2011

An evaluation of synchronization loss rate calculations on high speed networks

Aaron Tureau
Louisiana State University and Agricultural and Mechanical College, aturea2@tigers.lsu.edu

AN EVALUATION OF SYNCHRONIZATION LOSS RATE CALCULATIONS ON HIGH SPEED NETWORKS

A Thesis
Submitted to the Graduate Faculty of the
Louisiana State University and
Agricultural and Mechanical College
in partial fulfillment of the
requirements for the degree of
Master of Science

in

System Science

in

The Interdepartmental Program

in

Computer Science

by
Aaron Tureau
B.S., LSU, 2009
August, 2011

Acknowledgements

I would like to thank my parents, Joey and Sherri, and my little brother, Caleb, for their support during this whole process. My committe leader and major professor, Dr. Seung-Jong Park, who allowed me to have access to CRON and who pushed me to where i am today. Thanks to the CRON research team, Cheng Cui, Lin Xue, Praveenkumar Kondikoppa, and Chui-Hui Chiu, who have helped me with organization, hardware and software tutorials and setup, and general troubleshooting. I would also like to thank the rest of my thesis committee, Professors Rajgopal Kannan and Jian Zhang, who are taking time our of thier busy schedule to listen to my defense and the CSC department staff who kept me up to speed on what i needed finished and when. Finally I would like to thank my friends and especially my roommate, Willie Nettles, all of whom had to put up with me during this process and whom encouraged me and made sure i was on task event when i didn't want to be.

Table of Contents

List of Tables

List of Figures

Abstract

This paper is broken down into two parts:(1) discussion of current formulas that are used to calculate synchronized loss rates among concurrent TCP flows with the results of those equations on flows running through a bottleneck on a high speed emulated network and (2)steps to create revised forms of these equations that are more accurate and give a more reasonable estimation without having the shortcomings of the current equations.

This paper brings to light three equations that were previously proposed and were used in published research projects along with their strengths and shortcomings. Through the study of these equations a series of newly revised forms of these algorithms will be introduced that take into account their predecessors' pitfalls and gives a more affective and useful result under any set of networking conditions including large numbers of concurrent flows, multiple variations of TCP, and variations of receiving queue sizes among others. In this paper we will prove, through logical analysis of the equations structure, definition, and results they provide of network emulations on the CRON testbed, that these modifications to the existing formulas are necessary in order to provide a metric that depicts an accurate view of the synchronization loss rate of a given network.

Chapter 1

Introduction

In this paper, I will be focusing on two different ideas that build from one another: first, I will evaluate previously proposed algorithms to calculate the synchronization loss rate of concurrent flows, testing their equations on a series of high speed network emulations and finding their strengths and weaknesses; secondly, I will reveal newly renovated forms of these algorithms that do not contain the previous equations' shortcommings using the information gained from the performance of these algorithms.

1.1 Motivation

When dealing with high speed networks, one of the biggest issues facing researchers today is the ability to utilize as much bandwidth from high speed links as possible in order to achieve the networks desired throughput. However, current versions of TCP, including the most widely used version, Reno, variations with high link fairness, CUBIC, even versions that are able to reach and sustain high amount of bandwith, HSTCP, have a tendency to cause concurrent flows to simultaneously drop packets. This happens whenever the queue overflows causing incomming packets to become lost. According to TCP/IP protocol, every time a packet is received by the destination an ACK is sent to acknowldge that each packet was sent successfully. When a packet is dropped the destination will resend the previous ACK to signify the last in order packet that it received to the sender. If a sender detects three of the same ACKs then the sender is forced to reduce their congestion window (*cwnd*) size of its flow by some multiplicative amount. The *cwnd* is the variable used by the sender to determine the amount of data it sends through the network in the next RTT. When this occurs to more than one flow within the same RTT this is known as a synchronized loss event. Figure 1.1 shows the affect synchronization can affect link utilization.

These synchronized loss events cause high fluctuations in throughput especially when the majority of flows share in the synchronized loss event. It is logical to think when a loss event, that is an RTT in which at least one packet loss occurs [**?**], involves fewer flows, then fewer cwnd are reduced in value. With more flows sustaining their higher cwnd values the average throughput and link utilization will tend to be higher than situations where multiple flows reduce their *cwnd* in synch. Studies have been performed by [**?**] [**?**] [**?**] to map the inverse relationship to global synchronized loss rates and average throughput.

While link utilization is a very well know and easy metric to calculate there is no universal method to calculate global synchronized loss rates. Since the idea has only been in major discussions in recent years

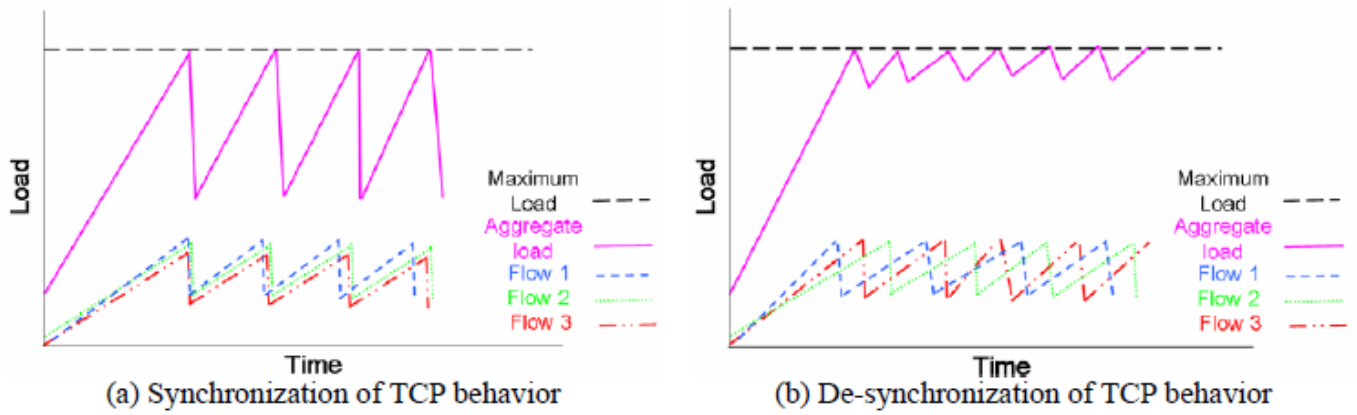

Figure 1.1: Affect of synchronization and desynchronization on link utilization

given the current spike in network throughput rates, there are actually very few published formulas used to track this phenomenon. These few formulas are affective in giving a solution for the synchronized loss metric for most cases, but each has its issues in becoming an affective generally accurate estimation for this problem including but not limited to limits on the number of flows the formula can consider at a single time, false positive results, and multi-level averaging that can reduce the confidence in the results.

By observing the theories, variable definitions, and performance of these previously proposed equations new formulas were composed. The formulas that are being proposed in this paper will not only give an accurate account of each individual flow's synchronized loss rate with respect to the rest of the simultaneously operating flows, they can also give global synchronized loss rate of the group of concurrently running flows as a whole. In this way, the newly proposed formulas can be used as general application formulas for the calculation of link synchronization.

Therefore the ultimate goal of this paper can be divided into two parts: define and observe the currently proposed equations to find the pros and cons to their methodology, refine these equations and give results and reasoning that show that the newly defined equations give a more accurate and valid result for the state of synchronization of each flow's participation in loss events. This can be achieved by conducting a series of tests on a network emulator given a variation of different network scenarios that have previously defined tendencies and collect results of congestion window values at close intervals to detect the moment loss events occur. Then, from these results we can calculate synchronization loss rates by using each equation and compare the results.

The rest of the paper is organized as follows. Chapter 2 discusses the experimental setup of the testbed that was used in the network emulation including the hardware and software components involved. Chapter 3 gives definition and evaluation to all of the previously proposed synchronization loss rate equations. Chapter 4 introduces and gives logical merit to the revised versions of the previously proposed formulas. Chapter 5 discusses the results obtained by the network emulation on the CRON testbed and how each equation faired in calculting the global synchronization loss rate. Chapter 6 concludes the paper.

Chapter 2

Emulation Methodology

In this chapter, I will explain the hardware and software setups for the experiment that were conducted to test the variations of the synchronized loss equation.

2.1 Hardware Testbed Setup

In order to test the effectiveness the various equations network emulations were created with use of the Cyberinfrastructure of Reconfigurable Optical Networking (*CRON*) testbed. CRON is a network emulator created by the Center for Computation and Technology (CCT) at LSU and can create emulations of real time network environments that can be dynamically customized to the specifications of many types of topologies and network setups. The exceptional aspect of this testbed is that it is able to emulate network speeds of up to 10 Gbps. CRON is available at [**?**].

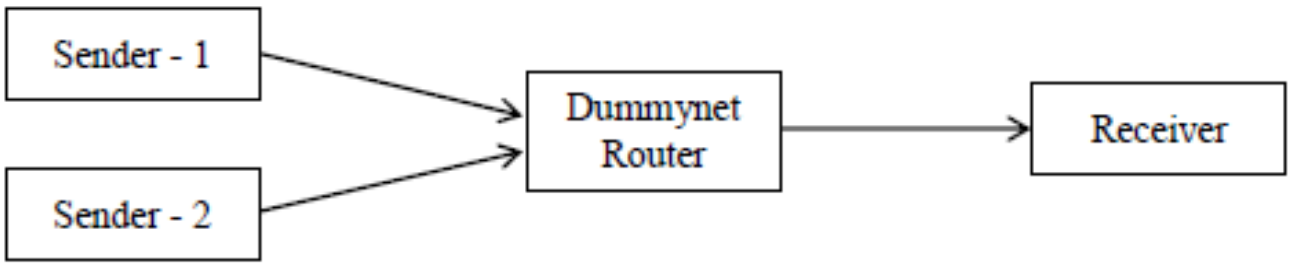

Figure 2.1: A view of the Y-Topology being emulated.

To start an experiment you must first have an account registered on the CRON webpage [**?**]. From there an expereiment can be created from the top menu under the Experiment drop down menu as shown below:

A Y-shaped topology(Figure 2.1) was created using an NS Script that was uploaded into the CRON website's test creation page. The topology consists of two sending nodes connected through a router to a shared receiving node. This network topology was used to emulate a routing situation where flows from the same and different sources would meet at a shared link. Each flow travels through a 10Gbps link with 7 other concurrent flows into a router where it is introduced to the other 8 flows. All 16 flows

Figure 2.2: Experiment Dropdown.

Figure 2.3: Experiment Creation Page.

continue into an 8 Gbps bottleneck and into a shared destination receiver. The extreme bottleneck is used to exaggerate the affects of the congestion and increase the frequency of packet losses as a result of overloading the link. The different synchronized loss rate values from flows of the same source and flows of the two different sources were both considered. A total of 7 physical nodes are needed to fully complete this topology, 2 senders, 1 receiver, 1 router, and 3 delay nodes.

2.2 Software Testbed Setup

Each link is controlled by a delay node which has a *FreeBSD OS 7.3* version image with *Dummynet*, a software emulator. *Dummynet* allows for the links to emulate the conditions of a link of various delays lengths and bandwidths. In the experiments that were conducted, a delay of 30ms was for all links giving each packet an average RTT of 120ms. By the Mathis equation, we know that as RTT increases the ability for a network to reach higher bandwidths decreases. The delay of 30ms was put into place to give reasonable delay to where packet drops would be stimulated to occur, but not too long to where the flows utilization of the link would be hindered too greatly. Figure 2.2 shows the introduction of delay nodes for each link. A total of 6 VLANs are created for this topology [**?**].

In *Dummynet*, the default queue size is 1 MB. This value is much too small to be used in such a high

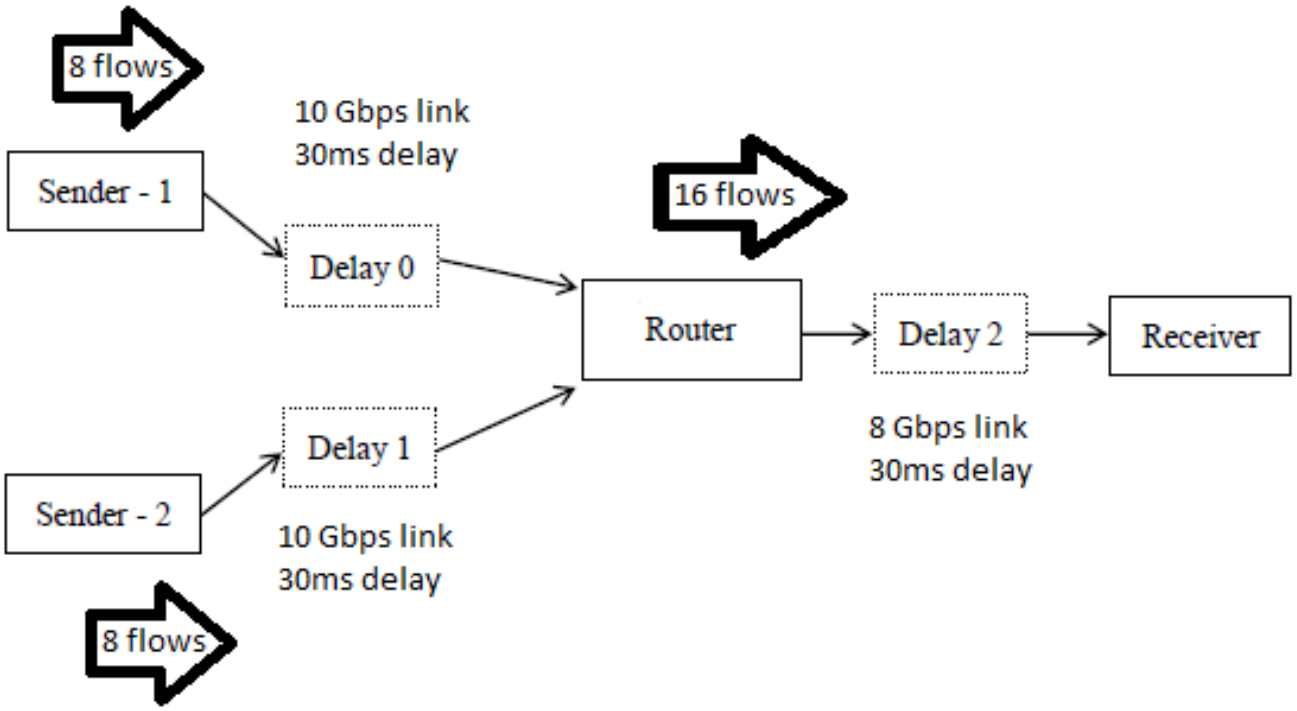

Figure 2.4: A view of the actual form of the Y-Topology used in CRON.

speed network as the one that was used in these experiments. This can be changed by manipulating the *sysctl* variable. Throughout the course of the experiment the queue size was varied with reference to percentages of Bandwidth Delay Product (BDP). BDP is the overall rule of thumb for setting up network queue sizes and can be calculated by multiplying the RTT with bandwidth of the bottleneck. With varying queue sizes the number of packet drops, and in turn the synchronized rate, should increase with the smaller queue sizes and decrease with larger queue sizes. Changing the queue size at the bottleneck should result in a distinct difference in synchronized loss rates when calculated under all variations of equations. Also for these experiments, all of the queues implemented a Droptail queuing policy because it is the most widely used queuing policy used today [?]. While a more active queuing policy such as RED or GRED are known to give lower synchronized loss rates, the affects of the variant of TCP and queue size should be enough to show the difference in synchronization needed to test the equations.

Using the same sysctl command the TCP variation can be changed as well. In this experiment three different variations were used: Reno, CUBIC and HSTCP. As stated in Chapter 1 TCP Reno was used as the default TCP since it is currently the most commonly implemented TCP algorithm. Reno is also less aggressive in its additive increase phase than most TCP variants, for each acknowledgement received it increases the *cwnd* by 1 [?], which should keep the ratio of packet drops over time very low. With a smaller amount of packet drops the chance of flows being desynchronized from other flows should be higher [?]. The other TCP variant that was used in these trials was CUBIC TCP. CUBIC was chosen for its mild aggressiveness in increasing its congestion window when polling for the link capacity. Because of this trait CUBIC TCP will cause the flows to suffer from packet drops at a slightly faster rate than Reno which will allow for more loss events within the same amount of time. Also because of CUBIC's tendency to share the link capacity in a more fair way than other TCP variants, this may cause much higher synchronized loss rates than other variants [?]. The third variant, HSTCP, is one of the most agressive yet stable form of TCP that is currently in use [?]. While having the ability to reach very high bandwidths very quickly is has mechanisms in place that allow for these high bandwidths to be sustained increasing the overall link utilization even if it sacrifices TCP fairness. Between these three

variations we can view many of the extreme traits of most TCP variants (Appendix A).

In order to generate and read packets, *iperf* was installed into all of the non-delay nodes (i.e. the sending, receiving, and routing nodes) on top of a 64-bit light-weight *Linux OS* image. *Iperf* is a well known and widely used packet generating software that we used to create and measure the emulation of TCP traffic within the network. We are able to do this by transferring memory buffers from a source machine to a destination machine. Using this software we can get overall throughput, link utilization and packet losses of emulated network use. It allows for the user to customized number of senders and the direction of traffic to help to see how the network behaves under many different traffic scenarios. The iperf software, as stated above, is broken down into two separate systems: transmitter and listener. The transmitter can specify the destination, number of flows, interval of calculating throughput, span of the tending time, the pipe that the packets are being sent through, etc. The destination node will set up the listener program using the option '-*s*'. The listener waits for specified packets that were sent from any connected node using iperf. Also from the information given by iperf the cwnd can be derived and will be written to as standard output at the receiver using the same option [**?**].

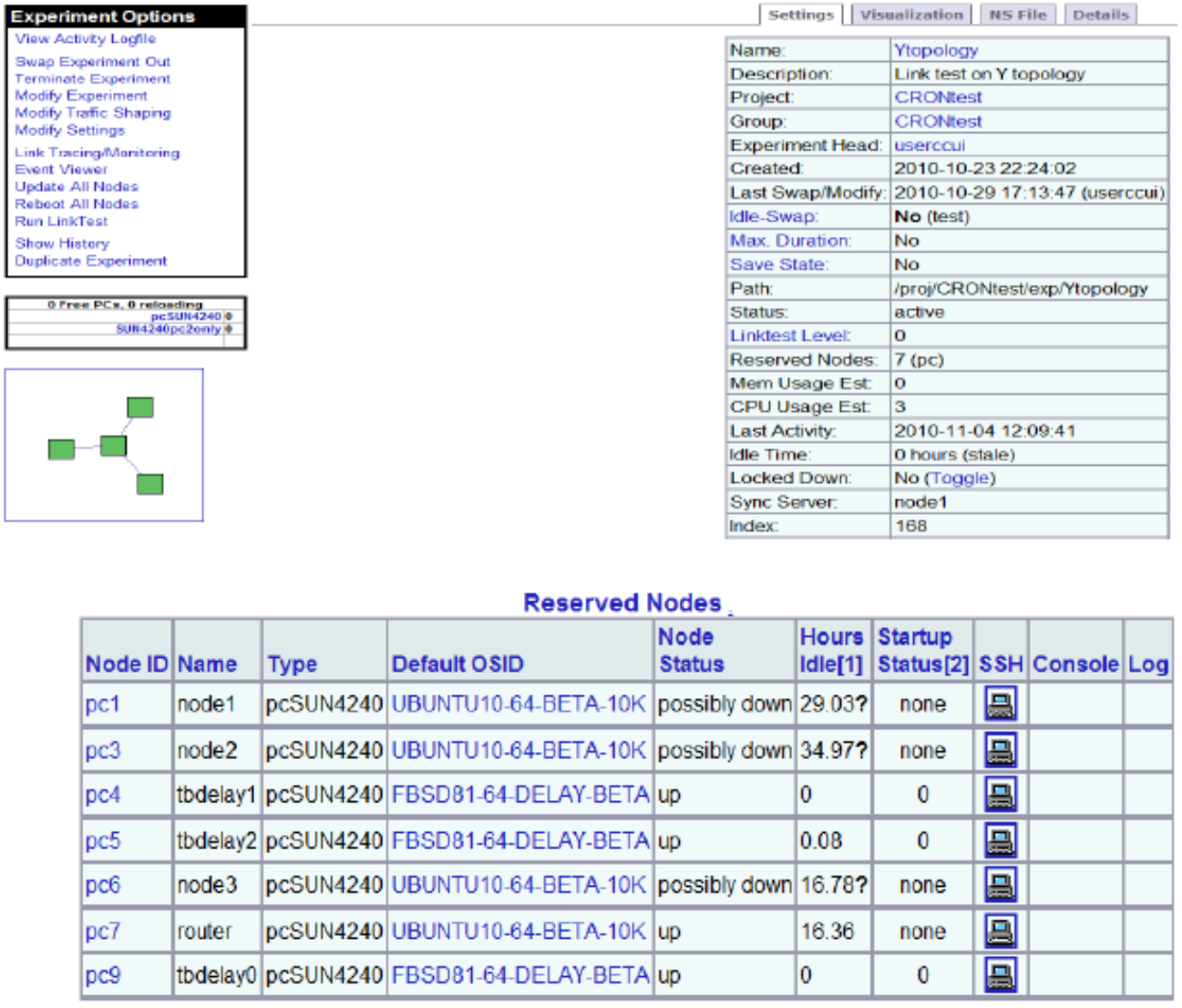

Figure 2.5: CRON Experimental information and node reservation as shown in cron.loni.org

Chapter 3

Observations on Existing Synchronization Loss Rate Equations

In data analysis, the thought and methodology behind choosing equations for metrics is critical to the validity of the work. With that in mind, this chapter will examine with a critical eye the use, strengths, and limitations of three variations of synchronization loss rate equations.

3.1 The Australian Equation

3.1.1 Equation Definition

In [**?**], the equation proposed by a research group based in the University of South Australia was based on the idea that if you have three flows with flow 1 and flow 2 are synchronized and flow 2 and flow 3 are synchronized then all flows 1, 2, and 3 are all synchronized. From this communities property an initial formula was proposed:

$$SR_i = \frac{n \times N_{Cor}}{\sum NL_i} \times 100\% \tag{3.1}$$

Where NL_i is the number of observed loss events for $flow_i$ and N_{Cor} is the number of synchronized loss events in a given flow where the synchronization involves the whole set of n flows. This equation gives the synchronization rate for a single flow. For a global synchronization loss rate they extended this equation to allow for equal representation for all SRs that were previously calculated:

$$SR = \frac{n \times \prod SR_i}{\sum((\prod SR_i)/SR_j)} \times 100\% \tag{3.2}$$

Where n is the number of concurrent flows and SR_i and SR_j are the calculated SR from 3.1 for each flow i or j.

3.1.2 Equation Evaluation

Networks that are handling a low number of flows such as 2 or 3, considering only loss events that involve the whole set of n flows or combination of pairs of flows may be reasonable. As this number of concurrent flows within a network goes up, however, the probability of all concurrent flow sustaining a loss event at the same time tends to dwindle. That is all that equation 3.1 is counting according to the definition given in [**?**]. In the experiments that were conducted for this paper, a total of 16 concurrent flows were observed. The number of loss events where all 16 flows were involved was less than 1% of the total number of loss events. Most of them involved a subset of the flows that usually consisted of 8 or less, but if this equation is used in true form to the definition then those particular events, which count for almost all of the loss events, go unnoticed.

Another way of looking at this equation is to consider it as a two tier equation. This means that there are two sets of equations involved to find the global synchronization loss rate. This can be an issue when dealing with a large number of concurrent flows or tests that last over a large time interval. Because of the nature of the algorithm, individual synchronization loss rate calculated for each combination of pairs of concurrent flow. Each of these combinations, for the most part, have equal weight in the calculation of global synchronization loss rate. This may not be a realistic evaluation of the performance of the network. It would only be completely correct if every combination of pairs of flows had a similar synchronization rate. If the results vary then the global synchronization loss rate will skew.

3.2 The Task Equation

3.2.1 Equation Definition

A more basic approach [**?**] was developed by the Internet Engineering Task Force based on thier definition of synchrnization loss rate. They cosidered synchronization loss rate to be the ratio of the degree at which two loss events happen symultaneously. The idea was that when one TCP flow of a pair has a loss event, the synchronization ratio is given by the fraction of those loss events for which the second flow has a loss event within one round-trip time. In other words the rate at which a $flow_i$ is synchronized with a $flow_j$, where $i \neq j$, can be calculated as the number of loss events that the pair of flows incounter simultaniously as N_{ij} divided by the total number of loss events that $flow_i$ encounters, N_i. That is

$$S_{ij} = N_{ij}/N_i \tag{3.3}$$

In this way $0 \leq N_{ij} \leq min(N_i, N_j)$ which means $0 \leq S_{ij} \leq 1$. Inversely $S_{j^i} = N_{ij}/N_j$ will give the synchronization loss rate for $flow_j$. Because of this fact they claim that the global synchronization loss rate can be claculated as:

$$S_{ij} = max(S_{ij}, S_{j^i}) \tag{3.4}$$

3.2.2 Equation Evaluation

The Task Equation is very simple and straightforward. The focus of this equation is the the rates that each flow encouters syncronized loss events and seperates them from the events where each flow comes across a loss event by itself.

Because of this equations very basic view on synchronization many of the problems it faces come when dealing with more complex networking scenarios. First, the equation can only evaluate two flows at a time. Under more realistic cercumstances, such as 16 concurrent flows, this can be very cumbersome. In order to get the compared rates of every combination of flow pairs $\binom{n}{2}$ calculations must be performed. For 16 flows 120 calculation combinations must be dealt with and recorded. This high number of calculations does not have the analytical integrity that is needed to depict a clear representation of the synchronization loss rate of either any given flow or network.

When trying to claculate global synchronization loss rates, this equation also stumbles. If we consider a situation where two concurrent flows share a bottleneck and $flow_1$ encounters a loss event every time $flow_2$ encouters one, but flow two encouters many more loss events that $flow_1$ does not, the global synchronization rate using equation 3.4 would be 1, which would mean that all flows would be completely synchronized within the given trial time. This would be a gross over estimation of the rate as considering the maximum compared value runs the risk of doing. Because of these issues the general usefulness of this equation greatly deminishes.

This equation does not come without some pros that should be considered. Like the Australian equation, there are no results of flows synchronized with itself counted as flase positives or false negatives in equation 3.3 since it actively states that it will not consider situations where $flow_i = flow_j$. Also the method in which it defines the variables needed to calculate the synchronization loss rate have merit as well.

3.3 The French Equation

3.3.1 Equation Definition

The third formula that was considered was derived by [?] at the Institut TELECOM in France. The authors of this paper define a loss event as an event that takes place every time one or more flows lose one or more packets in an interval Δ. With $T > 0$ indicating the total number of loss events that occur within a measuring time interval τ, M as the number of flows that share the bottleneck within the time period $\tau$and $d_{k,l}$ symbolizing whether a flow k drops one or more packets within the l-th lost event, where $d_{k,l} = 1$ if true and 0 otherwise, the equation is derived as such:

$$SR = \frac{1}{T}\sum_{l=1}^{T}\frac{1}{M}\sum_{k=1}^{M} d_{k,l} \tag{3.5}$$

The equation can also be manipulated to give the synchronization rate in terms of individual flows. Let S_{ij} be the fraction of loss events where flow i and flow j both suffer loss events. Also, let the number

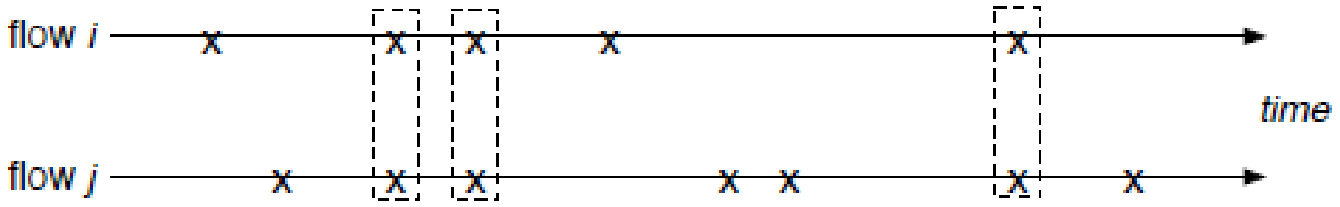

Figure 3.1: An example of synchronization.

of loss events where flow k loses one or more packets be denoted as $N_k \in [1,\ldots,\mathrm{T}] = \sum_{l=1}^{T} d_{k,l}$. So by definition:

$$\lambda_i = \frac{1}{T}\sum_{l=1}^{T} d_{i,l} \tag{3.6}$$

A value $\lambda = 1$ would stand for a flow I having all of its packet drops synchronized with one or more concurrent flows. Figure 3.1 Shows an example of two concurrent flows. Using equation 3.6 we can calculate that T =9, $\lambda = 5/9$ and $Lam = 7/9$.

3.3.2 Equation Evaluation

The French equation is a more directly focused and is functional under a wider breath of situation than the previous two equations. The theory behind the equation is sound, in the fact that it does give a practical definition of synchronization loss rates and created a metric that is both fairly accurate and dynamically intuitive to changes in network synchronization, in the sense that when flows drop packets with one or more flows the metric calculated by this equation is affected. The equation is also quite easy to use. Through a very simple add-on this equation can be added to the output of iperf with minimal memory use or any hindrance of the performance of the test or data recording mechanisms.

The issue comes in many of the details. First, the output of equation 3.5 only ranges from $R = [1/(M), 1]$. This means that if a flow is completely desynchronized with all other flows that share the bottleneck within the time interval τ, the lowest synchronization loss rate that any flow and in fact the network as a whole can receive is $1/M$. This is not a proper way of measuring synchronization loss rate when you compare it to its definition. According to the definition a flow i can only be considered synchronized with another flow $j \neq i$. This contradiction makes this equation inaccurate to the true synchronization loss rate. Counting a flows synchronization with itself also skews the overall calculation giving flows that have a high number of loss events more of a weight in calculating synchronization rate than those flows with less packet losses. This can cause synchronization loss rates to average to be much higher than they should be because flows that should be considered desynchronized are counted. On the other hand, loss events that involve more than one flow are given less weight which will shift the synchronization loss rate down since each of these scenarios are counted for less weight in the overall calculation than they should be.

Chapter 4

Defining the Revised Equation

In this chapter, based on the information gathered by the previous chapter we will modify the existing synchronization loss equations in a way that will keep their strengths while reducing the number of limitations.

4.1 Revised Australian Formula

As we saw in section 3.1, the main issue that was present when dealing with the Ausrtalian equation was dealing with the dividing by zero that occured when a pair of concurrent flows were completely desynchronized. To rectify this issue an $n \times n$ matrix was created to hold the individual synchronization loss rates for each flow i with respect to any other flow j, where i ,$j \in$(1,. . . ,n).

$$\begin{bmatrix} 1 & SR(1,2) & SR(1,3) & \dots \\ SR(2,1) & 1 & SR(2,3) & \dots \\ SR(3,1) & SR(3,2) & 1 & \dots \\ \dots & \dots & \dots & \dots \end{bmatrix}$$

Using this matrix a sub-global synchronization loss rate was calculated using the rates in the rows of the matrix which represented the synchronization loss rate each flow had with a particular flow i. An average was taken of these sub-global rates to get a global rate. Using this system the rates claculated along the main diagonal of the matrix were not used since these rates represent a flows synchronization with itself. These rates have no real meaning when considereing the metric of global synchronization rate and would only result in a mistakenly high reading. While this does give a result that is close to what the synchronization loss is, the number of times the rates must be manipulated has a tendency to skew the numbers in a way that can reduce the confidence in the resulting calculation.

4.2 Revised French/Task Merged Equation

When revising the French formula, the thing that is most important, and what should be the focal point the equation, is the way it defines the term being analyzed. This is the backbone that sets all variations of equations apart. the way that we chose to view synchronization loss rate is the percent chance that

anytime a $flow_i$ encounters a loss event there is a different $flow_j$, where j≠i, that encouters a loss event within a given time interval τ.

Through this definition a variation of the French synchronization loss equation was created as an adaptation of equations 3.3 and 3.6. The French equation was chosen because it returned a value that had the meaning closest to what we define to be the correct synchronization loss rate, but as we stated earlier in Chaper 3, the French equation counts a flow being synchronized only with itself as a synchronized loss event. The revised equation disregards these events, and, like equaiton 3.6, only focuses on the situations where two or more flows encounter loss events synchronously. Unlike 3.6 the revised French equation can consider more than two concurrent flows and does a better job at calculating global synchronization that is not as affected by outlying flows.

Let n be the number of concurrent flows that share a bottleneck within an interval T. At every smaller interval τthere can be a number of loss events that occur ranging from [1 ... n], but since we are not counting single loss events by one flow as a synchronized loss event, we will disregard one loss event which simulates the loss event encountered by the given $flow_i$. Lets have that number be represented by NL. Over the course of time interval T each $flow_i$ will encounter $T_{drop(i)}$ loss events which can range from [0, ..., T]. The calculation constructed to find the syncronization loss rate for any $flow_i$ is as such:

$$\lambda_i = \frac{\sum_{k=1}^{T} \frac{NL_i - 1}{n-1}}{T_{drop(i)}} \tag{4.1}$$

Just looking at the two equations, 3.6 has the same general structure as 4.1, but if you consider the same scenario that was proposed by Figure 3.1 the difference becomes quite noticeable. Under 3.6 λ_1 was calculated to be 5/9 and λ_2 was 7/9. Under the revised French equation λ_1 is 3/5 and λ_2 is 3/7.

In this way the revised French equation has more of a likeness to the Task equation in that it only counts the events that the concurrent flows in question share, but unlike 3.3, the revised French equation can be used to calculate the synchronization loss rate of a single flow as compared to any number of concurrent flows. It can also do this without a large number of calculations and comparisons. This makes it much easier to have general use while still providing a analytical metric that takes into account all possibilities of rates.

Now that we have n seperate synchronization rates corresponding to the n concurrent flows the question of finding a fair global calculation becomes an issue. The things we want to get from this equation is for it to give a reasonable solution while representing each of the smallest units equally, the smallest unit being the individual loss events. Seeing as our individual flow equation does all of these things, starting with that equation was the best first step. In order to account for all of the loss evetns individually we needed to find the ratio of sycnhronized loss events each flow encountered to total units loss events for each flow. With that in mind, the global equation is:

$$SR = \sum_{i=1}^{n} \frac{\left(\frac{\sum_{k=1}^{T} NL_i - 1}{T_{drop(i)}}\right)}{n(n-1)} \tag{4.2}$$

By calculating the average this way, each loss event has equal weight which will give a fair result that is more reflective of the events as a whole.

Chapter 5

Results

In this chapter we will view the results of the proposed equations on emulated network result.

5.1 Experimental Routines

With help from the CRON test bed, the network set up that was mentioned in Chapter 2 were emplemented and aseries of trials were run to procure emulated data that made it possible to test the various equation. Each trial was performed for 1800 seconds. This time frame was chosen because it gives enough time for even the smallest packet drop per minute set up, Reno with 100% queue size, to record enough packet drops to obtain a reasonable measurement. Because of the limitations of the *iperf* software, *cwnd* values and instant throughput readings were gathered every 0.5 seconds. Any finer grained calculations were either below *iperf*'s coded range or affected the performance of the network in a way that did not reflect the tuned specification that were wanted for the trial. Each variation of tests was performed 5 times and an average was given for each result.

There are essentially three ways to view the results of these experimental trials. The first is to compare the revised versions of each equation with its original predecessor. Understanding the differences that the logical changes that were performed on the previously proposed equations and how those changes affect the outcome of the synchronization loss rate is crutial to validating the accuracy and necessity of those changes. The Second way of viewing these results and the more general view is to look at all of the equations individually against each other. Each equation looks at the issue of calculating synchronization of loss events in a different light and because of that returns different answers in result. While some results are more precise to the reality of the network than others each does provide some insight. The third way of viewing these results, while is not directly tied into the scope of this paper, still gives interesting information that can be used in future testing. That is to compare the resutls of the different TCP variants. Looking at how each variant affects the overall synchronization loss rate under the same experimental situations can help to pinpoint some truths about the affects of synchronization to overall throughput produced by each of these variants.

Figure 5.1 shows the synchronization loss rate from the first set of trials with the TCP variation set as Reno calculated by four of the five methods. The graph spans over a range of queue sizes based on percentages of BDP(10%, 20%, 40%, 70%, and 100%). The increase in queue sizes gives more space for incoming packets to be held while they are waiting to be processed by the receiver which should decreases the frequency of loss events and, as shown by most of the results in the figure 5.1, decrease

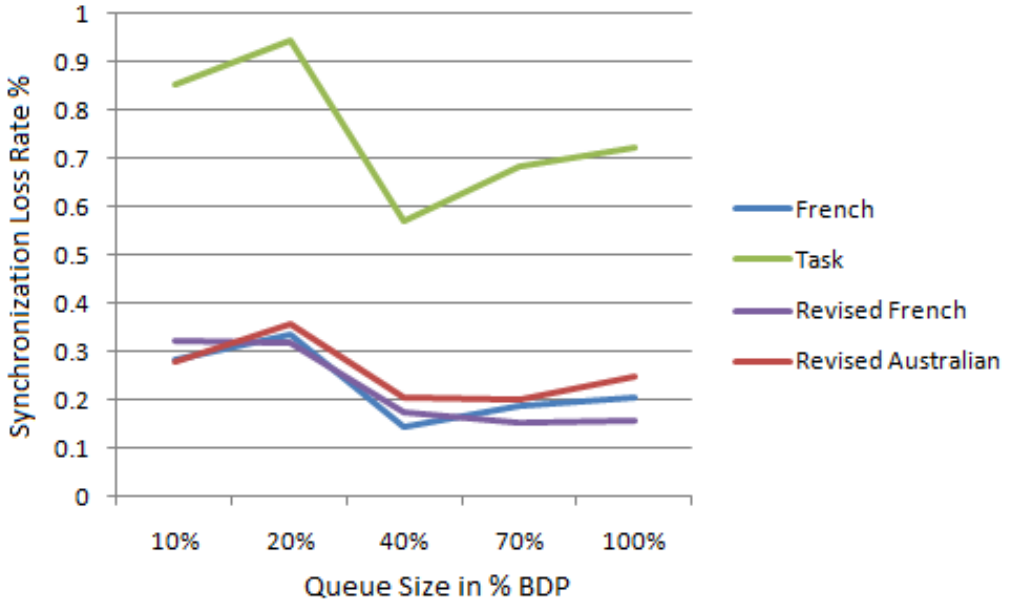

Figure 5.1: Results of the Synchronization Loss Rate Calculation with TCP Reno flows on the CRON testbed.

the chance for loss events of concurrent flows to be synchronized. Most notibly is the results for the Task equation. Since the global sychnronization loss rate is based largely on the highest value that is recorded, not only is the value much higher than the rest of the equation results, it is also the most inconsistant with the pattern that was previously stated involving the size of queues to synchronization rate. There are some results that show it as being completely synchronized which is a gross exageration.

It would be good to note that all of these calculation methods were performed on the same sets of data. The differences that are seen are based solely on the performance of the different equations. Also to note is that in figure 5.3 the results for the original version of the Australian equation are not represented. The method of using the equation proposed in [?] was highly sinsitive to combinations of flows that were completely desynchronized. A number of trials that used TCP Reno had this scenario occur and the results of that methodology were of no use (i.e. resulted in division by zero). Another thing that can be noticced is the increased stability of the newer version of the equation. The extreme changes in the original Australian formula has been found to come from the flows' diversity in pair synchronization rates. When the flow pairs all have rates within a relatively small range, the formula gives a more stable global rate, but if they are within a larger range then the accuracy of the global rate calculated using the original equation tends to become much lower than what can be tolerated for a synchronization loss rate.

In figure 5.1, other than the Task and original Australian equation, the results seem almost comparable. The newly revised French algorithm proposed in this paper tends to be always somewhere in between the other two equations with the original French equation having a little lower percentage and the Australian equation higher, but all giving ranges from the mid to upper twentieth percentile with larger queues to the upper fortith to the upper fiftieth percentile with lower queues.

The main motivation for the modification of the French equation was to accurately count only synchronized loss events. The original equation gave false results because it counted loss events that involved only one flow as synchronized loss events. By changing the focus to the number of flows any given flow is synchronized to, this problem is done away with. This change actually caused a second difference in the calculation results as well. When more than one flow encounter synchronized loss events, the original French equation would count each of these flows to find that average number of flows that dropped at

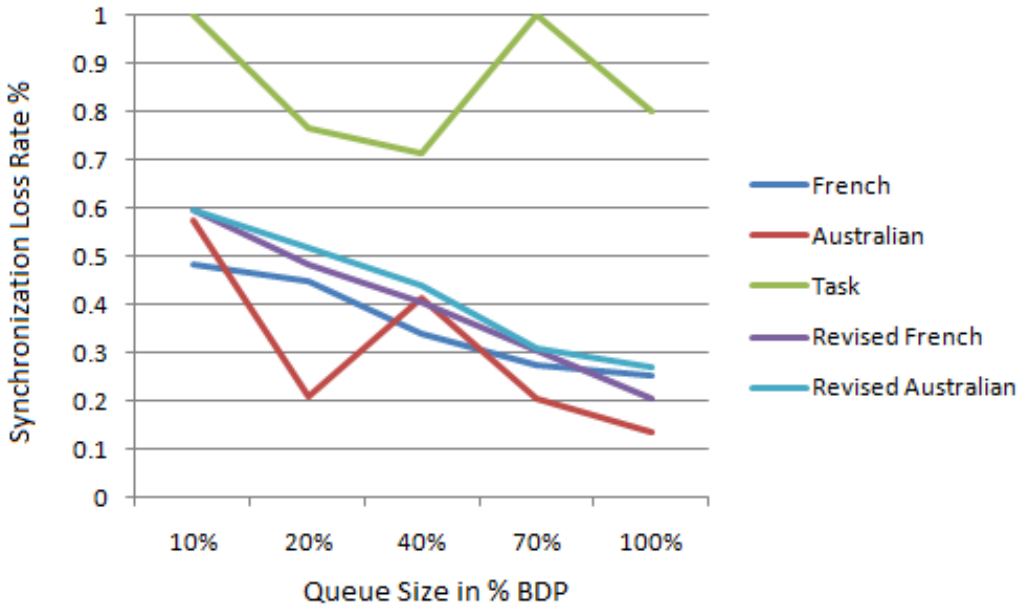

Figure 5.2: Results of the Synchronization Loss Rate Calculation with Cubic TCP flows on the CRON testbed.

every loss event. With the revised French equation counting only the number of flows any give flow synchronized with reduced the number of counted flows by 1 each time but raised each flows weight. That increase in weight for multiple synchronized loss events can account for the higher synchronization loss rates that are shown accross the board by the revised French equation.

As shown in figure 5.4 representing the Reno trials, completely desynchronized loss events, that is loss evetns where only one flow drops a packet, is 9% of the total number of loss events, in figure 5.5 representing the CUBIC trials, the percentage of completely desynchronized loss events is at 11%, and in fugure 5.6 representing the HSTCP trials the percentage was 11%. While these do represnt a small portion of the loss events, they do represent a significant amount of false positive readings that under highly desynchronized networks such as those shown in figure 5.1. These false positive readings are believed to be the cause of the irrregular pattern in the original French equation

Figure 5.2 shows the results from the synchronization loss rates from the second trials which used CUBIC TCP. As seen with the TCP Reno trials, the Task equation not only shows unrealistic results, it also shows no regular pattern. As we have stated before, this can be deamed as evidense that the Task equation is not a suitable equation that can give any reliable or workable data by itself solely because of it choses the maximum of the synchronization rates as the global synchronization loss rate.

The CUBIC trials are the first chance that the results of the original Australian equation gave results that did not suffer from flow pairs that were completely desynchronized. Even so the results are still quite volitile resulting in the similar irregular results that were seen in the Task equation results. This results, as stated above has to do with the vast range of results of synchronization rate among pairs of concurrent flows. The results of synchronization loss rates did tend to be higher when comparing flows that came from the same source and lower when comparing flows from different sources. This may have to do with some sycnronization that happened before the flows reached the bottleneck which could have skewed the results. This situation will have to be resolved in further testing later.

Another result that was similar to that of the Reno trials was that the other three eqautions are very similar in their results with the smaller queue sizes where they land in the upper twentieth to lower thirtieth percentile. There is a discrepancy in the results involving test using larger queue sizes. The

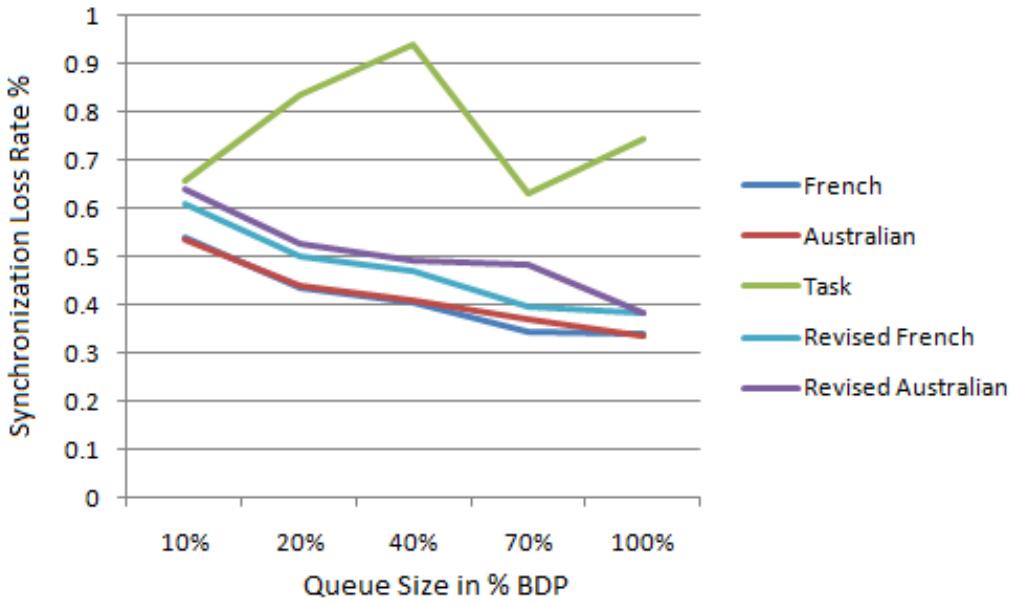

Figure 5.3: Results of the Synchronization Loss Rate Calculation with HSTCP flows on the CRON testbed.

results shown in figure 5.2 reveal that the original French equations has a slight rise in synchronization loss rate with reading in the mid to lower 20th percentile, while the revised French equation reports rates in the mid teens. This result is believed to be caused by a high amount of false positive results in these trials. Figure 5.5 shows how the number of flows that are participating in loss events is distributed during the CUBIC trials. The amount of desynchronized flows reach an average of almost 11%. What is not shown in this figure is that of the average 244 false positive results that were recorded during the CUBIC trials, and average of 94 of them resulted in the tests where the queue size was based on 100% BDP. During these sets of trials single flow loss events jumped to 25% of the total loss events. That means one in every four loss events that were counted in the original French equation were disregarded in the revised version. Even with this vast difference the recorded global synchronization loss rate are comparable to each other. This gives evidence of the first issue stated above regarding the change in distribution of weights for each flow.

The final set of trials that were performed for this paper involved HSTCP. Figure 5.3 show the results of the different equations with respect to the receiver's queue size. Other than the Task equation, which once again does not produce a reasonable result, the other equations give the most similar results they have given in the span of these trials. As the figure shows both the original French and original Australian equations are almost equal. One of the most noticable differences that can be seen with HSTCP is that there are a higher number of loss events that are encountered by more than 8 flows. This had a large affect on the higher synchronization loss rates that are recorded. Figure 5.6 shows that almost 20% of the loss events involved more than 8 flows. This is largely because of HSTCP's aggressive algorithm which causes cwnd values to increase rapidly each RTT. With this constant burst of cwnd values comes a constant occurance of loss events as the summation of these cwnd values tend to overlap the capacity of the link very quickly. In fact out of the three TCP variations as shown in Table 5.1 HSTCP encountered the most loss events throughout the course of the trials with CUBIC TCP in third and TCP Reno in last.

Along with having an unquestionable lead in the metrics of highest link utilization, highest throughput-put, and most loss events, HSTCP also dominated the charts in synchronization loss rate. Comparing the values calculated by the revised French equation, HSTCP averaged 47%. CUBIC TCP averaged a little under 40%. TCP Reno averaged only 22% synchronization loss rate. This was very predictable

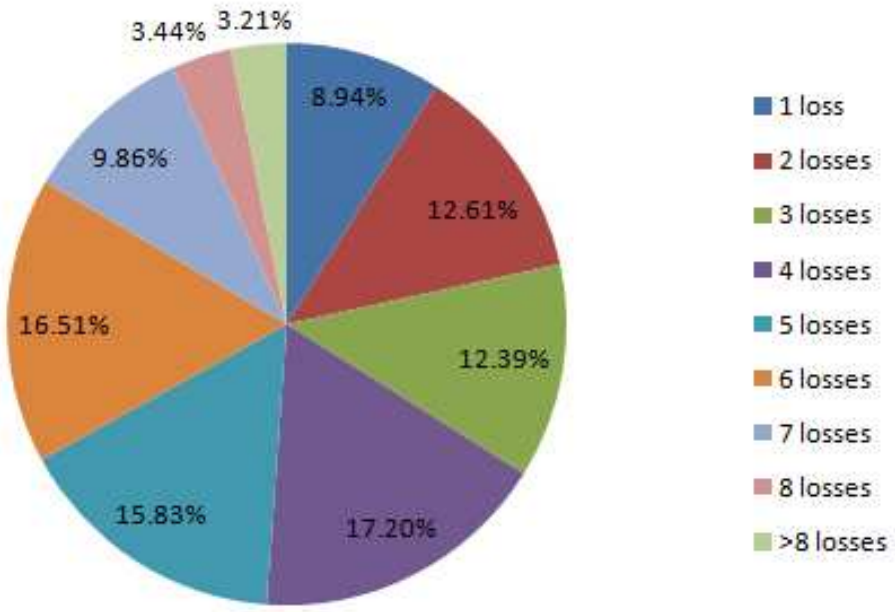

Figure 5.4: The distribution of loss events that occured throughout the TCP Reno tests categorized by the number of flows that participate.

Table 5.1: Average Loss Events

TCP Variant	Average
TCP Reno	438
CUBIC TCP	2662
HSTCP	4601

since each TCP variant developed synchronization of their losses more readily the more aggressive their cwnd increase became.

As was stated in the begining of this chapter all of these results come from a series of five test for each network scenario. Table 5.2 shows the standard deviation of the synchronization loss rate calculated by the five equations seperated by TCP variants. Through the standard deviation we can indicate the stability of the equation and how it reacts to changes in network patterns. As was in most of the other tests it was involved in, the Task equation stands out with the highest deviation per test. This is again because it is asking fot the maximum synchronized rate among all flow pairs. This value is extremely volitile which make the algorithm results fluctuate to a large degree. Also as seen in previous results, the original Australian equation returned unusable results in the Reno trials, but becoming less varied as the trials moved on to CUBIC and HSTCP. It was still one of the higher fluctuating result groups. Of the final three equations, the original French and revised Australian equations were off by just the slightest amount. The revised Australian equation had a 27.6% smaller standard deviation than its predicessor. The smallest standard deviation on average was given by the revised French equation an 11.3% smaller deviation than the original French equation.

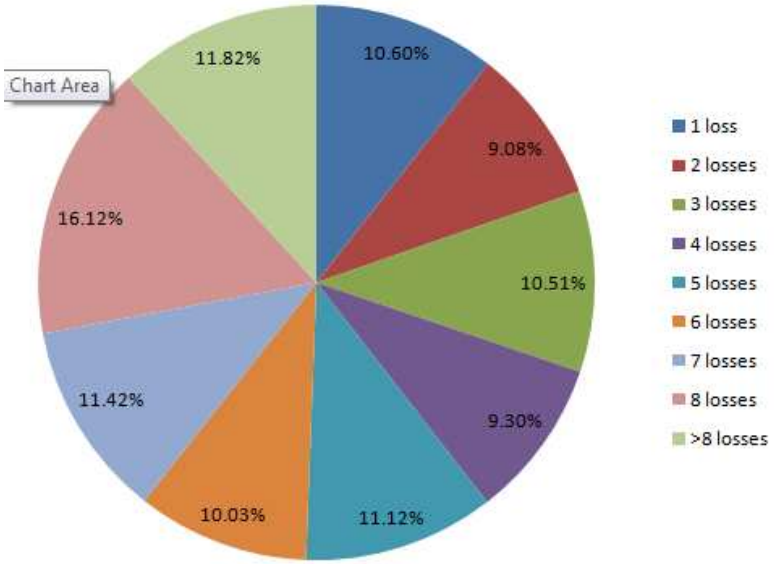

Figure 5.5: The distribution of loss events that occured throughout the CUBIC TCP tests categorized by the number of flows that participate.

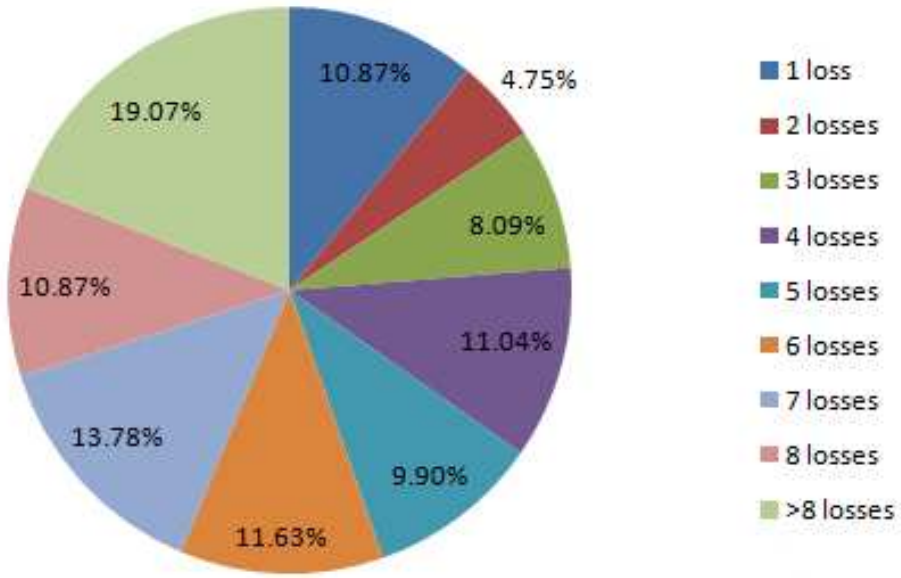

Figure 5.6: The distribution of loss events that occured throughout the HSTCP tests categorized by the number of flows that participate.

Table 5.2: Standard Deviation of Synchronization Loss Rate Equations

Equation	TCP Variation	10%	20%	40%	70%	100%
Original French	Reno	.031	.042	.034	.024	.018
	CUBIC	.033	.081	.039	.045	.053
	HSTCP	.042	.064	.046	.051	.054
Original Australian	Reno	N/A	N/A	N/A	N/A	N/A
	CUBIC	.091	.182	.093	.054	.083
	HSTCP	.052	.064	.052	.078	.052
Task	Reno	.082	.030	.212	.124	.218
	CUBIC	.000	.081	.129	.000	.153
	HSTCP	.207	.084	.045	.167	.144
Revised French	Reno	.028	.038	.031	.020	.015
	CUBIC	.029	.070	.035	.039	.047
	HSTCP	.039	.058	.037	.049	.048
Revised Australian	Reno	.036	.051	.034	.055	.041
	CUBIC	.041	.067	.043	.067	.043
	HSTCP	.042	.072	.042	.071	.049

Chapter 6

Conclusion and Future Work

With high speed networks running at and above 10Gbps becoming a more widely used hardware option, the ability to utilize this new technology to its full potential is one of the most critical topics in networking research today. Finding out the secrets behind optimum link utilization will take many man hours of testing and research, but all of that testing and research will be useless without having properly turned devises and metrics to base the results of these tests on. One of the leading theories behind increasing link utlilization is the reduction in synchronization of loss events for cuncurrent flows within a network, but until now has there been an equation created that provides a proper view of the synchronization of these loss events that could be used as a general formula for all network situations.

In this paper, A number of equations were evaluated through extensive network simulations using the CRON testbed to find their strengths and fault. Each equation gave a new perspective and chalenged the very definition of the metric but none gave the full picture. The problem either came from the equations only able to handle pairs of concurrent flows, counting false positive cases, or misrepresenting the weight given to loss event based on the frequency of loss events encountered by each given flow. Because of this, new variations of these equations were derived based on the results of this investigation which took into account what was found to be the proper elements needed to calculate a flows state of synchronization. Though these efforts more accurate testing may be done in the hope of tuning network synchronization in a way that help to decipher its validity as a key metric in improving link utilization, overall throughput and buffer optimization.

While this paper gives validation to the logic and definition of the revised metrics, they still can be improved upon. The tests were based on a very large grained scale which gave a good but not perfect representation of the status of the network. This was allowed because of limitations in the software's ability to keep track of the high amounts of information without affecting the performance of the traffic generation. Using a more fine turned system will allow for not only more precise calculations but also different ways of looking at the loss events which could allow for more accurate metric definitions.

Bibliography

[1] `http://cron.loni.org`.

[2] Marta Carbone and Luigi Rizzo. Dummynet revisited, April 2010.

[3] Arash Dana and Ahmad Malekloo. Performance comparison between active and passive queue management. *International Journal of Computer Science*, 7(3), 2010.

[4] Kevin Fall and Sally Floyd. Simulation-based comparisons of tahoe, reno, and sack tcp. *COMPUTER COMMUNICATION REVIEW*, 26:5–21, 1996.

[5] S. Floyd. Rfc 3649-highspeed tcp for large congestion windows. 2003.

[6] S. Floyd and E. Kohler. Tools for the evaluation of simulation and testbed scenarios. `http://tools.ietf.org/html/draft-irtf-tmrg-tools-05`, 2006.

[7] Qiang Fu and Philip Jay. A step towards understanding loss synchronisation between concurrent tcp flows. 2008.

[8] Sangtae Ha, Injong Rhee, and Lisong Xu. Cubic: a new tcp-friendly high-speed tcp variant. *SIGOPS Oper. Syst. Rev.*, 42:64–74, July 2008.

[9] Sofiane Hassayoun and David Ros. Loss synchronization, router buffer sizing and high-speed tcp versions: Adding red to the mix. In *LCN'09*, pages 569–576, 2009.

[10] Vishwanadh Raparthi. Experimentations of the high speed networks with hardware and software emulators, 2010.

[11] Chakchai So-in and Chakchai So-in. Loss synchronization of tcp connections at a shared bottleneck link, 2006.

[12] David X. Wei, Pei Cao, and Steven H. Low. A case for increasing tcp loss synchronization rate in cluster computation networks, 2009.

Appendix: TCP Variations

Thoughout this paper different variations of TCP were used to differentiate the characteristics and rates of cwnd value and frequency of packet drops. In this section, I will give a basic description of each variant allong with a closer view of thier performance in the trials. The three TCP variants that will be discussed are TCP Reno, CUBIC TCP and HSTCP.

Before going into the details of the variants, I would like to refresh the basic idea of TCP and the steps that all variants must contain. When a TCP flow starts, it is in the phase known as slow start [**?**]. In this phase the *cwnd* starts at one packet. It causes the cwnd to increase by 1 for every acknowledgement it receives from the destination. Once the cwnd reaches a predefined threshold it converts to the congestion avoidance phase, where the cwnd increases based on the set formula of the TCP variant. When the rate of transmission becomes too much for the network packets get dropped.

When the receiver realizes this either by waiting a time limit or recieving packets out of order it will send a *dup ack*, a duplicate of the previously send acknowledgement, to the sender. If three dup acks are received before the wanted packet reaches the receiver, the sender considers this as a lost packet and quickly sendes out the lost packet again. This phase is called *Fast Retransmit*. After this happens the TCP variant will reduce the cwnd by a predefined multiplicative amount and congestion avoidance starts again. Going to congestion avoidance phase instead of returning to a cwnd of 1 and starting slow start phase again is known as *Fast Recovery* [**?**].

TCP Reno

TCP Reno is one of the oldest, but still most widely used variation of TCP [**?**]. It, like all the other version of TCP, start at the slow start phase. Then when the predefined threshold is reached, it converts to the congestion avoidance phase where the cwnd increases by 1 after every RTT that does not result either three dup acks or a timeout. If three dup acks are received then it reduces its cwnd by half and enters fast retransmit then fast recovery. If a timeout occurs, which is when no acks are received within the given time interval, then the cwnd drops to 1 and slow start is inacted [**?**].

Figure 6.1 shows the performance of TCP Reno on the CRON Emulation testbed [**?**]. As the figure shows, TCP Reno has a very slow congestion avoidance process and any packet losses will only slow this process even further. Under the scenario of this experiment, 10Gbps netowrks with 120ms round trip time and 1000 byte packets, a single flow using TCP Reno would have to have a congestion window of 100,000 segments and a single packet loss every 5,000,000,000 packets. Because of this there are sources that believe it to not be a realistic approach to fully utilize high speed networks in a real time situation. This is the main motivation of research in high speed TCP variations. All of the versions of TCP that were used for this experiment are variations on Reno and more specifically focus on modifications of the congestion control phase [**?**].

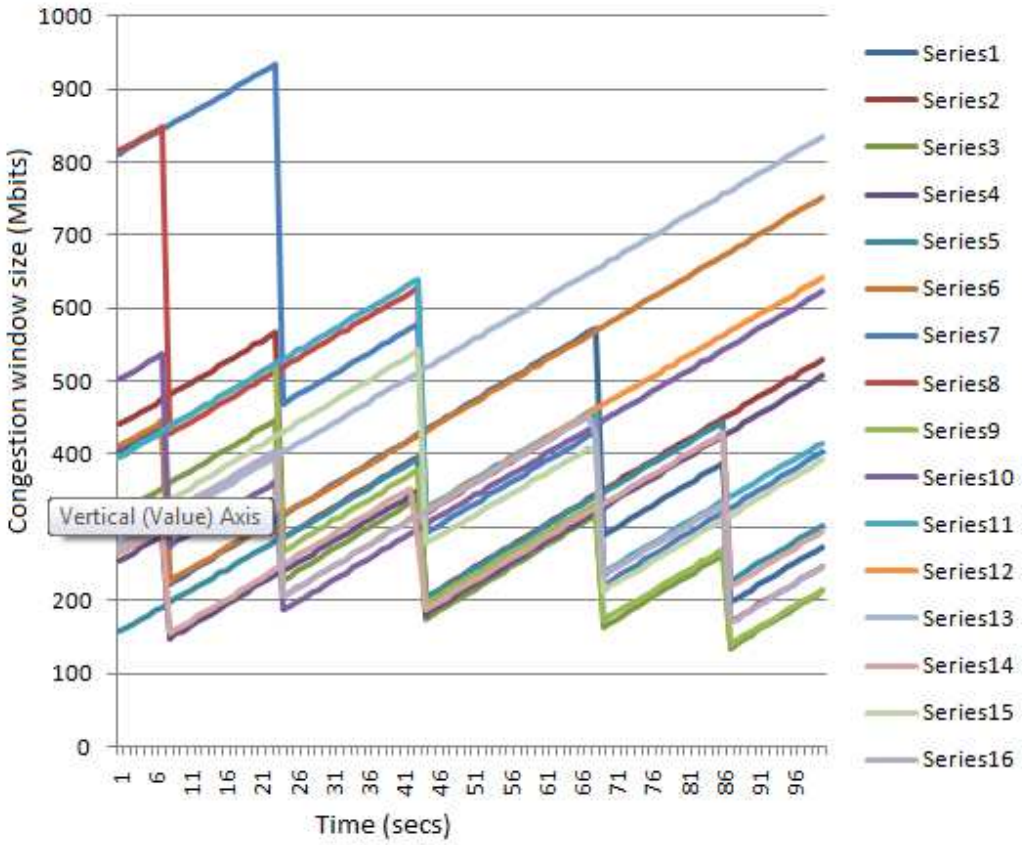

Figure 6.1: A snapshot of the performance of the 16 flows while using TCP Reno on the CRON testbed with a queue size of 10% BDP.

CUBIC TCP

CUBIC TCP is actually a second generation modifies version of TCP Reno. Its predicesor BIC, which sands for Binary Increase Congestion Control, was considered to be much too aggressive for TCP and caused a low production in throughput [**?**]. Also the window control algorithm was complex because of it having multiple phases. CUBIC takes its scalability and stability from methodologies that are present in BIC which cause a very impressive improvement in both TCP and RTT fairness compared to other TCP variants. This can be attributed to CUBIC's throughput in congestion avoidance to be only defined by packet losses. It is completely independent of RTT which helps CUBIC to perform well in situations with high packet loss rates which tend to happen which it is inacted in high speed networks [**?**].

The equation behind the cwnd growth durring the congestion avoidance phase works just as the name suggests. The function performs by growing on a cubic scale: [**?**]

$$W_{cubic} = C(t-K)^3 + W_{max} \tag{6.1}$$

C is a scaling factor that can be preset by the source, t is a the amount of time that has elapsed between the current RTT and the time at which the previous drop in cwnd occured, W_{max} is the window size just before the previous drop in cwnd, and $K = \sqrt[3]{W_{max}\beta/C}$, where $\beta$is a constant multiplication decrease factor applied for window reduction at the time of a loss event at which point the window reduces to βW_{max}. After the drop in cwnd size the window will tend to grow at a faster pace when it is father away from W_{max} and at a slow pace as it gets closer to W_{max} and almost no growth when it reaches

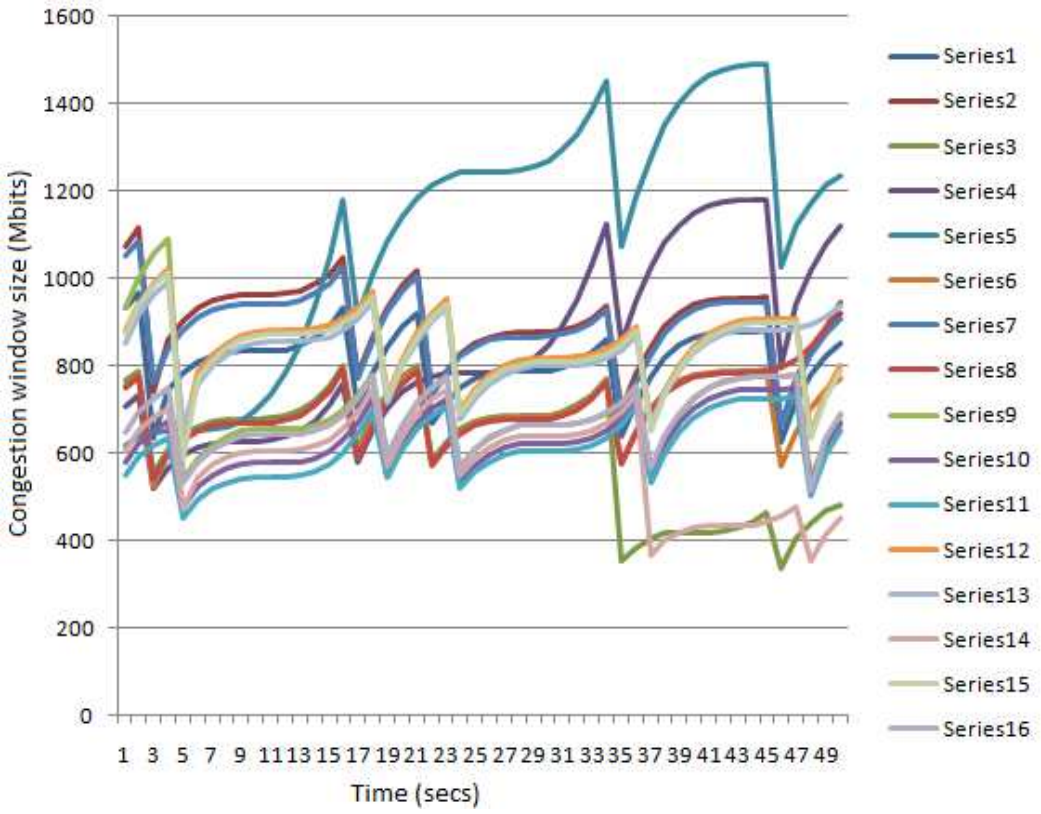

Figure 6.2: A snapshot of the performance of the 16 flows while using CUBIC TCP on the CRON testbed with a queue size of 10% BDP.

W_{max}. It will then start to rapidly increase its growth as it gets above and further away from W_{max} while it probes for more bandwidth. The scalability comes from the rapid growth after W_{max} is passed while the stability factor is achieved by the slowing of growth as the cwnd value gets closer to W_{max} while increasing the untilization of the network. [**?**].

Figure 6.2 shows a 50 second snapshot of the performance of the flows using CUBIC TCP on the CRON testbed. The most noticable difference is that there are a much higher number of loss events. This is caused by the fact that CUBIC TCP has a much more aggressive congestion avoidance algorithm. This allows for flows to reach higher levels of throughput at a much facter pace. The higher packet loss rate also can also be considered to contribute to the higher synchronization rates. With more packet losses happening the chance of concurrent flows dropping their congestion window in the same RTT is greatly increased.

HSTCP TCP

HSTCP stand for High Speed TCP. This alludes to the fact that its congestion control algorithm has been tailored to support high bandwitch networks. HSTCP was created by following five guidelines:

- Achieve high per-connection throughput without requiring unrealistically low packet loss rates.
- Reach high throughput reasonably quickly when in slow-start
- Reach high throughput without overly long delays when recovering from multiple retrandmit timeouts, or when ramping-up from a period with small congestion windows.
- No additional feedback from TCP receivers.

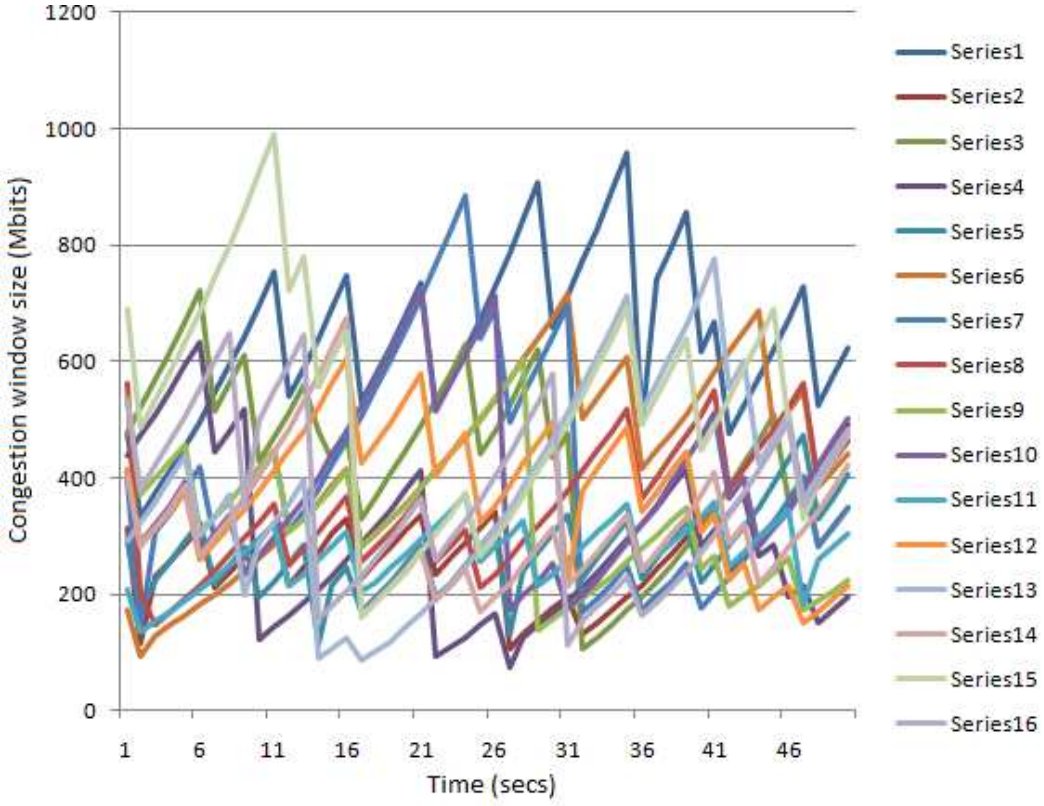

Figure 6.3: A snapshot of the performance of the 16 flows while using High Speed TCP on the CRON testbed with a queue size of 10% BDP.

- TCP-compatible performance in environments with moderate or high congestion.

The different approach this variant brings is a change in the increase and decrease parameters. The parameters used in this agorithm are *Low_window*, *High_window*, and *High_P*. The will be used to regulate the TCP response fundtion. There are two phases involved in the peformance of HSTCP. The first phase happens when the congestion window w is lower than Low_window [**?**]. In this case HSTCP performs standard TCP. Once w goes above Low_window the HSTCP changes the TCP response function which is

$$w = (\frac{p}{Low\_P})^S \times Low\_window \tag{6.2}$$

where *Low_P* is packet drop rate and S is a constant which is defined as

$$S = \frac{(logHigh\_window - logLow\_window)}{(logHigh\_P - logLow\_P)} \tag{6.3}$$

In order to surpass the stardard TCP principles of Additive Increase Multiplicative Decrease (AIMD) and increasing congestion window by 1MSS (Maximum Segment Size) for every RTT which makes it increasingly difficult to achieve and sustain the high speed response function HSTCP modifies their algorithm to moderate increase and timely decrease. HSTCP increases the cwnd by a(w) and decreases it by b(W). Both functions are dynamic formulas that are given below [**?**].

$$b(w) = \frac{(High\_Decrease - 0.5)(log(w) - log(W))}{(log(log(W\_1) - log(W)) + 0.5)} \tag{6.4}$$

$$a(w) = \frac{(w^2 * p(w) * 2 * b(w))}{(2 - b(w))} \tag{6.5}$$

Where High_Decrease is b(w) in case of w = High_window and p(w) is the packet drop rate for congestion window of size w.

While High Speed TCP allows for much higher bandwidths to be reached and sustained, one of its biggests weaknesses is that of TCP fairness. As figure 6.3 shows, There is a very wide spread in the utilization of bandwidth. While some flows are able to secure 800Mbps others are left with only 200Mbps. This is not an ideal situation in real life application where every user wants to have an equal share fo the bandwidth. However, this vast difference in fairness does not reduce the synchronization loss rate of the flows of HSTCP. In fact it has been shown to have some of the highest sychronization loss rate on average. This may be cause by HSTCP's aggressive nature.

Vita

Aaron M. Tureau is a native of Saint Amant, Louisiana. He graduated high school from St. Amant High and received his Bachelor of Science in computer science from Louisiana State University. He is planning on proceeding into a career in computer networking after he graduates in the Summer of 2011 with a Masters of Science in System Science.