A STUDY ON THE PERFORMANCE OF TRANSPORT PROTOCOLS COMBINING EXPLICIT ROUTER FEEDBACK WITH WINDOW CONTROL ALGORITHMS

By

AARTHI HARNA TRIVESALOOR NARAYANAN

Master of Science in Computer Science

Oklahoma State University

Stillwater, OK

2005

Submitted to the Faculty of the
Graduate College of the
Oklahoma State University
in partial fulfillment of
the requirements for
the Degree of
MASTER OF SCIENCE
December, 2005

A STUDY ON THE PERFORMANCE OF TRANSPORT

PROTOCOLS COMBINING EXPLICIT ROUTER

FEEDBACK WITH WINDOW CONTROL

ALGORITHMS

Thesis Approved:

Dr. V Sarangan

Thesis Advisor

Dr. J Thomas

Dr. N Park

A. Gordon Emslie

Dean of the Graduate College

ACKNOWLEDGEMENTS

My two and a half years stay at Oklahoma State University has been wrought with many changes in my life. Through it all, I am lucky to have the blessings of The Almighty and my parents without which nothing would have been possible.

I consider myself fortunate enough to have worked under my Advisor Dr. Venkatesh Sarangan whose able guidance; intelligent supervision and understanding helped me finish my thesis. I would also like to extend my sincere appreciation to my committee members Dr. Johnson Thomas and Dr. Nophill Park whose encouragement and support proved invaluable.

I would also like to thank my relatives and other well wishers back home for their love and support. I am fortunate to have a lot of good friends. I would like to thank them for their immense help, understanding and support they provided through all my good times and bad. I would like to thank my supervisors at work especially, Dr. Earl Mitchell, Dr. Yousif Sherif and Kristi Elrich for believing in me.

Finally, I would like to thank the Computer Science Department for supporting me during my two and a half years of study.

TABLE OF CONTENTS

LIST OF TABLES

Table Page

LIST OF FIGURES

Figure Page

I: INTRODUCTION

The Transfer control protocol (TCP) is a connection-oriented, reliable, byte stream, end-to-end transport layer protocol. TCP provides an interface between a network layer and the application layer above. During data transfer, TCP senders and receivers can alter the flow of data. This is called flow control, congestion control and/or congestion avoidance. TCP is a complex and an evolving protocol. For the last two decades, TCP [16] has been the most widely used protocol on the Internet. During these two decades, the network has evolved drastically with changes in power, network speeds and storage capacity. Over the years, multiple enhancements have been proposed to alleviate the various problems faced by TCP like Random Early Detection (RED) [5,6], Random Early Marking (REM) [7], Fast Recovery [8], Slow Start [8], Fast Retransmit [8], Forward Acknowledgement (FACK) [9,10], Selective Acknowledgement (SACK) [11], and High-speed TCP [12] being some of them. Although significant improvements have been made to it over the years, due to increase in the use of TCP in the Internet for various applications, a need for better methods to avoid undue congestion has become a necessity.

TCP uses packet drops as a way of detecting congestion and thereby limiting its sending rate. This implicit signal gives very little information on the type of congestion in the network. Thus, even if the congestion level is quite low, irrespective of how many packets are dropped, the window size is decreased drastically. Such a binary nature of feedback restricts the utilization of high-bandwidth links. Another limitation of basing the control laws on packet drops is that apart from not giving any feedback, but it is also not possible to detect the packet loss quickly either. As timeout is generally equal to or greater than the RTT of the network, TCP will use longer time to detect a packet loss, especially in high delay networks where the RTT is high. This process limits bandwidth utilization and the problem only worsens with increased bandwidth and delay.

TCP typically uses a sliding window protocol that provides handling during both timeouts and retransmissions. TCP make use of sliding window algorithms as a part of flow control to reliably handle loss, minimize errors, manage congestion and perform efficiently in high-speed environments. SWP (Sliding Window Protocol) is a connection-less protocol. It allows data to be sent in one direction between a pair of protocol entities, subject to a maximum number of unacknowledged messages. If SWP is operated with a window size of 1, it is equivalent to the Alternating Bit Protocol. Both hosts use a window for each connection. A congestion window in TCP is not only determined by the receiver but also by the congestion in the network. The sender has the receiver's advertised window as well as the congestion window size. The actual window size would be the minimum of these two.

Actual window size= minimum (receiver window size, congestion window size)

During steady-state, TCP uses Congestion Avoidance algorithms to linearly increase the value of the congestion window (w). The standard *Additive Increase and Multiplicative Decrease window algorithm* (AIMD) [16] is one used widely by TCP.

The control law for AIMD algorithm is given in the equation 1.1 below from [16].

Increase: $w(t+rtt) = w(t) + \alpha \qquad \alpha>0$ (1.1)

Decrease: $w(t) + \delta = w(t) - \beta\, w(t) \qquad 0<\beta<1$

where, w(t) is the window size at time t

δ is the time to detect packet loss since the last update.

The *Increase rule* increases the congestion window size by one when a positive acknowledgement is received and the *Decrease rule* roughly halves the window size when congestion occurs. The disadvantage of the AIMD control law is that it prevents TCP from quickly grabbing the available bandwidth as it only allows TCP to increase its congestion window by one packet every round trip time. When a packet drop occurs, TCP will reduce its congestion window multiplicatively and from that window size increase additively by sending one additional packet every RTT. Thus, with networks having high delay, TCP might take a long time to grab the spare bandwidth. These theoretical disadvantages of AIMD led to the need for better control laws. Binomial algorithms [2], Multiple Increase/ Multiple Decrease (MIMD) [26] and Square Increase Multiplicative Decrease (SIMD) [4] are steps in that direction.

Window Control Algorithms

The stability in the Internet is b large directly affected by the presence of well designed flow control algorithms. However, huge differences in the Internet traffic conditions have led to the need to develop different window control algorithms for catering to the different data needs present in the Internet today. Some of the window control algorithms are studied below and tested with the eXplicit Congestion Control protocol and their performance is evaluated.

Binomial Algorithms

The increased diversity of Internet application requirements has spurred the need for transport protocols with flexible transmission controls. Hence, a new class of *non-linear* congestion control algorithms called *Binomial Algorithms* [2] was introduced for Internet transport protocols and applications. These classes of algorithms aimed at generalizing the familiar class of linear control algorithms of which AIMD is one example. Thus, the control equation in (1) can be generalized as given below in Equation 1.2.

Increase: $w(t+ rtt) = w(t) + \alpha/w(t)^k \quad \alpha>0$ (1.2)

Decrease: $w(t) + \delta = w(t) - \beta\, w(t)^l \quad 0<\beta<1$

where w(t) is the window size at time t and

δ is the time to detect packet loss since the last update.

These control equations are also called as *Binomial Congestion Control Algorithms* [2]. These rules generalize the class of all linear control algorithms and other control laws with non-linear controls. They are called Binomial Algorithms because their control expressions involve the addition of two algebraic terms with different exponents.

For the Additive Increase and Multiplicative Decrease algorithm, the value of k=0 and l=1. They generalize TCP-style additive increase by increasing inversely proportional to the power of k of the current window size and TCP-style multiplicative decrease by decreasing proportional to the power of l of the current window. Thus, there are infinite numbers of deployable TCP-compatible binomial algorithms that converge to fairness under synchronized-feedback assumption. The family of increase and decrease algorithms can be obtained by changing the values of *l* and *k* as shown in the table below.

Control Law	***K***	***L***
AIMD	0	1
MIMD	-1	1
AIAD	0	0
MIAD	-1	0
IIAD	1	0
SQRT	½	½

Table I Table of Control Equations [2]

Binomial algorithms were mainly proposed because of its use in applications such as Internet audio and video that does not react well to drastic rate reductions. In [2], the k+l rule is adopted to show how binomial algorithms are TCP-Compatible. The paper shows how the throughput λ of a flow in related to the packet loss-rate *p* as λ α S/(R√p), where S is the packet-size and R is the connection's round-trip time [2] and depicts how an algorithm is said to be TCP-Compatible only if its throughput λ α S/(R√p) with the same constant of proportionality as for a TCP connection with the same packet size and round-trip time. The *k*+*l* rule states that the binomial algorithms are TCP-compatible only if *k*+*l*=1 and *l*<=1 for suitable α and β. Of this wide (k, l) space, [2] have proposed two binomial algorithms.

SQRT Binomial Algorithm

One of the interesting TCP-Compatible algorithms in the (k,l) space is the Square Root algorithm (SQRT) algorithm. It is so called because both its increase is inversely proportional and the decrease is proportional to the square root of the current window.

Increase: $w(t+rtt) = w(t) + \alpha/\sqrt{w(t)} \quad \alpha>0$ (1.3)

Decrease: $w(t) + \delta = w(t) - \beta\sqrt{w(t)} \quad 0<\beta<1$

where w(t) is the window size at time t and

δ is the time to detect packet loss since the last update.

SQRT is seen from [2] to be less oscillatory and have a faster convergence to fairness than TCP for similar network conditions.

Inverse Increase Additive Decrease Binomial Algorithm

The Inverse increase and Additive Decrease algorithm (IIAD), where the increase is inversely proportional to the current window and its decrease is additive is another interesting algorithm in the family of binomial equations. Its increase/decrease equations are shown below.

Increase: $w(t+rtt) = w(t) + \alpha/w(t) \quad \alpha>0$ (1.4)

Decrease: $w(t) + \delta = w(t) - \beta \quad 0<\beta<1$

where w(t) is the window size at time t and

δ is the time to detect packet loss since the last update.

The IIAD is seen to be converging to fairness under synchronized fairness assumption if $k+l=1$, at least k or l are positive and the throughput of a binomial algorithm λ is given as

$\lambda \alpha 1/p^{1/k+l+1}$ where λ is the throughput of the flow and p is the loss rate it encounters. This *k+l* rule represents the fundamental tradeoffs between probing aggressiveness and congestion responsiveness.

Although the binomial algorithms have good smoothness and almost negative oscillations, they are much slower in responding to bandwidth changes and hence are less aggressive in probing for extra bandwidth.

Multiplicative Increase/ Multiplicative Decrease Algorithm (MIMD)

The Multiplicative Increase/ Multiplicative Decrease algorithm is another algorithm that can be obtained from the generalized binomial equation shown above by having k=-1 and l=1. In this algorithm, the window size in increased upon positive acknowledgement proportional to the current window size and similarly, the window size is also decrease upon packet drops to a value proportional the current window size. It has the following control law:

Increase: $w(t+rtt) = w(t) + \alpha * w(t) \quad \alpha>0$ (1.5)

Decrease: $w(t) + \delta = w(t) - \beta * w(t) \quad 0<\beta<1$

The MIMD control law is used in the *slow start* phase of the TCP window control. The MIMD is seen to be more stable than the other binomial control laws like AIAD and MIAD. MIMD is also seen to be more aggressive than other control laws.

Binomial Algorithms Convergence to Fairness

The fairness of the binomial congestion control algorithms were studied by [2]. A network of two sources (x1,x2) that implement the same binomial algorithm with $k,l \geq 0$

is shown by [2]. It is seen that these converge to a fair and efficient operating point (x1=x2=1/2), provided that (k+l>0). It is also assumed that the network provides synchronized feedback about the congestion to all the sources. It is supposed that if x1<x2, then the area above the x2-x1 equi-fairness line is seen. Now a window increase is plotted for these two sources and compared with a window increase of a AIMD algorithm where k has a value of 0.

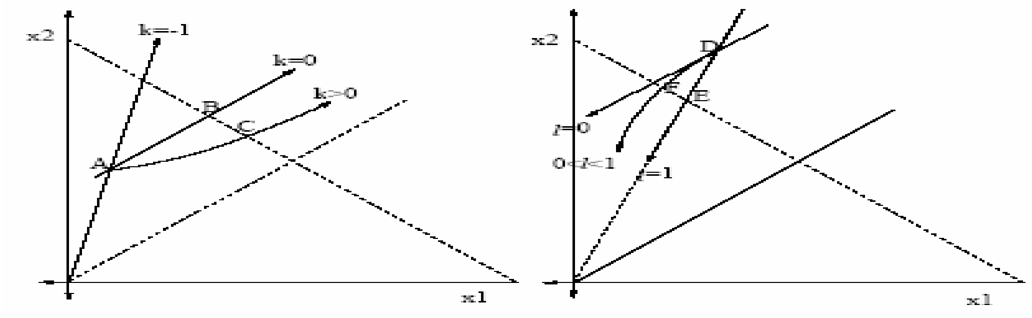


Figure 1 Sample path showing convergence to fairness for a binomial algorithm [2]

Now, with window reduction, when l=0 (additive decrease), the window reduction occurs along the 45°-line (DE) which actually worsens the fairness [2]. When l=1, the window decrease is multiplicative and therefore the fairness remains unchanged. Now, for the binomial algorithms which l value lies between 0 and 1, the decrease occurs as a curve as shown in the Figure [1]. Thus, this intersects the maximum-utilization line at a point which is closer to the fair-allocation point that is present at the intersection of the two lines that the other window algorithms. Another window control algorithm that has been proposed is the Square Increase/ Multiplicative Decrease Algorithm (SIMD).

Square Increase/ Multiplicative Decrease Algorithm (SIMD)

The author in [4] introduced another window control mechanism that is quite different from the existing laws and which shows a distinct improvement over the standard control laws. *Square Increase and Multiplicative Decrease (SIMD)* algorithm is one such algorithm that uses *history information* in addition to current window size in the fairness controller of XCP. It has been shown in [4] that SIMD has allowed for a more progressive expansion of the window based algorithms with just a simple addition to the control laws. SIMD is shown in [4] to be fundamentally different from AIMD and Binomial algorithms as SIMD uses as history information, the window size at the time of detecting last loss which is helpful in improving transient behavior. Also, [4] has shown how SIMD, being TCP- friendly, has shown *high smoothness, aggressiveness and responsiveness.* We say that the Binomial and other control mechanisms are memory-less because their control laws only considers the current window size w(t) while the SIMD control laws consider the size of the window after the last decrease w(0). The control laws of SIMD are as given below.

$$\begin{aligned} &Increase: w(t+rtt) = w(t) + \alpha\sqrt{w(t) - w_0}; \alpha > 0 \\ &Decrease: w(t+\delta) = w(t) - \beta w(t); 0 < \beta < 1 \end{aligned} \tag{1.6}$$

Unlike AIMD, in the increase law, SIMD considers both the size of the window at present and also the window size just before congestion, i.e. reduction in window size. In the congestion control Equation 3 given above, the SIMD scheme takes in the window size before the last congestion epoch as w_{max} and w_0 is taken as the window size after the last decrease. The value of α_{SIMD} too is not a constant as in other control schemes but is a factor of w_{max} which keeps changing after each *congestion epoch*. A *congestion epoch* is defined in [4] as a sequence of window increments followed by one window decrement. Hence, the sequence of window increments is the value of w_{max} and the value of the window size right after the decrement is taken as $w_{0.}$ α, unlike AIMD and other memory-less control equations, does not remain constant but rather varies with the value of w_{max}. The value of α_{SIMD} has been gives in [4] as:

$$\alpha_{SIMD} = \frac{3\sqrt{\beta}}{(1-\frac{2\beta}{3})\sqrt{2w_{\max}}} if\beta << 1, \cong \frac{3\beta}{\sqrt{2w_{\max}}} \tag{1.7}$$

The different window control algorithms were then compared with a few general parameters to check their performance based on certain criteria namely Aggressiveness, Responsiveness, Fairness, Smoothness, Link Utilization and Multiple Connections. On comparing AIMD, the two Binomial Algorithms namely SQRT and IIAD and SIMD, it is seen that in a TCP-like situation, if these algorithms are plugged in instead of TCP's AIMD that exist currently, then SIMD is seen to perform better especially in terms of fairness and aggressiveness.

PROPERTIES	**AIMD [1]**	**BINOMIAL [2]**	**SIMD [4]**
Aggressiveness	AIMD is not very aggressive in probing for extra bandwidth as window increase is linear.	Least Aggressive in probing for extra bandwidth as window increase is sub-linear.	Most aggressive as window increase is super-linear. Grows aggressive with time.
Responsiveness	Upon congestion, AIMD reduces window size multiplicatively.	Least Responsive as when *l*=0, window decreases only by a constant.	Similar to AIMD, window decrease is multiplicative.

Fairness	AIMD has good convergence to fairness.	IIAD converges very slowly to fairness when compared to AIMD and SIMD.	SIMD converges faster and closer to the fairness line than AIMD or the binomial algorithms.
Smoothness	Least Smooth. AIMD has noticeable amount of oscillations even when the loss rate is constant.	Good Smoothness. Binomial Algorithms have negligible oscillations but are slower to respond to bandwidth changes.	High Smoothness. SIMD can provide high smoothness in steady state.
Link Utilization	AIMD has average link utilization but does not grab the spare bandwidth fast.	Binomial Algorithms have good link utilization and show higher long term throughput than AIMD.	SIMD has good link utilization and allows two competent flows to quickly and smoothly transit to steady state.
Multiple Connections	AIMD is fair to other AIMD connections in the network.	IIAD is fair to other IIAD and also to TCP connect - ions except at high loss rates, where TCP gains.	SIMD is also fair to other competing flows in the network but at high loss rates, SIMD gains.

Table II Comparison of Different Window Control Algorithms

As shown in Table 1.1, SIMD is superior to AIMD because of high smoothness in steady state and faster convergence to fairness compared to AIMD. It is also more aggressive and more responsive especially when the network conditions change frequently and drastically. The history information used by SIMD is a good indicator for the congestion in the network at present and it can be used to predict the congestion that is going to occur in the network in the future.

SIMD's Convergence to Fairness and Efficiency

SIMD is shown to have better convergence behavior than the memory-less AIMD in [4]. In [4], the authors use the vector space used in [15] to show how multiple users with synchronized feedbacks can converge to fairness when using the SIMD control scheme. First, they illustrate the convergence to fairness for two users shown on the vector-space as x_1 and x_2. The two-user resource allocation is represented by a point $X(x_1, x_2)$, where x_i is the resource allocation for the i^{th} user, i=1,2. [4] defines the fairness index as $\max(\frac{x_1}{x_2}, \frac{x_2}{x_1})$. The line $x_1 = x_2$ is the *fairness line* and the line $x_1 + x_2 = 1$ is the *efficiency line*. Thus, every control scheme would try to bring the system to an intersection between these two lines.

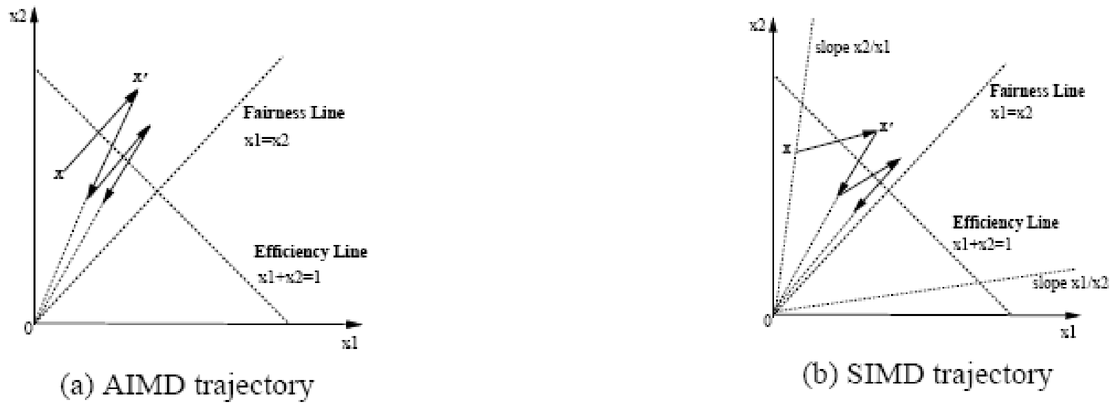


Figure 2 Convergence to Fairness [4]

A comparison between SIMD and AIMD is shown above where each of the congestion windows are analyzed on their increase laws. The multiplicative decrease law does not affect the fairness. If the fairness index of either law is closer to unity, then the resource allocation is said to be fairer. It is seen from Fig. 2 that as the system evolves, with increases and decreases, the SIMD trajectory moves much closer to the fairness line as compared to the AIMD line for the same two-users. As the system evolves, both the algorithms bring the resource allocation point towards fairness but the SIMD outperforms AIMD in that it is able to converge to fairness faster than AIMD and then oscillate around the efficiency line, maintaining the fairness while it takes AIMD longer time to converge to fairness.

SIMD's Convergence Speed

In [4], SIMD's convergence speed is analyzed. A case of $l - \frac{1}{k+1} = -1$ is chosen, satisfied by the Square Increase, Multiplicative Decrease (SIMD) algorithm which has values of k=-0.5 and l=1. [4] shows intuitively as well as analytically how SIMD converges more quickly to fairness than AIMD. X(x_1, x_2) is considered to be under-utilized with $x_1 + x_2 < 1$ and $x_1 < x_2$. The point of intersection of the efficiency line and the trajectory for AIMD is shown to be $(\frac{1+x_1-x_2}{2}, \frac{1-x_1+x_2}{2})$ and that of SIMD is shown to be $(x_1 - x_2 + \frac{x_2}{x_1+x_2}, x_2 - x_1 + \frac{x_1}{x_1+x_2})$. Then, the authors of [4] compare then compare the fairness index of these two schemes, and show that intuitively, SIMD reaches a more fair intersection when $x_1 + x_2 > 1/3$.

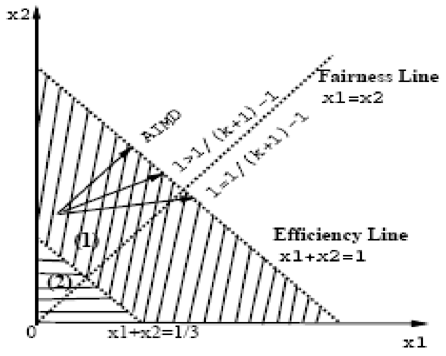


Figure 3 Convergence Speed [4]

The Figure 3 from [4] shows that three categories have been plotted; AIMD, l>1/(k+1)-1 and l=1/(k+1)-1. The third line, *l*=1/(k+1)-1 i.e. l=1 line depicts the SIMD control law. As the area (1) where $x_1 + x_2 > 1/3$, is much larger than area (2), it can be seen intuitively that SIMD converges much faster than AIMD and other control laws that fall into the *l*>1/ (k+1)-1 category.

eXplicit Congestion control: Explicit Router Feedback

Another quite different approach to avoid congestion was also proposed. [3] Introduces *XCP: eXplicit **C**ongestion **C**ontrol*, which on the other hand, uses a precise congestion control feedback where the source is *explicitly* informed when congestion occurs. [3] gives a new architecture for Internet congestion control that decouples the control of congestion from the bandwidth allocation policy. It alters the window size based on the *level of congestion* in the network and not just because of a simple packt loss as an indication of congestion. The window equations of XCP are given below in Equation 4.

Increase: $w(t+ rtt) = w(t) + \alpha(t) \qquad \alpha>0$ (1.8)

Decrease: $w(t) + \delta= w(t) - \beta(t)*w(t) \; 0<\beta<1$

Where, α(t) and β(t): feedback parameters that change when the congestion level of the network varies.

Here, every packet sent is given a congestion header that gives sufficient information to the routers along the path about the connection. The method given in [3] has been shown to be highly efficient due to its marked difference from the TCP's standard AIMD. AIMD algorithms use a black and white feedback mechanism as there are only two states that are indicated i.e. positive acknowledgement or congestion. Thus, TCP is unable to distinguish whether the congestion is from the losses or the link errors in the network. Hence, unnecessary drops result due to link errors. XCP on the other hand, can explicitly gauge the level of congestion in the network. It is used efficiently in high bandwidth-delay product environments as it can correctly estimate how much spare bandwidth is available and the window size can increase accordingly.

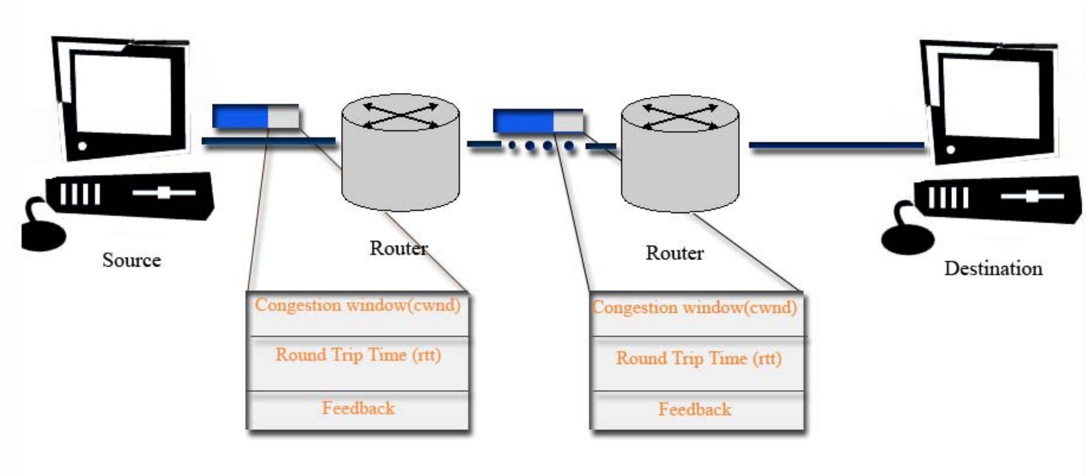


Figure 4 Explicit Router Feedback [3]

In XCP, the routers along the path from the sender and the receiver play an important role by computing the per-packet feedback and estimating the positive and negative feedback in a control interval. Here, it uses two controllers called the *Efficiency Controller (EC)* and the *Fairness Controller (FC)* at each router. The fairness controller uses the control equations to compute the total feedback at each router. In [3] the standard Additive Increase and Multiplicative Decrease (AIMD) control law is used to distribute this feedback fairly among all flows. Here, this document proposes to use The *Square Increase and Multiplicative Decrease (SIMD)* algorithm proposed in [4] in the fairness controller (FC) of the XCP. It can be shown through equations and with ns-simulations that using XCP coupled with SIMD would produce a far more superior means to detect and avoid congestion.

II: Background and Related Work

This chapter gives a brief overview on the general picture of the Internet Congestion Control and also gives some of the previous work that has already been done in congestion control and the relevance of those in this thesis.

Internet Congestion Control

With the introduction of fiber networks and Gigabit Ethernets, the Internet congestion as we know of has changed. There is now a deluge of data that is being sent from different parts of the world. Thus, the standard TCP/IP protocol that was designed when the network demands were quite different from what is present today; it does not remain very competent for the current network conditions. The control of Internet congestion is done with two different entities working and co-operating together. The end-systems (senders and the receivers) are the end-hosts that manage the data. But, data along the network path is sent from one host to another with the aid of the routers present in the network. Traditionally, the responsibility of the routers has been limited forwarding the packets along the network path and if the traffic is too great and it is unable to handle that many packets at a time, then the packets are dropped. The senders have a congestion control protocol that monitors the packet drop and when a packet drop occurs, the sender takes that as a sign of congestion in the network and reacts to it by decreasing the sending rate (decreasing the sending window size). In the current Internet, predominantly this type of congestion control exists. Drop-Tail routers that drop packets upon their buffer becoming too full and networks that run TCP protocol is very predominant in today's network.

The research community is now slowly moving towards placing more control in the routers. Some such proposed changes are Random Early Detection (RED) [22], Active Virtual Queue (AVQ)[23], Random Early Marker (REM)[24], ARED[25] etc. These are collectively referred to as Active Queue Management schemes (AQMs).

Transmission Control Protocol (TCP)

The Transport Control Protocol (TCP) is the backbone of Internet today. The TCP is a set of rules that is used in tandem with the Internet Protocol (IP) to send and receive data as packets between end hosts over the internet. TCP is responsible for sending each small unit of data that a message gets divided into from the sender to the receiver. TCP is responsible for selecting the most reliable, smallest and in general most efficient routing across the Internet.

TCP is a connection-oriented protocol. Here, a connection is established by means of *Handshaking* process where the sender initiates a communication and if the receiver is ready to accept communication from the sender, it sends an acknowledgement. Once the acknowledgement is received by the sender, it can then start sending data across. This *Handshaking* process is very important as the two end hosts settle different parameters of which the receiver's buffer size is also one. This receiver's buffer size is set as the sender's maximum window size. This is done to ensure that the sender does not overwhelm the receiver with more data than it can process because sometimes the sender's processing speed is greater than the receiver's processing speed, then the receiver's buffer size becomes full and the receiver cannot accept any more data packets from the sender. This is called *Flow Control.*

TCP uses Sliding Windows, a technique which is also called as *windowing* as a method to control the flow of data packets from the sender to the receiver. As TCP is a reliable protocol, it requires every packet to be acknowledged by the receiving host. The original specification of TCP included the window based scheme of controlling the sender's flow as a means of governing the amount of data sent by the sender to the receiver. The original TCP did not include congestion control schemes. Van Jacobson [16] beginning in the year 1986, developed a congestion avoidance scheme that have not become a part of the standard TCP implementation. These mechanisms operate on the host systems and when congestion is encountered in the network, the TCP connections back-off or the sender reduces the sending rate to alleviate the congestion. These are called the TCP congestion avoidance algorithms. These provided the fix that prevented the Internet "meltdown" and ensures that the network does not collapse due to congestion.

III: METHODOLOGY

The evolving Internet has made it essential to look for better schemes to avoid congestion in the network and to respond to congestion with minimal delays and maximum utilization of the resources. Studies have shown that the current TCP with AIMD window control scheme is not very suitable for the current Internet with its high-bandwidth-delay product environments and increasing users and traffic. AIMD is not quite aggressive in probing for extra bandwidth and it also suffers from a low responsiveness and poor convergence to fairness. Also, the Binomial algorithms proposed in [2], though it does have some improvements over the basic AIMD algorithms, it does not respond well to sudden increase in congestion-responsiveness and it is not very aggressive in probing for spare bandwidth due to tradeoffs in the value of k and l for throughput. It has been shown in [3] that regardless of the queuing scheme; TCP becomes unstable and inefficient when the bandwidth-delay product increases. The XCP-SIMD coupling, on the other hand, has shown to be far more superior in all the aspects mentioned above although AIMD or binomial algorithms may have some attributes that outdo SIMD in some cases. The XCP-SIMD coupling is a step towards exploring the space between memory-less control schemes and equation-based schemes with precise and *explicit* congestion feedback as an indication for current and future congestion.

XCP Router using Different Window Control Algorithms

The XCP, which generalizes the Explicit Congestion Notification Proposal [5], uses routers to inform senders about the degree of congestion in the network. It also decouples *fairness control* from *utilization control.* Efficiency involves monitoring only the aggregate traffic's behavior whereas fairness involves monitoring the relative throughput of the flows in the link. Utilization in XCP is done by using the *spare bandwidth* and the feedback delay to adjust its aggressiveness in probing for bandwidth. This prevents oscillations, provides stability and helps make efficient use of the network resources especially in high bandwidth-delay product environments. Fairness control is maintained by taking away the extra bandwidth from those flows that utilize more than their fair share and allocating them to other flows. Decoupling fairness and utilization control opens up different methods for service differentiation using schemes that provide desired bandwidth allocations, yet are too aggressive or too weak for controlling congestion. Also, congestion losses and error losses in the links can then be easily identified. If modifications need to be made in any one of the controllers, it can be done without affecting the other controller. It makes design and analysis of these controllers simpler and efficient. In XCP, the router has both an *Efficiency Controller (EC)* and a *Fairness Controller (FC).* Senders maintain their congestion window (*w)* and round trip time (rtt*)* and send this information to the routers along the path by means of a congestion header placed in every packet.

In the basic XCP framework, the senders maintain their congestion windows cwnd and the average round trip time (rtt). The throughput is sent to each router from these two values i.e. cwnd/rtt. RTT is also sent in the congestion header to each of the routers. Routers monitor the input traffic rate to each of their output queues. Now, the difference between the link bandwidth and the input rate of traffic will provide the router with a good picture of the network performance. As each router along the path calculates this difference and updates the feedback field present in the congestion header, then the feedback field would contain the value of the most congested router in the network i.e. it will have the feedback from the bottleneck along the network path from the sender to the receiver.

H_cwnd (set to sender's current cwnd)
H_rtt (set to sender's rtt estimate)
H_feedback (set to the sender's desired increase value)

Figure 5 The congestion Header[4]

This congestion header is used to communicate the flow's state to the routers and the feedback from the routers onto the receivers. Each time a packet is sent; the sender sets the H_cwnd field to its current cwnd and sets the H_rtt field to its estimate. The routers along the path monitor the input traffic and after obtaining the difference between the link bandwidth and its input traffic rate, each router modifies the congestion header according to the level of congestion to indicate to the other flows sharing the link to either increase or decrease their congestion windows. This feedback is then placed in the H_feedback field. The routers along the path can modify only this field. This feedback then reaches the receiver, which then changes its congestion window (cwnd) accordingly and sends its acknowledgement to the sender. When the sender receives a positive acknowledgement, it increases its congestion window while a negative acknowledgement makes the sender to reduce its congestion window. The congestion window can be estimated as [12]:

$$cwnd = max\ (cwnd + H\_feedback, s) \tag{3.1}$$

where, cwnd : current congestion window

H_feedback : Field in the congestion header which the sender initializes to request its desired window increase

s : packet size

The feedback sent by the router is computed using the fairness and utilization controller. The XCP controllers make a single control decision every average RTT. [14] defines some basic parameters in order to explain the system. An estimation interval of T_e is defined as the interval over which the routers estimate all the parameters and it is usually an estimate of the average rtt of the flows in the link. Thus, T_e is dynamic and keeps changing with the average rtt. The control interval, $T_{c,}$ on the other hand is a constant interval during which a single decision is applied. After the end of one $T_{c,}$ the router reads all the estimators and decides what kind of feedback it should send in this control interval. The XCP routers take into account the input traffic rate, which the total number of bits that is sent to a queue in one control interval divided by the interval duration and spare bandwidth, which is the difference between the link capacity and the input traffic rate. The router also estimates the average rtt. The average rtt computed in one estimation interval gives the duration for the next T_e. The rtt and the cwnd of each flow are normally provided by the sender itself in the congestion header. The average rtt is computed as:

$$\overline{rtt} = \frac{\sum rtt_i * \frac{s_i}{r_i}}{\sum \frac{s_i}{r_i}} \tag{3.2}$$

where s_i is the packet size in bytes and r_i is the throughput in bytes/sec.

The router maintains a per-link estimation-control timer that has a value equivalent to the average RTT of the flows traversing the link.

Traditionally, the efficiency and the fairness have been grouped together under the same control law. Conceptually, however, they are different. While Efficiency involves only the traffic behavior as a whole where it concerns lie with the input traffic rate, fairness on the other hand, involves the relative throughput of the flows sharing a network. A scheme is said to be fair, only when all the flows in the link have the same throughput, whether congestion exist or not. [16].

Thus, in [16] a decoupling of these two fundamentally different approaches is done. One of the major advantages of such a decoupling is that t also simplifies the design of each controller where any one of them can be modified without affecting the other's performance. Thus, two controllers, namely Efficiency Controller (EC) and Fairness Controller (FC) exist.

Efficiency Controller (EC)

The efficiency controller is used to maximize link utilization and to minimize the aggregate traffic. The XCP-EC computes the desired increase or decrease in the number of bytes in an average RTT. [16] gives a method to compute this feedback φ in each average RTT as:

$$\boldsymbol{\Phi = \alpha.\ S - \beta.\ Q/d,} \tag{3.3}$$

where: α, β are constant parameters,

d is the average RTT

S is the spare bandwidth and

Q is the persistent queue size.

The spare bandwidth is calculated as the difference between the input traffic rate and the link capacity. The persistent queue size is calculated by taking the minimum queue seen by a packet that has just arrived during the last propagation delay. Thus, the feedback φ is proportional to the spare bandwidth S. It is also proportional to the persistent queue size Q. So, when the link is underutilized, S>=0, a positive feedback is sent and when S<0, the link is over utilized i.e. congestion and a negative feedback is sent. This gives the feedback φ in bytes. Thus, to drain the persistent queue, the feedback is made proportional to the persistent queue too. Finally, because the feedback is in bytes/sec, to match the unit of Q, the Q value is divided by the control interval d. This is also necessary as scaling down Q by a factor of d ensures that the system remains stable for any feedback delay. The efficiency controller does not bother about fairness of all the flows in the network at all but is only concerned that the link is utilized to the maximum while the drop rate and persistent queues are at a minimum [16].

α and β are constant and set to values of 0.4 and 0.226 respectively based on stability analysis [16]. Thus, this value of feedback is then assigned to the H_feedback field present in the congestion header. Now, as each router along the link traversing from the sender to the receiver changes the H_feedback field, it will at the end contain the feedback from the bottleneck router. [16]

Fairness Controller (FC)

The fairness controller in [16] used Additive Increase and Multiplicative Decrease (AIMD) [1] control law to converge to fairness and to allocate feedback to individual packets. But, as was indicated by the comparison chart in Section 1, different window control algorithms can be used instead of AIMD and its performance can be seen. The brief description is its implementation in XCP's fairness control is as described here. The job of the fairness controller is to divide the total feedback equally and to allocate them among the individual flows to achieve fairness. The fairness controller in [16] uses AIMD control law. Thus, for such a control law, intuitively, the following policy can be applied:

If $\Phi>0$, allocate feedback equally among all the flows

If $\Phi<0$, allocate feedback to flows proportional to the current throughput [16]

Fairness Controller with SIMD

In order to allocate positive and negative feedback to individual packets, the Fairness Controller can use the Square Increase, Multiplicative Decrease Algorithm that was explained in Sec. 1.1. This algorithm increases its congestion window proportional to the square of the time elapsed since the detection of the last loss event [4] and decreases multiplicatively.

$$\begin{aligned} &Increase: w(t+rtt) = w(t) + \alpha\sqrt{w(t) - w_0}; \alpha > 0 \\ &Decrease: w(t+\delta) = w(t) - \beta w(t); 0 < \beta < 1 \end{aligned} \tag{3.4}$$

where $\alpha = \dfrac{3\sqrt{\beta}}{(1-\frac{2\beta}{3})\sqrt{2w_{\max}}} if \beta << 1, \cong \dfrac{3\beta}{\sqrt{2w_{\max}}}$

Here, w_{max} is given the value of the window size at the end of the last congestion epoch, which is a series of increments followed by one decrement and w_0 is the window size of the decrease. Thus, with the decrease rule, w_0 can be obtained from w_{max} and vice versa. For example, $w_0 = (1-\beta)w_{\max}$.

Thus, the feedback policy can be slightly altered to suit SIMD control law.

If Φ>0, allocate feedback equally among all the flows by square root increase

If Φ<0, allocate feedback to flows proportional to the current throughput

As the router should calculate the feedback and place it in the congestion header of each packet, the router needs to calculate the *positive feedback (pi)* and the *negative feedback (ni)* separately and then it would be easier to compute the actual per-packet feedback H_Feedback which would be the difference between the positive and negative feedback.

$$H\_Feedback=p_i-n_i$$

And the shuffled traffic can be computed as h=max(0, γ.y-| *Φ*|*) [16]*

where y is the input traffic rate and γ is a constant equal to 0.1

Per-packet Positive Feedback

The per-packet positive feedback is computed when the feedback is positive (*Φ>0*). When the feedback is positive, then the throughput needs to be increased by the same amount to ensure fairness. As according to the SIMD increase equation shown above, for a flow to increase its throughput every rtt proportionally to $\sqrt{r_i(t) - r_0}$, then the sum of the positive feedback for all the flows together that it receives in a rtt should also be proportional to $\sqrt{r_i(t) - r_0}$, where $r_i(t)$ is the current Throughput of the flow and r_0 the Throughput of a flow after a congestion epoch i.e. after a decrease. Therefore, the positive feedback per-packet should be proportional to $\sqrt{r_i(t) - r_0}$. Now, the total change in the throughput of a flow is the sum of the per-packet feedback it receives. Hence, the per-packet feedback can be calculated by dividing the change in throughput by the expected number of packets from a flow i that a router sees in a control interval d[16]. The expected number of packets in a control interval is proportional to the flow's throughput divided by the packet size $\frac{r_i}{s_i}$. As d is a constant during a control interval, the per-packet positive feedback is inversely proportional to its throughput divided by its packet size $\frac{1}{r_i / s_i}$. Hence, the per-packet positive feedback can be given as:

$$p_i = \xi_p * \frac{s_i}{r_i} * \sqrt{r_i = r_0} \qquad (3.5)$$

where ξ_p is a constant.

The total increase in the aggregate traffic rate is h+max(*Φ,0),*where max(*Φ,0)* makes sure that the positive feedback is only being computed. Now, this value is equal to the sum of the increase in the rates of all the flows, which is nothing but the positive feedback of a flow. [16]

$$h + \max(\Phi, 0) = \sum^{L} p_i \tag{3.6}$$

where L is the number of packets seen by the router during a control interval d. From this ξ_p can be derived.

$$\xi_p = \frac{\mathrm{h} + \max(\phi, 0)}{\sum \frac{s_i}{r_i}} \tag{3.7}$$

Thus, the per-packet positive feedback is computed using the Square Increase, Multiple Decrease algorithm where a flow is proportional to the square root of the difference in the throughputs.

As the α value in SIMD does not remain constant and keeps changing with the value of the maximum throughput, the per-packet positive feedback needs to be altered to accommodate that too. Thus, for a flow to increase its throughput every rtt proportionally to $\sqrt{r_i(t) - r_0}$, then the sum of the positive feedback for all the flows together that it receives in a rtt should also be proportional to $\sqrt{r_i(t) - r_0}$, where $r_i(t)$ is the current Throughput of the flow and r_0 the Throughput of a flow after a congestion epoch i.e. after a decrease. Now, the throughput of a flow is also proportional to the α_{SIMD}. Thus the per-packet positive feedback is also proportional to $\frac{3\sqrt{\beta}}{(1 - \frac{2\beta}{3})\sqrt{2r_{\max}}}$ where $r_{\max}$ is the window size just before the decrease and β is a constant set to 1.236, computed after stability analysis. Thus, the per-packet positive feedback is now computed as:

$$p_i = \xi_p * \frac{s_i}{r_i} * \sqrt{r_i - r_0} * \frac{3\sqrt{\beta}}{(1-\frac{2\beta}{3})\sqrt{2r_{\max}}} \tag{3.8}$$

Per-packet Negative Feedback

The per-packet negative feedback can be computed similar to the per-packet positive feedback. The SIMD has a multiple decrease similar to the original XCP-AIMD and hence the negative feedback calculation remains unchanged.

The per-packet negative feedback is calculated when feedback is negative (*Φ<0*). To compute the negative feedback, the throughput of any flow I should be proportional to its current throughput (i.e. $\Delta r_i \alpha\ r_i$). The desired per-packet feedback is the change in the throughput divided by thr expected number of packets that a router will see in a control interval. As d is constant for a control interval, it will be inversely proportional to $\frac{r_i}{s_i}$.

Thus, the r_i values cancel out each other and per-packet negative feedback is proportional to the packet size.

$$n_i = \xi_n * s_i \tag{3.9}$$

where ξ_n is a constant.

Similar to the positive feedback computation, the total decrease in the aggregate traffic is the sum of the decrease in the rates of all flows.

$$h + \max(-\Phi,0) = \sum^{L} n_i \qquad (3.10)$$

where L is the number of packets seen by the router during a control interval d. From this ξ_p can be derived.

$$\xi_p = \frac{\mathrm{h} + \max(-\phi,0)}{\sum s_i}$$

Thus the per-packet negative feedback is calculated using multiplicative decrease. All the parameters for the positive and negative feedback can be easily obtained by the router. All these parameters are present in the congestion header of the packet.

The Fairness Controller tracks the total amounts of positive and negative allocation when a control interval starts. It stops computing the positive feedback when the sum of the positive feedback that has been allocated in that control interval becomes equal to h+ max($\Phi,0$) and the stops computing negative feedback when the negative feedback allocated in that control interval becomes equal to h+ max($-\Phi,0$). This is done to ensure that the allocation error is bounded and that it follows the decision of the Efficiency Controller.

Fairness Controller with Binomial Algorithms

There are two main binomial algorithms that have been taken into consideration here; the Square Root Algorithm (SQRT) and the Inverse Increase, Additive Decrease Algorithm (IIAD).

The general binomial equations are:

Increase: $w(t+rtt) = w(t) + \alpha/w(t)^k \quad \alpha>0$

Decrease: $w(t) + \delta = w(t) - \beta\, w(t)^l \quad 0<\beta<1$

(i) SQRT

The fairness controller is plugged in with the Square Root binomial algorithm. This has the following control law:

Increase: $w(t+rtt) = w(t) + \alpha/\sqrt{w(t)} \quad \alpha>0$

Decrease: $w(t) + \delta = w(t) - \beta\sqrt{w(t)} \quad 0<\beta<1$

Per-packet Positive Feedback:

The per-packet positive feedback can be calculated with the increase in the throughput of all flows to be proportional to $1/\sqrt{ri}$.

The expected number of packets in a control interval is proportional to the flow's throughput divided by the packet size $\frac{r_i}{s_i}$. As d is a constant during a control interval, the per-packet positive feedback is inversely proportional to its throughput divided by its packet size $\frac{1}{r_i/s_i}$. Hence, the per-packet positive feedback can be given as:

$$p_i = \xi_p * \frac{s_i}{r_i * \sqrt{r_i}} \tag{3.11}$$

where ξ_p is a constant.

The total increase in the aggregate traffic rate is h+max(*Φ,0),*where max(*Φ,0)* makes sure that the positive feedback is only being computed. Now, this value is equal to the sum of the increase in the rates of all the flows, which is nothing but the positive feedback of a flow. [16]

$$h + \max(\Phi,0) = \sum^{L} p_i \ ,$$

where L is the number of packets seen by the router during a control interval d. From this ξ_p can be derived.

$$\xi_p = \frac{\mathrm{h} + \max(\phi,0)}{\sum \frac{s_i}{r_i * \sqrt{r_i}}}$$

Per-packet Negative Feedback:

To compute the negative feedback, the throughput of any flow I should be proportional to its current throughput (i.e. $\Delta r_i \alpha \sqrt{r_i}$). The desired per-packet feedback is the change in the throughput divided by the expected number of packets that a router will see in a control interval. As d is constant for a control interval, it will be inversely proportional to $\frac{r_i}{s_i}$.

$$n_i = \xi_n * \frac{s_i * \sqrt{r_i}}{r_i} \tag{3.12}$$

where ξ_n is a constant.

Similar to the positive feedback computation, the total decrease in the aggregate traffic is the sum of the decrease in the rates of all flows.

$$h + \max(-\Phi,0) = \sum^{L} n_i \ ,$$

where L is the number of packets seen by the router during a control interval d. From this ξ_n can be derived.

$$\xi_n = \frac{\mathrm{h} + \max(-\phi, 0)}{\sum \frac{s_i * \sqrt{r_i}}{r_i}}$$

(ii) IIAD

The fairness controller is plugged in with the Inverse Increase Additive Decrease binomial algorithm. This has the following control law:

Increase: w(t+ rtt) =w(t) + α/w(t)   α>0

Decrease: w(t) + δ= w(t) – β   0<β<1

Per-packet Positive Feedback:

The per-packet positive feedback can be calculated with the increase in the throughput of all flows to be proportional to $1/r_i$

The expected number of packets in a control interval is proportional to the flow's throughput divided by the packet size $\frac{r_i}{s_i}$. As d is a constant during a control interval, the per-packet positive feedback is inversely proportional to its throughput divided by its packet size $\frac{1}{r_i / s_i}$. Hence, the per-packet positive feedback can be given as:

$$p_i = \xi_p * \frac{s_i}{r_i^2} \tag{3.13}$$

where ξ_p is a constant.

The total increase in the aggregate traffic rate is h+max(*Φ,0),*where max(*Φ,0)* makes sure that the positive feedback is only being computed. Now, this value is equal to the sum of the increase in the rates of all the flows, which is nothing but the positive feedback of a flow. [16]

$$h + \max(\Phi,0) = \sum^{L} p_i \, ,$$

where L is the number of packets seen by the router during a control interval d. From this ξ_p can be derived.

$$\xi_p = \frac{\mathrm{h} + \max(\phi,0)}{\sum \frac{s_i}{r_i^{\,2}}}$$

Per-packet Negative Feedback:

To compute the negative feedback, the throughput of any flow I should be proportional to its current throughput (i.e. $\Delta r_i \alpha$ constant). The desired per-packet feedback is the change in the throughput divided by the expected number of packets that a router will see in a control interval. As d is constant for a control interval, it will be inversely proportional to $\frac{r_i}{s_i}$.

$$n_i = \xi_n * \frac{s_i}{r_i} \tag{3.14}$$

where ξ_n is a constant.

Similar to the positive feedback computation, the total decrease in the aggregate traffic is the sum of the decrease in the rates of all flows.

$$h + \max(-\Phi, 0) = \sum^{L} n_i \,,$$

where L is the number of packets seen by the router during a control interval d. From this ξ_n can be derived.

$$\xi_n = \frac{\mathrm{h} + \max(-\phi, 0)}{\sum \frac{s_i}{r_i}}$$

Fairness Controller using MIMD

The fairness controller is plugged in with the Multiplicative Increase, Multiplicative Decrease algorithm. This has the following control law:

Increase: $w(t+rtt) = w(t) + \alpha * w(t) \quad \alpha > 0$

Decrease: $w(t) + \delta = w(t) - \beta * w(t) \quad 0 < \beta < 1$

Per-packet Positive Feedback:

The per-packet positive feedback can be calculated with the increase in the throughput of all flows to be proportional to r_i

The expected number of packets in a control interval is proportional to the flow's throughput divided by the packet size $\frac{r_i}{s_i}$. As d is a constant during a control interval, the per-packet positive feedback is inversely proportional to its throughput divided by its packet size $\frac{1}{r_i / s_i}$. Hence, the per-packet positive feedback can be given as:

$$p_i = \xi_p * s_i \tag{3.15}$$

where ξ_p is a constant.

The total increase in the aggregate traffic rate is h+max(*Φ,0),*where max(*Φ,0)* makes sure that the positive feedback is only being computed. Now, this value is equal to the sum of the increase in the rates of all the flows, which is nothing but the positive feedback of a flow. [16]

$$h + \max(\Phi,0) = \sum^{L} p_i \ ,$$

where L is the number of packets seen by the router during a control interval d. From this ξ_p can be derived.

$$\xi_p = \frac{\mathrm{h} + \max(\phi,0)}{\sum s_i}$$

Per-packet Negative Feedback:

To compute the negative feedback, the throughput of any flow I should be proportional to its current throughput (i.e. $\Delta r_i \alpha\ r_i$). The desired per-packet feedback is the change in the throughput divided by the expected number of packets that a router will see in a control interval. As d is constant for a control interval, it will be inversely proportional to $\frac{r_i}{s_i}$.

$$n_i = \xi_n * \mathrm{s}_\mathrm{i} \tag{3.16}$$

where ξ_n is a constant.

Similar to the positive feedback computation, the total decrease in the aggregate traffic is the sum of the decrease in the rates of all flows.

$$h + \max(-\Phi,0) = \sum^{L} n_i \ ,$$

where L is the number of packets seen by the router during a control interval d. From this ξ_n can be derived.

$$\xi_n = \frac{\mathrm{h} + \max(-\phi, 0)}{\sum \mathrm{s_i}}$$

IV: FINDINGS

Simulation of XCP using *ns-2*

The main XCP protocol suite was downloaded from the ISI website [18]. The suite had a set of C++ codes along with a few *ns* test simulations. The implementation of XCP was first tested in Red Hat 2 [20] and Fedora Core 2 [21] Linux platforms. As Red Hat does not have some basic functionality needed to run *ns-2* Fedora was then chosen. Lately, the C++ codes and the *ns* simulations were tested on *Cygwin [19]* a UNIX platform that can be run on Windows. The code that was downloaded from the XCP website had many errors and many other files that were not part of the package were also needed. All the errors and bugs were fixed and any new files were found on the internet and the program was recompiled with the complete list. Once the original XCP was compiled it was made to run on the different platforms. Next, the XCP codes were changed to operate using different Window Control Algorithms. New *ns* simulations were written to make XCP run on different network topologies.

For different Window Control algorithms, the performance of XCP was tested and different graphs plotted. A simple topology having 3 XCP flows and 1TCP flow sharing a bottleneck link was created.

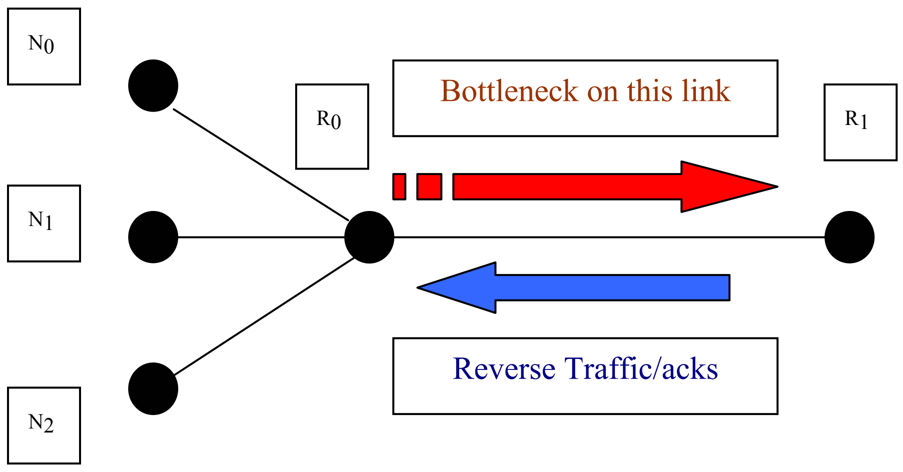


Figure 6 A simple XCP topology involving a bottleneck

N0, N1 and N2 are three XCP flows that share a network path from R0 to R1. Packets are sent from N0, N1 and N2 to their respective sinks through the link R0 and R1. This link R0 to R1 is also shared by a TCP flow. Thus, the link from R0 to R1 not only has the traffic from the three XCP flows, it also has traffic from a TCP flow. Also the reverse traffic and acknowledgements from each of the destinations (sinks) back to the sources need to be considered. The statistics of the link are as follows:

The qType is set to XCP, the bandwidth of the bottleneck as well as the as that of the each of the nodes are set to 20MB, the delay is set to 10ms, the buffer size is always set to the bandwidth-delay product and the data packet size is set to 1000 for all the XCP flows as well as the TCP flow. The XCP flows and the TCP flow is started at different times to emulate a real time network with bottleneck.

Simulations are run for these four flows and the performance of XCP with different Window Control Algorithms is evaluated with each other and with the TCP flow.

Graphs

XCP-AIMD

The original XCP flow with Additive Increase and Multiplicative Decrease was run for the topology mentioned above and the graphs were plotted. Four graphs were plotted:

(i) Comparing the congestion Window of the different XCP flows with the TCP flow for the duration of 30sec.

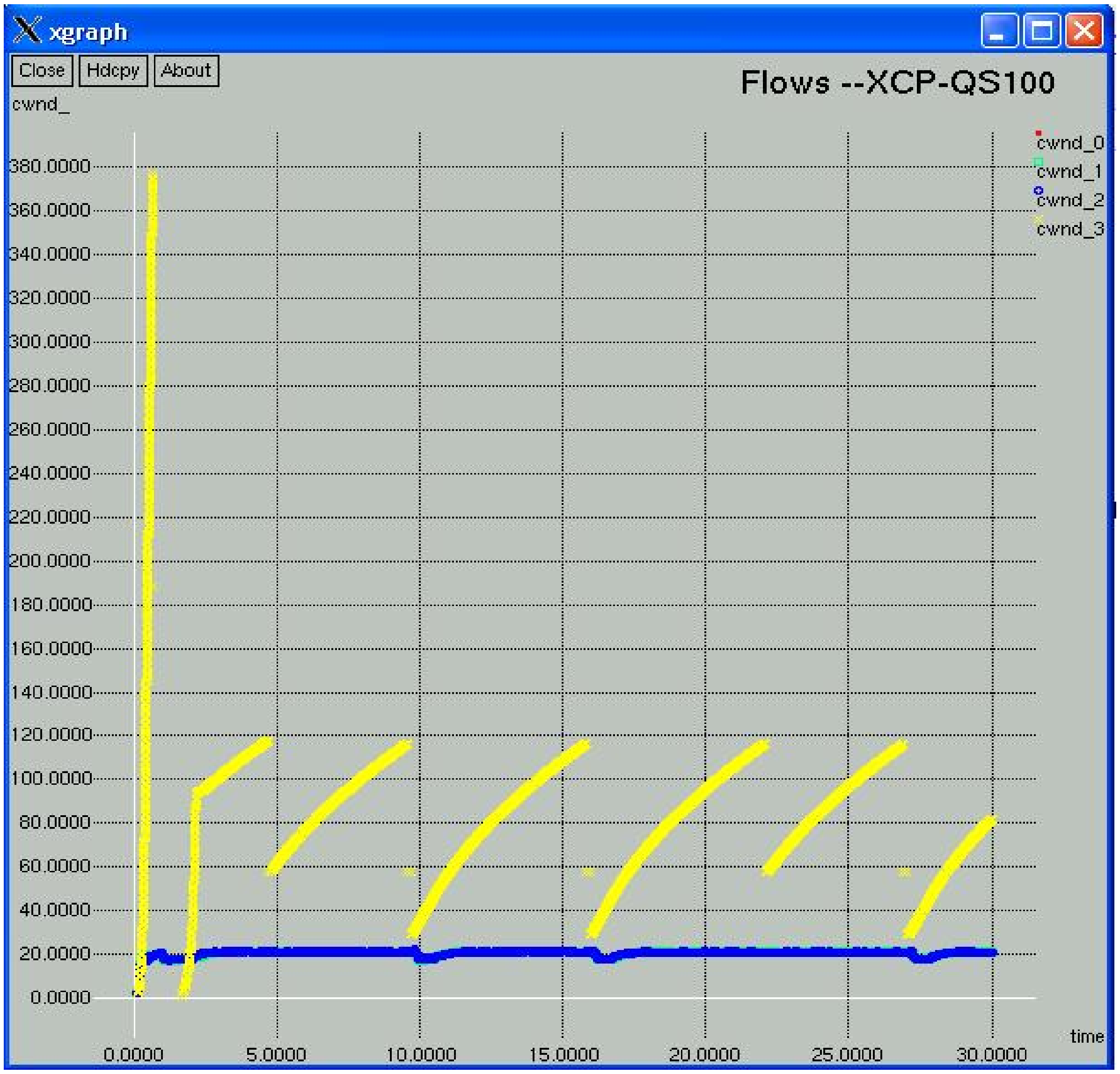


Figure 7 Comparison of XCP-AIMD Congestion window with TCP flow

(ii) Comparing the sequence numbers of the packets sent for the different XCP flows with the TCP flow for the duration of 30sec.

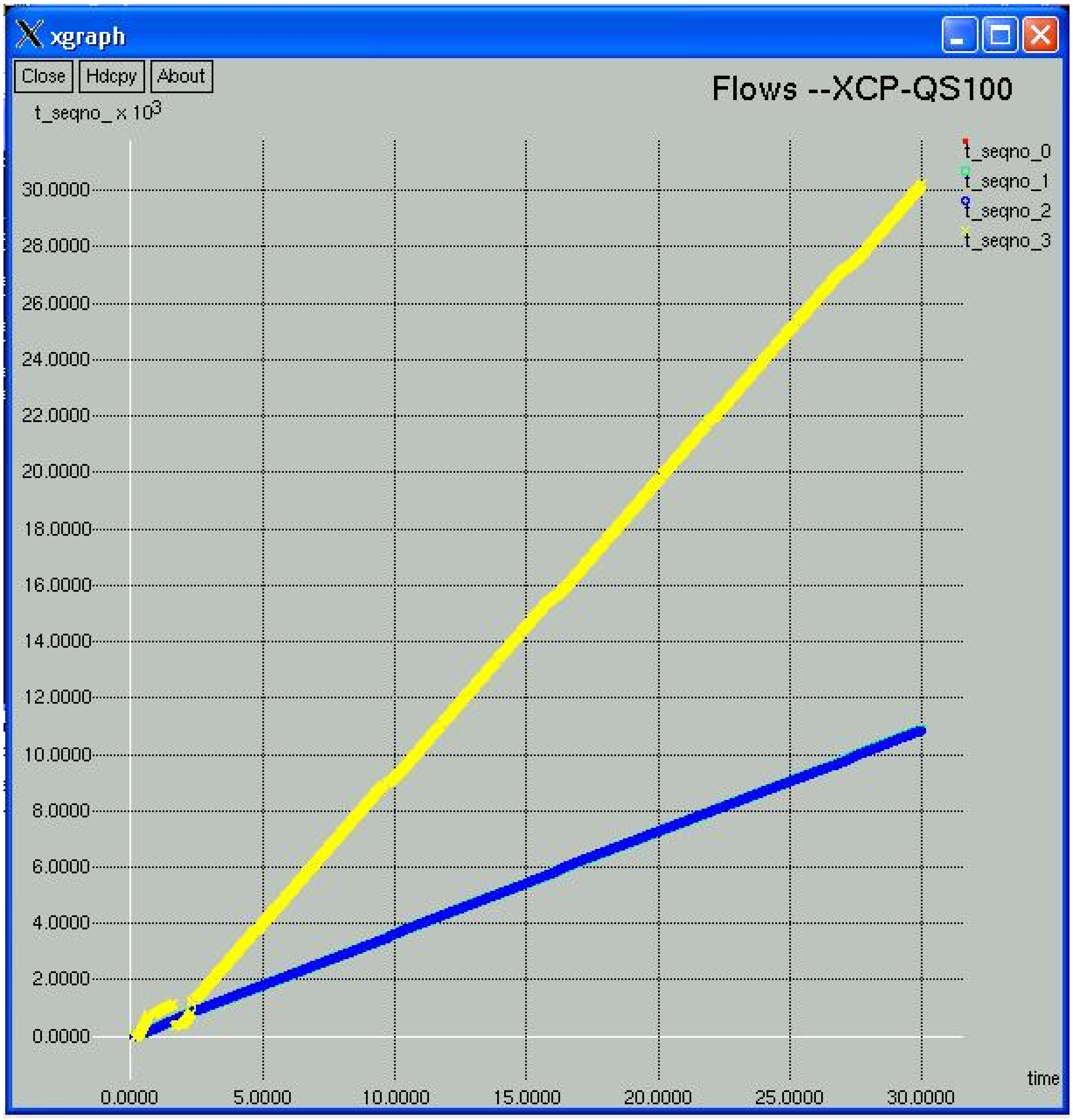


Figure 8 Comparison of XCP-AIMD Sequence Number with TCP flow

(iii) Comparing the throughput of the different XCP flows for duration of 30sec.

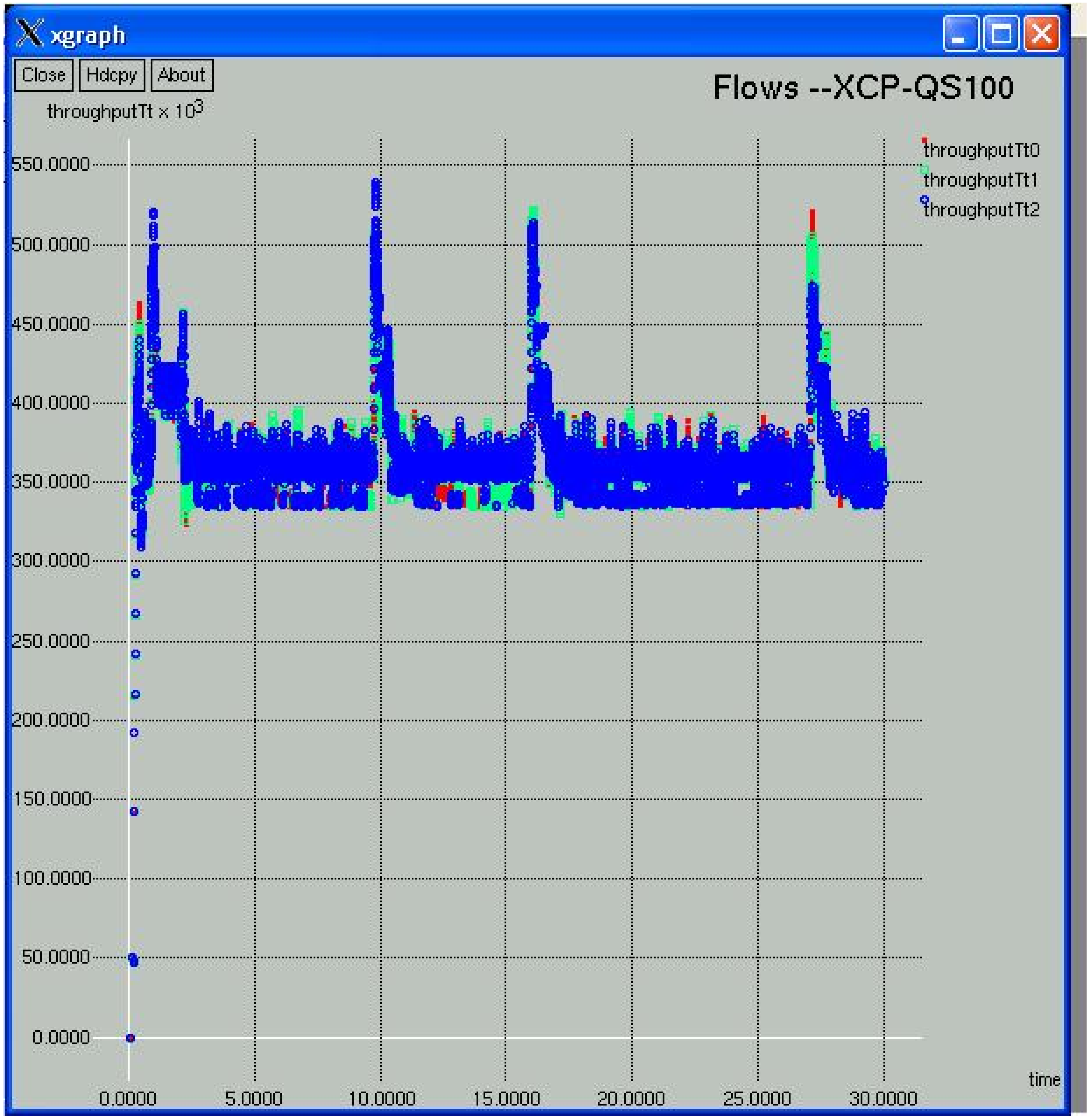


Figure 9 Comparison of XCP-AIMD Throughput

(iv) The utilization of the XCP flows for a duration of 30sec.

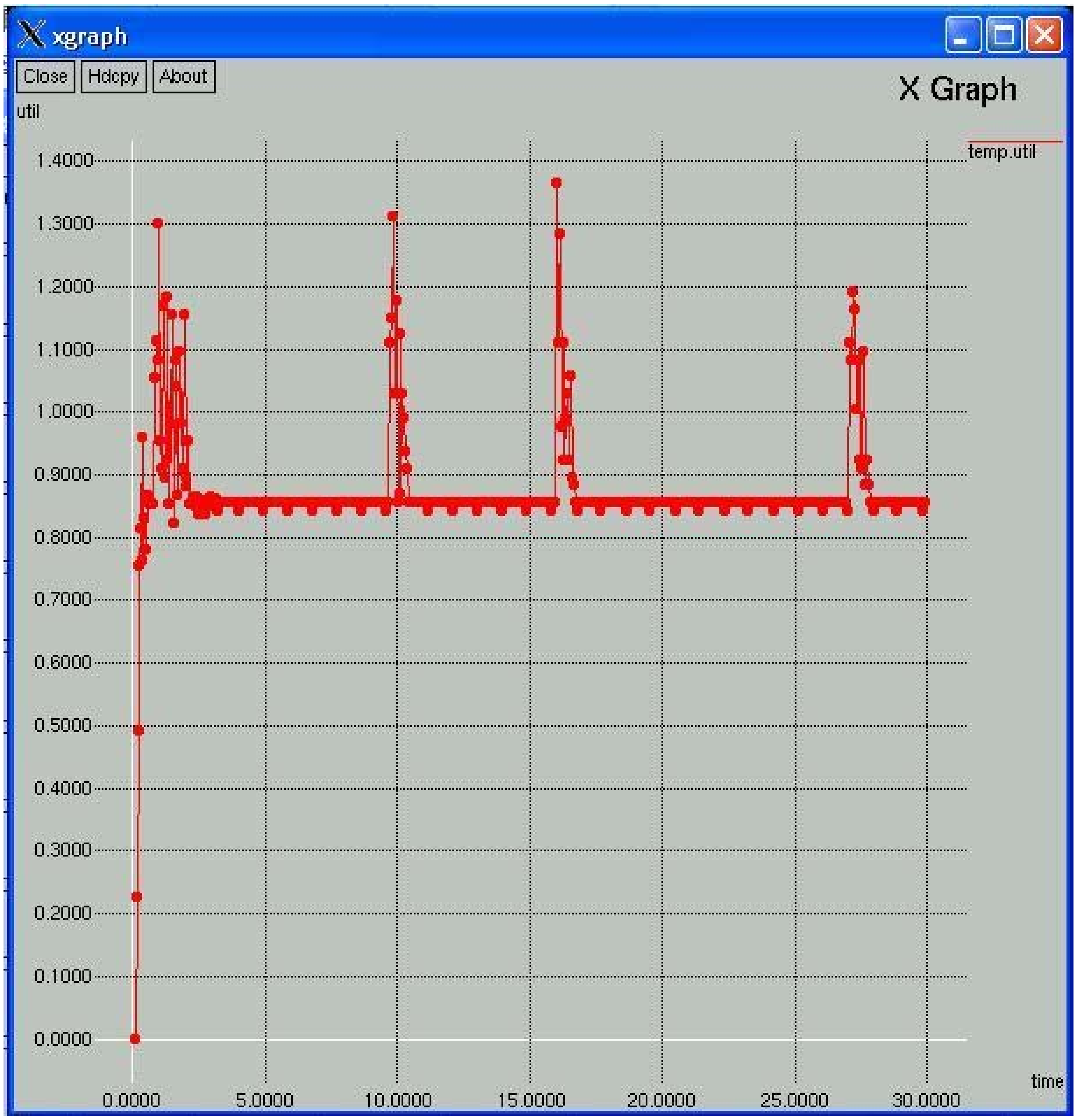


Figure 10 XCP-AIMD Utilization

XCP-modified SIMD

The XCP was next plugged in with the control equations of SIMD. It was seen that the SIMD equations with their square increase component did not improve the overall performance of the XCP when compared to the existing AIMD graphs. After trying different algorithms, the fourth root increase component and alpha value corresponding to that, and a beta value of 1.267 was found to be optimal. The results are shown below.

(i) Comparing the congestion Window of the different XCP flows with the TCP flow for the duration of 30sec.

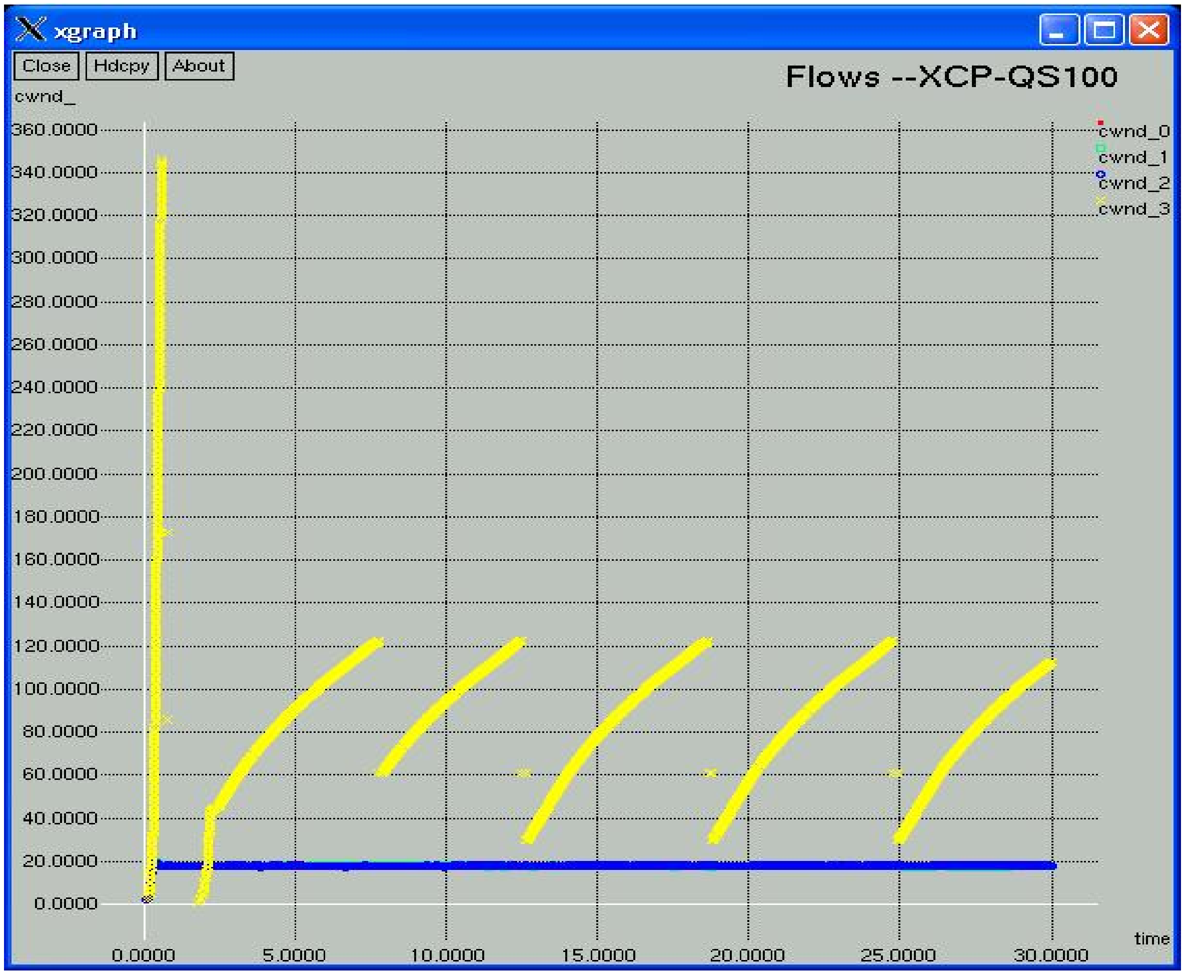


Figure 11 Comparison of XCP- fourth root SIMD Congestion window with TCP flow

(ii) Comparing the sequence numbers of the packets sent for the different XCP flows with the TCP flow for the duration of 30sec.

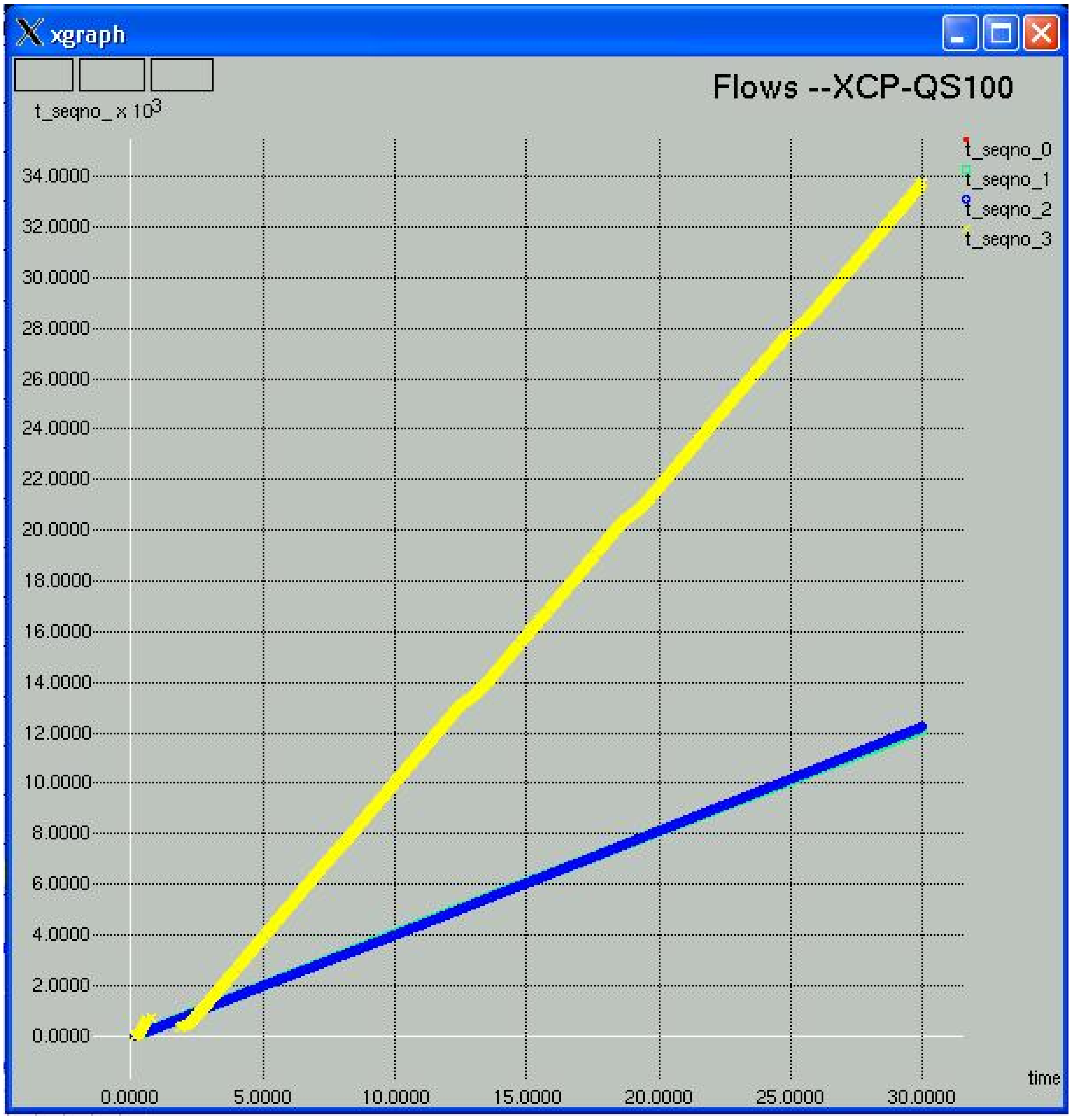


Figure 12 Comparison of XCP- fourth root SIMD Sequence Number with TCP flow

(iii) Comparing the throughput of the different XCP flows for a duration of 30sec.

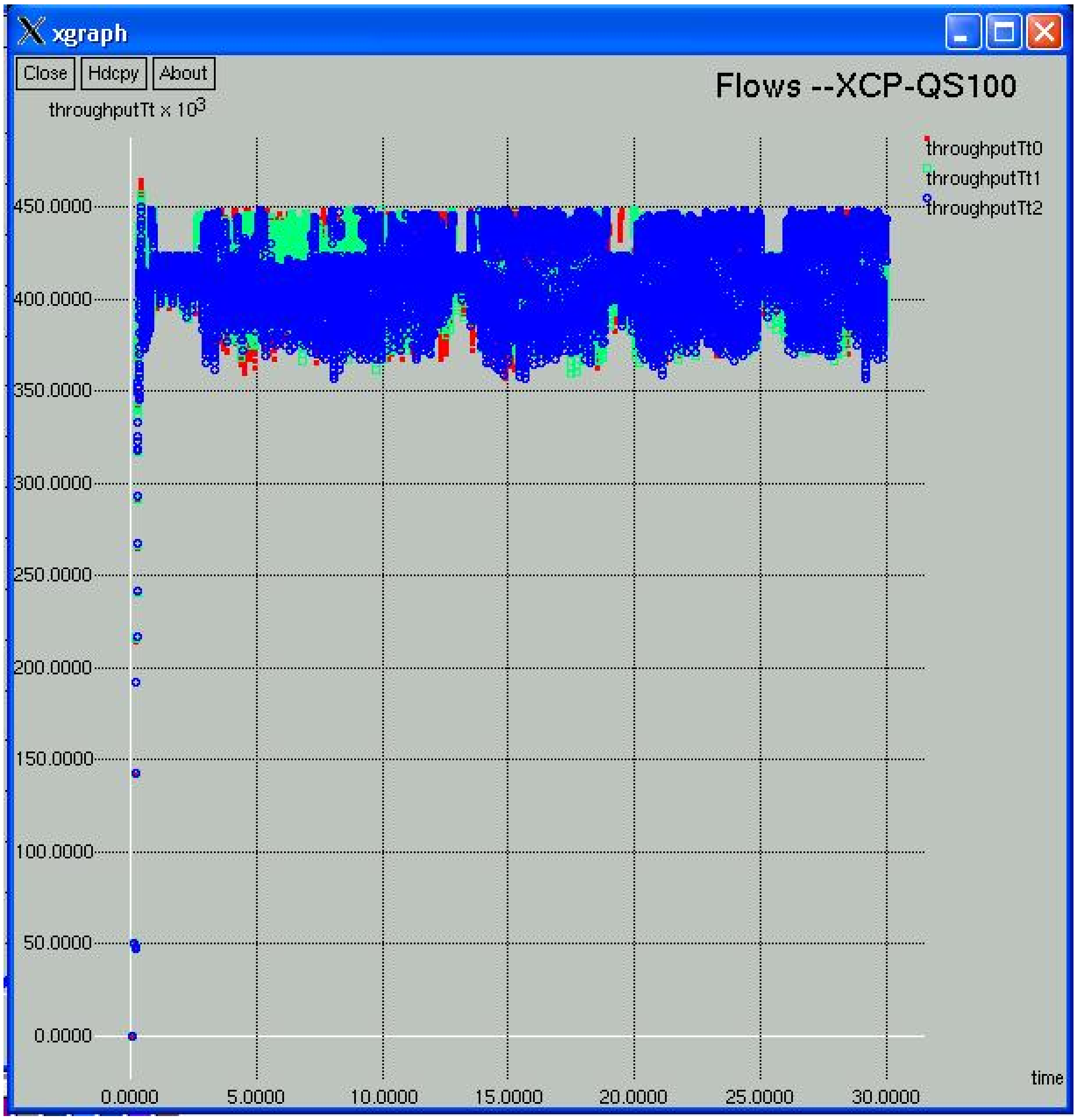


Figure 13 Comparison of XCP-fourth root SIMD Throughput

(iv) The utilization of the XCP flows for a duration of 30sec.

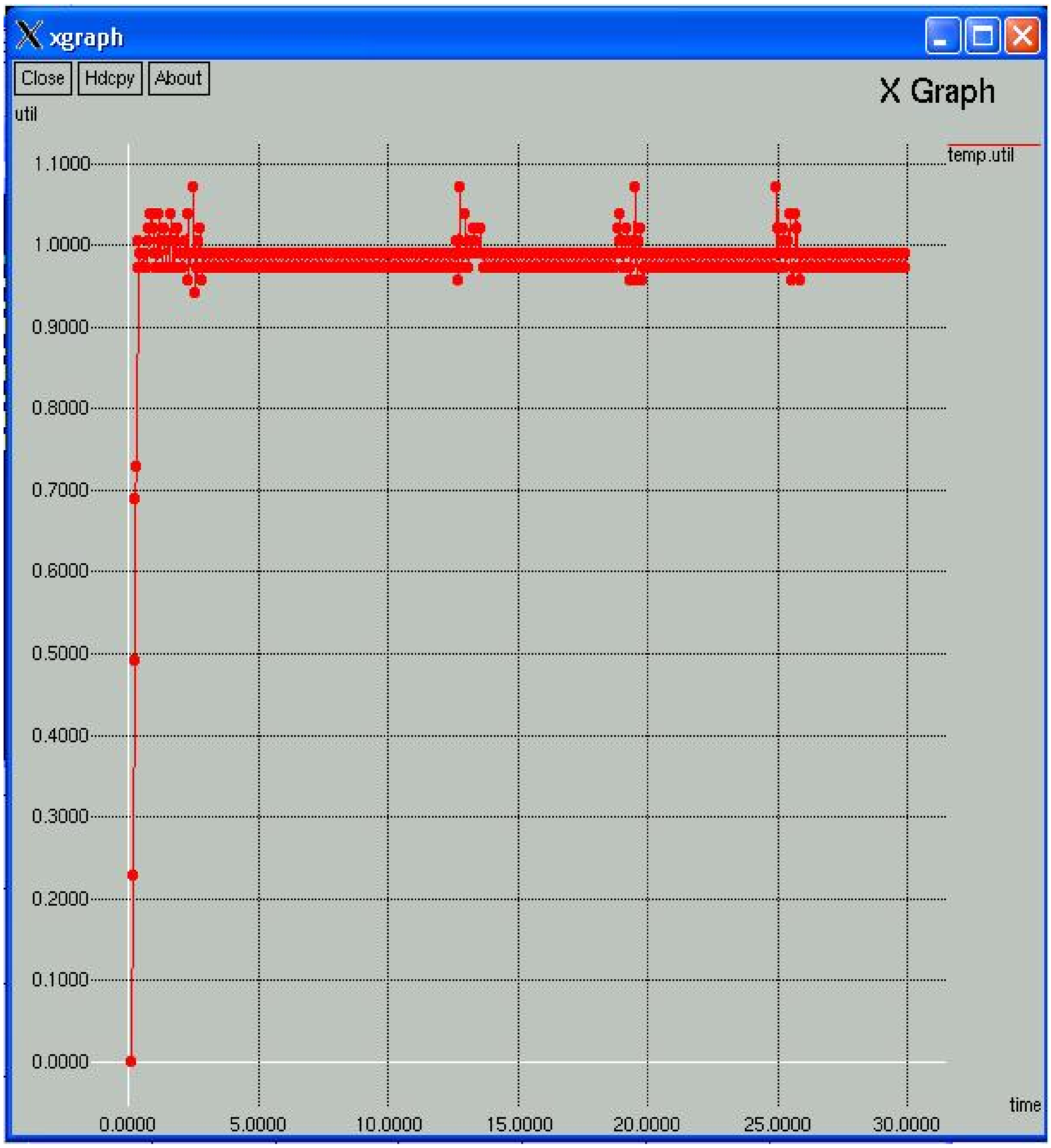
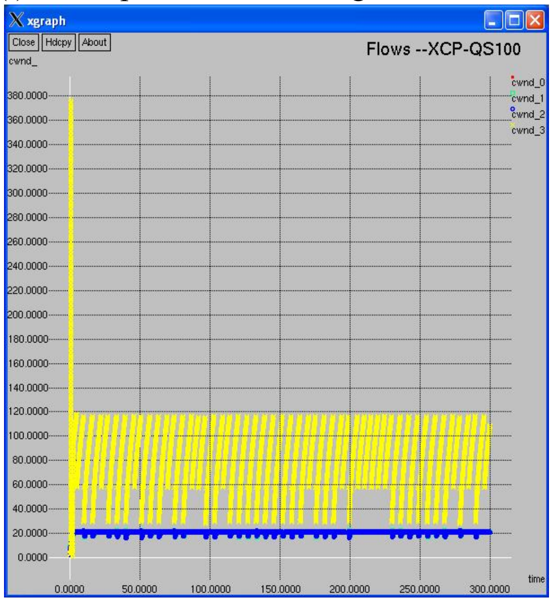


Figure 14 XCP-fourth root SIMD Utilization

XCP-AIMD and fourth root SIMD run for 300 seconds

The XCP with both AIMD and fourth root SIMD is run for 300 seconds to monitor their behavior over a longer period of time and the graphs are plotted.

(i) A comparison of the congestion window (cwnd) of the different XCP flows with TCP.



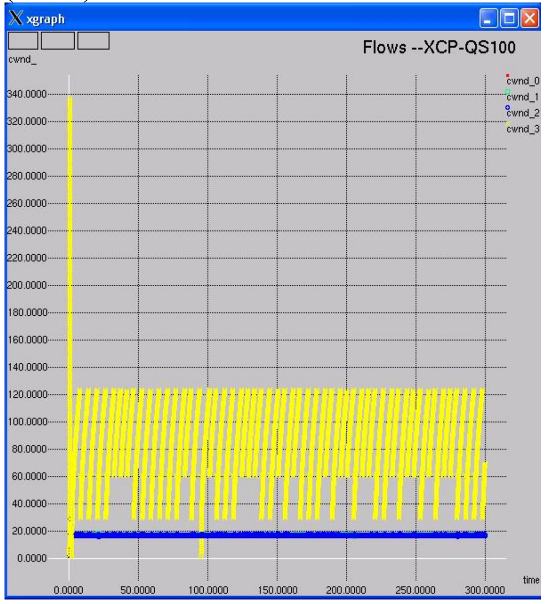
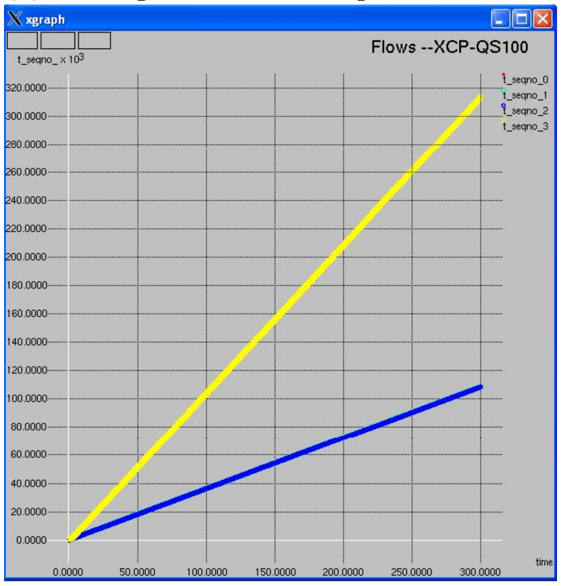


Figure 15 Cwnd comparison for XCP-AIMD(300 sec)

Figure 16 Cwnd Comparison for XCP-fourth root SIMD(300 sec)

(ii) A comparison of the sequence number of the different XCP flows with TCP flow.



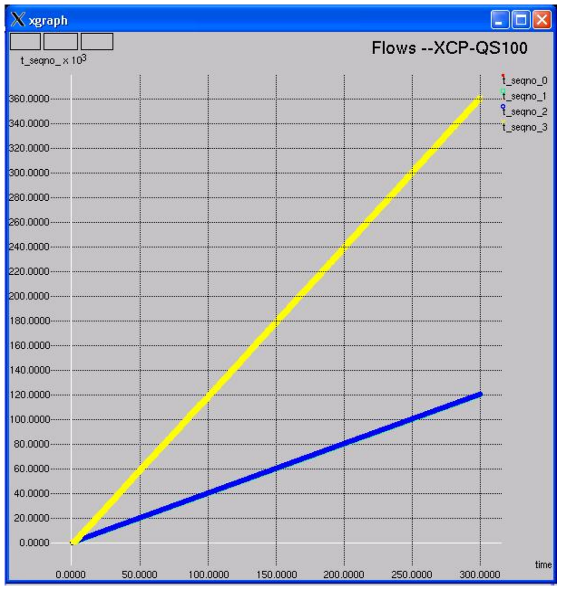
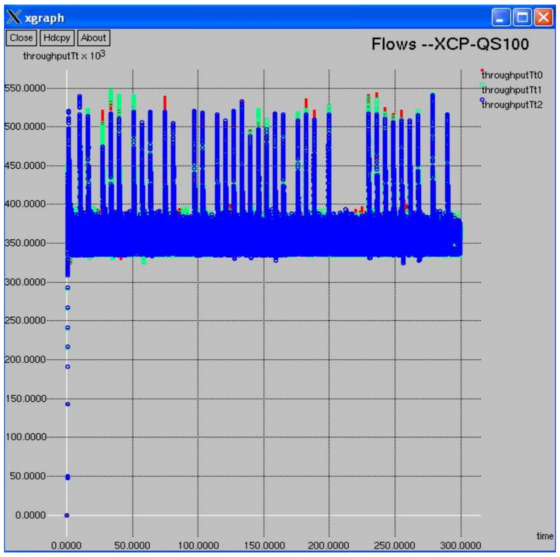


Figure 17 Sequence Number Comparison for XCP-AIMD(300 sec)

Figure 18 Sequence Number Comparison for XCP-fourth root SIMD(300 sec)

(iii) A comparison of the throughputs of the different XCP flows.



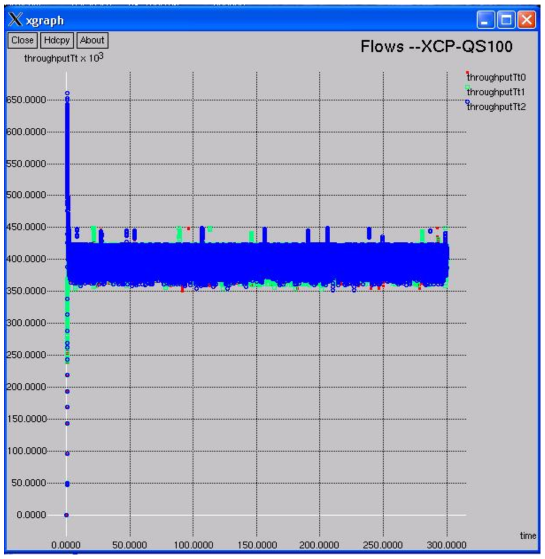
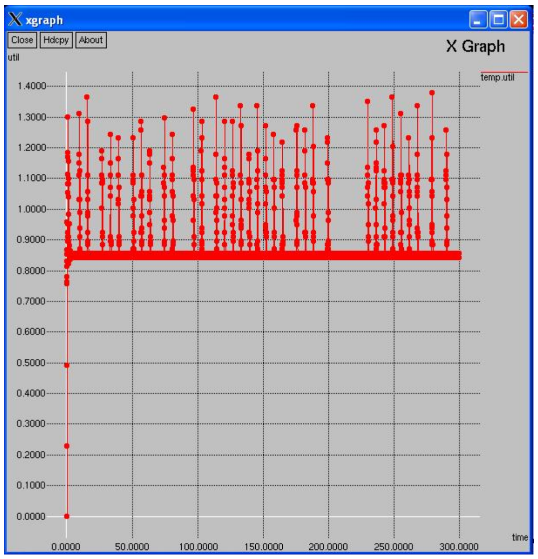


Figure 19 Throughput Comparison of XCP-AIMD(300 sec)

Figure 20 Throughput Comparison of XCP-fourth root SIMD(300 sec)

(iv)Utilization for the different XCP flows.



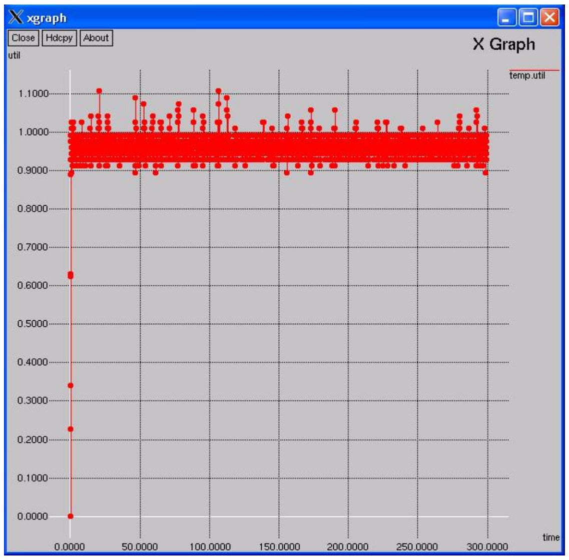


Figure 21 Utilization of XCP-AIMD(300 sec)

Figure 22 Utilization of XCP-fourth root SIMD(300 sec)

From the XCP graphs that are plotted for both the Additive Increase Multiplicative Decrease (AIMD) algorithm as well as the fourth-root Square Increase Multiplicative Decrease (SIMD) algorithm that is implemented after modifying the existing SIMD (with square root) algorithm, it is seen that the change to fourth-root, modifying w(t) and settling on a beta value of 1.267 has proven favorable in terms of performance and operability. In the XCP scenario, it has been observed that the modified SIMD algorithm performs better than the existing XCP-AIMD algorithm. From the graphs in Figures 7-22, an overall improvement is seen in the graphs that were obtained for XCP-fourth root SIMD when compared to XCP-AIMD. There is an increase to the congestion window (cwnd) in XCP-fourth root SIMD and there are also lesser oscillations. Also, the number of packets being sent (sequence number) is more in XCP-fourth root SIMD when compared to XCP-AIMD as seen from the graph where the XCP-AIMD graph shows only 11 packets at the maximum being set in the duration of 30 seconds whereas in XCP-fourth root SIMD, more than 12 packets are being sent for the same amount of time. The Throughputs of XCP-fourth root SIMD is also shown to be far more superior to XCP-AIMD. XCP-fourth root SIMD is seen to utilize the bottleneck resources much better than XCP-AIMD with XCP-fourth root SIMD's utilization being around 1.0 and XCP-AIMD's utilization around 0.8. There figures remain the same when the simulations were run for a longer period of time (300 seconds) and the behavior of the algorithms remain unmodified with XCP-fourth root SIMD emerging the obvious winner in terms of performance over XCP-AIMD that is being used at present in the XCP protocol.

XCP with Binomial Algorithms

The XCP protocol was then plugged with the two main binomial algorithms, SQRT and IIAD and the performance analyzed.

(i) A comparison of the congestion window (cwnd) of the different XCP flows with TCP.

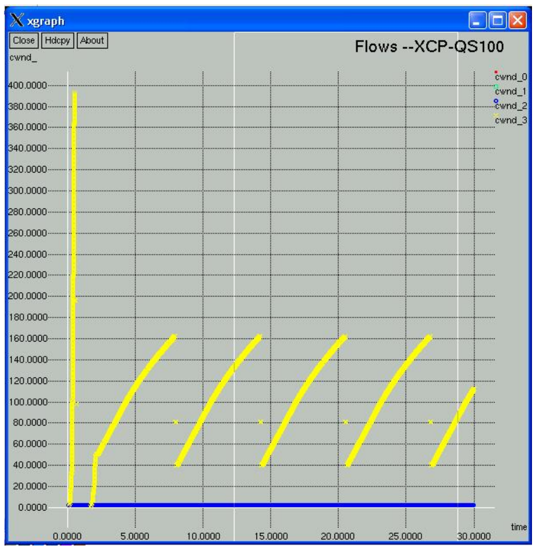


Figure 23 Cwnd Comparison of XCP-SQRT

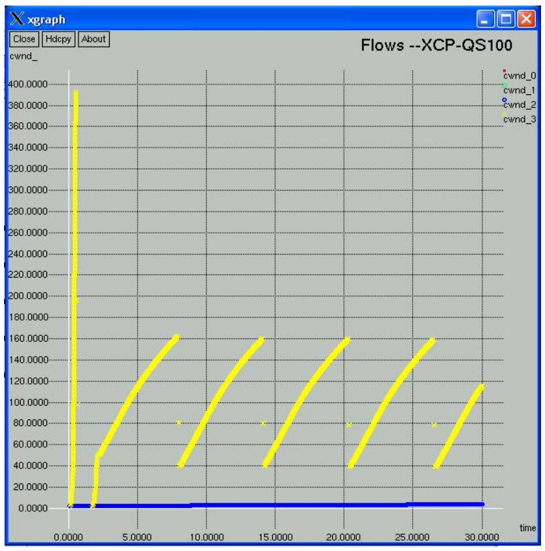


Figure 24 Cwnd Comparison of XCP-IIAD

(ii) A comparison of the sequence number of the different XCP flows with TCP flow.

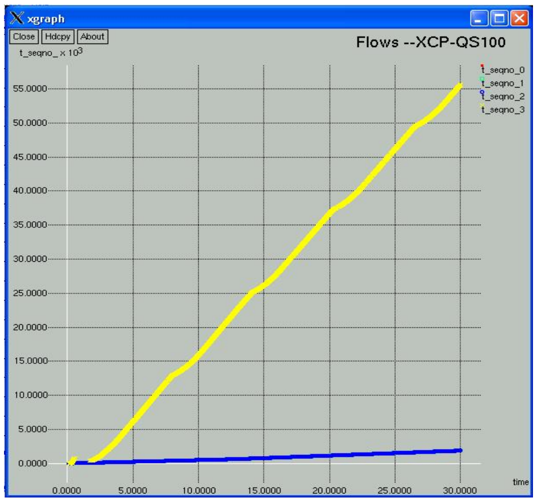


Figure 25 Sequence Number Comparison of XCP-SQRT

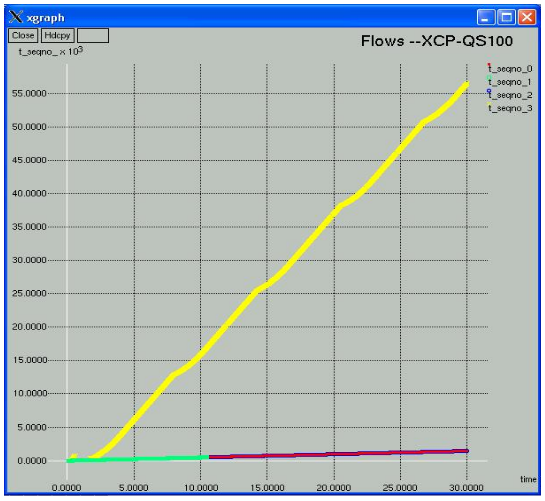
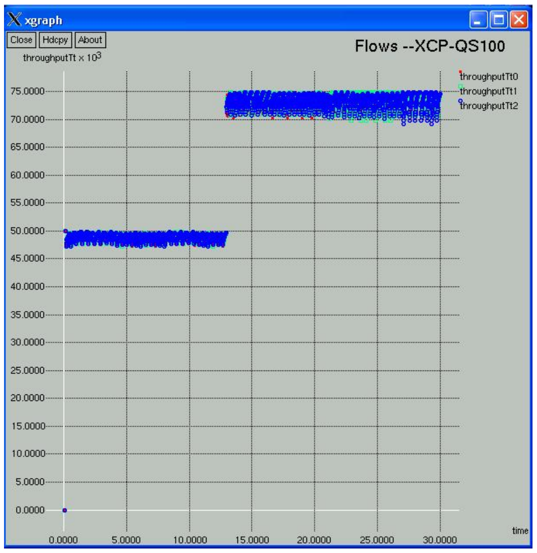


Figure 26 Sequence Number Comparison of XCP-IIAD

(iii) A comparison of the throughputs of the different XCP flows.



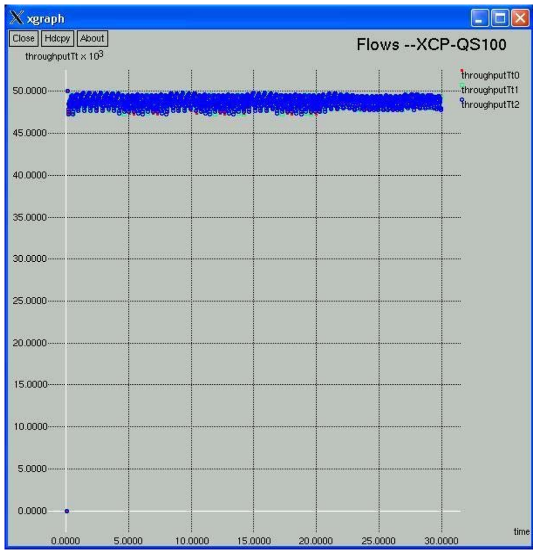
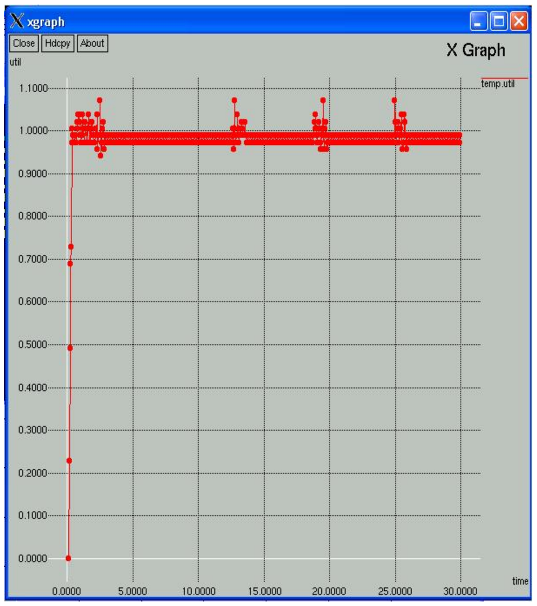


Figure 27 Throughput Comparison of XCP-SQRT

Figure 28 Throughput Comparison of XCP-IIAD

(iv)Utilization for the different XCP flows.



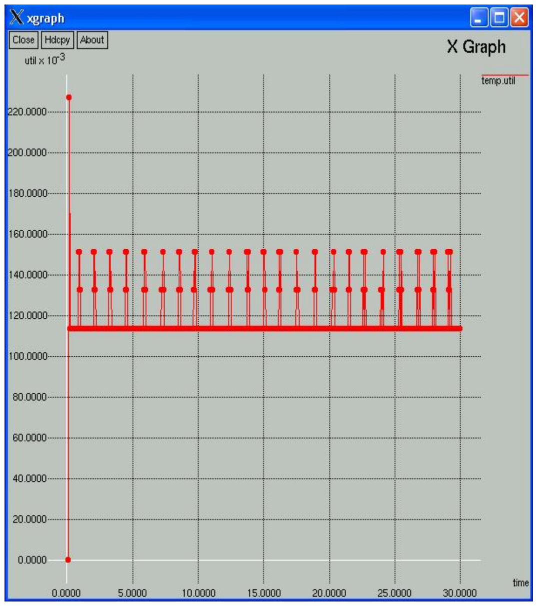


Figure 29 Utilization of XCP-SQRT

Figure 30 Utilization of XCP-IIAD

It can be seen from the two different graphs of XCP-SQRT and XCP-IIAD, that although the XCP-SQRT is slightly better in performance when compared to XCP-IIAD, they are not better than XCP-AIMD or XCP-fourth root SIMD.

XCP with MIMD

The XCP protocol was then plugged with the Multiplicative Increase, Multiplicative Decrease algorithm and the performance was studied.

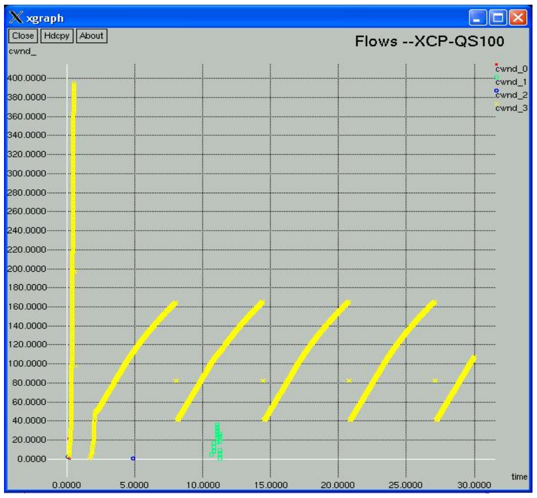


Figure 31 Cwnd Comparison of XCP-MIMD

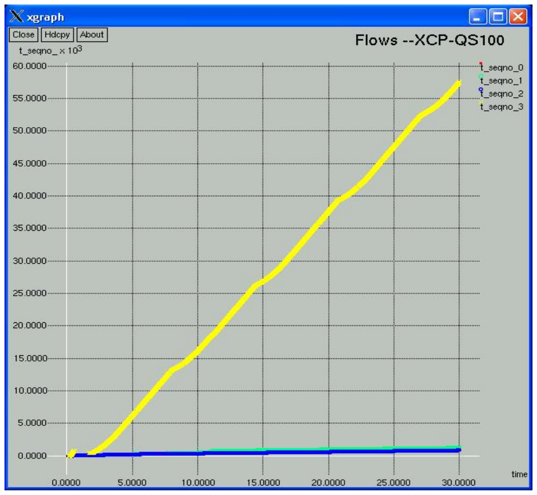
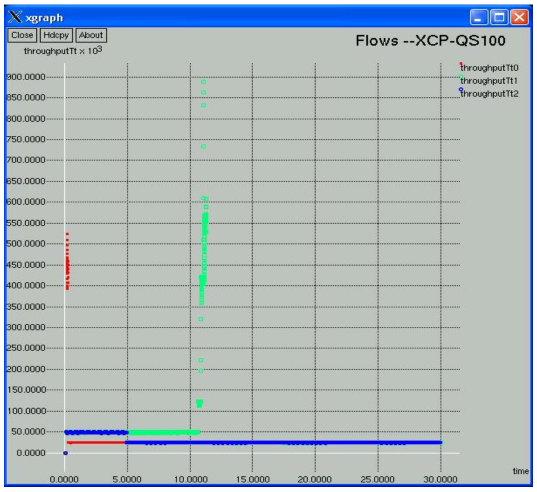


Figure 32 Sequence Number Comparison of XCP-MIMD



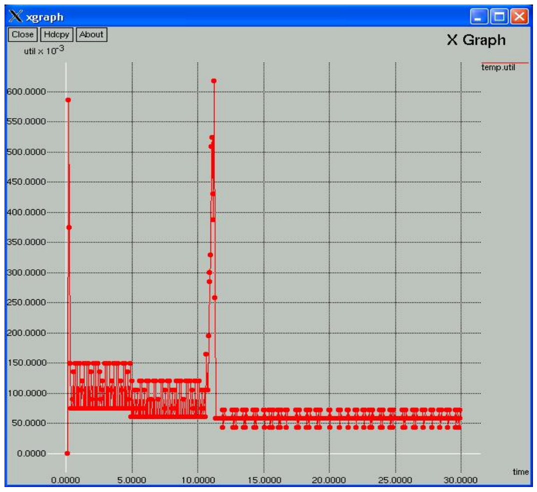


Figure 33 Throughput Comparison of XCP-MIMD

Figure 34 Utilization of XCP-MIMD

It can be seen that the performance of the MIND algorithm is also not that efficient. It only worsens the existing performance provided by the XCP-AIMD algorithm.

V: Future Work and Conclusion

Future Work

The SIMD paper [4] defines a *congestion epoch* as a series of congestion window increments followed by one window decrement. The SIMD was proposed for TCP where an acknowledgement signals a window increment and a packet drop signals a window decrement. TCP uses an implicit and binary feedback and hence the concept of history information as proposed in [4] proved vital in improving its performance. XCP, on the other hand, has an explicit feedback that is sent from the router to the sender and with the help of that feedback the sender either increases or decreases its window size. Hence, the perception of the *congestion epoch* and its significance in XCP is altered. While plugging in the SIMD equations in XCP, thus, the translation of the *congestion epoch* with its congestion window zero (w_0) and maximum window (w_{max}) might not have been accurate. A more detailed study of this is needed to be done.

The SIMD equations when plugged in directly into XCP did not yield favorable results. Hence, by trying out different combinations of the algorithm, it was seen that when at time *t* instead of using the current congestion window (w_t), when the previous congestion window (w_{t-1}) is used, along with a fourth root to compute the difference between w_{t-1} and w_0, with beta value set as 1.267, it yields better performance results. Further work needs to be done it terms of understanding theoretically how this might be better than the SIMD equation proposed in [4] and using control theory to substantiate the modified algorithm.

Conclusion

This paper studies the Explicit Control Protocol proposed in [3] and the detailed version in [16]. XCP protocol was introduced as a new way to improve the congestion control, bandwidth utilization and the congestion avoidance algorithm in TCP. Unlike other improvements that were proposed for TCP, XCP does not try to be backward compatible with existing TCP implementations. XCP has been shown to outperform TCP, especially in dedicated peer-to-peer networks that boast of high bandwidth and perhaps even high delays. This paper also studies the performance of different window control algorithms that have been newly proposed in the research community and gives a brief description of the coupling of an explicit router feedback with Binomial Algorithms (SQRT, IIAD), MIMD and SIMD and to study their performance in terms of better fairness, higher smoothness, aggressiveness and responsiveness upon congestion. Simulations in *ns2 [17]* with the XCP protocol were done and the results were studied. It was seen that the XCP with the original AIMD algorithm had a high performance ratio. The SIMD equations when plugged in as such did not seem to yield higher performance. Another disadvantage in using SIMD is that the SIMD paper [4] does not clearly enunciate what a *congestion epoch* is and hence there is a gray area while translating the equations for the XCP fairness controller. But, when a fourth root for the calculation of alpha is used in the denominator, it results in a slightly better XCP performance.

REFERENCES

[1] R. Jain, K. Balakrishnan and D. Chiu. Congestion Avoidance in Computer Networks with a Connectionless Network Layer. *Technical Report DEC-TR-506, Digital Equipment Corporation, August 1987*

[2] D. Bansal and H. Balakrishnana. Binomial congestion control algorithms. *In Proceedings of IEEE INFOCOM, April 2001.*

[3] D. Katabi, M. Handley and C. Rohrst. Congestion Control for High Bandwidth-Delay Product Networks. *In Proceedings of ACM SIGCOMM, August 2002.*

[4] S. Jin, L. Guo, I. Matta and A. Bestavros. TCP-friendly SIMD Congestion control and its Convergence Behaviour. *In Proceedings of ICNP 2001, November 2001.*

[5] Floyd S., Jacobsen V., "Random Early Detection Gateways for Congestion Avoidance", In *IEEE/ACM* Transactions on Networking, 1(4):397-413, August 1993

[6] Low S.H., Paganini F., Wang S., Adlakah S., Doyle J.C., "Dynamics of TCP/RED and a scalable control". *In Proc. Of IEEE INFOCOM, June 2002.*

[7] Athuraliya S., Li V. H., Low S. H., Yin Q., "REM: Active Queue Management", *IEEE Network Magazine, vol. 15, pp. 48-53, May 2001*

[8] Stevens W. "TCP Slow Start, Congestion Avoidance, Fast Retransmit, and Fast Recovery Algorithms", *Internet Request For Comments, RFC 2001, January 1997*

[9] Clark D., "Window and acknowledgment strategy in TCP", *Internet Request For Comments, RFC 813, July 1982*

[10] Mathis M., Mahdavi J., "Forward Acknowledgment: Refining TCP Congestion Control", *Proceedings of ACM SIGCOMM, Stanford, CA, August, 1996*

[11] Mathis M., Mahdavi J., Floyd S., Romanov A., "TCP Selective Acknowledgment Options", *Internet Request For Comments, RFC 2018, October 1996*

[12] Floyd S. "HighSpeed TCP for Large Congestion Windows", *Internet Request for Comments, RFC 3649, December 2003*

[13] Katabi D., Handley M., "Using Precise Feedback for Controlling Congestion in the Internet", *Technical Report MIT/LCS/TR-820, May 2001*

[14] Katabi D., "Decoupling Congestion Control and Bandwidth Allocation Policy with Application to High Bandwidth-Delay Product Networks", *Masters Thesis, MIT, March 2003*

[15] Chiu D., Jain R., "Analysis of Increase and Decrease Algorithms for congestion avoidance in Computer Networks", *Computer Networks and ISDN systems, June 1989*

[16] Van Jacobson, "Congestion avoidance and control", *Proceedings of SIGCOMM, August 1988.*

[17] The Network Simulator ns-2. http://www.isi.edu/nsnam/ns.

[18] XCP@ISI-"The Explicit Control Protocol" http://www.isi.edu/isi-xcp/

[19] Cygwin http://www.cygwin.com/

[20] Red Hat Linux http://www.redhat.com/

[21] Fedora Core 2 http://fedora.redhat.com/

[22] Floyd S., Jacobson V., "Random Early Detection for congestion avoidance", *IEEE/ACM Transactions of Networking, August 1993*

[23] Athuraliya S., Li V.H., Low S.H., Yin Q., "Rem: Active Queue Management" *IEEE Network, June 2001*

[24] Kunniyur S., Srikat R.,"Analysis and design of an adaptive virtual queue" *Proceedings of ACM SIGCOMM, April 2001*

[25] Floyd S., Gummadi R., Shenker S., "Adative Red: An Algorithm for increasing the robustness of red", *September 2001*

[26] Altman E., Acrachenkov K., Barakat C., Kherani A.A., Prabhu B.J., " Analysis of MIMD congestion control algorithm for high speed networks" *Computer Networks, August 2005*

VITA

Aarthi Harna Trivesaloor Narayanan

Candidate for the Degree of

Master of Science

Thesis: A STUDY ON THE PERFORMANCE OF TRANSPORT PROTOCOLS COMBINING EXPLICIT ROUTER FEEDBACK WITH WINDOW CONTROL ALGORITHMS

Major Field: Computer Science

Biographical:

Education: Graduated with Bachelors of Technology in Information Technology from University of Madras, Chennai, India in May 2003. Completed the requirements for the Master of Science degree with major in computer Science at Oklahoma State University in December 2005.

Experience: Employed as a graduate research assistant at the Department of Biochemistry and as an IT Lab Assistant at the CEAT labs, Oklahoma State University, 2004-Present.

Professional Memberships: Association of Computing Machinery, Computer Society of India