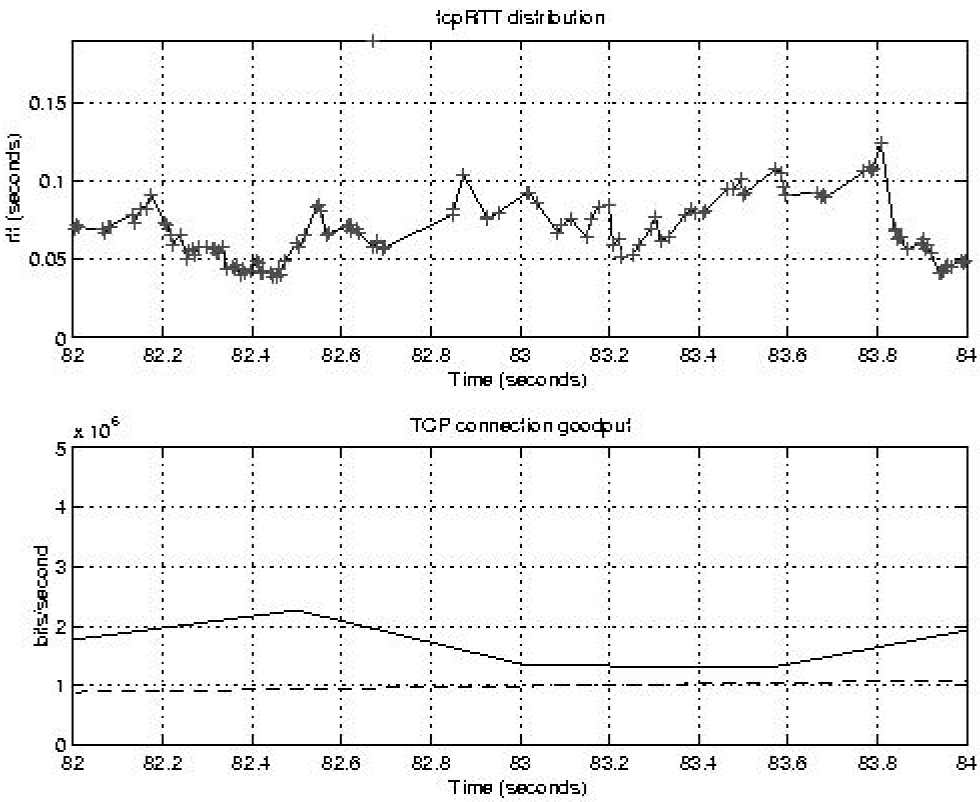

Figure 5-2.   Details of the path 1 single run dynamics

The correlation indication data associated with the run was:

P[SampledRTT(2) > windowAVG(5)] = .42

P[SampledRTT(2) > windowAVG(20)] = .39

P[sampledRTT(2) > rttAVG] = .41

The metric indicates that loss is preceded by an increase in the *tcpRTT* samples 40% of the time. The loss conditioned delay correlation metric shown in Figure 5-3 confirms that for this run there is essentially no correlation between an increase in RTT and packet loss. The loss correlated delay CDF metric illustrated in Figure 5-4 shows that *sampledRTT* and *tcpRTT* distributions are essentially identical.

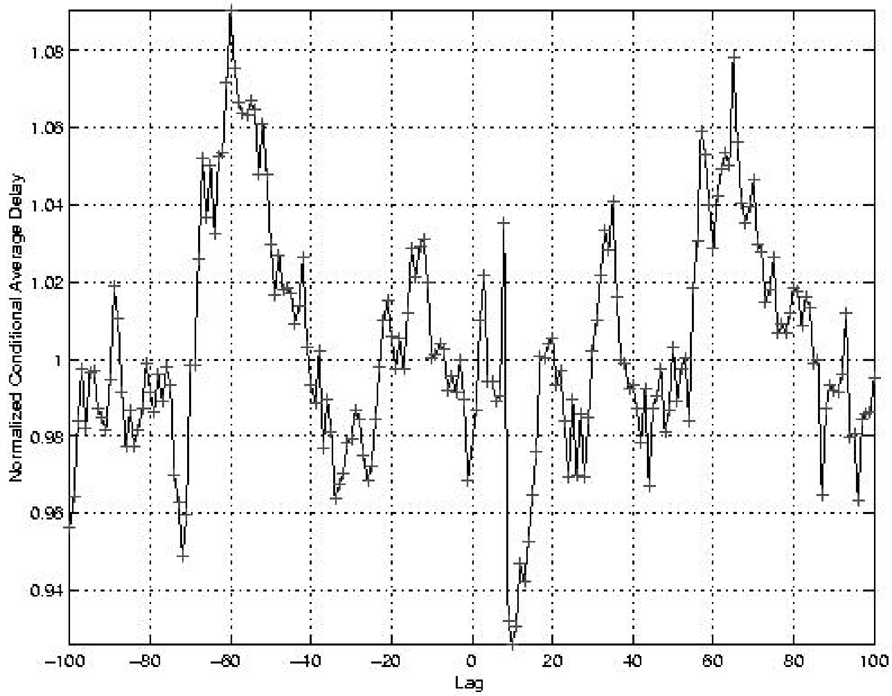

Figure 5-3. Loss conditioned delay correlation delay for path 1single run

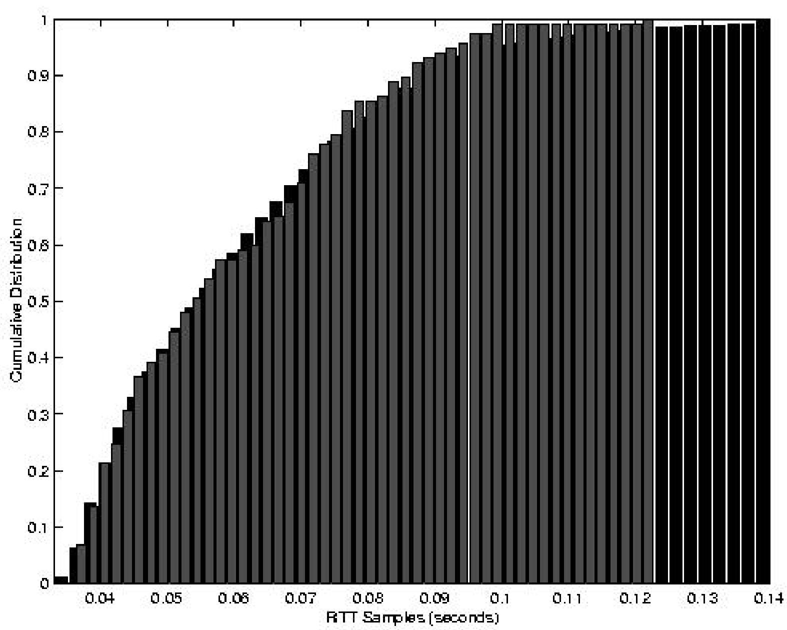

Figure 5-4. Loss correlated delay CDF for path 1 single run

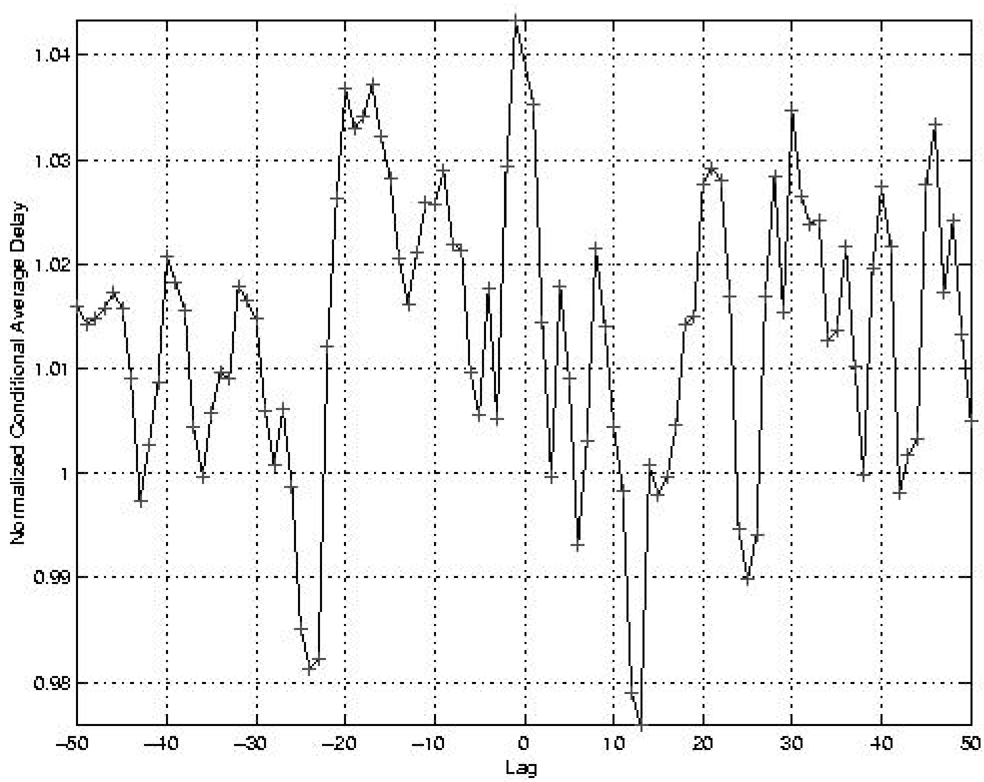

Figure 5-5. Loss conditioned correlation delay for path 3 single run

Path 1 represents a well behaved high speed Internet path. For the run represented by Figures 5-1 through 5-4, the level of correlation between an increase in the *sampledRTT* and packet loss is weak. Other runs over the path exhibit slightly higher levels of correlation. Several of the other paths show more correlation than path 1 (we will see this in the next section when we present the statistical results). For example, Figure 5-5 illustrates the loss conditioned correlation metric for a run over path 3 which shows a distinct peak in the lag –1 loss conditioned delay value.

We next show an example run from a much more congested path (path 7). We chose path 7 because it crosses the MAE-WEST Network Access Point (NAP) which is frequently heavily congested. While several years ago, a path exhibiting this level of performance was more common, it is the exception today. However, we wanted to evaluate the path to see if the dynamics are conducive to DCA. We expected the answer to be negative and that is what the results show. The correlation indication values associated with a run over path 7 are:

P[sampledRTT(2) > windowAVG(5)] = .35

P[sampledRTT(2) > windowAVG(20)] = .32

P[sampledRTT(2) > rttAVG] = .23

The correlation indication values are surprisingly low. Looking at the *tcpRTT* time series associated with the run (as plotted in the top curve of Figure 5-6), we see almost random *tcpRTT* fluctuations that clearly reflect very brief time scale congestion. It turns out that the router located at the MAE NAP suffers from sustained congestion (if we ping www.eas.asu.edu in the middle of the night we see an average delay on the order of 70 milliseconds). We assume that the RTT fluctuation being detected by the *tcpRTT* samples reflects bursty congestion associated with packet loss and is generally of a time scale smaller than 1 RTT.

Figure 5-7 illustrates the loss conditioned delay correlation metric for the run and Figure 5-8 illustrates the loss conditioned CDF for the run. The loss conditioned delay correlation plot (Figure 5-7) supports the correlation indication results indicating that there is generally not an increase in RTT in the lags prior to loss. However the metric detects a significant increase at lag +1 (i.e., immediately after loss). This actually helps explain the low correlation indication data values that we find. The *tcpRTT* samples closer to the time when loss occurs will tend to reflect the queue delay that is associated with the loss. And likewise, the *tcpRTT* samples generated 1 RTT prior to loss (which are the *sampledRTT* samples used by the congestion indication metric) will not detect the increase and are more likely to be lower than the average *tcpRTT* value. To provide further insight into the analysis, we conclude this section with a more detailed view of the dynamics associated with the *tcpRTT* samples and loss events.

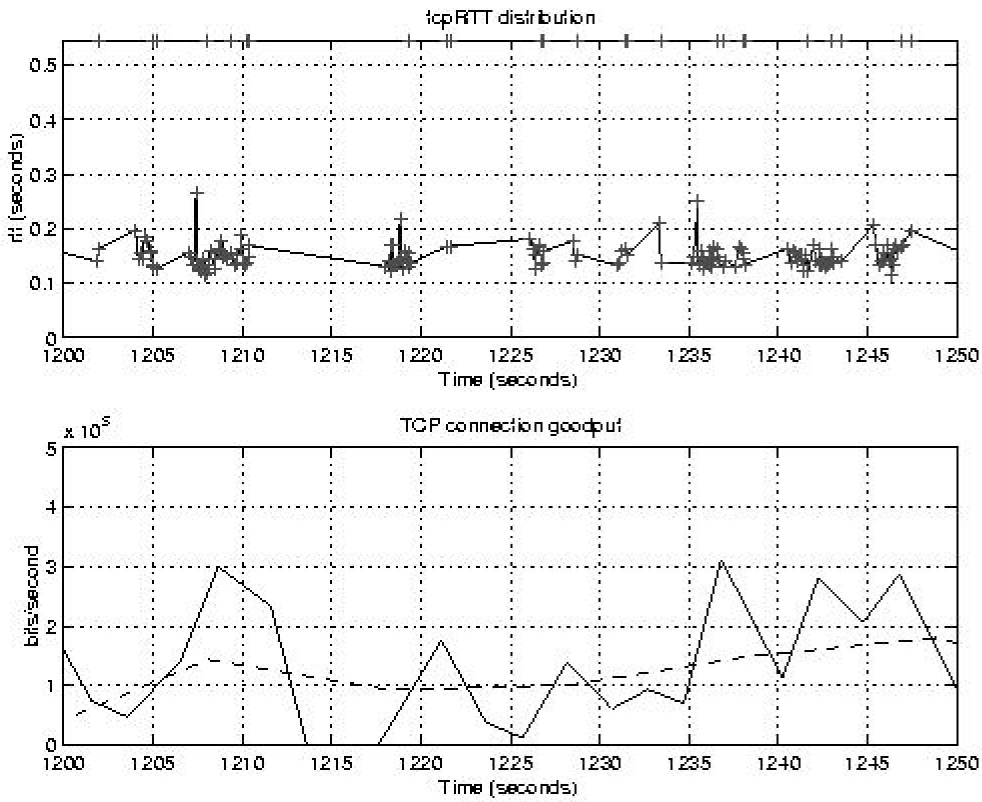

Figure 5-6. Performance of a single run over the ASU path

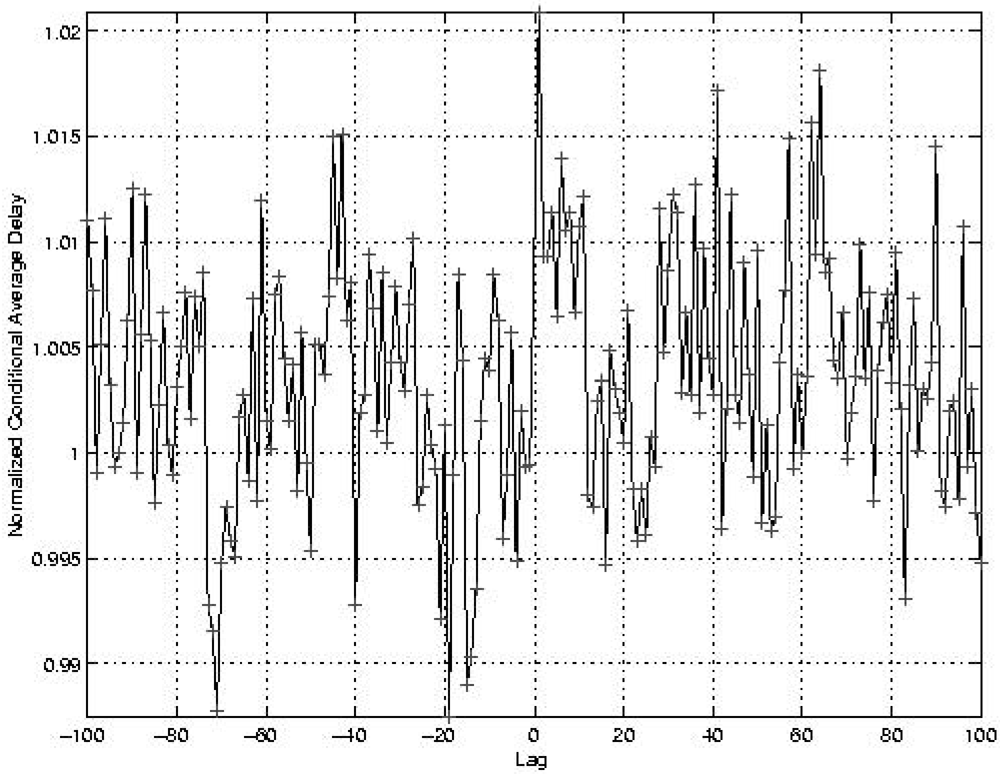

Figure 5-7. Loss conditioned correlation delay over ASU run

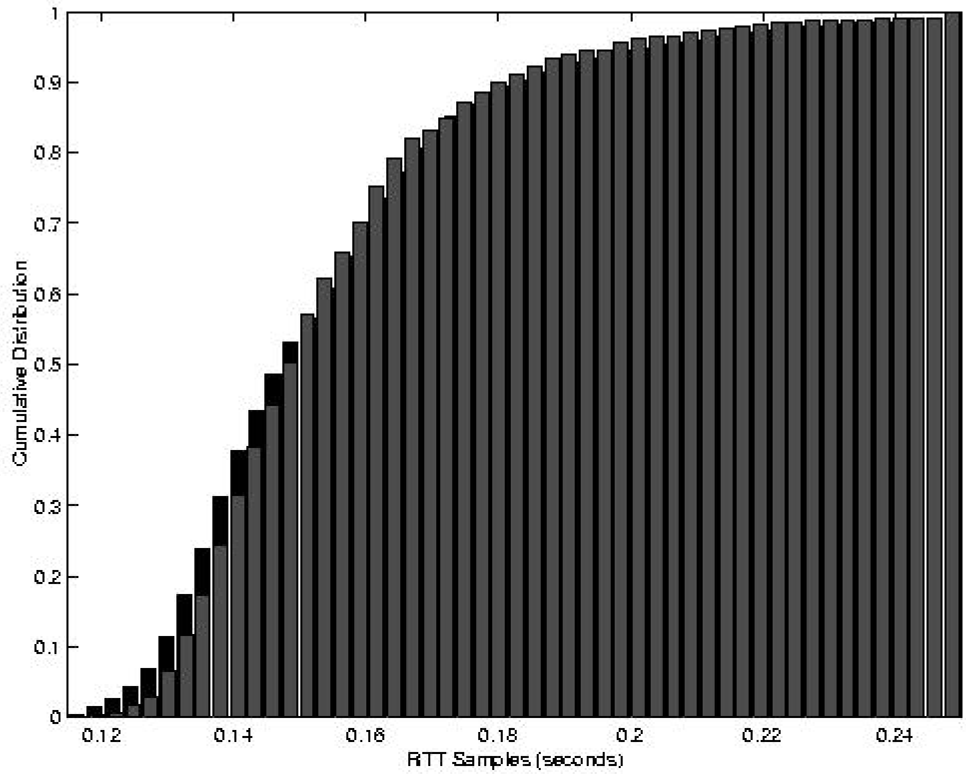

Figure 5-8. Loss conditioned CDF over ASU run

We first show in greater detail an example where the *tcpRTT* samples prior to loss increase early enough so that a DCA algorithm can potentially predict the loss based on an observed increase in *tcpRTT*. Figure 5-9 provides more details of the dynamics associated with the ASU run by plotting the *tcpRTT* curve along with the sequence/acknowledge plot for a 3 second time period. The lower curve shows data packet departures (indicated with a "+" in the lower curve) and acknowledgement arrivals (indicated with a "O"). TCP tends to send packets in bursts and can be modeled as an on/off source where it sends a number of packets (constrained by the window) followed by an "off" time that is on the order of a RTT. This behavior is evident in Figure 5-9. The upper curve shows that the *tcpRTT* samples are generated roughly every RTT such that each sample corresponds to a burst by the sender. If the time scale associated with the queue buildup that precedes loss is large (notice the slow buildup between time 162.5 and 163.5 in Figure 5-9), then the *tcpRTT* sample prior to loss (at time 163.1 in Figure 5-9) might reflect the additional delay.

The next example shows the more common case where the increase in RTT is detected to late too avoid the packet loss. Figure 5-10 illustrates two graphs corresponding to the dynamics associated with the Emory run pictured in Figures 5-1 through 5-4. The top graph plots the *tcpRTT* time series and the lower graph plots the packet departures (i.e., the “+” signs) and acknowledgement arrivals (i.e., the “o” signs). The lower curve reflects the TCP “on/off” behavior discussed earlier such that the *tcpRTT* samples tend to be grouped in clumps separated by a RTT. At time 81.075 seconds, the lower curve shows three ACKs arriving, each leading to two additional segments being sent. Essentially we see a burst of 6 segments (for the discussion, we will identify these as segments 1 through 6). With each ACK arrival, we obtain a *tcpRTT* sample that reflects an increase in RTT (an increase from 36 milliseconds to about 45 milliseconds). The sequence/acknowledgement graph shows that segment 6 gets dropped (we see the duplicate acknowledgements begin to arrive at time 81.2 followed by the retransmission at 81.22 seconds). The two *tcpRTT* samples (at time 81.14 seconds) correspond to segments 2 and 4 respectively. Because these segments travel through the congested router at roughly the same time as segment 6 (i.e., the segment that gets dropped), it is reasonable to expect that the two *tcpRTT* samples are larger because they reflect the additional queueing delay associated with the loss.

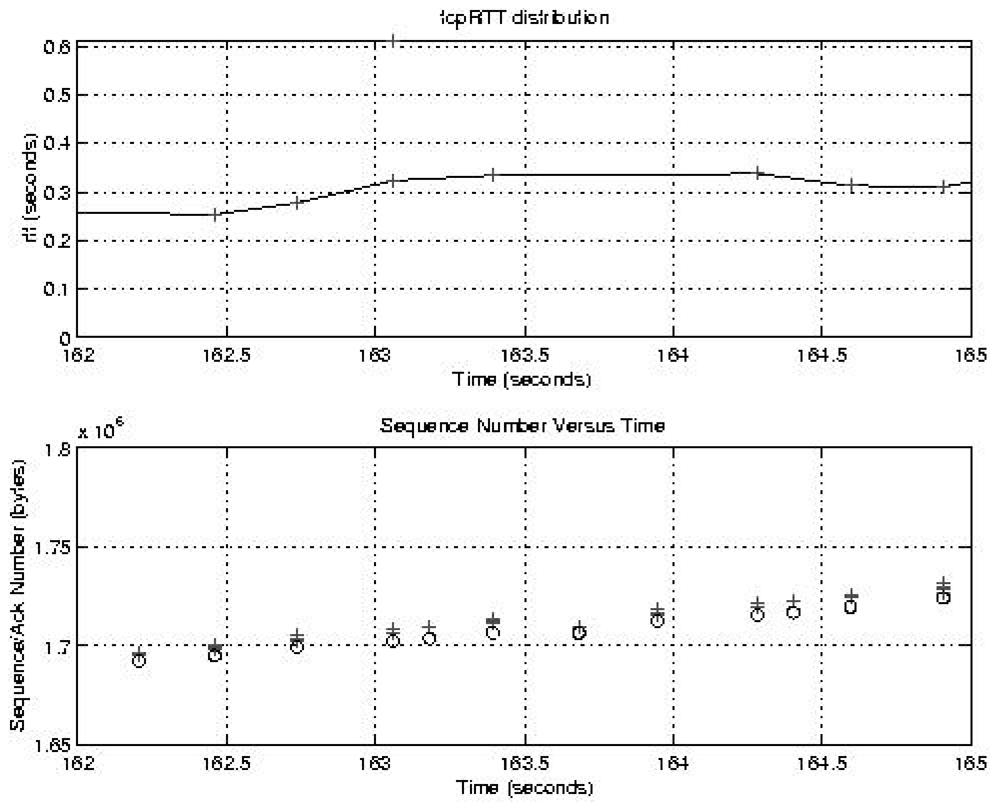

Figure 5-9. Dynamics of single run over path 3

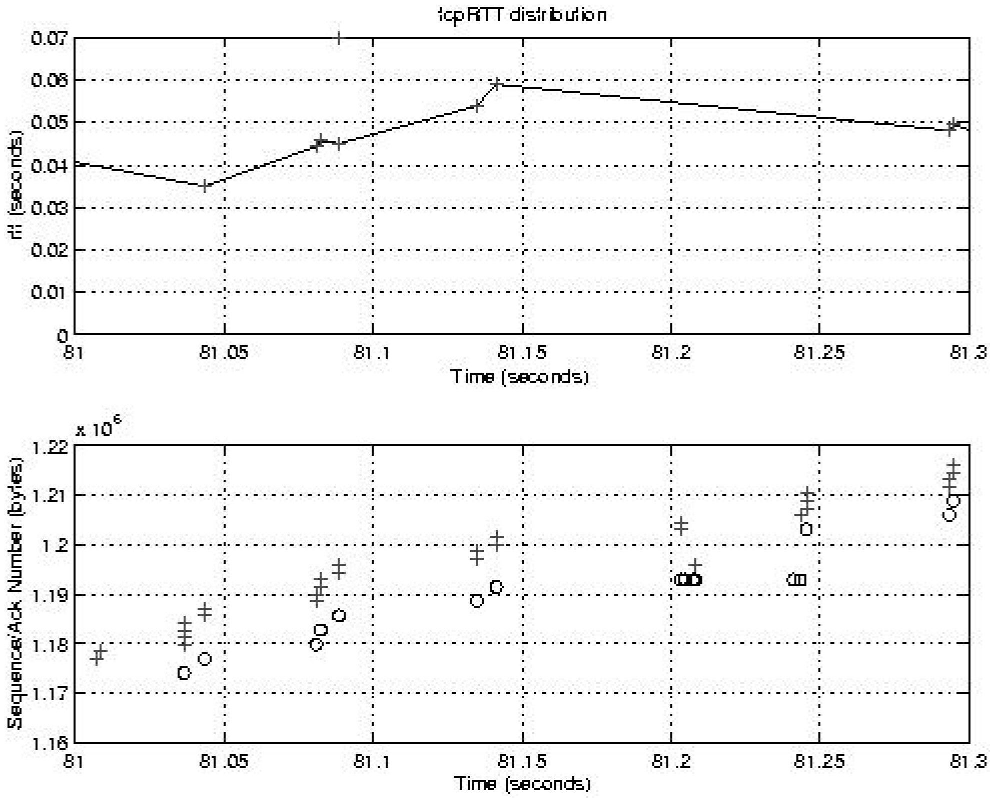

Figure 5-10. Dynamics of single run over path 1

Table 5-6 provides the results of additional analysis that we perform based on the measured data. The goal is to derive an understanding of the congestion dynamics that are active at a congested router within the network. From the *tcpRTT* plot in Figure 5-10, we observe that the RTT increases from .035 seconds to .06 seconds in roughly .1 seconds. If we assume that the link speed associated with the output queue is 10 mbps, this translates to an increase in queue size on the order of 31250 bytes as a result of a traffic arrival rate on the order of 2.5Mbps (i.e., 31250 bytes arriving in .1 seconds). If we assume the bottleneck link capacity is 45 mbps, we see the corresponding queue increase and traffic arrival intensity is much greater. It is clearly not possible to say for certain which of the two scenarios are likely. If we could identify the location of the congestion (i.e., identify the router along the path), a tool such as *pathchar* might be able to indicate if the bottleneck link is closer to the 10 mbps or 45 mbps (or some thing else). We explore this in more detail in Part 2 of this dissertation when we develop simulation models based on the measured data.

Table 5-6. Hypothetical queue dynamics at the bottleneck router

Link Speed	Queue increase (Bytes)	Rate of Queue Growth(bps)
10 mbps	31250	2.5Mbps
45 mbps	140,625	11.25Mbps

5.2 Statistical Analysis

In this section, we show the results of the three metrics applied to the measured data. As described in the previous section, we traced multiple TCP connections over 7 paths over the course of 5 days (5 runs per day for a total of 25 runs per path). The duration of each run ranged from several minutes up to an hour depending on the congestion level). The results are the basis for our claim that an increase in the *tcpRTT* samples prior to loss is not strongly correlated to loss events. We rely on the metric results from the *ns* simulation run to define the baseline for a "strong" level of correlation (refer to section 4.2.4). The results from the analysis in this section indicates that although packet loss tends to occur when the RTT is above some threshold, the level of correlation between uniquely identifiable increases in RTT and packet loss is weak. We will show in the next section that DCA is unable to improve TCP throughput because of this result.

5.2.1 Correlation Indication Metric

For each run, we compute the correlation indication metric values. Then, for each path, we find the mean and standard deviation using the metric values from the 25 runs. Table 5-7 illustrates the results. Each row of the table corresponds to the paths identified in Table 5-1. The table reflects three different runs of the metric each using different (*x,w*) combinations. The *P[sampledRTT(x)>rttAVG]* values (which is not shown in the table) is similar to the P*[sampledRTT(x)>windowAVG(w)]* values (i.e., in the 30-50% range) which implies that the *tcpRTT* samples prior to loss is actually more likely to be lower than the average of all *tcpRTT* samples (i.e., the *rttAVG*). This result is further supported by the loss conditioned delay CDF metric results which we present in a subsequent section.

We have added several additional variations of the correlation indication metric. We define the event *[sampledRTT(x)>windowAVG(w) + std]* to signify when the *sampledRTT(x)* prior to loss is greater than the moving window average by a threshold amount (the standard deviation associated with the *windowAVG(w)*). The purpose of this event is to see how effectively a filtered congestion detection decision can predict future packet loss events. The results of this are shown in the fourth and fifth columns corresponding to the (2,20) and (8,20) runs respectively. Using the (2,20) run of path 1 as an example (i.e., the third column), the data indicates that loss is preceded by an increase in the *tcpRTT* samples 40% of the time. Furthermore the amount of increase was greater than one deviation over the moving window average (i.e., the fourth column) 17% of the time. The results indicate that path 6 exhibits the highest level of correlation although it also had the highest variation. If we increase the threshold to twice the standard deviation, no loss events are detected (this result is not shown in the table).

Table 5-7 Aggregate Correlation Indication Metric Results: mean(standard deviaton)

Path #	**P[sampledRTT (2)> windowAVG(2)]**	**P[sampledRTT(2) > windowAVG(20)]**	**P[sampledRTT(2)> windowAVG(20) + std]**	**P[sampledRTT(8) > window(20)AVG + std]**
1	.42(.07)	.40(.06)	.17(.06)	.02(.02)
2	.43(.07)	.38(.07)	.14(.04)	.003(.007)
3	.37(.12)	.37(.12)	.11(.09)	.006(.014)
4	.32(.13)	.28(.09)	.11(.08)	.0004(.002)
5	.45(.16)	.42(.15)	.18(.11)	.004(.011)
6	.50(.17)	.48(.18)	.13(.17)	.005(.021)
7	.30(.05)	.27(.05)	.07(.018)	.000(.000)

Table 5-7 also shows the metric when the size of the moving window associated with the *sampledRTT(x)* samples are increased to 8 *tcpRTT* samples (i.e., the last column). The idea is to see if the level of correlation increases as the moving window associated with the estimate of the “instantaneous“ RTT level increases. Effectively this filters the sampling process making it less responsive to short term RTT variations. The effect is significant (the fifth column in the table shows that very few of the loss events could be predicted) and suggests that the time scale associated with an increase in RTT prior to packet loss is on the order of several *tcpRTT* samples (or less). We also ran the algorithm using other combinations of window sizes (e.g., using *x* and *w* combinations of (2,200) and (1,20)). A *w* value of 20 samples limits the assessment of the RTT average to the most current which is sufficient to allow the algorithm to detect increases quickly. As the *w* value increases, the standard deviation tends to increase which decreases the number of losses that are detected. In other words, because the dynamics associated with the *tcpRTT* time series varies significantly over time, an average based on a short history of *tcpRTT* values provides the most useful information for a loss prediction algorithm. A common pattern for all paths is that a small *sampledRTT* window (of 2) along with a *windowAVG* window of 20 had the highest *P[sampledRTT(x)> windowAVG(20) + std]* values.

From Table 5-7, we see that a TCP constrained per-packet probe generally can detect the queue buildup that precedes packet loss only 30-50% of the time. More significantly, if the congestion detection algorithm attempts to filter out minor RTT increases from larger increases (where a large increase is defined to be when: *sampledRTT(x)>windowAVG(w) + std*), only 7-18% of packet loss events would have had a chance of being detected and avoided. In terms of DCA algorithms, an algorithm that reacts whenever the condition *sampledRTT(x)>windowAVG(w)* is true will react very frequently to the variations in *tcpRTT*. If we modify the congestion decision by reacting only when the *sampledRTT(x)* exceeds a threshold (i.e., a standard deviation over the moving window average) the algorithm will not react as frequently although it will reduce the number of times that it is able to avoid a loss. We address the impact of frequent DCA congestion reactions to TCP throughput in the next section.

Table 5.8. P[*sampledRTT(2) > windowAVGRTT(20) + std*] with standard deviation

Path #	9:00AM	11:00AM	1:00PM	3:00PM	5:00PM
1	.186(.1)	.2(.036)	.137(.068)	.162(.034)	.149(.022)
2	.19(.044)	.137(.04)	.135(.028)	.15(.022)	.107(.039)
3	.065(.047)	.142(.064)	.14(.058)	.14(.14)	.08(.1)
4	.34(.06)	.11(.06)	.1(.017)	.132(.165)	.099(.07)
5	.21(.08)	.16(.07)	.21(.16)	.128(.022)	.19(.17)
6	.34(.29)	.1(.06)	.047(.036)	.054(.035)	.097(.077)
7	.081(.011)	.072(.015)	.073(.022)	.063(2.0)	.068(.023)

Table 5-8 shows the results of the P[*sampledRTT(2)>windowAVG(20)+std]* metric for each path based on runs grouped by time of day. We see high variability in the metric values over each path. Path 4 and path 6 both show extremely high values at the 9:00 AM run. We will see later in this chapter that even if DCA is able to avoid a large percentage of loss, the connection will still suffer from degraded throughput.

The results from Tables 5-7 and 5-8 assume that the reference point between a loss event and an RTT measurement begins at the *tcpRTT* sample that precedes the transmission of the segment that is dropped. We ran the correlation indication metric assuming a reference point further in the future (i.e., at the *tcpRTT* samples that precede the retransmission of the segment that was dropped). Table 5-9 illustrates that the level of correlation increases significantly. The reason is as we discussed in the previous section. TCP's burstiness makes the RTT-based probing process very granular. Because the queue buildup associated with packet loss at high speed switches is likely to be quite fast, a congestion probe (or a set of back-to-back probes) with a frequency of 1 RTT will likely not detect the bursty congestion that precedes the loss. Typically not all the packets in the burst will be dropped which means that there is a chance that a *tcpRTT* sample might be generated from a segment that survives the queue overflow will likely reflect the queueing delay associated with the packet loss. Based on Table 5-9, we see that when we move the reference point, 50-70% of all loss events observed are preceded by an increase in RTT. As an example, for path 1, when the reference point is considered to be the time of the initial transmission, we found that 40% of loss events were preceded by an increase in RTT (for the 2,20 moving average window sizes), Table 5-9 shows that when the reference point moves, 52% loss events are detected (i.e., column 2). We deduce from this data that 12% of the loss events associated with the path 1 traces are preceded by queue buildup that occurs in less than one round trip time.

Based on the results from the correlation indication metric, we observe that the time scale associated with the RTT increase that precedes packet loss can be very brief (i.e., on the order of 1-2 RTT's or less). The following pieces of evidence support this.[2]

- Only 30-50% of loss events observed in the traces are preceded by an increase in the *tcpRTT* sampling process.
- The filtered prediction algorithm is more effective when the congestion decision is based on a *sampledRTT* value that consists of 2 *tcpRTT* samples rather than 8.
- When the reference point is moved forward in time from the time of the initial transmission of a segment that is dropped to the time it is retransmitted, we see a 10-20% increase in the number of loss events that are detected.

Table 5-9 Correlation Indication when the referenced event is the retransmission : mean (standard deviation)

Path #	P[sampledRTT(2) > window(5)AVG]	P[sampledRTT (2)> window(20)AVG]	P[sampledRTT (2)> window(20) AVG + std]
1	.57(.08)	.52(.09)	.26(.06)
2	.52(.09)	.46(.08)	.20(.08)
3	.62(.08)	.61(.11)	.20(.14)
4	.57(.14)	.47(.15)	.24(.13)
5	.71(.15)	.66(.18)	.35(.23)
6	.60(.19)	.66(.16)	.24(.21)
7	.50(.03)	.45(.03)	.12(.02)

5.2.2 Loss Conditioned Delay Correlation (LCDC) Metric

We concatenate the 25 *tcpRTT* times series associated with each path and run the loss conditioned delay correlation algorithm on this data. The reference point for the metric (i.e., lag 0) is the time when the segment that is dropped is first transmitted. Most of the paths (except for paths 1 and 6) show that the loss

[2] The coarseness of a TCP-constrained congestion probe is also a factor. However, we believe that the results do reflect a tendency for the increase in RTT that precedes loss to occur very quickly.

conditioned delay correlation appears inflated over many lags centered around the loss event. For example Figure 5-14 reflects a significant increase in RTT over both long and very short time scales before and after packet loss. This indicates that packet loss occurs more frequently at higher RTT's. In the measured data, each path typically had several runs that exhibited much higher *tcpRTT* samples and loss rates. The metric can be misleading because the results can be skewed by a few very large RTT samples. In spite of this, the correlated data does provide an interesting visualization of the magnitude and time scale associated with increases in RTT surrounding loss events.

The two most significant results we derive from the data are:

- The time scale of the increase in RTT that is associated with loss is on the order of –1 to –5 lags (i.e., 1 to 5 *tcpRTT* samples prior to the transmission of a packet that gets dropped). This confirms the results from the correlation indication metric that suggests that the time scale associated with queue buildup that precedes loss is very short.
- Most paths reflect the tendency for there to be a larger increase in RTT following loss (rather than prior to loss).

The LCDC results confirms that there is a weak level of correlation between an increase in RTT and loss. However the results suggest that there are complications that will negatively impact a DCA algorithm. For example, the results associated with path 1 and 6 (Figures 5-11 and Figure 5-16) suggest that the increase in RTT prior to loss is difficult to detect. This is because the increase in RTT prior to lag 0 is not unique (as it is in the correlated benchmark simulation run illustrated in Figure 4-11). The results associated with paths 2, 3, 4, 5 and 7 suggest that the correlation is actually stronger after lag 0 which implies that the *tcpRTT* samples that are generated after the dropped packet is first transmitted reflects a higher increase in RTT. The results in Figure 5-16a (5-16b expands the view) indicate that there is a significant drop in correlation after the loss. Figure 5-16b shows that the loss conditioned delay value at lag 3 (i.e., the third *tcpRTT* sample that is generated after the dropped segment gets transmitted) drops significantly. The delay returns to about the previous level and then it drops again between lags 9 – 14. This appears to be the result of some sort of synchronized behavior, however it is not possible to explain the cause.

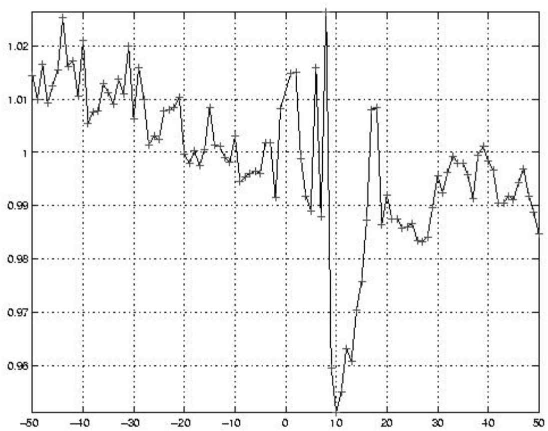

Figure 5-11. Loss conditioned delay correlation metric for path 1

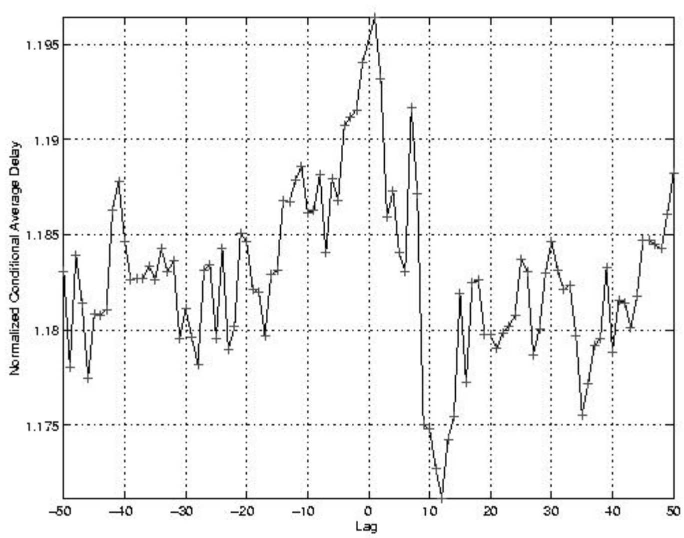

Figure 5-12. Loss conditioned delay correlation metric for path 2

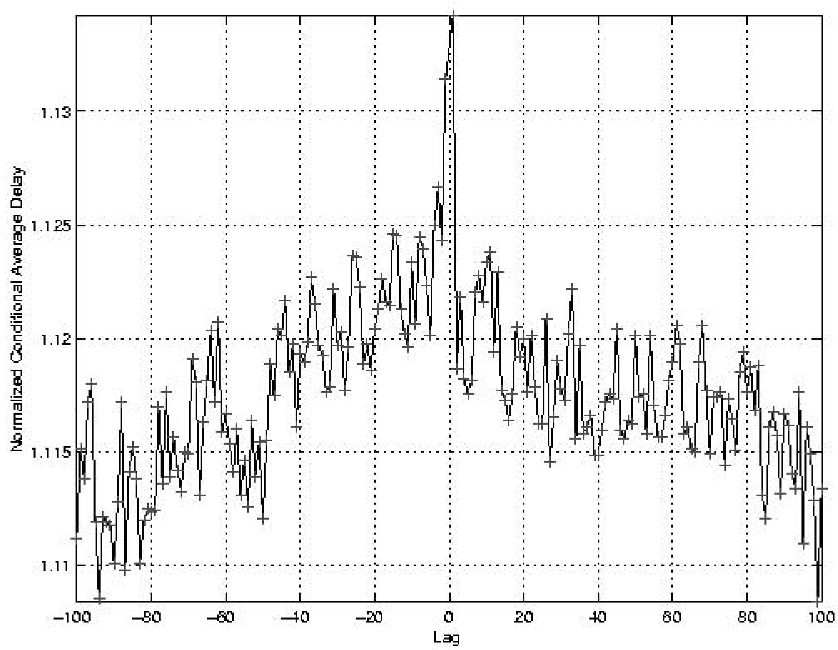

Figure 5-13. Loss conditioned delay correlation metric for path 3

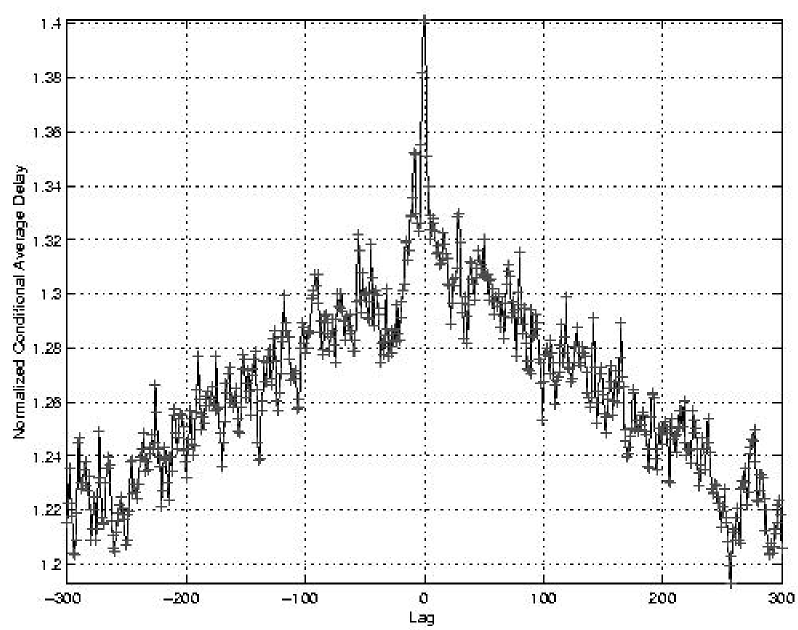

Figure 5-14. Loss conditioned delay correlation metric for path 4

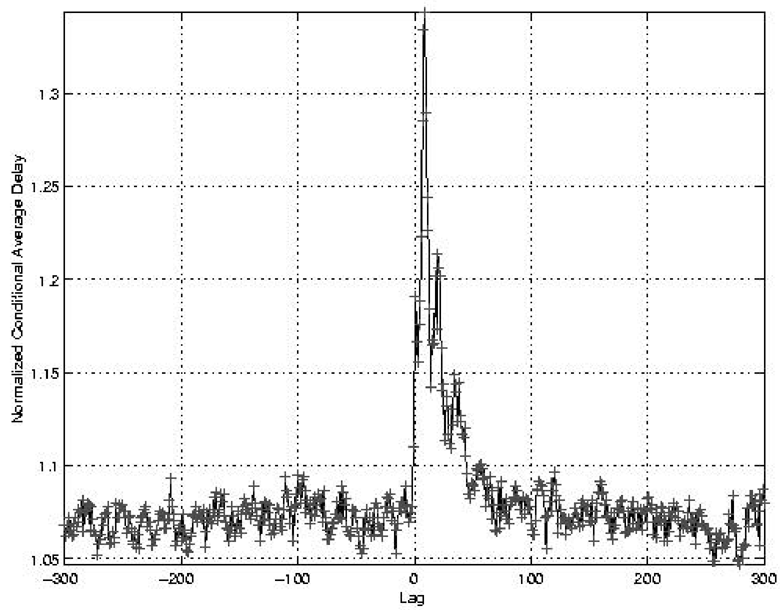

Figure 5-15. Loss conditioned delay correlation metric for path 5

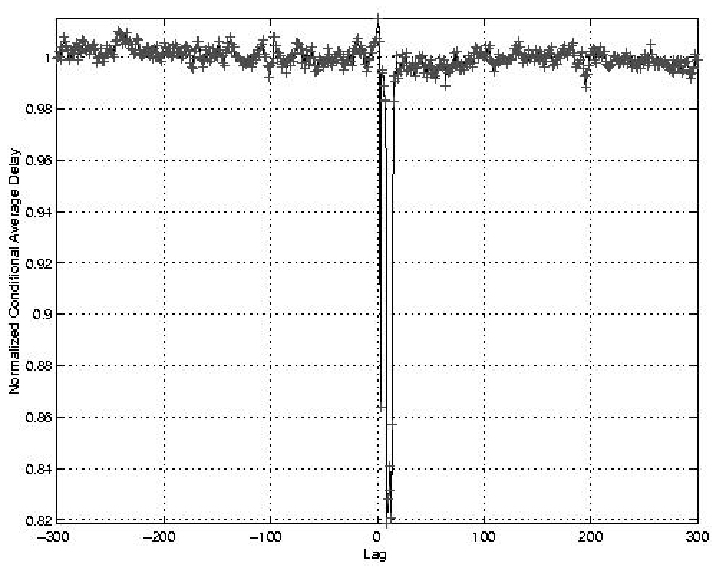

Figure 5-16a. Loss conditioned delay correlation metric for path 6

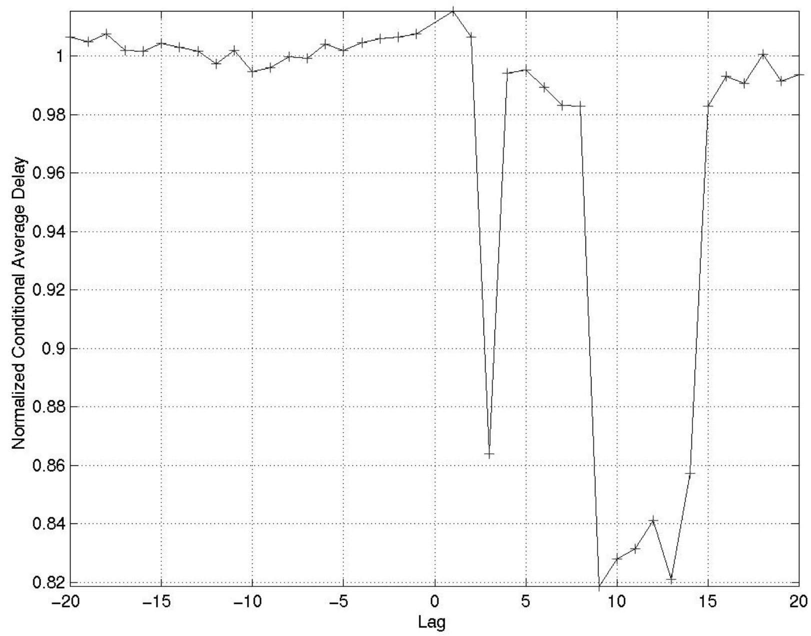

Figure 5-16b. Closer view of the metric for path 6

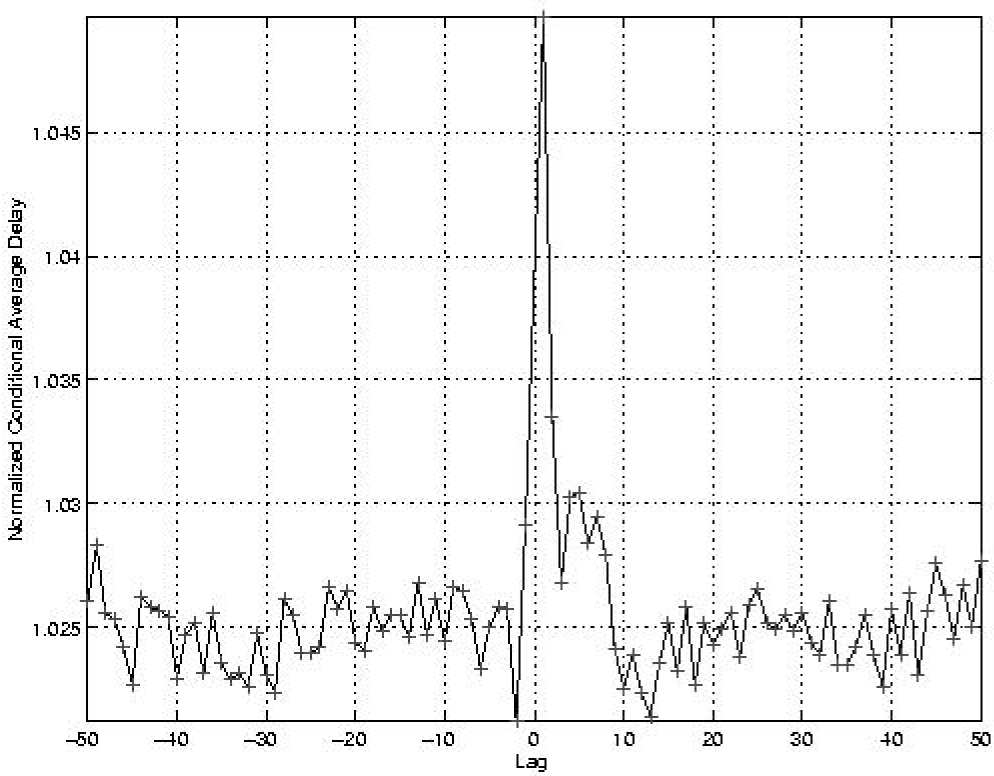

Figure 5-17. Loss conditioned delay correlation metric for path 7

5.2.3 Loss Conditioned Delay CDF

Using the concatenated *tcpRTT* time series of all runs associated with a path, we apply the loss conditioned delay CDF metric. Figures 5-18 through 5-25 illustrates the results. To illustrate the relationship between the PDF's of the two distributions, Figure 5-19 shows the PDFs of the *tcpRTT* and *sampledRTT* time series associated with the aggregate data for path 1. The results of the metric can be summarized as follows:

- Most paths exhibit a threshold RTT value (i.e., *thresholdRTT*) below which no loss occurs.
- Most *tcpRTT* values are greater than the *thresholdRTT* which indicates that increases in RTT occurs very frequently.

- Above the *thresholdRTT* the distributions of the *tcpRTT* and the *sampledRTT* appear very similar (with the exception of paths 2 and 3). The implication is that loss does not tend to occur more frequently for higher values of *tcpRTT*. From each CDF curve, we estimate the probability *P[tcpRTT<sampledRTTAVG]*. A value of .5 indicates that half of the *tcpRTT* values were lower than the average RTT value that precedes loss events. As the probability approaches 1, this indicates that the RTT increase prior to loss tends to be large. The correlated simulation run from section 4.2.4 (i.e., our baseline) shows a value of .75. Based on Figures 5-18 through 5-25, the probability values for each path are:
  - Path 1, *P[tcpRTT<sampledRTTAVG]* = .5
  - Path 2, *P[tcpRTT<sampledRTTAVG]* = .75
  - Path 3, *P[tcpRTT<sampledRTTAVG]* = .7
  - Path 4, *P[tcpRTT<sampledRTTAVG]* = .58
  - Path 5, *P[tcpRTT<sampledRTTAVG]* = .6
  - Path 6, *P[tcpRTT<sampledRTTAVG]* = .55
  - Path 7, *P[tcpRTT<sampledRTTAVG]* = .54

Rather than a measurement of the correlation between increases in RTT and loss events, the loss conditioned delay CDF metric is really an indicator of the uniqueness of the increase in RTT that precedes loss. The *sampledRTT* distributions illustrated in Figures 5-18 through 5-25 clearly show that loss tends to be accompanied by an increase in RTT. However, the metric results for some paths (path 1, 6 and 7) indicate that loss occurs uniformly spread over the *tcpRTT* distribution. Other paths (path 2 and 3) show that there is a slight tendency for loss to occur with larger *tcpRTT* values.

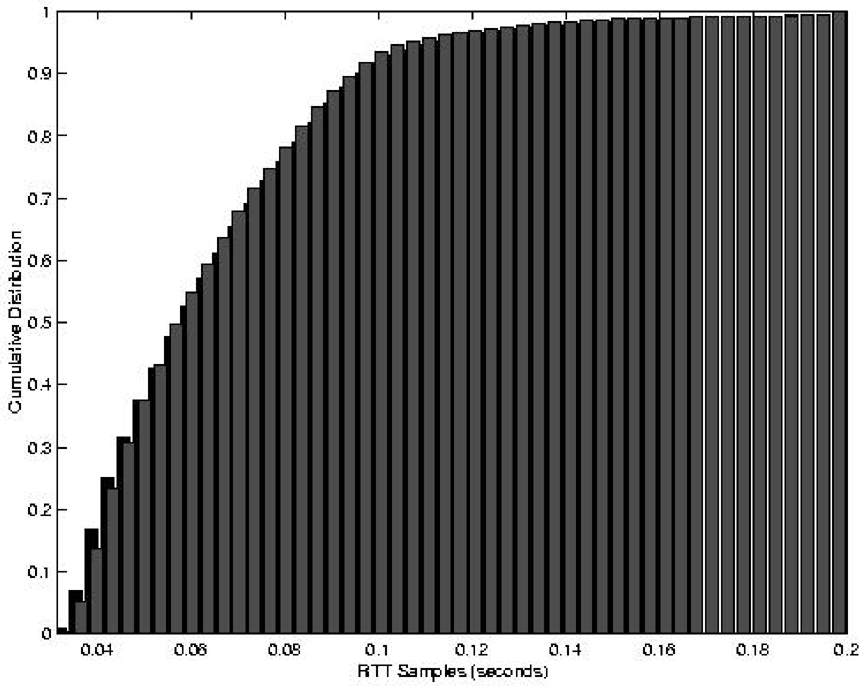

Figure 5-18 Loss Conditioned Delay CDF for Path 1

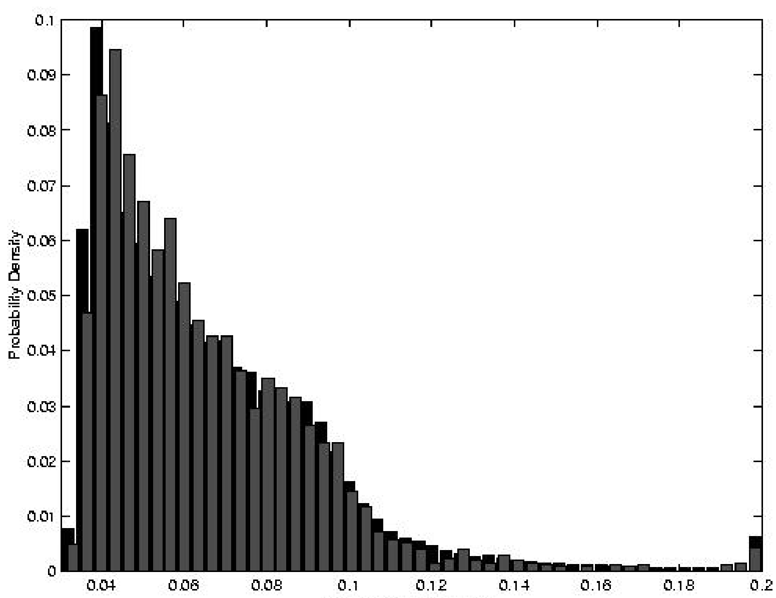

Figure 5-19 Loss Conditioned Delay PDF for path 1

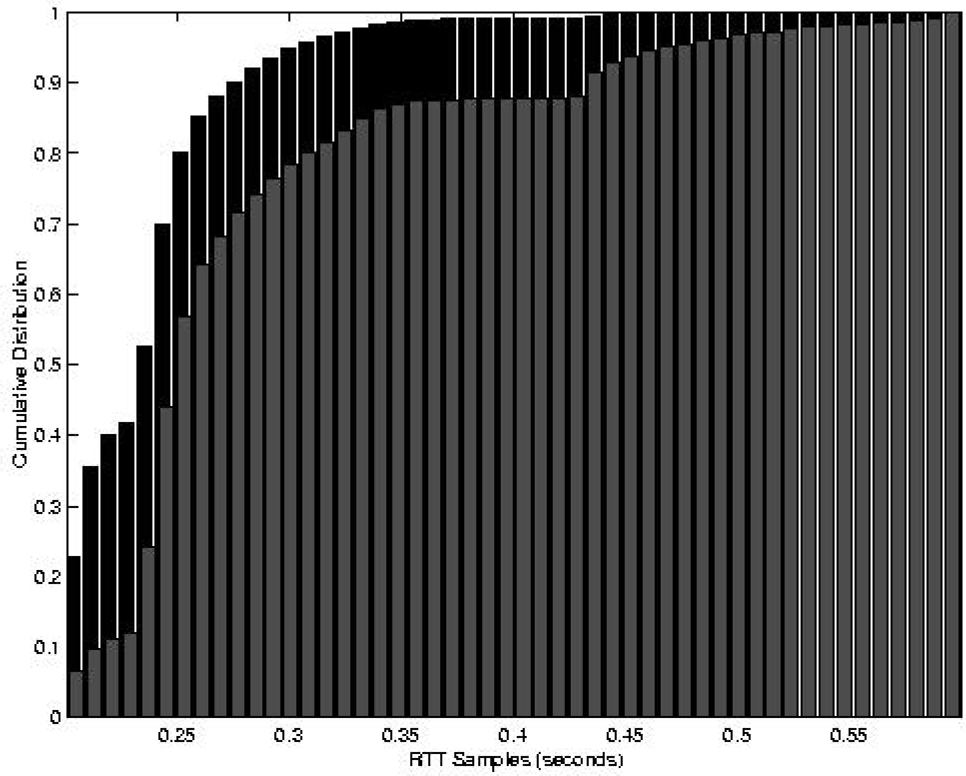

Figure 5-20 Loss Conditioned Delay CDF for Path 2

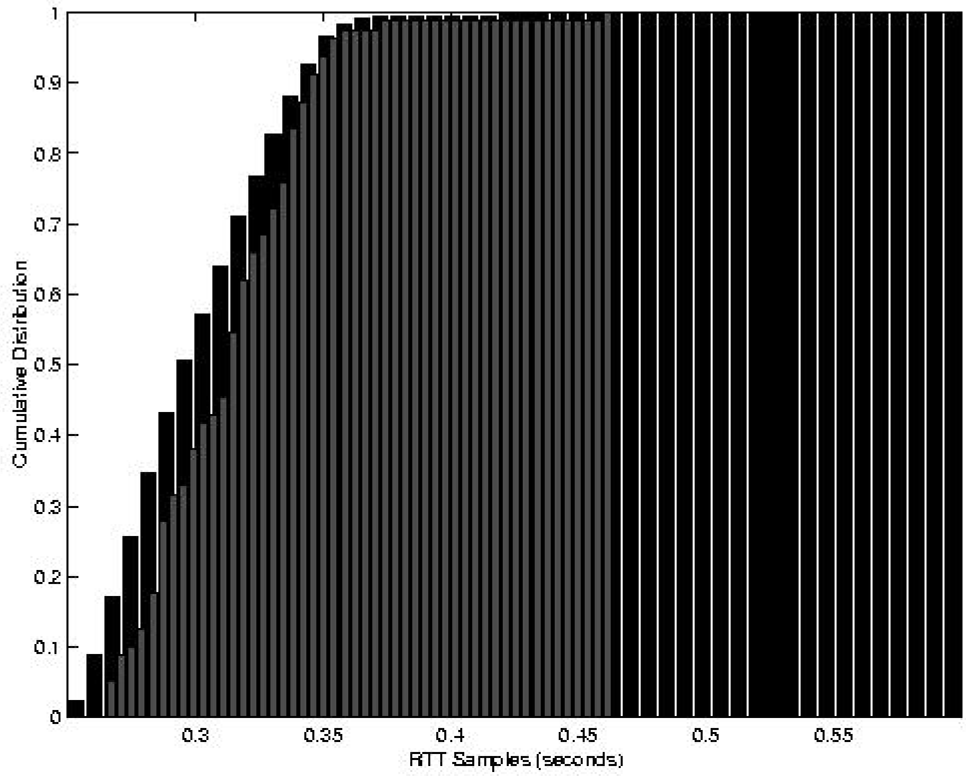

Figure 5-21 Loss Conditioned Delay CDF for Path 3

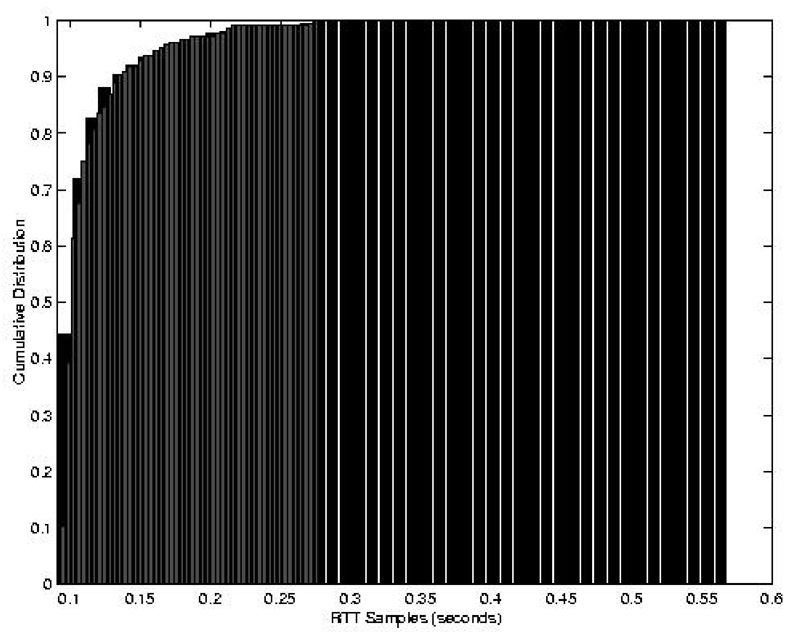

Figure 5-22 Loss Conditioned Delay CDF for Path 4

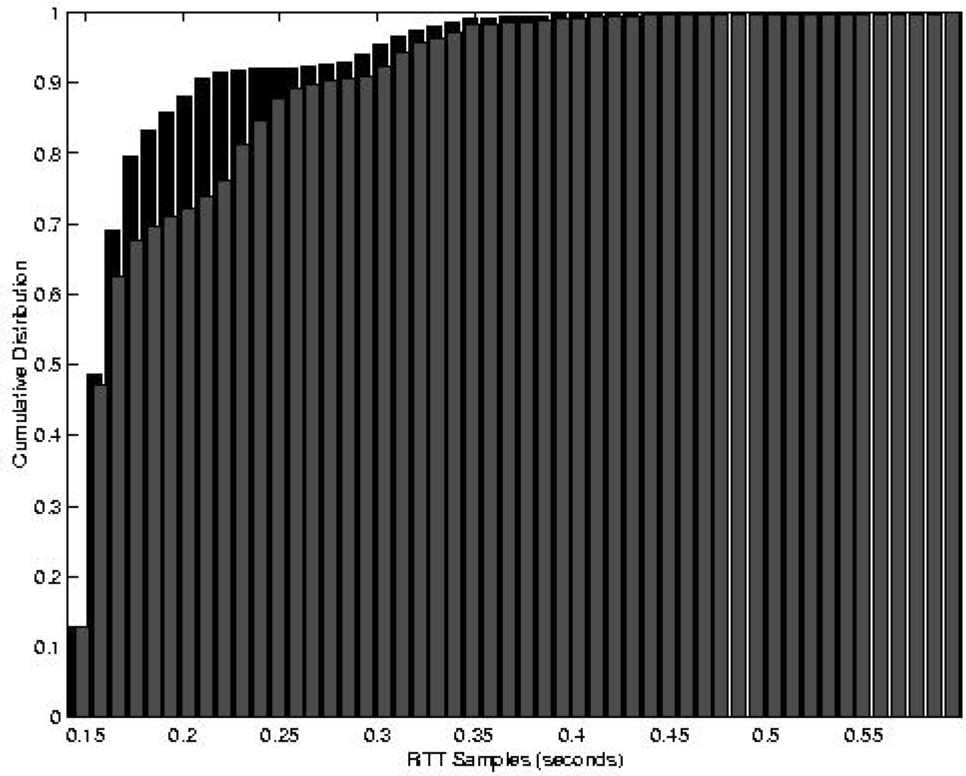

Figure 5-23 Loss Conditioned Delay CDF for Path 5

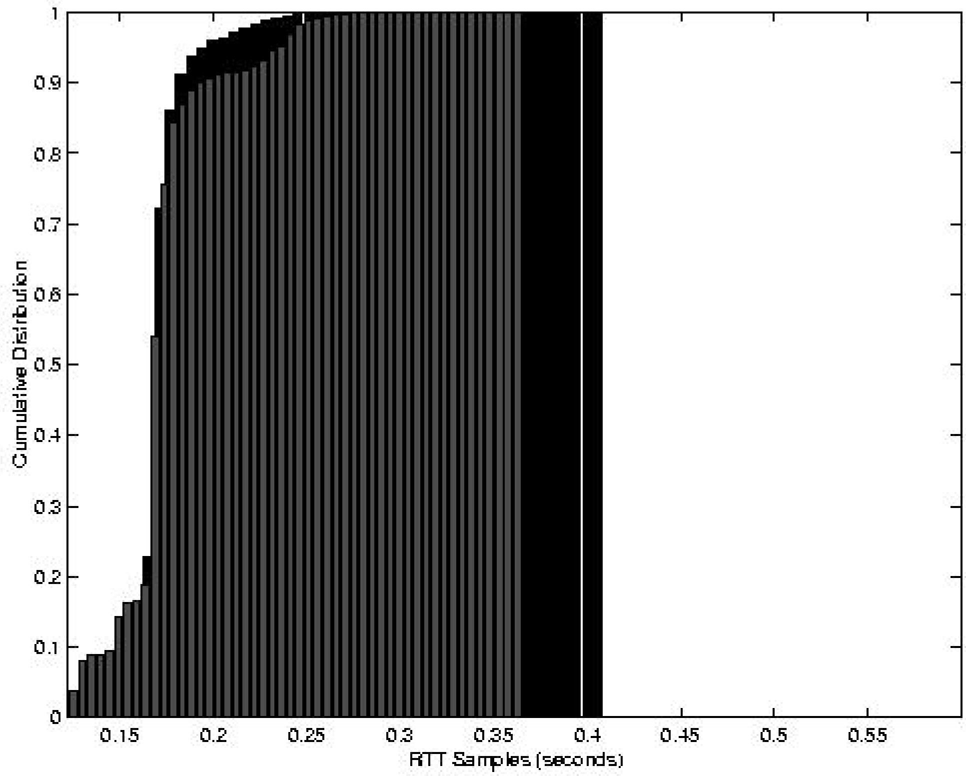

Figure 5-24 Loss Conditioned Delay CDF for Path 6

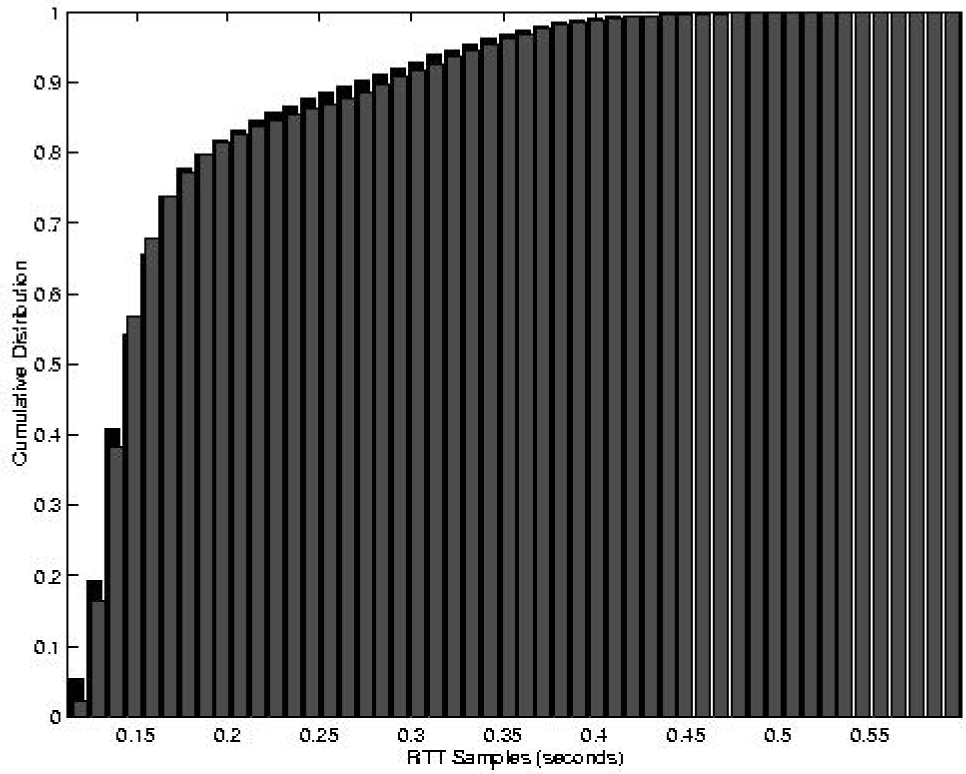

Figure 5-25 Loss Conditioned Delay CDF for Path 7

5.3 Hypothetical Throughput Analysis of DCA

In this section, we assess the impact DCA has on TCP throughput. As we have stated earlier, DCA will improve throughput as long as the throughput loss caused by the congestion reactions is offset by the gain in throughput achieved by reducing the level of packet loss and timeouts. Using an analytic throughput model of DCA in conjunction with the measured data, we show that DCA will degrade rather than improve TCP throughput. Even though RTT increases are associated with some losses, DCA does not improve its throughput. This is because DCA cannot distinguish from the increases associated with loss and those that are not.

5.3.1 DCA Throughput Analysis: Overview and Assumptions

In this section we define the hypothetical DCA algorithm (which we call TCP/DCA) that we use in the throughput analysis and will run the DCA algorithm on the *tcpRTT* time series extracted from the measured TCP traces. We quantify how well the algorithm is able to "hypothetically" avoid the losses experienced by the TCP connection (the loss information associated with the traced TCP connection is encoded in the *tcpRTT* time series data). We also quantify how frequently the DCA algorithm reacts "unnecessarily" (i.e., when the algorithm reacts to congestion that would not have lead to loss if the algorithm did not react). We feed this estimate of the rate of DCA congestion reactions to an analytic throughput model of TCP/DCA. The results of this analysis indicate that will degrade rather than improve the throughput. This approach assumes that a reaction by a DCA flow (i.e., all reactions including the "unnecessary" reactions) will not affect the congestion processes that are active over the path. The assumptions supporting the validity of our approach are discussed in the end of this section.

The key aspects of TCP/DCA are:

- **congestion decision**. We define the congestion decision as follows:

  $$sampledRTT(x) > windowAVG(w) + threshold$$

  As described in the previous section, the *sampledRTT* is an estimate of the "instantaneous" RTT level and the *windowAVG* is a smoothed average of RTT. The *threshold* value is intended to control the frequency of DCA reactions. For most of the analysis we use a *threshold* set to the standard deviation associated with the RTT samples in the *windowAVG* moving window. To confirm the selection of the *threshold*, we also run the analysis using a *threshold* of 0 and also set to twice the standard deviation. A *threshold* value of 1 standard deviation proves to be the most effective at predicting loss.
- **congestion decision frequency**. This determines how frequently the algorithm will run the congestion decision algorithm. For most of the analysis, we limit the frequency to once per congestion epoch. A congestion epoch is defined as a period of time during which the *tcpRTT* samples are greater than some minimal level (i.e., the RTT reflects queueing delay).

- **increase/decrease algorithm**. When the congestion decision algorithm indicates a reaction is necessary, the TCP/DCA sender reacts aggressively by reducing the *cwnd* by 50%. Other DCA algorithms (e.g., TCP/Vegas and TCP/Dual) chose lower reduction amounts based on simulation results which showed that over T1 speed links a small reduction was effective at both avoiding loss and reducing queue oscillations caused by synchronized TCP connections. We select a reduction of 50% because the measurement results indicate that the time scale associated with the RTT increase that precedes loss is roughly 1 RTT. A reduction of 50% is equivalent to the TCP reaction to a loss that is recovered by the arrival of three duplicate acknowledgements (an event referred to as a TD event). The benefit of DCA becomes significant if the algorithm is able to avoid a loss that would otherwise require a timeout for recovery. Here the potential gain in throughput would be tremendous compared to the throughput loss caused by a window reduction if the reduction avoids future packet losses. Avoiding a single timeout can offset quite a few "incorrect" congestion decisions where the DCA algorithm reacts but unnecessarily.

We adapt a TCP throughput model [PADH98] to help quantify the tradeoff involving a reduced packet loss rate at the expense of additional send rate reductions due to congestion indications. In the discussion below, a DCA congestion reaction implies a reaction based on an observed increase in RTT. A key to the analysis is that a DCA congestion reaction is equivalent to a reaction to a triple duplicate acknowledgement recovery (i.e., reduce the *cwnd* by 50%). We rely on simulation analysis (presented in the second part of this dissertation) to study the effect of different reactions by DCA to TCP throughput.

The work of [PADH98] presents a technique known to estimate TCP throughput based on packet losses and RTT (refer to Appendix A for a summary of the model). The model accounts for the throughput loss caused by triple duplicate acknowledgement recovery via the TCP fast recovery/fast retransmit algorithm and by recovery via timeout. The model also handles window size limitations due to the maximum TCP send and receive windows which are an input to the model. Triple duplicate acknowledgement recovery is a random event based on the packet loss probability. A timeout recovery is based on a further probability that a loss event results in a timeout. The model generates the timeout probability by modeling the event that multiple packets are lost within an RTT period and is therefore based on the same packet drop probability. We modify the model to reflect separate loss probabilities for recovery by fast recovery and for recoveries by timeout (*p(loss)* and *p(timeout)* respectively). Because a DCA congestion reaction is equivalent to a reaction caused by fast recovery, we can use the model to illustrate the impact to throughput as the number of DCA reactions increases (i.e., as the *p(loss)* increases) while the frequency of timeouts decreases (i.e., as the *p(timeout)* decreases).

We assume that the assumptions made by Padhye et., al., are valid in our application of the model. These assumptions are supported by an array of previous work as described in [PADH98]. Since Padhye et. al, validate their model over paths similar to those that we are interested in, it seems reasonable to accept these assumptions. The base assumptions are:

- The model ignores TCP's slow-start and fast recovery algorithms.
- The model assumes that packet loss events within an RTT time period are correlated and that losses that occur greater than one RTT apart are considered independent. This implies that RTT increases that are separated by at least one RTT period are also independent.
- The round trip time experienced by TCP packets is independent of the TCP *cwnd* size.
- All TCP/Senders implement the TCP/Reno protocol as documented in [ALLM99].

Our throughput analysis requires these additional conjectures and assumptions:

- We conjecture that we will see similar results for different variations of the DCA algorithm. We validate this in Part 2 of the dissertation.
- We assume that a reaction by DCA will not significantly impact the congestion process of the observed paths. This is an extension of the assumption by Padhye et. al., indicating that RTT is independent of the TCP *cwnd* value. The more fundamental assumption is that the load placed on a network by a single flow (whether TCP/Reno or TCP/DCA) is only a fraction of the total traffic that exists over the path. This assumption is supported by recent measurement studies supports which indicate that Internet backbone switches are subject to thousands of active flows at any given time [CLAF97,THOM97]. We provide additional validation for this in Part 2 of the dissertation.

5.3.2 DCA Throughput Analysis Approach

Our approach is as follows. For each traced TCP connection associated with the measurement analysis described in the previous section, we run the TCP/DCA algorithm over the *tcpRTT* time series. We count the number of times the algorithm reacts to increases in RTT (we refer to this as the *congestionReactionCount*). We also count the number of times that the actual losses could have been detected (we refer to this as the *lossAvoidedCount*). So the *lossAvoidedCount* includes observed losses that are recovered by either a TD event or by a timeout. Assume that for a given run there are a total of *packetsSent* data packets transmitted and that a total of *numberRetransmits* packets are retransmitted (by either a timeout or a TD event recovery). Therefore the actual packet loss rate can be approximated as *numberRetransmits/packetsSent*.

For each run, we find the following probabilities:

- *DCA Congestion Reaction Rate (DCRR)*: this takes the measured loss rate, adjusts it since we will be avoiding some packet loss and further adjusts it since we will be performing additional send rate reactions due to the DCA algorithm. The DCRR will be the *p(loss)* parameter used in the throughput model and is defined as:

  DCRR : (numberRetransmits + congestionReactionCount – lossAvoidedCount) / packetsSent

- *Adjusted Packet Loss Rate (APLR)*: this takes the measured loss rate and adjusts it for the number of losses that are avoided by the DCA algorithm. This loss rate is used to account for the throughput reduction caused by packet loss recovered via timeouts. The APLR will be the *p(timeout)* parameter used in the throughput model and is defined as:

  APLR : (numberRetransmits– lossAvoidedCount) / packetsSent

The throughput generated by the model will reflect the tradeoff associated with DCA's ability to reduce the number of timeouts while increasing the number of fast recovery reactions (due to the congestion indication reactions). The original model used a single packet loss rate. The modified model accepts the

DCRR and the APLR (and along with the maximum window size and the measured average RTT) and returns the predicted throughput. Because both adjusted probabilities are based on measured data, the results obtained via this method will be reflective of what we can expect from a DCA algorithm when operated over high speed Internet paths.

We process each *tcpRTT* sample in the time series looking for the beginning of a congestion epoch. A congestion epoch begins when the RTT begins to increase:

```
If ( (sampledRTT > lastUncongestedRTT)) && (setState == CongestionEpochNotActive) )
  lastUncongestedRTT = sampledRTT(x);
  setState = CongestionEpochActive;
End
```

A congestion epoch ends when the RTT returns to the level it was at when it first entered the epoch:

```
If ((sampledRTT(x)< lastUncongestedRTT) && (setState==CongestionEpochActive))
  SetState = CongestionEpochNotActive
End
```

During an epoch if the congestion detection algorithm triggers, we increment the *congestionReactionCount* by one.

```
If ((sampledRTT(x)< windowAVG(w) + threshold) && (setState==CongestionEpochActive))
  CongestionReactionCount ++
End
```

Once the DCA algorithm decides to react, we look ahead in the trace by some amount of time (referred to as the *lookAheadThreshold* which is specified by a number of RTTs) and search for loss events that are potentially avoided by the DCA reaction. For all losses that occur within this time period following a DCA reaction, we increment the *lossAvoidedCount* by one. We run the algorithm two ways: a best case where the send rate reduction is sufficient to avoid all the predicted losses, and a “more likely” case where the loss is avoided 50% of the time. We refer to the two variations as *DCA-best* and *DCA-likely*. We make sure that the algorithm does not react more than once per epoch (or more than once per RTT period, whichever is smaller).

We use one of two values for the *lookAheadThreshold* parameter: a single RTT or a number of RTT's that is reflective of the average window. For the latter case, the average window value, *W(p)*, is an estimate of the average window experienced by the connection and is based on the packet loss probability *p*. It is defined as:

$W(p) = 2/3 + \sqrt{\frac{4*(1-p)}{3p} + 4/9}$ where *p* is the measured packet loss rate.

Using a *lookAheadThreshold* value of 1 RTT period implies that while a DCA reaction does not affect the congestion processes that are active over the path, it can reduce the chances of the flow experiencing a loss within 1 RTT period. The basis of this is our observation that if loss is accompanied by an increase in RTT, the time scale of the queue buildup is brief (i.e., on the order of a RTT). We relax this assumption with by also performing the analysis with the *lookAheadThreshold* set to a significantly larger value (i.e., the *W(p))*. In this case, the assumption is that if a DCA flow reacts to an increase in RTT, it potentially will avoid any loss event that occurs for the duration of time until the *cwnd* is restored to its original value before the send rate reduction. The following explains how this time period is calculated.

If we assume that the receiver acknowledges every other segment, the rate of growth within each RTT will be ½ a segment. When a sender reduces its window from *W* to *W/2* segments, it will take *W* RTT's to restore the window from *W/2* back to *W*. By definition, the average value of *W* will be *W(p)*. Thus, once the DCA decides to react, we look *W(p)*sampledRTT(x)* amount of time in the future for loss events that might be avoided.

The use of the *W(p)* as the choice for the *lookAheadThreshold* along with the *DCA-best* variation overestimates the abilities of the DCA algorithm. For some of the runs, the analysis predicts that more than 50% of all the actual loss events will be avoided. Other aspects of the algorithm further bias's in favor of the DCA algorithm. Limiting the frequency of congestion decisions to once per epoch minimizes the number of unnecessary congestion reactions. For example, we modified the algorithm so that it was allowed to react more than once per epoch (i.e., once per RTT). This significantly increases the number of congestion reactions without increasing the number of packet loss events avoided. Essentially the technique overestimates DCA's abilities to avoid loss while minimizing the price paid for "unnecessary" DCA congestion reactions. We did this so that we could evaluate the performance of DCA when it is at its best to measure its maximum benefits.

5.3.3 Demonstrating the DCA Throughput Model Analysis

Appendix A overviews the TCP throughput model and shows the changes that we have made as we adapted it to model DCA. In this section, we utilize the throughput model analysis approach described above on the low speed simulation run that was used in the previous chapter (i.e., the baseline analysis described in section 4.2.4). The objective is to validate the analysis methodology by showing that in an environment that is conducive to DCA, the DCA throughput model will predict an improved TCP throughput.

As indicated in section 4.2.4, the correlation metrics indicate that the dynamics associated with the simulation as such that packet loss and increases in RTT are highly correlated. The loss rate associated with the run was 2% (17% of the loss events required a timeout for recovery) and the observed average throughput was 59.8 Kbytes/seconds. The original TCP model predicts a throughput of 54.8Kbytes/second.

After running the TCP/DCA algorithm on the *tcpRTT* time series extracted from the TCP/Reno connection that was under observation, we arrive at the following adjusted loss probabilities are:

$$DCRR = 2.4\%$$

$$APLR = .4\%$$

The TCP/DCA throughput model predicts an increase in throughput to 85.176 Kbytes/second which reflects a 35.6% increase. This is based on a *lookAheadThreshold* of 1 RTT with the congestion decision *threshold* value set to one standard deviation. The increase in throughput is primarily because the TCP/DCA algorithm is able to avoid a large amount of loss and therefore a significant number of timeouts. The point here is to simply provide some level of assurance on the accuracy of the TCP/DCA throughput model analysis.

5.3.4 DCA Throughput Analysis Results

In this section, we show the results of the throughput model analysis. The results are based on a trace concatenated with the data from the 25 traces associated with each path. The *lookAheadThreshold* is set to 1 RTT and the *DCA-best* variation is assumed. The length of the moving windows used in the congestion decision are 2 and 20 (*x* and *w* respectively) as these produced the best loss prediction results. We show the results using two different congestion decisions: a congestion decision with a *threshold* set to one standard deviation and a decision with a *threshold* of 0. Thus, we utilize a filtered and an unfiltered congestion decision:

filtered decision: P[sampledRTT(2) > windowAVG(20) + std]
unfiltered decision: P[sampledRTT(2) > windowAVG(20)]

We begin by showing two metrics (which we did not describe in the previous chapter) that provide an indication of the relative frequency associated with the DCA congestion decisions. The first metric is defined as:

congestionReactionCount /numberRetransmits.

The results using the filtered and unfiltered congestion decision are shown in the second column of Table 5-10. Values greater than 1 implies that there are more congestion indication reactions than actual loss events. As expected, the metric values are larger for the unfiltered congestion decision. This is because the algorithm makes no attempt to limit the frequency of congestion decisions. In general, the higher the value of the metric, the larger the throughput degradation. Path 7 is unusual in that the values are below 1. Because the path exhibits a high loss rate, the relative number of congestion decisions will be small (in fact less than the number of losses). Based on this, we would expect the amount of throughput degradation to be smaller for this path compared to the other paths.

The second metric is defined as:

congestionReactionCount /lossAvoidedCount

The results are shown in the third column of Table 5-10. Values larger than 1 implies more DCA congestion reactions and will tend to degrade throughput more. The highest value does not necessarily correspond to the largest reduction in throughput however. For example, path 6 had the highest metric value but as we will see shortly, it did not suffer the most throughput degradation. It had the lowest loss rate so that the *lossAvoidedCount* was very small compared to other paths.

Table 5-10. Relative frequency of DCA congestion reactions (filtered decisions, unfiltered decisions)

Path #	CongestionReactionCount / numberRetransmits	CongestionReactionCount / lossAvoidedCount
1	4.2, 6.4	18.5, 14
2	4.3, 6.8	15.7, 13
3	2.1, 3.8	10.2, 9.4
4	11.4, 18.9	62.6, 62.5
5	12.5, 22.6	51.1, 41.6
6	23.7, 43.4	77.2, 82.6
7	.27, .51	2.8, 1.7

Table 5-11 summarizes the measured results from the previous chapter that are utilized in the throughput analysis. The *modelThroughput* column indicates the expected throughput using the unchanged TCP throughput model. Notice that the predicted model throughput is consistently lower than the measured throughput. The difference ranged from 3.5% to 32.9% (paths 5 and 2 respectively). The model makes some simplifying assumptions (e.g., it does not model the amount of time spent in slow-start nor does it model fast recovery) which will tend to underestimate TCP throughput.

Table 5-12 illustrates the results of the analysis based on the modified TCP throughput model with the *lookAheadThreshold* set to a value of 1 RTT. The first two columns represent the DCRR and APLR values described earlier using the *DCA-best* variation. Each table entry represents two runs of the analysis using the filtered and unfiltered congestion decision algorithms respectively. The adjusted throughput column represents the results of the TCP throughput model using the adjusted loss probabilities. The final column simply indicates the percentage of throughput loss between the adjusted throughput and the original model throughput. As an example, the first row of Table 5-12 indicates that the Emory path (i.e., path 1) will experience a 50.5% reduction in throughput when using the filtered congestion decision algorithm.

Table 5-11. Aggregate measured statistics

Path#	Avg Loss Rate (%)	Avg RTT (seconds)	Avg Throughput (Kbytes/sec)	Model Throughput (Kbytes/sec)
1	.8	.066	185.1	137.6
2	1.7	.249	46.4	31.1
3	6.5	.32	12.8	9.3
4	.89	.117	23.8	22.7
5	.65	.174	71.7	69.2
6	.55	.171	46.2	41.9
7	10.5	.179	10.8	7.9

Table 5-12. Aggregate results: throughput model results (lookAheadThreshold = 1 RTT) : (filtered, unfiltered congestion decision)

Path#	DCA congestion reaction rate	Adjusted packet loss rate	Adjusted throughput (Kbytes/sec)	% throughput reduction
1	.021,.027	.0067,.0047	68.0,54.3	50.5%,60.5%
2	.04,.053	.014,.011	17.1,13.9	45.0%,55.3%
3	.083,.09	.06,.046	7.8,7.5	16.1%,19.3%
4	.029,.042	.008,.006	13.0,9.35	42.7%,58.8%
5	.029,.039	.005,.005	28.6,22.7	58.7%,67.1%
6	.024,.039	.005,.003	27.3,22.1	34.8%,47.2%
7	.12,.122	.096,.077	7.2,7.3	7.5%,8.9%

Table 5-13 illustrates the results of each path based on the time of day associated with the trace. The table provides a more detailed view of the impact of DCA. Each entry represents the average (and standard deviation) associated with the runs at a given time. To provide some statistical assurance, we provide the 95% confidence interval associated with the data. We see that some paths might occasionally not experience a degradation although that should occur infrequently.

Table 5.13. Percentage of throughput degradation: mean (standard deviation) with 95% confidence intervals

Path #	9:00AM	11:00AM	1:00PM	3:00PM	5:00PM
1	-54(13) (-70,-40)	-50(9) (-60,-40)	-53(17) (-73,-34)	-33(15) (-50,-16)	-44(22) (-70,-20)
2	-50(14) (-66,-33)	-46(14) (-62,-29)	-47(17) (-67,-26)	-59(7.6) (-68,-50)	-50(27) (-81,-19)
3	-20(14) (-36,-4)	-20(6) (-28,-13)	-26(11) (-39,-13)	-25(19) (-46,-3)	-7(11) (-20, 5.5)
4	-39(15) (-56,-21)	-50(2) (-64,-36)	-41(12) (-55,-28)	-49(9.4) (-60,-38)	-54(11) (-66,-41)
5	-43(20) (-66,-20)	-57(6) (-63,-50)	-59(6.1) (-66,-52)	-56(10) (-67,-45)	-62(2) (-65,-59)
6	-41(12) (-55,-27)	-45(8) (-54,-36)	-33(12) (-47,-20)	-20(17) (-39,0)	-31(16) (-49,-13)
7	-19(9) (-29,-9)	-13(7) (-21,-4)	-5.4(6.6) (-13,2.2)	-4.2(2.1) (-6,-2)	-12(10) (-24,-1)

The results from Tables 5-12 and 5-13 are not surprising. The reduction in throughput increases as the loss rate decreases. The results in Table 5-14 reflect the analysis results when the *lookAheadThreshold* is increased to *W(p)* RTT's beyond the congestion decision. The results reflect the filtered congestion decision. As in Table 5-12, there are two runs displayed in the table. The first value represents the DCA-best variation where all loss events within a W(p) time period following a DCA congestion reaction are assumed to be avoided. The second value represents the "more likely" scenario where 50% of the loss events that occur within the time period are assumed to be avoided. As the fifth column in Table 5-14 indicates, we see a slight improvement (i.e., less throughput degradation) compared to the case when the *lookAheadThreshold* time is one RTT. The conclusion to be drawn from Table's 5-12, 5-13 and 5-14 is that the benefit of avoiding packet loss is far outweighed by the frequent send rate reductions that are not associated with loss. As one might expect, this effect is observed to a greater extent over lower loss paths.

Table 5-14. Aggregate throughput model results (lookAheadThreshold = W(p)): (DCA-best,DCA-likely)

Path#	DCA congestion reaction rate	Adjusted packet loss rate	Adjusted throughput (Kbytes/sec)	% throughput reduction
1	.018, .020	.0038,.0061	78.79,70.03	42.7,49.1
2	.035,.04	.009,.013	19.0,17.4	39,44
3	.08,.085	.058,.061	8.02,7.74	13.8,16.8
4	.027,.028	.0058,.0073	13.93,13.24	38.6,41.7
5	.026,.028	.002,.004	31.46,29.51	54.5,57.3
6	.023,.024	.004,.005	27.66,27.33	34,34.8
7	.11,.12	.093,.099	7.46,7.05	5.5,10.7