LTCP-RC: RTT COMPENSATION TECHNIQUE TO SCALE HIGH-SPEED PROTOCOL IN HIGH RTT LINKS

A Thesis

by

SAURABH JAIN

Submitted to the Office of Graduate Studies of
Texas A&M University
in partial fulfillment of the requirements for the degree of

MASTER OF SCIENCE

August 2005

Major Subject: Computer Engineering

LTCP-RC: RTT COMPENSATION TECHNIQUE TO SCALE HIGH-SPEED PROTOCOL IN HIGH RTT LINKS

A Thesis

by

SAURABH JAIN

Submitted to the Office of Graduate Studies of
Texas A&M University
in partial fulfillment of the requirements for the degree of

MASTER OF SCIENCE

Approved by:

Chair of Committee,	A. L. Narasimha Reddy
Committee Members,	Pierce Cantrell
	Deepa Kundur
	Riccardo Bettati
Head of Department,	Chanan Singh

August 2005

Major Subject: Computer Engineering

ABSTRACT

LTCP-RC: RTT Compensation Technique to Scale
High-Speed Protocol in High RTT Links. (August 2005)
Saurabh Jain, B.Tech., Indian Institute of Technology Bombay
Chair of Advisory Committee: Dr. A.L. Narasimha Reddy

In this thesis, we propose a new protocol named Layered TCP with RTT Compensation (LTCP-RC, for short). LTCP-RC is a simple modification to the congestion window response of the high-speed protocol, Layered TCP (LTCP). In networks characterized by large link delays and high RTTs, LTCP-RC makes the LTCP protocol more scalable. Ack-clocked schemes, similar to TCP, suffer performance problems like long convergence time and throughput degradation, when RTT experienced by the flow increases. Also, when flows with different RTTs compete, the problem of unfairness among competing flows becomes worse in the case of high-speed protocols. LTCP-RC uses an RTT Compensation technique in order to solve these problems. This thesis presents a general framework to decide the function for RTT Compensation factor and two particular design choices are analyzed in detail. The first algorithm uses a fixed function based on the minimum RTT observed by the flow. The second algorithm uses an adaptive scheme which regulates itself according to the dynamic network conditions. Evaluation of the performance of these schemes is done using analysis and ns-2 simulations. LTCP-RC exhibits significant performance improvement in terms of reduced convergence time, low drop rates, increased utilization in presence of links with channel errors and good fairness properties between the flows,. The scheme is simple to understand, easy to implement on the TCP/IP stack and does not require any additional support from the network resources. The

choice of parameters can be influenced to tune the RTT unfairness of the scheme, which is not possible in TCP or other high-speed protocols. The flexible nature of the analysis framework has laid the ground work for the development of new schemes, which can improve the performance of the window based protocols in high delay and heterogeneous networks.

To my parents

ACKNOWLEDGMENTS

I am very grateful to my advisor, Dr. A. L. Narsimha Reddy, for giving me the opportunity to work with him. I owe him gratitude for showing me the direction for the research. Without his constant guidance, suggestions and encouragement, this work would not have been possible. He has answered all my questions very patiently and supported and encouraged me, whenever I needed him. The informal group meetings organized by him have been a constant source of knowledge and inspiration. I also want to thank him for all the facilities and support provided to me. Thanks a lot, Dr. Reddy, for everything.

I would also like to express my sincere acknowledgment to Sumitha Bhandharkar. Her constant support, valuable comments and guidance have helped me throughout the course of my master's study. She has helped me in learning new things, given time to discuss the problems and has been a source of inspiration all along.

I would also like to thank my parents and brother, who taught me the value of hard work by their own example. I would like to share my moment of happiness with them. Without their encouragement and confidence in me, I would have never been able to pursue and complete my master's study.

Last but not the least, I would like to thank all my friends, who directly and indirectly supported and helped me, in completing this thesis.

TABLE OF CONTENTS

CHAPTER Page

LIST OF TABLES

LIST OF FIGURES

CHAPTER I

INTRODUCTION

The current version of the TCP protocols (Tahoe, Reno, NewReno) suffer performance problems in connections characterized by relatively high error rates and long propagation delays, such as those that encompass terrestrial and satellite radio links. Changes in the communication networks over the last few years have led to ever-increasing availability of the network bandwidth and the deployment of high-speed links for high-delay transatlantic communication. This has posed a serious challenge for the AIMD algorithms used for congestion control in TCP. Over the past few years, several solutions and new protocols have been put forth for solving this problem and improving the performance of TCP in high-speed networks. HighSpeed TCP [1], Scalable TCP [2], Fast TCP [3], XCP [4], BIC-TCP [5], H-TCP [6] and LTCP [7] are some of the examples.

Most of the above protocols modify the congestion window response function of the TCP at the sender side and do not require any additional support from the network. They use the window-based transmission algorithm, which is triggered by incoming acknowledgments (ACK) from the receiver. It must be stressed that many high-speed networks run over long distances, connecting several organizations around the world, and their round trip times (RTTs) can rise beyond 200 milliseconds [5]. High RTTs reduce the congestion window growth rate, which results in significant throughput degradation. Even when two flows experience the same RTT, they may take long time to converge when they start at different time intervals. Moreover, when flows with different RTTs compete over the same bottleneck link, the flow with

---

The journal model is *IEEE Transactions on Automatic Control*.

longer RTT is penalized in terms of reduced throughput and unfair sharing of the network bandwidth. Most of the high-speed protocols suffer from this *RTT unfairness* problem, when the window increase rate gets larger as the congestion window grows [5]. RTT unfairness problem for high speed networks is exacerbated by drop tail routers where losses are highly synchronized [5].

In light of these problems, there has been a surge of interest for schemes which will scale the TCP protocol for both high-speed and long distance networks, and also reduce the RTT unfairness problem experienced by the high-speed schemes. In this thesis, we have tried to study the performance problems which occur due to increase in link delays and RTT observed by the flow. We propose a new scheme, *LTCP-RC*, which use an RTT Compensation technique to improve the performance of high-speed protocol, LTCP, in high RTT links. LTCP-RC scales the congestion window response of the LTCP protocol by using the RTT Compensation factor based on the RTT observed. The work presented in this thesis provides a detailed analysis framework and new set of techniques which can help to solve the RTT disparity problem.

A. Related Work

Effects of the increase in round trip time on the performance of TCP have been studied extensively in literature. An analytical model of TCP throughput was developed by J. Padhye et. al. [8], which provides the equation for the throughput obtained, in terms of loss rate and RTT observed by the flow. The equation given by this model can be represented in simplified form as,

$$BW = \frac{\sqrt{\frac{3}{2}}}{RTT\sqrt{p}} \tag{1.1}$$

Equation 1.1 shows that, for a given loss probability p and congestion window W, an increase in the observed RTT, results in a proportionate decrease in the throughput obtained by the TCP flow. In [9], Golestani et. al. have studied the dependency of the window increase rate on the round trip time. The authors have also investigated the impact of this dependence on the fairness properties of the algorithms. The authors introduced the notion of *window-oriented* and *rate-oriented* fairness and concluded that TCP shows window-oriented fairness. The paper states that for a TCP flow, window size at the equilibrium point is independent of the round trip time and only depends on the loss probability.

When two flows with different RTTs compete over the same bottleneck link then the TCP algorithm, by its design is biased against the longer RTT flow. Several schemes have been proposed to reduce this bias. Sally Floyd proposed a constant-rate window increase algorithm in [10]. The paper presents an algorithm in which, each flow increases its congestion window by $a * RTT^2$, where a is a fixed constant. Thus, each flow increases its window by a packets per second, such that flows with different RTTs achieve the same sending rate. TCP Hybla [11] provides an extension of Floyd's scheme [10] and aimed at providing a protocol to solve the RTT disparity problem of TCP. TCP Hybla modifies the congestion window update algorithm on the receipt of an acknowledgment. On a successful receipt of an acknowledgment, the congestion window is updated using the relation,

$$W_{i+1} = \begin{cases} W_i + 2^\rho - 1, & \text{during Slow Start} \\ W_i + \rho^2/W_i, & \text{during Congestion Avoidance} \end{cases} \tag{1.2}$$

Simulation results presented in the paper show that TCP Hybla flows are fair to each other during random link losses and when flows with different RTTs compete with each other.

The problem of RTT unfairness in high-speed protocols has been suggested by Rhee et. al. [5]. The authors state that, in order to scale the protocol, the window increase rate of most of the high-speed schemes gets larger as the window grows. This makes the RTT unfairness problem of high-speed protocols more severe. Several schemes have been proposed to scale TCP for both high speed and long distance networks. Below we present a brief review of two such schemes.

1. BIC TCP: Binary Increase Congestion Control

Rhee et. al. proposed BIC protocol to scale TCP for fast, long distance network in [5]. The primary goal of this scheme is to probe the available bandwidth aggressively initially and then, become less aggressive when the window gets closer to the maximum possible window. The algorithm uses the combination of *binary search increase* and *additive increase* to control the congestion window at the sender. In the binary increase mode, the sender keeps track of the maximum window (window at which packet drop occurred) and the minimum window (window at which there are no losses). The binary search is employed by calculating the target window as the mid-point between the maximum and the minimum. The congestion window is increased to this target window, if packet loss is not observed for a RTT. Then, the minimum is set to the new current window and the target is calculated again. If the distance between the target and the minimum window is larger than a threshold value, then additive increase is used. In this phase, congestion window is increased by a fixed amount at every RTT.

On a packet loss, BIC uses multiplicative decrease similar to TCP. But the decrease factor is 0.125 for BIC as compared to 0.5 for TCP. To make the convergence of two flows faster, BIC uses the fast convergence algorithm in which, it keep track of the window size at which packet drop occurred for two consecutive loss events.

By comparing the window size at which drop occurred between two consecutive loss events, it can be inferred whether the current window is larger or smaller than fair share. If window is in a downward trend, then the current window is larger than the fair share. The maximum is readjusted to the new target window and a new target is recalculated. Otherwise, window is increased using normal BIC algorithm. A flow with the larger window increases by a smaller rate, as compared to the flow with smaller window. This leads to faster convergence in BIC protocol. Due to the use of binary search increase, the BIC algorithm reduces its window increase rate when the current window gets closer to the target window size. Therefore, it results in low packet loss rates.

This paper also presents the issue of RTT unfairness and the authors state that *synchronized losses* makes the RTT unfairness problem more severe. Simulation results presented show that, in the case of drop tail routers, the number of synchronized loss can be quite substantial. This paper has studied the RTT unfairness of HSTCP [1] and STCP [2] through analysis and simulations on ns-2 simulator. Both the schemes exhibit serious RTT unfairness problem and performance worst than standard AIMD schemes. At larger window sizes, the RTT unfairness of the BIC is similar to the AIMD schemes but at lower window, BIC performs worse than AIMD. Results presented in the paper confirmed that BIC has good bandwidth utilization, better RTT unfairness properties as compared to HSTCP and STCP. With respect to TCP Friendliness, the performance of BIC is better at higher bandwidth but worse at lower bandwidth, when compared with HSTCP and STCP protocols.

2. H-TCP: TCP for High-Speed and Long-Distance Networks

This is another scheme [6] that adjusts the rate at which congestion window is increased on the sender side, in order to scale conventional TCP for high-speed and

long distance networks. The key idea in this scheme, is to use the time elapsed since the last packet drop experienced by the source, to decide the rate, α, at which source inserts packet into the network. The protocol uses two modes of operation. In the low-speed mode, it behaves as conventional TCP in order to maintain backward compatibility. In the high speed mode, α is calculated using a function of time elapsed since the last packet drop. The function proposed in the paper is given by,

$$\alpha^H(\Delta) = 1 + 10(\Delta - \Delta^L) + (\frac{\Delta - \Delta^L}{2})^2 \tag{1.3}$$

where Δ^L is the threshold at which the switch between the low-speed and the high-speed mode occur and α^H is the rate at which window is increased in the high-speed mode.

On observing a packet loss, the H-TCP algorithm uses an adaptive backoff mechanism to achieve maximum throughput and link utilization. The protocol keep track of the throughput, B^-, obtained by the flow, just before a congestion event. If the difference between B^- in the current and the previous loss event is greater than a threshold, then the window decrease factor β is set to 0.5. Otherwise, it is set to $\frac{RTT_{min}}{RTT_{max}}$ in order to achieve maximum throughput. This can be represented as,

$$\beta(k+1) = \begin{cases} 0.5 & |\frac{B^-(k+1) - B^-(k)}{B^-(k)}| > 0.2 \\ \frac{RTT_{min}}{RTT_{max}} & \text{otherwise.} \end{cases} \tag{1.4}$$

In order to achieve constant convergence time, H-TCP algorithm scale 'α' by the round-trip time observed by the flow. Mathematical intuition behind this scaling is not presented in this paper but it will make the throughput obtained by H-TCP flow proportional to RTT. This paper has not presented any results for higher RTT flows or flows competing in different RTT links. However, ns-2 simulation results presented reveal good fairness properties between two H-TCP flows. Other aspects like RTT

unfairness, TCP friendliness are not explored.

B. Motivation

With the increase in the available bandwidth and inter-continental communication, several researchers made an effort on improving the performance of TCP. There has been substantial amount of research in the area of improving TCP performance, when RTT increases. To reduce RTT unfairness, most of the schemes emphasized on scaling the window increase function by squared RTT to obtain fair bandwidth share. The key contribution of this thesis will be an RTT Compensation technique which can be tuned to achieve desired fairness properties. A detailed analytical framework is presented and different solutions are investigated. We also observe that a fixed solution might not be sufficient for all performance requirements. An adaptive technique which makes the protocol more or less aggressive according to dynamic network conditions is also explored. We aimed at obtaining a better understanding of the effects of different network factors and conditions on our scheme.

C. Organization

The rest of the thesis is organized as follows. In chapter II, we provide a brief background of the LTCP protocol. In Chapter III, an analysis of the problems faced by LTCP at high RTT and heterogeneous links are presented. In Chapter IV, LTCP-RC design, using a fixed function for RTT Compensation is proposed. Chapter V presents an adaptive algorithm for LTCP-RC, which regulate itself according to dynamic network conditions. Finally, Chapter VI, presents conclusions and recommended future work.

CHAPTER II

BACKGROUND

A. LTCP: Layered Transport Control Protocol

LTCP is a simple layering technique for the congestion window response of TCP to make it more scalable in high-speed networks [7]. The original LTCP scheme proposed a sender-side modification, which aimed at improving the performance of TCP only on high-bandwidth links. It uses a two-dimensional congestion control framework. The macroscopic control, employed layering to quickly and efficiently make use of the available bandwidth whereas microscopic control, extends the existing AIMD algorithm of TCP to determine the per-ack behavior. The scheme can be thought of as an emulation of multiple flows at the transport level, with the key contribution that the number of virtual flows adapt to dynamic network conditions.

LTCP algorithm is described as follows. When a flow operates at a higher layer it increases its congestion window faster than the flow operating at a lower layer. Initially, all new LTCP connections start with the first layer. If congestion is not observed for an extended period of time, it adds more number of layers. Operating at a particular layer K, the LTCP flow increases its congestion window more aggressively as compared to normal TCP. The congestion window is increased by $K/cwnd$ packets for each incoming ack, or equivalently, it is increased by K on the successful receipt of one window of acknowledgments. Each layer K is also associated with a step-size δ_K. When the current congestion window exceeds the window corresponding to the last addition of a layer (W_K) by the step size δ_K, a new layer is added. The layers in LTCP can be formalized as,

$$W_1 = 0, \quad W_2 = W_1 + \delta_1, \quad \ldots \quad W_K = W_{K-1} + \delta_{K-1} \tag{2.1}$$

Therefore, LTCP flow operates at a layer K, when the congestion window W lies between W_K and W_{K+1} as shown in Figure 1.

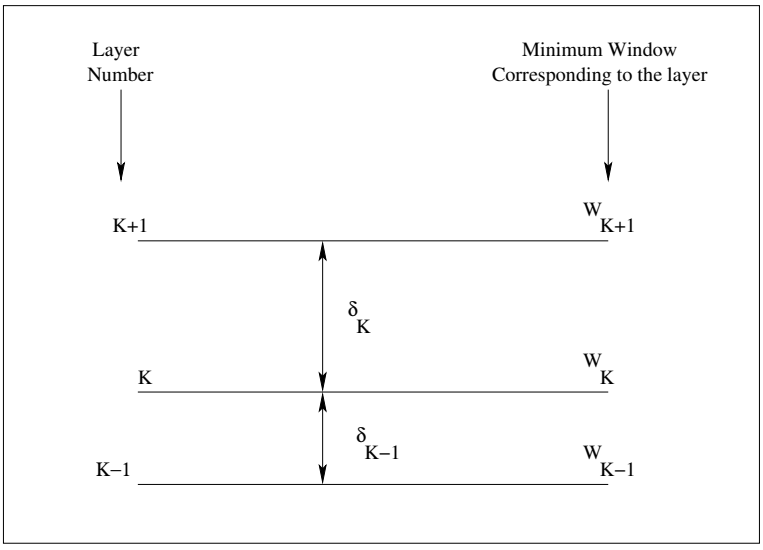


Fig. 1. Graphical Representation of Layers in LTCP

On a congestion drop event, LTCP reduces its congestion window using a multiplicative decrease β similar to TCP. The window reduction WR on the receipt of 3-duplicate acknowledgments can be represented as,

$$WR = \beta * W \tag{2.2}$$

The primary objective of the LTCP protocol design was to scale TCP in high speed links while making sure that the two LTCP flows, operating at same RTT, should be fair to each other. In order to ensure that two LTCP flows with same RTT, but starting at different times, converge, the number of RTTs taken by the larger flow to regain the lost bandwidth after a congestion event, should be larger than the recovery time of the smaller flow. This would make sure that the smaller flow

will grab the bandwidth faster than the larger flow and two flows will converge. To formulate this mathematically, we assume that the two flows are operating at layers K_1 and K_2 ($K_1 > K_2$) and WR_1 and WR_2 are the window reductions for each flow, upon a packet loss. After the packet drop, suppose the flows operate at layers $K_1^{'}$ and $K_2^{'}$, respectively. The flows take $\frac{WR_1}{K_1^{'}}$ and $\frac{WR_2}{K_2^{'}}$ RTTs respectively to regain the lost bandwidth. Following the above reasoning we can write the inequalities as,

$$\frac{WR_1}{K_1^{'}} > \frac{WR_2}{K_2^{'}} \tag{2.3}$$

[7] states that in order to allow smooth layer transitions after a window reduction due to a packet loss, at most one layer can be dropped. Therefore, a flow operating at layer K before the packet loss, should operate at layer either at layer K or *(K-1)*, after the window reduction. Keeping this in mind, the worst case for the convergence of two flows arises, when two flows operate in adjacent layers and the larger flows does not reduce a layer but the smaller flow does i.e. $K_1^{'} = K$ and $K_2^{'} = (K_2 - 1) = (K - 2)$. This gives the inequality,

$$\begin{aligned} \frac{WR_1}{K} &> \frac{WR_2}{K-2} \\ \Rightarrow \frac{W^{'}}{K} &> \frac{W^{''}}{K-2} \end{aligned} \tag{2.4}$$

The second equation above is derived from Equation 2.2 by substituting for *WR*. Again the worst case will occur, when the window $W^{'}$ will be close to transition to the layer *(K+1)* and the window $W^{''}$ has recently transitioned into layer *(K-1)* i.e. $W^{'} = W_{K+1}$ and $W^{''} = W_{K-1}$. For this worst case, Equation 2.4 can be re-written as,

$$W_{K+1} > \frac{K}{K-2} W_{K-1} \tag{2.5}$$

Based on the above inequality, the increase behavior for LTCP is chosen conser-

vatively as,

$$W_K = \frac{K+1}{K-2} W_{K-1} \tag{2.6}$$

With this choice of W_K and starting layers at $W_2 = W_T$, we have,

$$W_K = \frac{K(K+1)(K-1)}{6} W_T \tag{2.7}$$

where W_T is the window threshold parameter and is set to 50, in the current implement. By defining, $\delta_K = W_{K+1} - W_K$, the size of layer K is given by,

$$\delta_K = \frac{K(K+1)}{2} W_T \tag{2.8}$$

The above analysis is based on the assumption that after a window reduction due to a packet drop, at most one layer is dropped. In order to ensure this, parameter β should be chosen appropriately. That means, that the window reduction at layer K should be smaller than the size of the layer below, δ_{K-1}. [7] suggests use of smaller value of $\beta = 0.15$ for LTCP as compared to $\beta = 0.5$ for TCP. This value is chosen in order to support $K = 19$ and to maintain a full link utilization for a 2.4Gbps link with an RTT of 150ms (where window size can grow to 30,000 packets).

Results obtained from analysis and simulations have shown that LTCP protocol shows very high bandwidth utilization, low drop rates, excellent convergence and fairness properties. But being an ack-clocked, window controlled scheme similar to TCP, the aggressiveness of LTCP depends on RTT. Performance reduces as the link delay and RTT observed by the flow increases. In the next chapter, we present the problems which arise due to increase in RTT and following chapters will provide possible solutions.

CHAPTER III

PROBLEM AT HIGH RTT

LTCP congestion control algorithm can be viewed as a feedback system, where the input is the information about the congestion in the network and the output is the sending rate or the congestion window of the flow. As the link delay increases, the acknowledgments from the receiver are delayed. This reduces the congestion window growth rate and leads to the significant throughput degradation. According to the throughput analysis for LTCP given in [7], the throughput BW obtained by an LTCP flow on a network with round trip time RTT, loss probability p, is given by,

$$BW = \frac{\sqrt{\frac{K'}{\beta}(1 - \frac{\beta}{2})}}{RTT\sqrt{p}} \tag{3.1}$$

where, β is the factor by which the congestion window is reduced and $K^{'}$ is the layer at which the flow operates after the packet drop.

Equation 3.1 shows that the throughput obtained by the LTCP flow is inversely proportional to the RTT. Since $BW = W/RTT$, for a given loss probability p and a fixed window size W, an increase in RTT will result in a proportionate reduction in the throughput obtained by the LTCP flow. Equation 3.1 can also be interpreted in other way: for a given link bandwidth BW, as RTT increases, the congestion window W required to fill the pipe had to increase. The larger target for the congestion window coupled with the increase in RTT, will increase the time required to fully utilize the available bandwidth. In the following section, we present an analysis for interaction of two LTCP flows. The behavior of LTCP is analyzed in terms of RTT unfairness and convergence time.

A. Interaction of Two LTCP Flows

In this section, the behavior of LTCP when two flows compete over the same bottleneck bandwidth is analyzed. Since, the behavior depends on whether the flows experience same or different RTTs, these two cases are analyzed separately. The analysis presented in this section is based on the assumption of synchronized loss model. In networks with drop-tail routers or high speed links with low multiplexing, the fraction of synchronized losses are quite significant [5]. Therefore, this assumption is made to simplify the analysis.

1. Case 1: Flows Competing at Different RTTs

The analysis presented in this section is based on the analysis presented in [5] for RTT unfairness. Assuming synchronized losses, the time between two drop events t, will be same for both the flows. For a flow i with round trip time RTT_i and packet loss probability p_i, the average number of packets sent by the flow between two drop events is given by $1/p_i$. Also, the number of RTTs between two consecutive loss events is t/RTT_i. Therefore, average window size of the flow i is given by,

$$W_i = \frac{1/p_i}{t/RTT_i} = \frac{RTT_i}{tp_i} \tag{3.2}$$

From Equation 3.1, the bandwidth of LTCP flow is given by,

$$\begin{aligned} BW &= \frac{W_i}{RTT_i} = \frac{\sqrt{\frac{K_i'}{\beta}(1-\frac{\beta}{2})}}{RTT\sqrt{p}} \\ \Rightarrow p_i &= \frac{K_i'C}{W_i^2} \quad where \quad C = \frac{1}{\beta}(1-\frac{\beta}{2}) \end{aligned} \tag{3.3}$$

By substituting p_i in Equation 3.2 we get,

$$W_i = \frac{tK_i'C}{RTT_i} \tag{3.4}$$

Equation 2.7, gives the relationship between the window size W and the operating layer K for the LTCP flow as,

$$W \propto K^3 \quad or \quad K \propto W^{1/3} \tag{3.5}$$

Substituting this relationship in Equation 3.4, we get,

$$W_i \propto (\frac{tC}{RTT_i})^{3/2} \tag{3.6}$$

RTT unfairness is defined as the ratio of the throughput obtained by two flows in terms of the ratio of the RTTs. RTT unfairness for LTCP can be found by dividing Equation 3.6 for two flows. By synchronized loss model, time t will be same for two flows. Also, the factor C is constant for a given value of β and will remain same for the two flows. Hence, RTT unfairness for LTCP is given by,

$$\begin{aligned} & \frac{(\frac{W_1}{RTT_1})(1-p_1)}{(\frac{W_2}{RTT_2})(1-p_2)} \simeq \frac{\frac{W_1}{RTT_1}}{\frac{W_2}{RTT_2}} \propto (\frac{RTT_2}{RTT_1})^{5/2} \quad since, \quad p \ll 1 \\ \Rightarrow \quad & \frac{BW_1}{BW_2} \propto (\frac{RTT_2}{RTT_1})^{5/2} \end{aligned} \tag{3.7}$$

Analysis done on similar lines for TCP gives RTT unfairness equation as,

$$\frac{BW_1}{BW_2} \propto (\frac{RTT_2}{RTT_1})^2 \tag{3.8}$$

Comparing Equation 3.7 and 3.8, we observe that the RTT unfairness of the LTCP is slightly worse than that of the TCP. In case of LTCP, a flow with the larger window size operates at a higher layer as compared to a flow with the smaller window. The larger flow increases its window at the rate corresponding to its operating layer, which is larger than the rate of window increase for the smaller flow. In case of TCP, both the flows increase window at the same rate. Difference in the rate of increase in congestion window coupled with difference in RTTs, make the RTT

unfairness of LTCP worse than that of TCP. We have verified this analysis through ns-2 simulations. Implementation details and results of the simulations will be given in the following chapters.

2. Case 2: Flows Competing at Same RTT

This analysis is also based on a synchronous loss model. In this section, the average time of convergence of two LTCP flows is analyzed and a relationship is established between the convergence time and the RTT. The behavior of congestion window for two LTCP flows, competing with each other is shown in Figure 2. The design of the protocol ensures that after a drop event, the smaller flow claims the lost bandwidth faster than the larger flow. Let W_{max} represent the combined window of the two flows at which packet drop occur and t be the time between two loss events. Following parameters are defined for the flow i,

$K_i^{'}$: Average layer number in which flow operates between two drop events,

$W_i^{'}$: Congestion window just before the first drop event,

$W_i^{''}$: Congestion window just before the second drop event

If we assume that the link bandwidth does not change then the sum of the windows of the two flows, just before the packet drop event, will remain same and equal to W_{max}. This can be represented as,

$$W_{max} = W_1^{'} + W_2^{'} = W_1^{''} + W_2^{''} \tag{3.9}$$

Between two loss events, the congestion window increases at an average rate of $K_i^{'}$ per RTT. Equation for $W_i^{''}$ can be written as,

$$W_1^{''} = (1 - \beta)W_1^{'} + K_1^{'}t/RTT \tag{3.10}$$

$$W_2^{''} = (1 - \beta)W_2^{'} + K_2^{'}t/RTT \tag{3.11}$$

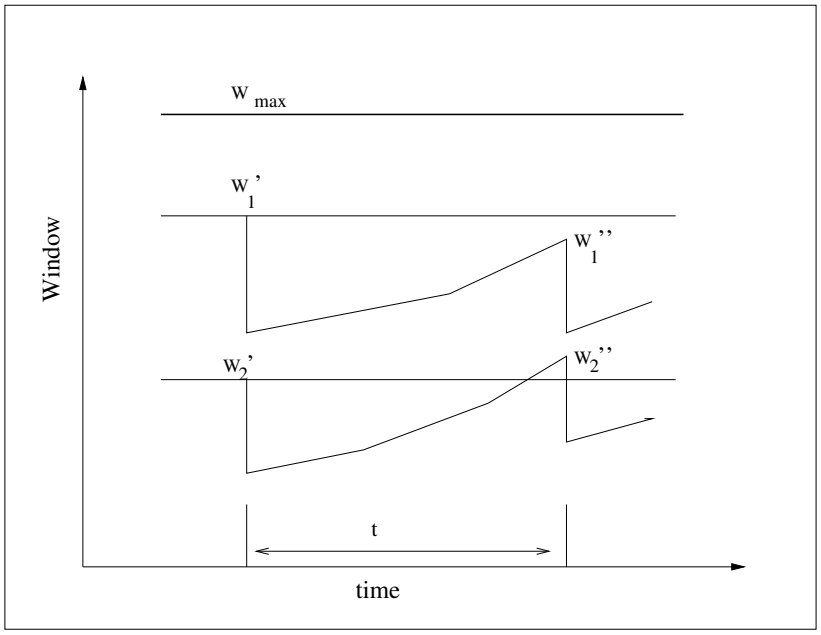


Fig. 2. Convergence of LTCP Flows

Adding the above equations and substituting from Equation 3.9, we get,

$$\begin{aligned} W_{max} &= W_1^{''} + W_2^{''} \\ &= (1-\beta)(W_1^{'} + W_2^{'}) + (K_1^{'} + K_2^{'})t/RTT \\ &= (1-\beta)W_{max} + (K_1^{'} + K_2^{'})t/RTT \end{aligned} \tag{3.12}$$

From this, the time between drop events as,

$$t = \frac{\beta W_{max}}{(K_1^{'} + K_2^{'})} RTT \tag{3.13}$$

Equation 3.13 reveals that the time between two loss events is directly proportional to RTT. The value of t can be used to calculate the congestion window $W_1^{''}$ as,

$$W_1^{''} = (1-\beta)W_1^{'} + K_1^{'}\frac{\beta W_{max}}{(K_1^{'} + K_2^{'})} \tag{3.14}$$

The congestion window after a packet drop is independent of the RTT. Therefore, increase in RTT will not affect the congestion window after the drop event. Hence, the total number of drop events will not change. But, an increase in RTT will result in a proportionate increase in the time between two loss events. Therefore, total time of convergence of two flows, which depends on the time between two drop events will also increase. This analysis is verified using simulations on ns-2 simulator and the results are presented in the following chapters.

Analysis presented above shows that the LTCP protocol suffers from performance deterioration when the RTT increases. These problems are direct consequence of the LTCP design and cannot be resolved without introducing modifications to the existing LTCP algorithm. In the following, we present LTCP-RC, a simple extension to the LTCP high-speed protocol, which makes LTCP more scalable in high RTT networks. LTCP-RC employs an RTT Compensation technique using a factor, K_R which is based on the RTT observed by the flow. This technique improves the convergence time, RTT unfairness properties while preserving the fairness properties of the original LTCP protocol. Following chapters provide implementation details, analysis and simulations results of the LTCP-RC scheme.

CHAPTER IV

LTCP-RC I: USING A FIXED RTT COMPENSATION TECHNIQUE

In this design, LTCP-RC employs an RTT Compensation factor K_R based on the RTT observed by the flow. The RTT Compensation technique modifies the congestion window update algorithm, on the receipt of an acknowledgment, for LTCP. In the design presented in this chapter, a flow always employs the same function of RTT for K_R. In the next chapter, we will present a technique that adapt to the changes in the network.

Our first goal in the design of LTCP-RC is to preserve the convergence and fairness properties of the basic LTCP scheme. Therefore, LTCP-RC should not alter the fairness and convergence equations for LTCP presented in Chapter II. Specifically, Equation 2.3 should hold. In order to ensure this, RTT Compensation use a scaling factor K_R to the basic LTCP window increase function. On a successful receipt of one window of acknowledgments, LTCP-RC will increase its congestion window by $K_R * K$ packets, instead of K packets. For this design, Equation 2.3 can be re-written as,

$$\frac{WR_1}{K_{R_1} * K_1'} > \frac{WR_2}{K_{R_2} * K_2'} \tag{4.1}$$

For two flows with same RTT, the value of K_R will be same and thus will cancel out in the above Equation. The above Equation will reduce to Equation 2.3 for LTCP and convergence will hold.

Our second aim of scalability requires that increase in the RTT should make the LTCP-RC more aggressive. Therefore, K_R should be directly proportional to the Round Trip Time. But, increased congestion in the network leads to higher queuing delay and larger round-trip time. If K_R is chosen on the basis of instantaneous RTT then the value of K_R will also increase. This will make the load on the network

buffers worse. To take care of this problem, propagation delay is used instead of instantaneous RTT to calculate the value of K_R. Since, propagation delay is difficult to measure in actual implementation, the minimum RTT observed by the flow is used as a measure for the propagation delay. The value of K_R is updated whenever the minimum RTT change.

For the sake of simplicity, we choose an RTT Compensation function of the form,

$$K_R = c(RTT)^{\alpha} \qquad (c < 1) \tag{4.2}$$

where, c and α are constants and RTT is measured in milliseconds. Next few sections examine the performance of the protocol using different mathematical analysis. Results obtained from the analysis and simulations, will be used to decide the values for c and α.

A. RTT Unfairness

The analysis presented in this section is similar to the analysis presented in Chapter III. The assumption of synchronous loss model is made. As explained above, the LTCP-RC flow operating at a layer K, will increase the congestion window at the rate of $K_R * K$ packets every RTT. The throughput equation for LTCP-RC, corresponding to Equation 3.1 can be derived as,

$$BW = \frac{\sqrt{\frac{K_R * K'}{\beta}(1 - \frac{\beta}{2})}}{RTT\sqrt{p}} \tag{4.3}$$

Using $BW = W/RTT$ we will get,

$$\begin{aligned} BW &= \frac{W_i}{RTT_i} = \frac{\sqrt{\frac{K_{R_i} K_i'}{\beta}(1 - \frac{\beta}{2})}}{RTT\sqrt{p}} \\ \Rightarrow p_i &= \frac{K_{R_i} * K_i' C}{W_i^2} \quad where \quad C = \frac{1}{\beta}(1 - \frac{\beta}{2}) \end{aligned} \tag{4.4}$$

Substituting the value of p_i in Equation 3.2 from Chapter II, the average window size between two loss events can be calculated as,

$$W_i = \frac{tK_{R_i}K_i'C}{RTT_i} \tag{4.5}$$

RTT unfairness can be calculated by dividing the above equation for two flows and using approximation that $p \ll 1$,

$$\frac{BW_1}{BW_2} = \frac{\frac{W_1}{RTT_1}}{\frac{W_2}{RTT_2}} = (\frac{RTT_2}{RTT_1})^2(\frac{K_1'}{K_2'})(\frac{K_{R_1}}{K_{R_2}}) \tag{4.6}$$

Substituting $W_i \propto (K_i')^3$ from Equation 3.5 we get,

$$\frac{BW_1}{BW_2} = \frac{\frac{W_1}{RTT_1}}{\frac{W_2}{RTT_2}} = (\frac{RTT_2}{RTT_1})^{\frac{5}{2}}(\frac{K_{R_1}}{K_{R_2}})^{\frac{3}{2}} \tag{4.7}$$

By substituting $K_R = c(RTT)^\alpha$ and simplifying we get,

$$\frac{BW_1}{BW_2} = (\frac{RTT_2}{RTT_1})^{(\frac{5}{2}-\frac{3\alpha}{2})} \tag{4.8}$$

The above equation shows that RTT unfairness of LTCP-RC depends on the value of α. For example, choosing $\alpha = 1/3$, the RTT unfairness of LTCP is given by,

$$\frac{BW_1}{BW_2} = (\frac{RTT_2}{RTT_1})^2 \tag{4.9}$$

This is similar to the RTT unfairness of the AIMD scheme used in TCP. Table I, presents the RTT unfairness of LTCP-RC protocol for different values of α.

Choosing $\alpha = 1$ makes the congestion windows of two LTCP-RC flows independent of RTT and gives window oriented fairness [9]. Similarly, $\alpha = 5/3$ results in rate-oriented fairness, when the effect of RTT is completely eliminated and the protocol behaves as a rate-oriented, scheme independent of RTT. Thus, by influencing the value of α, RTT unfairness of the LTCP-RC protocol can be controlled. This provides a mechanism to tune the protocol and desired performance can be achieved.

Table I. LTCP-RC I: RTT Unfairness at Different Values of α

α	RTT unfairness $(\frac{BW_1}{BW_2})$
1/3	$(\frac{RTT_2}{RTT_1})^2$
1/2	$(\frac{RTT_2}{RTT_1})^{7/4}$
1	$(\frac{RTT_2}{RTT_1})$
5/3	constant

In our current implementation, we have used the value of $\alpha = 1/3$ to make the RTT unfairness of LTCP-RC same as that of TCP. The RTT Compensation function is then represented as, $K_R = c(RTT)^{1/3}$.

B. Convergence Time for Two Flows

In this section, we analyze the convergence time for two LTCP-RC flows when the second flow starts after the first flow has reached a steady state. Both the flows observe the same RTT. The analysis again is based on the assumption of synchronous loss model. Between two loss events, congestion window will increase at an average rate of $K_R * K_i^{'}$ per RTT. Two flows, experiencing same RTT, will have the same value for K_R. Using $K_i^{'}, W_i^{'}$ and $W_i^{''}$, as defined in Chapter III Section 2, we can re-write equation for the increase in congestion window with time as,

$$W_1^{''} = (1-\beta)W_1^{'} + K_R * K_1^{'}t/RTT \tag{4.10}$$

$$W_2^{''} = (1-\beta)W_2^{'} + K_R * K_2^{'}t/RTT \tag{4.11}$$

Assuming that W_{max} does not change, we can add the above two equations and

substitute for W_{max} from Equation 3.9 to get,

$$\begin{aligned} W_{max} &= W_1^{''} + W_2^{''} \\ &= (1-\beta)(W_1^{'} + W_2^{'}) + \frac{K_R(K_1^{'} + K_2^{'})t}{RTT} \\ &= (1-\beta)W_{max} + \frac{K_R(K_1^{'} + K_2^{'})t}{RTT} \end{aligned} \tag{4.12}$$

The time between two loss events is re-calculated as,

$$t = \frac{\beta W_{max}}{K_R(K_1^{'} + K_2^{'})} RTT \tag{4.13}$$

From the above equation, it is clear that for two LTCP-RC flows with same RTT (and hence same K_R), the time between two loss events is reduced by a factor of K_R. If we calculate the congestion window at the second loss event by substituting the value of t in equation for $W_i^{''}$ and simplifying we get,

$$\begin{aligned} W_1^{''} &= (1-\beta)W_1^{'} + K_R * K_1^{'} t/RTT \\ &= (1-\beta)W_1 + \frac{K_R K_1^{'} \beta W_{max}}{K_R(K_1^{'} + K_2^{'})} \qquad (4.14) \\ &= (1-\beta)W_1 + \frac{K_1^{'} \beta W_{max}}{(K_1^{'} + K_2^{'})} \qquad (4.15) \end{aligned}$$

The above equation shows that the window $W_1^{''}$ at next loss event is independent of K_R. RTT Compensation does not affect the windows achieved by the flows between two loss events. Thus for LTCP-RC, the number of loss events required convergence will remain same as LTCP. But, the time between loss events will be reduced by a factor of K_R. Therefore, for LTCP-RC, the overall convergence time will be reduced by a factor of K_R as compared to LTCP. RTT Compensation makes the protocol more aggressive by increasing the rate at which the flow increases its congestion window.

C. Effect of Random Drops

The assumption with synchronized losses might not hold in all network scenarios. Real networks are characterized by random losses and channel errors especially in the case of wireless links. In this section, we analyze the behavior of LTCP-RC assuming a random loss model. The analysis is based on similar lines presented in [9] and [12]. LTCP-RC, update the congestion window w on a successful packet delivery or reduce it on observing a packet loss. Let *A(w, RTT)* and *B(w, RT)* represent the congestion window response functions on a successful delivery of packet and observation of a packet loss, respectively. Here, w represents the size of congestion window and RTT is the round trip time. For LTCP-RC, these functions are given by,

$$\begin{aligned} A(w, RTT) &= \frac{K_R * K}{w} \sim \frac{K_R}{w^{1/3}} \\ B(w, RTT) &= \beta w \end{aligned} \tag{4.16}$$

The above Equation is derived using approximation, $K \propto w^{1/3}$. If p denotes the probability of packet loss then, the expected change of congestion window, Δw is given by,

$$\begin{aligned} E\{\Delta w\} &= p \cdot E\{\Delta w| \text{ packet loss}\} + (1-p) \cdot E\{\Delta w| \text{ successful transmission}\} \\ &= -p \cdot B(w, RTT) + (1-p) \cdot A(w, RTT) \end{aligned} \tag{4.17}$$

Let W_s represents the average window at statistical equilibrium. At the equilibrium point, expected change in window will be zero or $E\{\Delta w\} = 0$. By substituting this in the above equation and using Equation 4.16, we get,

$$\begin{aligned} p &= \frac{A(W_s, RTT)}{A(W_s, RTT) + B(W_s, RTT)} \\ &= \frac{1}{1 + \frac{\beta W_s^{5/3}}{K_R}} \end{aligned}$$

$$= \frac{1}{1 + \frac{\beta W_s^{5/3}}{c(RTT)^{1/3}}} \tag{4.18}$$

The terms in the above equation can be re-arranged to calculate the value of W_s as,

$$\frac{\beta W_s^{5/3}}{c(RTT)^{1/3}} = \frac{1-p}{p} \approx \frac{1}{p} \tag{4.19}$$

$$\Rightarrow W_s \propto \frac{RTT^{0.2}}{p^{0.6}} \tag{4.20}$$

The above Equation is similar to the throughput equation obtained for LTCP-RC, confirming the validity of our analysis.

Equation 4.18 can be used to establish fairness statement for two LTCP-RC flows, in the presence of random losses and channel errors. When two flows compete on the same bottleneck link then both the flows experience the same loss probability. Equating p_i, from Equation 4.18 for two flows we get,

$$\begin{aligned} \frac{\beta W_{s_1}^{5/3}}{c(RTT_1)^{1/3}} &= \frac{\beta W_{s_2}^{5/3}}{c(RTT_2)^{1/3}} \\ \Rightarrow \frac{W_{s_1}}{W_{s_2}} &= (\frac{RTT_1}{RTT_2})^{0.2} \\ \Rightarrow \frac{BW_{s_1}}{BW_{s_2}} &= (\frac{RTT_2}{RTT_1})^{0.8} \end{aligned} \tag{4.21}$$

The above equation shows that the throughput obtained by LTCP-RC at equilibrium point is inversely proportional to $RTT^{0.8}$. An analysis on similar lines for TCP gives the ratio of throughput at equilibrium point as,

$$\frac{BW_{s_1}}{BW_{s_2}} = \frac{RTT_2}{RTT_1} \tag{4.22}$$

TCP shows window oriented fairness and the throughput obtained by the TCP flow is proportional to the inverse of RTT. The throughput obtained by the LTCP-RC flow will be more than the throughput obtained by the TCP flow. LTCP-RC flows

will also be more fair to each other as compared to TCP flows for the same ratio of RTT. Hence, LTCP-RC will perform better than TCP in the presence of random losses in the network. The analysis presented will be verified by simulations using channel error rates in later sections.

D. Effect of Large Queuing Delay

In this section, we analyze the deterioration in the performance of LTCP-RC when the buffering delay becomes large. We study the effect of very large buffer size on the amount of data lost by the protocol. The analysis is done on the similar lines as presented in [13]. When buffer sizes become large, then the propagation delay could become negligible compared to the queuing delay. Schemes similar to TCP use packet loss as an indication of congestion. They continue to increase the congestion window even if the combined sending rate of all the flows on the bottleneck link exceeds the link capacity. For large buffers (hence large feedback delays), the end-users will eventually overshoot the bottleneck link and experience bursty packet losses. This is due to the excess data sent into the network before congestion is detected. In this analysis, we analyze the relationship between the amount of lost data during each overshoot and the buffering delay.

LTCP-RC increases its window size by $K_R * K/W(t-1)$ for each incoming acknowledgment, where *W(t-1)* represents the congestion window at time *(t-1)*. If C represents the bottleneck link capacity then after the link is saturated, acknowledgments from the receiver arrive at the rate of C pkts/sec. This is the rate at which the bottleneck link will transfer the data to the receiver. The increase in the congestion window at the sender can be written as,

$$\frac{dW}{dt} = \frac{C(K_R * K)}{W}$$

$$\begin{aligned} &= \frac{CK_R}{W^{2/3}} \text{ , since , } K \propto W^{1/3} \\ \Rightarrow W(t) &= (\frac{5}{3}CK_R t + \delta)^{3/5} \end{aligned} \tag{4.23}$$

where δ is the integration constant. The equation above gives the evolution of the congestion window for LTCP-RC with respect to time. Assuming that the bottleneck link starts overflowing at time, *t = 0*, the amount of extra packets injected in the network per acknowledgment are given by $\frac{K_R * K}{W(t)}$ or $\frac{K_R}{W(t)^{2/3}}$. Therefore, the amount of extra packets *S(t)* sent into the link during time *[0, t]* can be modeled as,

$$S(t) = S(t-1) + \frac{K_R}{W(t-1)^{2/3}} \tag{4.24}$$

Since, acknowledgments arrive at the rate of C pkts/sec, we can write the differential equations as

$$\begin{aligned} \frac{dS(t)}{dt} &= \frac{CK_R}{W(t)^{2/3}} \\ &= \frac{CK_R}{(\frac{5}{3}CK_R t)^{2/5}} \end{aligned} \tag{4.25}$$

By integrating t from [0, D], where D is the total buffering delay, we can simplify to get,

$$S(D) \sim \frac{5}{3}(CK_R D + \delta)^{3/5} \tag{4.26}$$

In the above equation, since K_R depends only on minimum RTT (i.e. only on propagation delay), it is taken as constant during integration. For LTCP, the amount of lost data $S(D) \propto D^{0.6}$. For TCP, $S(D) \propto D^{0.5}$ and for Rate-based AIMD $S(D) \propto D$ [13]. Thus, LTCP-RC results in slightly higher losses than TCP, but still performs better than Rate-based schemes. Slightly higher losses in LTCP-RC can be attributed to the aggressive nature of the LTCP scheme which is essential for scaling the protocol at high-speed and high RTT networks.

E. Simulation Results

The LTCP-RC design is evaluated using experiments conducted on *ns-2* network simulator [14]. All simulations are conducted using a dumb-bell network topology as shown in Figure 3. One common bottleneck link connects n sources to n corresponding receivers. Unless otherwise specified, the bottleneck link capacity is set to 1Gbps with a delay of 40ms. Links that connect senders and receivers to the routers are set to a bandwidth of 2.4Gbps and a delay of 10ms. Thus, end-to-end RTT for each flow is set to 120ms, unless specified. The default queue size at the routers is set to be equal to the product of bottleneck link bandwidth and delay. Drop-tail queue management scheme is used at the routers. The protocol is implemented by introducing a new *window option* in the basic TCP code in the file *tcp.cc* in ns-2. All the simulations use *TCP/Sack1* agent for the sender and *TCPSink/Sack1* agent for the receiver. Unmodified *TCP/Sack1* is used for the TCP simulations. FTP traffic is used between the senders and receivers. All the readings are taken for 1000 seconds and data for initial 300 seconds is discarded, to ensure that steady state is reached.

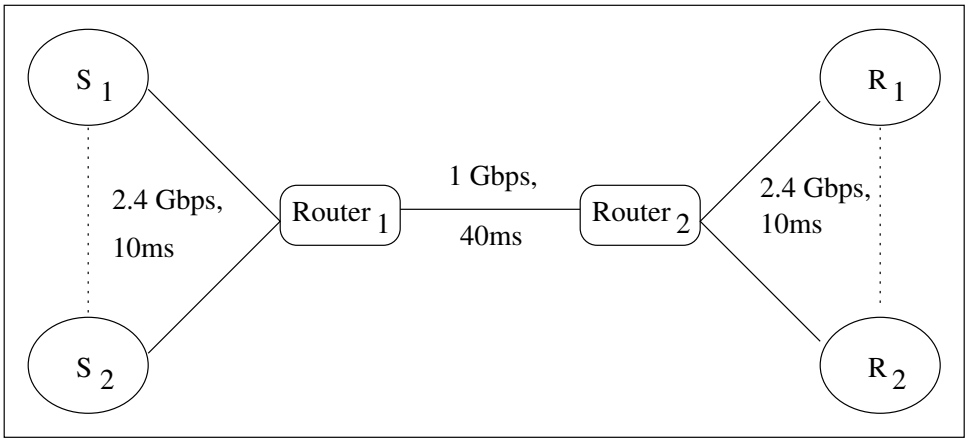


Fig. 3. LTCP-RC I: Simulation Topology

For comparison purposes, simulations for H-TCP and BIC high-speed protocols are also conducted using the ns-2 patches and example scripts available on authors websites [15], [16]. For LTCP-RC, the parameter β is set to 0.15 and W_T is set to 50 packets. For H-TCP flows, window option was set to *-10*, to simulate the complete H-TCP algorithm. For BIC flows, the parameter β is set to 0.875, S_{max} is set to 32, S_{min} is set to 0.01, window option is set to 12, low_window is set to 14, log_factor is set to 2 and fast convergence is turned on. Simulations for BIC use *TCP/SackTS* agent for sender.

1. Effect of Parameter c

In our implementation of LTCP-RC, the function for the factor K_R, is given by $c(RTT)^{1/3}$, where RTT is in milliseconds. Increase in the value of c will make the value of K_R larger and this will make the protocol more aggressive. As shown in the analysis for convergence time of two flows, larger value of K_R will reduce the convergence time but it might result in higher packet losses. We study the effect of parameter c on the convergence time and the packet drop rate in this experiment. The simulation consists of two flows competing on the same bottleneck link and observing same RTT. The second flow was started 300 seconds after the start of the first flow. This allows the first flow to grab the available bandwidth and reach a steady state. Following regions are defined as shown in Figure 4. Region 1 is defined from 100-299 seconds when the first flow is operating in a steady state. Region 2 is defined from 800-999 seconds when both the flows have converged and reached a steady state value.

Link drop-rates are measured in both Region 1 and Region 2. To measure the convergence time, the maximum congestion window attained by the first flow in Region 1 is measured. The maximum of the time taken (after the second flow is

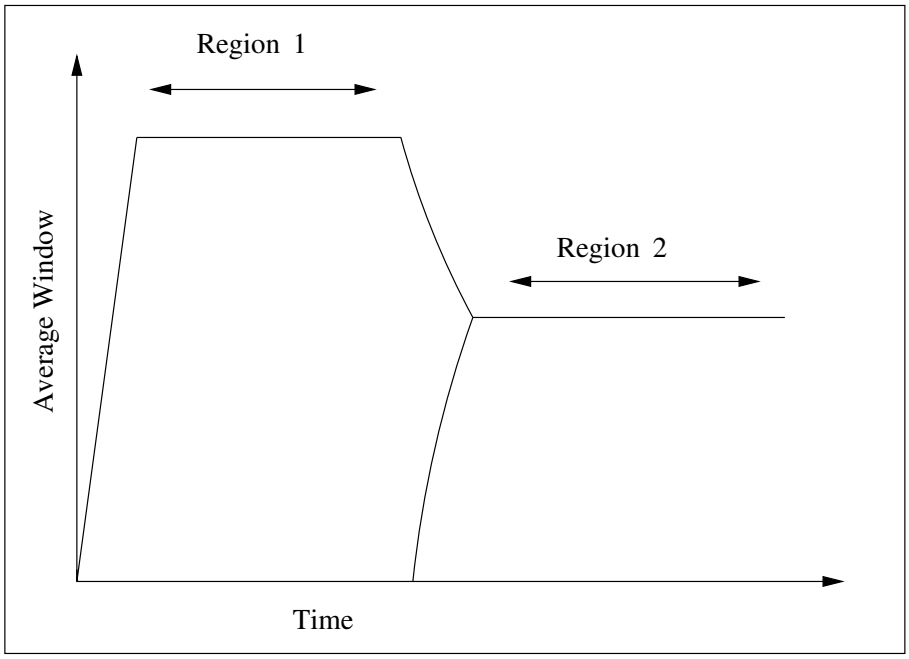


Fig. 4. LTCP-RC I: Definition of Region 1 and Region 2

started), by the first flow to reduce its congestion window to 55% of the maximum value or time taken by the second flow to increase by 45% of the maximum value, is calculated. This is termed as the time for 45-55% convergence. Table II, shows the droprates in Region 1 and 2 and the time of 45-55% convergence with varying values of c. It also presents the result of simulation with basic LTCP, BIC and H-TCP high speed protocols.

Table II, shows that increase in the value of c result in an increase in the drop rates in both Region 1 and Region 2. Results also reveal corresponding reduction in the convergence time. The choice of parameter c decides the operating point of the protocol. The results show that the basic LTCP protocol takes a long time to converge and the RTT Compensation technique has resulted in a significant reduction in convergence time. H-TCP protocol scales its 'window increase' by the round-trip

Table II. LTCP-RC I: Comparison of Drop Rates and Convergence Time

	Drop Rates (%)		Time for 45-55%
	Region 1	Region2	Convergence (seconds)
c = 0.4	0.00127	0.00325	202.2
c = 0.5	0.00193	0.00513	173.6
c = 0.6	0.00284	0.00723	154.1
c = 0.7	0.00365	0.00998	139.8
c = 0.8	0.00487	0.01286	123.2
LTCP	0.00035	0.00089	415.3
BIC	0.00147	0.00606	151.3
H-TCP	0.00620	0.01152	33.6

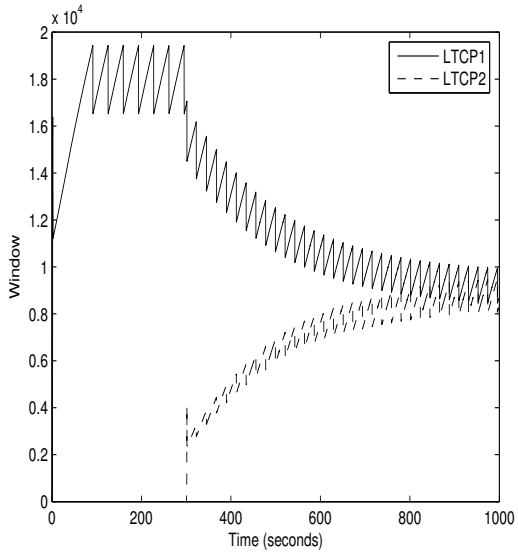
time to achieve the convergence time which is independent of RTT. But this scaling makes the H-TCP protocol very aggressive and leads to high drop rates.

In our current implementation, we have used $c = 0.5$ because it offers a good trade-off between convergence time and drop rate. Remaining simulations in this chapter use, $K_R = 0.5(RTT)^{1/3}$ (unless specified), where RTT is in milliseconds.

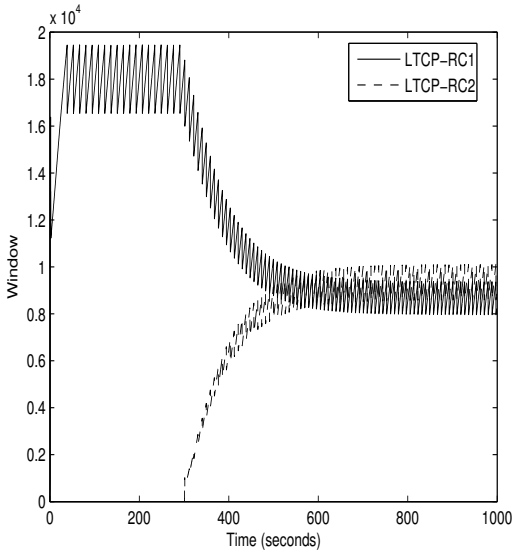
2. Fairness Among Multiple Flows

In this simulation, co-existence of flows of same protocol with each other is studied. As a first part of the experiment, the results from previous experiment is studied in more detail. Figure 5 plots the graph of the congestion window of two flows with respect to time for different protocols.

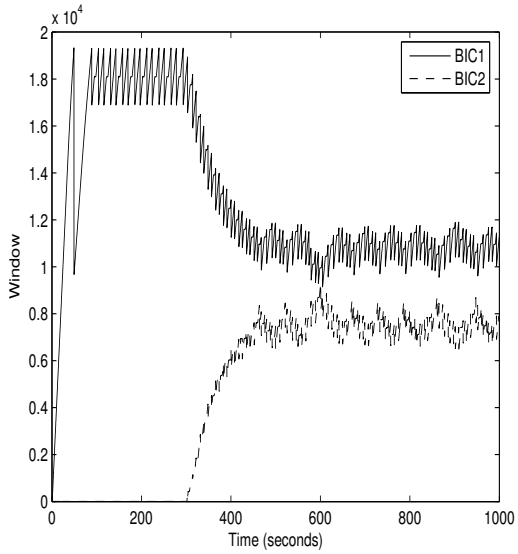
Although 45-55% convergence time of BIC protocol is low, but it exhibits overall convergence problems. Plot of the congestion window for BIC protocol reveals diver-



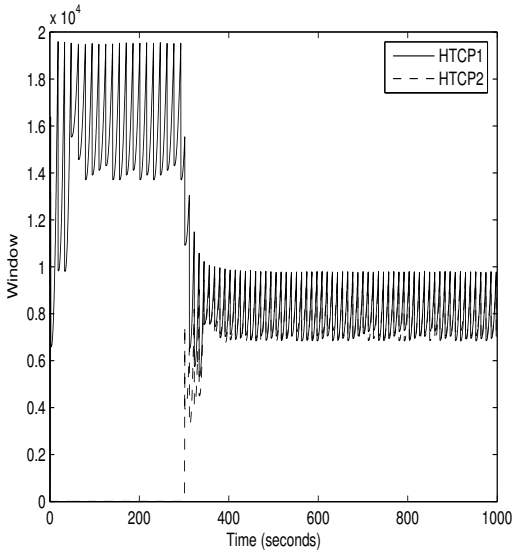
(a) LTCP



(b) LTCP-RC (with c=0.5)



(c) BIC



(d) H-TCP

Fig. 5. LTCP-RC I: Convergence of Different Protocols

gence between the congestion window. LTCP on the other hand, requires long time to convergence. As H-TCP protocol is designed for constant convergence time, it convergence very quickly. LTCP-RC clearly takes smaller convergence time as compared to LTCP and maintains fair share across different flows.

Table III. LTCP-RC I: Fairness Among LTCP-RC Flows

No. of Flows	Avg. per-flow Throughput (Mbps)	Min. per-flow Throughput (Mbps)	Max. per-flow Throughput (Mbps)	Standard Deviation	Jain's Fairness Index
2	480.77	480.34	481.20	0.60	1.00
4	240.39	240.28	240.55	0.12	1.00
6	160.26	160.14	160.37	0.10	1.00
8	120.19	120.16	120.31	0.05	1.00
10	96.15	95.75	99.55	1.19	0.99

Experiments are also conducted to measure the fairness of LTCP-RC flows with each other. Varying number of flows are started at the same time and average per-flow bandwidth of each flow is measured. The fairness index proposed by Jain et. al. in [17] is used as the measure of fairness. Table III presents the maximum, minimum and average per-flow throughput, standard deviation and Jain's Fairness Index for varying number of Flows. Results reveal that even if number of flows become large, the maximum and minimum throughput remains close to the average value. The fairness index also remain close to 1. Therefore, it can be inferred that LTCP-RC protocol maintains fairness among different flows and share the available network bandwidth equitably.

3. RTT Unfairness

In this experiment, the RTT unfairness of LTCP-RC protocol is studied. Two flows with different round trip times are started at the same time on a common bottleneck link. RTT unfairness is measured as the ratio of throughput obtained by each flow. The ratio of RTTs for two flows is varied from 2 to 4 for different runs of the experiment. The RTT of the shorter-delay flow is kept at 40ms. The bottleneck link has a bandwidth of 1Gbps and delay of 10ms.

Table IV. LTCP-RC I: RTT Unfairness

RTT Ratio	TCP	LTCP	LTCP-RC	BIC	H-TCP
2	3.14	3.84	3.35	9.73	1.85
3	7.35	12.99	7.63	16.63	2.78
4	14.78	28.61	13.65	26.88	3.68

Table IV, shows that the RTT unfairness of LTCP-RC is less than that of LTCP protocol. Due to the use of $K_R = c(RTT)^{1/3}$ function for RTT Compensation, RTT unfairness of LTCP-RC is almost similar to that of TCP. BIC has RTT unfairness worse than both TCP and LTCP-RC. The values of RTT unfairness for H-TCP is same as the ratio of RTT. This is due the use of scaling factor, proportional to RTT, by H-TCP protocol.

From the earlier analysis, we expected the ratio of throughput for two LTCP-RC flows proportional to RTT^2. The values obtained from simulations is little less than the value expected from analysis. This is due to the approximations for $W \propto K^{1/3}$ and use of propagation delay instead of instantaneous RTT. Moreover, measurement of minimum RTT might include queuing delay and which will change the ratio of

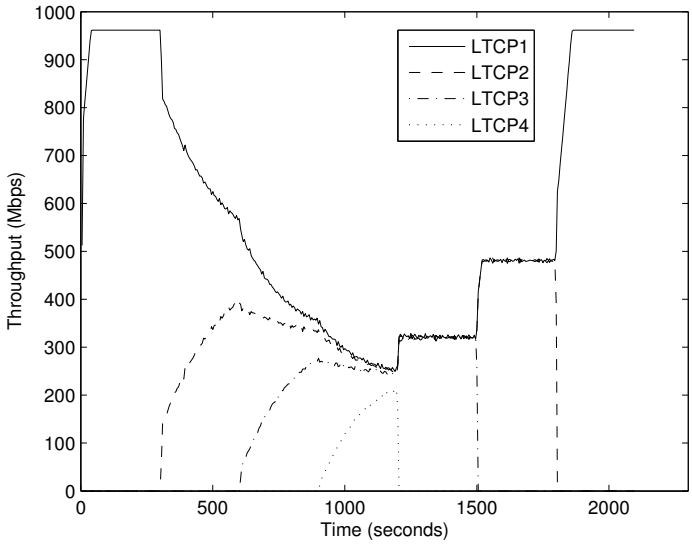
RTTs.

4. Dynamic Link Sharing

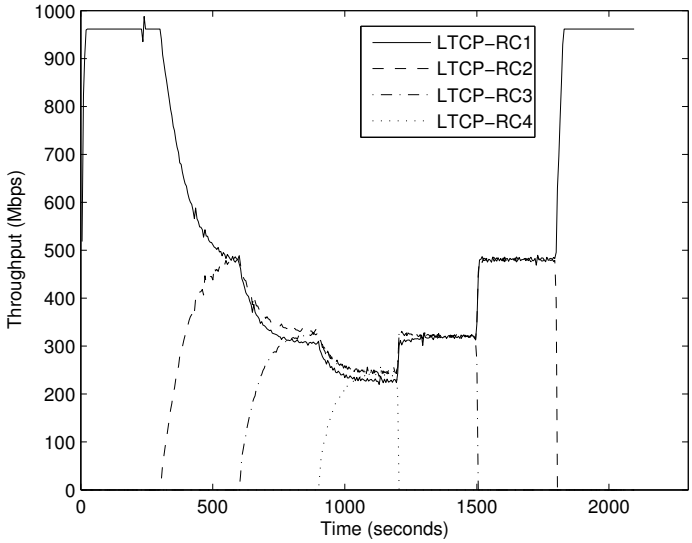
In this experiment, we evaluate the response of LTCP-RC to the changes in the network conditions due to the arrival of new flows and the departure of old flows. We study the performance improvement of LTCP-RC as compared to LTCP protocol. Four flows are started at regular intervals of 300 seconds. First flow is active from t=0 to t=2100 second, the second flow is active from t=300 to t=1800 seconds, the third flow from t=600 to t=1500 second and the fourth flow is active from t=900 to t=1200 seconds. Figure 6 shows the plot of the throughput obtained by each flow with respect to time for LTCP and LTCP-RC protocols. The figure reveal that in both the cases when a new flow is started, the existing flow gave up the bandwidth until all the flows reach the fair utilization level. When an existing flow stops sending data, the remaining flows ramp up and reach the new fair share level, utilizing the available link bandwidth. But, LTCP-RC converges much faster than the basic LTCP scheme. LTCP flow takes a long time to converge to the equilibrium steady state rate. This can be specially observed when flow 2 and flow 4 enters the network. In the plot for LTCP older flows did not converge to the fair share even after 300 seconds. Whereas in the case of LTCP-RC, the flows quickly reach to the common equilibrium rate.

5. Effect of Random Drops

The bandwidth equation for TCP has shown that TCP requires extremely low packet loss rates in order to maintain high link utilization. In this experiment, we compare the performance of LTCP with other high-speed protocols in the presence of channel errors. The simulation is conducted using a single FTP transfer over a bottleneck



(a) LTCP



(b) LTCP-RC

Fig. 6. LTCP-RC I: Dynamic Link Sharing

link of 1Gbps and RTT of 120ms. Random drops are introduced in the bottleneck link using an uniform error model. We measure the average throughput obtained by the flow, during the steady state. Figure 7 shows the plot of throughput obtained by different protocols at varying channel error rates. The packet loss rates in the graph do not account for the congestion losses (which might occur when the sender rate exceeds the bottleneck capacity). The data in the figure include only channel error rates at the bottleneck link, as specified in the simulation error model.

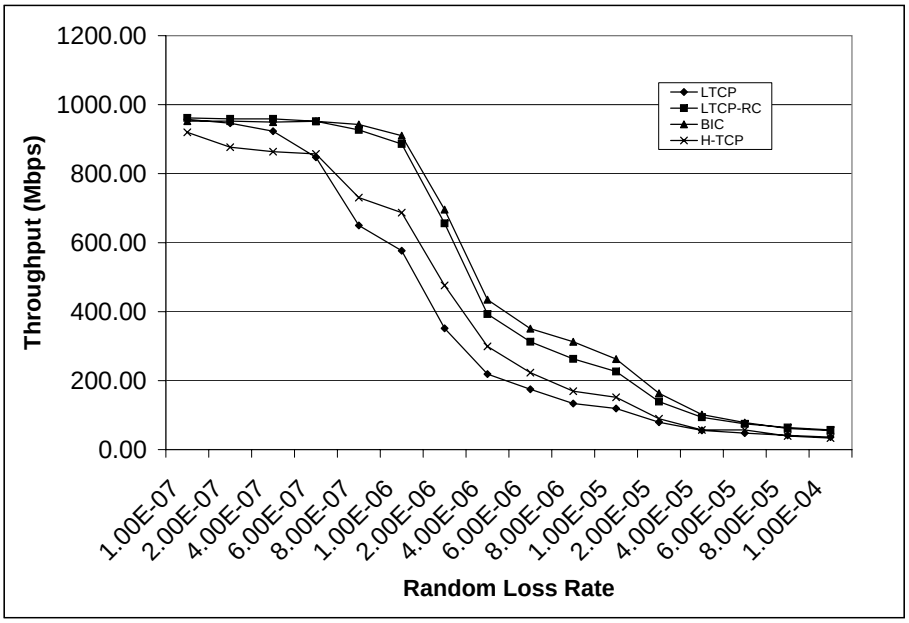


Fig. 7. LTCP-RC I: Throughput vs Error Rates

The graph reveals that all the protocols maintain high utilization at very low loss rates. But increase in the channel errors deteriorates the utilization. The utilization obtained by LTCP-RC is better than LTCP and H-TCP protocols. The utilization

Table V. LTCP-RC I: Effect of Channel Errors

Packet Loss Rate (%)	LTCP Throughput (Mbps)	LTCP-RC Throughput (Mbps)	BIC Throughput (Mbps)	H-TCP Throughput (Mbps)
1.00E-07	957.31	961.52	952.38	919.61
1.00E-06	576.40	886.18	909.96	687.05
1.00E-05	119.43	226.39	262.37	151.49
1.00E-04	36.01	57.01	55.25	33.49

of BIC is close to that of LTCP-RC. Table V, shows the sample data for the average throughput obtained by different protocols.

F. Conclusion

Our analysis and simulation results have revealed that the LTCP scheme suffers from performance problems at higher RTTs. RTT Compensation technique presented for LTCP-RC improves the performance of basic LTCP scheme considerably. Mathematical analysis and results obtained from simulation have shown improvement in terms of convergence time and RTT unfairness. But, the RTT Compensation technique employed in LTCP-RC makes the protocol more aggressive, resulting in higher drop rates. It offers a trade-off between the required convergence time and tolerable drop rates. In the next chapter, an adaptive scheme is proposed which will regulate itself according to the dynamics of the network.

CHAPTER V

LTCP-RC II: USING AN ADAPTIVE RTT COMPENSATION TECHNIQUE

In the previous chapter, we presented a RTT Compensation Technique which uses a fixed function for the scaling factor K_R. The fixed function make the LTCP-RC protocol more aggressive, resulting in higher loss rates but low convergence time. In this new design, we aimed at improving the LTCP-RC design further. In the new design, two modes of operations for LTCP-RC are defined. In the *steady state*, the protocol need not be aggressive. Therefore, we set $K_R = 1$ or turn it *off*. This will reduce the drop rates in steady state. In the *transient state*, a flow is probing for the available bandwidth and needs to be more aggressive. Therefore, we turn *on* K_R. This design is named as LTCP-RC$_{on-off}$. The scheme can be represented as,

$$K_R = \begin{cases} 1, & \text{during steady state } (off) \\ 0.5(RTT)^{1/3}, & \text{during transient state } (on) \end{cases} \tag{5.1}$$

To decide about the state of the protocol, the layering scheme, inherent to the design of LTCP is used. The decision about the steady state or the transient state is made on the basis of the layer at which a flow is operating. We measure and record the layers, at which last three packet loss events occurred. A steady state is assumed, if last three loss events occur in the same layer. In this case, K_R is turned *off*. When the congestion window is increased and the size of the current window becomes larger than the layer boundary (at which last drop occurred), then it is assumed that bandwidth is available. In this case, K_R is turned *on*.

Further, in order to achieve faster convergence, when two flows are competing over a bottleneck link, ideally, K_R should be turned *off* for the flow with larger window size and *on* for the flow with smaller window. This makes the smaller flow

more aggressive. After a packet drop, the smaller flow will regain the lost bandwidth faster than the flow with larger window and will lead to faster convergence. Therefore, If layer at which loss event occurs is smaller than the layer at which the last drop occurred, then it shows that the window is in a decreasing trend and is giving up bandwidth. Hence, K_R is turned off in this case.

A. Implementation Details

Below, we present the pseudo-code of the LTCP-RC$_{on-off}$ design. A packet drop event is characterized by the receipt of triple-duplicate acknowledgments from the receiver. The following parameters are used,

$current\_layer$	:	layer at which packet drop event occurred.
$last\_layer$	:	layer at which last packet drop event occurred.
$second\_last\_layer$	:	layer at which second last packet drop event occurred.
K	:	current operating layer.
W_K	:	window corresponding to the layer, K
$stored\_K_R$	:	stored value of K_R which is calculated at the start of the flow and updated whenever minimum RTT changes.

Initialization:

$second\_last\_layer = last\_layer = current\_layer = 1;$
$stored\_K_R = 0.5(RTT)^{1/3};$

On receiving 3 duplicate acknowledgments, decrease congestion window:

$second\_last\_layer = last\_layer;$
$last\_layer = current\_layer;$
$current\_layer = K\ ;$

if *(second_last_layer* $\geq$ *last_layer && last_layer* $\geq$ *current_layer)* ***then***
  $K_R = 1$;
else
  $K_R = stored\_K_R$;
  $cwnd\_ = (1 - \beta) * cwnd\_$;
  while $cwnd\_ < W_K$ ***do***
    $K = K - 1$;
  end while
end if

On receipt of an acknowledgments, increase congestion window:

$cwnd\_ = cwnd\_ + (K_R * K)/cwnd\_$;
while $cwnd\_ > W_{K+1}$ ***do***
  //window crosses the current layer boundary. Increase number of layers
  $K++$;
  if $K >$ *current_layer* ***then***
    // layer crosses the layer at which last drop occurred.
    $K_R = stored\_K_R$;
  end if
end while

B. An Alternate Design Choice

The LTCP-RC$_{on-off}$ design can create problem of overshoot when two flows operate in the same layer and each has it's K_R turned *off*. The flow which crosses the layer boundary faster will turn *on* its K_R, while the other flow still has its K_R turned *off*. The first flow will start increasing its congestion window at the rate of $K_R * K$ packets/RTT as compared to the other flow, which will increase its congestion window at the rate of K packets/RTT. For example, let's consider two flows operating near boundary of layer 10 at the RTT of 120ms. Then, the flow with K_R turned off will increase the congestion window at the rate of 10 packets per/RTT. On the other hand, the flow with K_R turned on will increase congestion window at the rate of 24 packets/RTT ($K_R * K = 0.5(120)^{1/3} * 10$). Due to the large difference in the rate of congestion window increase, it might lead to short term unfairness between the two flows.

To reduce the difference in the rate of congestion window increase for the two flows, we propose an alternate design named as LTCP-RC$_{full-half}$. In this algorithm, instead of turning the K_R off completely, we reduce it to a lower value at the steady state. We use the same function, $c(RTT)^{1/3}$ for K_R but modify the value of c, whenever K_R needs to be changed . In steady state, $c = 0.3$ is used while in transient state, $c = 0.5$ is employed. The rationale behind choosing these values is to make the value of K_R greater than 1, for RTT larger than 40ms and reduce the difference between the value of K_R in steady and transient state. The algorithm can be represented as,

$$K_R = \begin{cases} 0.3(RTT)^{1/3}, & \text{during steady state } (\textit{half}) \\ 0.5(RTT)^{1/3}, & \text{during transient state } (\textit{full}) \end{cases} \tag{5.2}$$

C. Stability Analysis

Due to dynamic network conditions, the value of K_R might change in the adaptive RTT Compensation techniques. In this section, we analyze the affect of change in the value of K_R on the convergence of two flows. Both the flows are operating at same RTT and will have same value for the function $c(RTT)^{1/3}$. The exact mathematical analysis becomes complicated due to the discrete nature of the layers in the protocol. Here, a simplified intuitive assessment for the convergence of two flows is presented. It is assumed, that the flow 1 is operating at a higher window as compared to flow 2. Each flow can have it's K_R either turned on or off. Therefore, there are four possible states for the two flows, as shown in Figure 8.

In states A and D, both the flows have the same value of K_R. The convergence analysis presented in Chapter IV Section B, will hold true and the two flows will converge. In the state C, K_R is turned off for the higher flow and turned on for the smaller flow. After a packet drop, the convergence equation of LTCP-RC can be

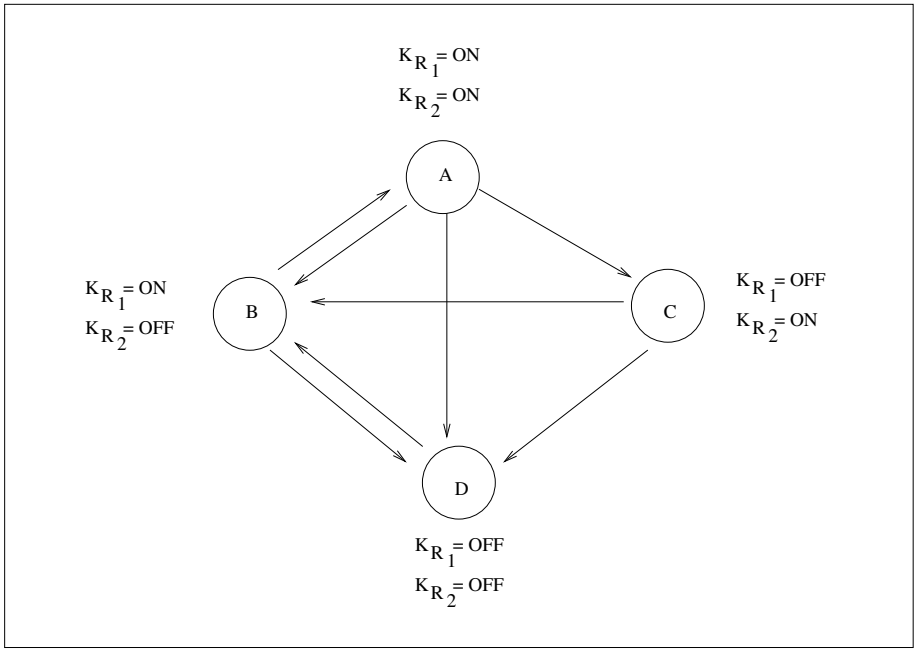


Fig. 8. LTCP-RC II: State Diagram for Convergence of Two Flows

re-written as,

$$\begin{aligned} \frac{WR_1}{K_R * K_1} &> \frac{WR_2}{K_R * K_2} \\ \Rightarrow \frac{WR_1}{K_1} &> \frac{WR_2}{K_{R_2} K_2} \end{aligned} \tag{5.3}$$

By LTCP-RC design, $WR_1/K_1 > WR_2/K_2$ and $K_R > 1$. Therefore, the above equation holds true and the two flows will convergence in state C also. Hence, states A, C and D represent the stable states, where the two flows will convergence. Figure 8 also shows the possible transition between different states. Only, the state B is an unstable state, where higher flow turns on it's K_R, while the smaller flow turns off the K_R. In this case, the time taken by the higher flow to regain the lost bandwidth, on a packet drop, might be smaller then the time taken by the smaller flow. This may lead to divergence between the two flows. But, since the network bandwidth is finite,

the higher flow cannot increase its congestion window infinitely. Also by design of the protocol, with increasing window, the size of the layer also increases. Due to the effect of either the finite maximum window or large size of the operating layer, the higher flow will eventually observe three consecutive drops in same layer. It will then transition to either state A or D, which are stable states. This shows that the state B is not permanent. In any of the remaining states, it is not possible for a flow to operate at a higher window, when a competing flow is operating at a smaller window.

Thus, two flows may experience temporary divergence, when they enter state B, but then they will start converging back again, when they return back to either of the remaining state.

The institutive explanation presented in this section shows that although we might have short term unfairness and oscillations, but we will observe overall convergence behavior.

D. Simulation Results

The algorithm for adaptive LTCP-RC (*on-off* and *full-half*) is implemented in ns-2 simulator by modifying the source code for TCP agent in files, *tcp.cc* and *tcp-sack1.cc*. Simulation topology for the experiments presented in this section, is same as shown in Figure 3. The bottleneck bandwidth is set to 1Gbps and a RTT of 120ms, unless otherwise specified. Experiments are conducted to evaluate both the designs namely, LTCP-RC$_{on-off}$ and LTCP-RC$_{full-half}$. In rest of the chapter, we will use LTCP-RC to represent the scheme from Chapter IV, with fixed function for K_R.

1. Convergence Time and Drop Rates

Our initial motivation for adaptive LTCP-RC scheme is to reduce the drop rates while keeping low convergence time. In this experiment, we study the convergence time and drop rates for two flows competing over the same bottleneck link. The second flow is started 300 seconds after the start of the first flow, so that the first flow grabs the available link bandwidth and reach a steady state. Region 1 and Region 2 used are same as those defined in Chapter IV Section 1. Convergence time is measured as the time for 45-55% convergence. Table VI, presents the results obtained from the experiment.

Table VI. LTCP-RC II: Comparison of Drop Rates and Convergence Time

	Drop Rates (%)		Time for 45-55%
	Region 1	Region2	Convergence (seconds)
LTCP	0.00035	0.00089	415.3
LTCP-RC$_{on-off}$	0.00035	0.00081	109.6
LTCP-RC$_{full-half}$	0.00075	0.00191	109.6
LTCP-RC	0.00193	0.00513	173.6
BIC	0.00147	0.00606	151.3
H-TCP	0.00620	0.01152	33.6

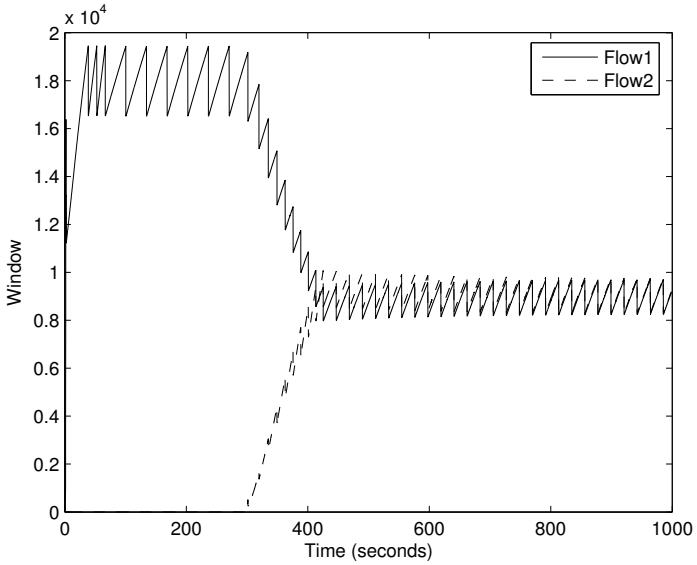
Results presented in Table VI, shows that drop rates for LTCP-RC$_{on-off}$ is similar to the basic LTCP scheme in both Region 1 and Region 2. In both these regions, the two flows reach a steady state and turn *off* their K_R. Thus, both of them behave exactly like LTCP flow and show same drop rates. On the other hand, drop rates for LTCP-RC$_{full-half}$ are slightly higher as compared to LTCP and LTCP-RC$_{on-off}$.

Still the drop rates observed for LTCP-RC$_{full-half}$ is very low as compared to LTCP-RC(with fixed function), BIC and H-TCP.

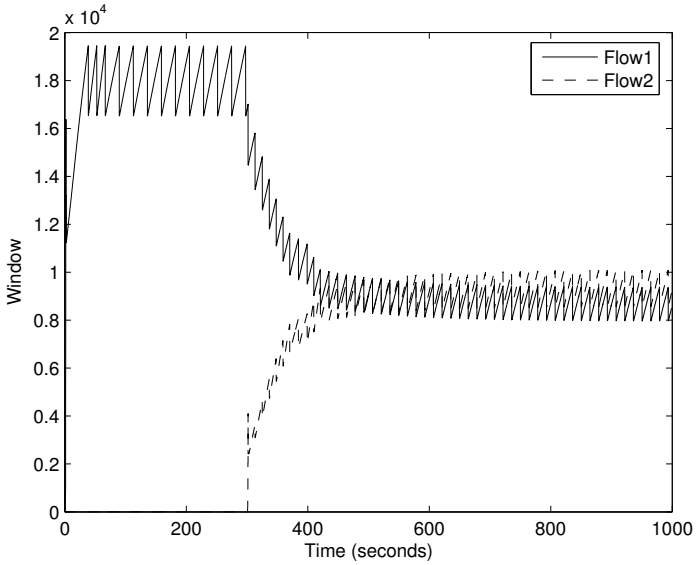
Convergence time for both adaptive LTCP-RC schemes is the same and much smaller than LTCP, LTCP-RC and BIC high-speed protocols. When the second flow is started, the flow with the larger window size observes consecutive drops either in the same or lower layers. On the other hand, the flow with the smaller window size regains the lost bandwidth faster and observe drops in increasing number of layers. Thus, the smaller flow turns on K_R and the larger flow turns off K_R, leading to smaller convergence time for the adaptive schemes.

2. Drop Events

The data for this experiment are taken from the simulations conducted for the previous section. In this experiment, we study the evolution of the window and the total number of drop events observed in different schemes. Figure 9 shows the plot of the congestion window with respect to time for LTCP-RC$_{on-off}$ and LTCP-RC$_{full-half}$ protocol. Figure 10 shows similar plot for LTCP-RC. The graph clearly shows the decrease in convergence time for LTCP-RC$_{on-off}$ and LTCP-RC$_{full-half}$ as compared to LTCP-RC. It is also evident from the graph, that the LTCP-RC$_{on-off}$ incurs less number of drop events, in both Region 1 and Region 2, as compared to LTCP-RC$_{full-half}$. LTCP-RC$_{full-half}$, on the other hand, experiences less drop events compared to LTCP-RC scheme. Results from Table VI, show that the LTCP-RC$_{on-off}$ and the LTCP-RC$_{full-half}$ observe lower drop rates compared to LTCP-RC. Decrease in the drop rates coupled with the reduced number of drop events will make the total number of packets lost by the LTCP-RC$_{on-off}$ and LTCP-RC$_{full-half}$ smaller than the packets lost by LTCP-RC.



(a) LTCP-RC$_{on-off}$



(b) LTCP-RC$_{full-half}$

Fig. 9. LTCP-RC II: Evolution of Window for Two Flows

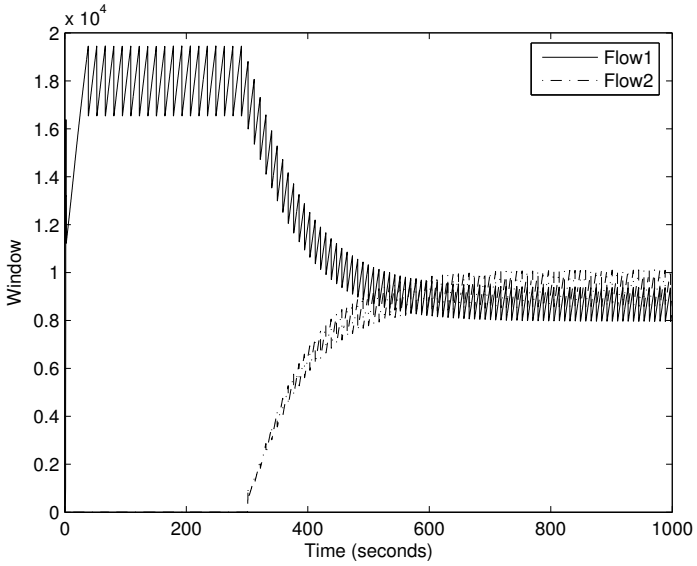


Fig. 10. LTCP-RC II: Evolution of Window for LTCP-RC

3. RTT Unfairness

In this experiment, we study the RTT unfairness for the schemes with adaptive RTT Compensation. At steady state, the LTCP-RC$_{on-off}$ flow uses $K_R = 1$. Therefore, an LTCP-RC$_{on-off}$ is expected to have RTT unfairness similar to an unmodified LTCP. LTCP-RC$_{full-half}$, on the other hand, uses $K_R \propto (RTT)^{1/3}$ at steady state. Hence, its RTT unfairness is expected to be similar to that of LTCP-RC. We conduct simulations to measure the RTT unfairness for two flows, sharing a same link but observing different RTTs. Simulations are conducted on a bottleneck link of 1Gbps and RTT of shorter-delay flow is set to 40ms. RTT unfairness is measured as the ratio of throughput obtained by two flows, with varying the ratio of RTT. Table VII, shows the result obtained from the simulations. The result verifies our arguments presented above. RTT unfairness of the LTCP-RC$_{on-off}$ is similar to that of LTCP. However, the LTCP-RC$_{full-half}$ shows performance similar to LTCP-RC.

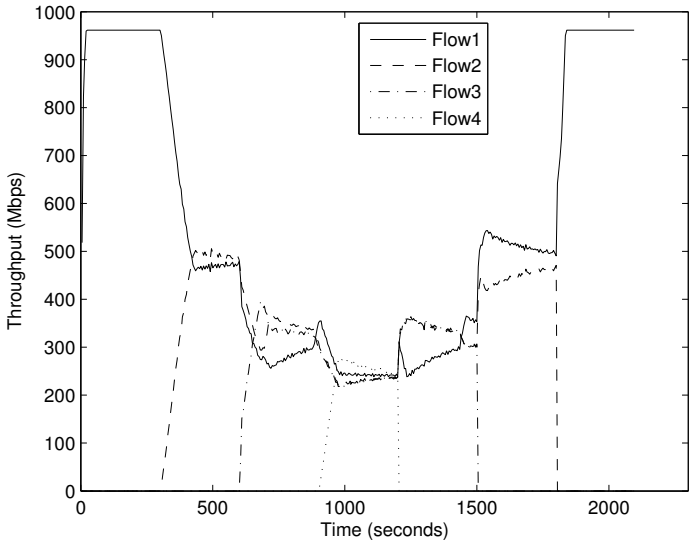
Table VII. LTCP-RC II: RTT Unfairness

RTT Ratio	LTCP-RC$_{on-off}$	LTCP-RC$_{full-half}$	LTCP	LTCP-RC
2	3.05	2.72	3.84	3.35
3	12.92	7.38	12.99	7.63
4	26.18	13.20	28.61	13.64

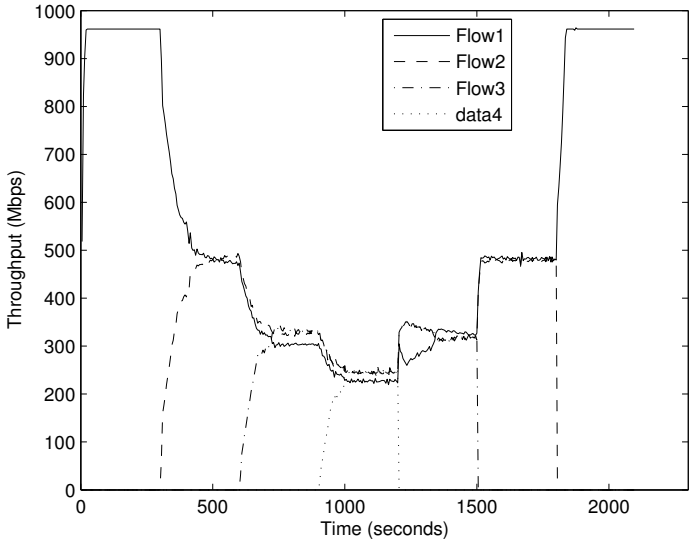
4. Dynamic Sharing

In this experiment, we study the response of the protocols to the dynamic network conditions, when different flows join and leave the network. Dynamic network conditions cause changes in the value of K_R and may affect co-existence of the flows with each other. Simulations are conducted using four flows, each joins and leaves the network at regular interval of 300ms. The bottleneck link capacity is set to 1Gbps, with an RTT of 120ms for the each flow. Simulations are conducted for both LTCP-RC$_{on-off}$ and LTCP-RC$_{full-half}$ protocol. Figure 11 shows the plot of the throughput obtained by each flow, with respect to time.

Figure 11 shows that, in both the schemes, when a new flow joins the network, the existing flows turns off K_R and quickly gave up the bandwidth. This leads to faster convergence. Similarly, when a flow leaves the network, the remaining flows quickly ramp up to grab the available bandwidth. But the graph also presents the problem of short-term unfairness in the adaptive schemes. This occurs when the flows reach a fair share and some flows turn on the K_R while others turn it off. Short-term unfairness can also arise when a flow leaves the network. If one of the flow turns on its K_R earlier than the other flows, then it can lead to temporary divergence. Short-term unfairness is much more prominent in LTCP-RC$_{on-off}$ scheme as compared to



(a) LTCP-RC$_{on-off}$



(b) LTCP-RC$_{full-half}$

Fig. 11. LTCP-RC II: Dynamic Link Sharing

LTCP-RC$_{full-half}$. It can be observed in Figure 11, when the third flow joins or one of the flows leaves the network. The third flow overshoots the first two flows by a large amount in LTCP-RC$_{on-off}$. But, the graph also shows that the offshoot and divergence of the flows is only temporary. The flows with larger window turn off their K_R quickly and all the flows converge back again to the fair share level.

E. Conclusion

The schemes presented in this chapter propose an adaptive RTT Compensation technique, which changes the value of K_R according to dynamic network conditions. This helps in faster convergence and lower drop rates at steady state. But simulation results reveal short term unfairness problems with adaptive schemes. But the divergence is not permanent and the adaptive RTT Compensation techniques will observe fair utilization over long intervals of time.

CHAPTER VI

CONCLUSION AND FUTURE WORK

In this thesis, we propose a new design for improving the performance of window-based schemes in networks characterized by long-delay and high RTTs. We have provided the ground work for a new protocol set termed *LTCP-RC*. The protocol uses a set of *RTT Compensation techniques* to tune the performance of high-speed protocols in high RTT networks. We have studied the RTT Compensation technique specifically with respect to the LTCP high-speed protocol but the framework presented in this thesis can also be extended to other high-speed protocols. One of the goals of this thesis was to understand the performance problems faced by window based schemes, due to the increase in link delays. Our analysis shows that the LTCP algorithm suffers from the problem of long convergence time and RTT unfairness worse than TCP. LTCP-RC scales the congestion window by using an RTT Compensation Factor, K_R. This factor is a function of the minimum RTT observed by the flow. Two design options for LTCP-RC are proposed and studied in detail. The choice of the functions for LTCP-RC is made on the basis of simplicity and feasibility of implementations. The detailed mathematical analysis framework presented in this thesis provides a mechanism to explore and select other design choices.

We have presented through analysis and simulations that LTCP-RC exhibits low convergence time and considerable speed up in claiming bandwidth and packet loss recovery times as compared to TCP. Our design choice makes the RTT unfairness, of LTCP-RC similar to TCP. Extensive analysis and simulation results, presented in this thesis, have also shown that the LTCP-RC can perform better or similar to BIC and H-TCP protocols, while maintaining the time tested AIMD characteristics. LTCP-RC maintains good utilization in the presence of random drops, adaptability

to dynamic network conditions and exhibits good fairness properties. The adaptive RTT Compensation techniques presented in Chapter V, provide low drop rates and very small convergence times. These techniques suffer from short term unfairness, but they provide improved solutions for networks characterized by low multiplexing or flows with long session time.

Analysis presented in Chapter IV and V are based on the assumption of drop tail queues at the routers and low multiplexing at high speed links. Future work will study the performance of LTCP-RC protocol in highly multiplexed links and with Active Queue Management (AQM) schemes like RED [18]. Another possible area for research is study the effect of the router buffers. More work is required in this area, in order to understand the effect of buffer sizes on link utilization and performance of the protocol. LTCP-RC uses the value of minimum RTT observed in order to calculate the RTT Compensation factor. In real implementation, the granularity and accuracy of timers used for estimation of RTT affects the performance of the scheme. We observe that better understanding of these issues will help us to formulate a design which will lead to a more stable and better performing protocol.

REFERENCES

[1] S. Floyd, "HighSpeed TCP for Large Congestion Windows," in *RFC 3649*, Experimental, December 2003. Available at www.icir.org/floyd/hstcp.html

[2] T. Kelly, "Scalable TCP: Improving Performance in HighSpeed Wide Area Networks," in *ACM Computer Communications Review*, Volume 33, Issue 2, pp. 83-91, April 2003.

[3] C. Jin, D. X. Wei and S. H. Low, "FAST TCP: motivation, architecture, algorithms, performance," in *Proc. of IEEE INFOCOM 2004*, Hong Kong, March 2004.

[4] D. Katabi, M. Handley and C. Rohrs, "Congestion Control for High Bandwidth-Delay Product Networks," in *Proc. of ACM SIGCOMM'02*, Pittsburgh, pp. 89-102, August 2002.

[5] L. Xu, K. Harfoush and I. Rhee, "Binary Increase Congestion Control for Fast Long Distance Networks," in *Proc. of IEEE INFOCOM 2004*, Hong Kong, March 2004.

[6] D. Leith and R. Shorten, "H-TCP: TCP for high-speed and long-distance networks," in *Proc. of PFLDnet 2004*, Illinouis, USA, February 2004.

[7] S. Bhandarkar, S. Jain and A. L. N. Reddy, "Improving the Performance of TCP in High Bandwidth High RTT Links Using Layered Congestion Control," in *Proc. of PFLDNet 2005*, France, February 2005.

[8] J. Padhye, V. Firoiu, D. Towsley and J. Kurose, "Modelling TCP Throughput: A simple Model and Its empirical validation," in *Proc. of ACM SIGCOMM'98*, Vancouver, pp. 303-314, October, 1998.

[9] S. J. Golestani and K. K. Sabnani, "Fundamental Observation on Multicast Congestion Control in the Internet," in *Proc. of IEEE INFOCOM*, March 1999.

[10] S. Floyd, "Connections with multiple congested gateways in packet-switched networks part 1: one-way traffic," *ACM SIGCOMM Computer Communication Review*, vol. 21, issue 5, pp. 30-47, October 1991.

[11] C. Caini and R. Firrincieli, "TCP Hybla: a TCP enhancement for heterogeneous networks," in *International Journal of Satellite Communication and Networking 2004*, vol. 22, pp. 547-566, September 2004.

[12] S. J. Golestani and S. Bhattacharyya, "A Class of End-to-End Congestion Control Algorithms for the Internet," in *Proc. of ICNP*, Austin, USA, pp. 137-150, October 1998.

[13] Y. Zhang and D. Loguinov, "Oscillations and Buffer Overflows in Video Streaming under Non-Negligible Queuing Delay," in *Proc. of NOSSDAV'04*, Ireland, June 2004.

[14] "ns-2 Network Simulator," Department of Electrical Engineering and Computer Sciences, University of California at Berkely, CA. Available at http://www.isi.edu/nsnam/ns, Accessed in November, 2003.

[15] "ns implementation of H-TCP," Hamilton Institute. Available at http://www.hamilton.ie/net/htcp.zip, Accessed in August, 2004.

[16] "BI-TCP Implementation for NS-2," Department of Computer Science, North Carolina State University. Available at http://www.csc.ncsu.edu/faculty/rhee/export/bitcp/bitcp-ns/bitcp-ns.htm, Accessed in November, 2004.

[17] D. Chiu and R. Jain, "Analysis of the Increase and Decrease Algorithms for Congestion Avoidance in Computer Networks," in *Computer Networks and ISDN Systems*, vol. 17, no. 1, pp.1-14, June 1989.

[18] Floyd S. and Jacobson V., "Random Early Detection gateways for Congestion Avoidance," in *IEEE/ACM Transactions on Networking*, vol. 1, no. 4, pp. 397-413, August 1993.

VITA

Saurabh Jain was born on February 19, 1980 in Kota, India. He received his Bachelor of Technology Degree in electrical engineering from Indian Institute of Technology Bombay, Mumbai, India in August 2002. His undergraduate research focused on the protocol for grouping the receivers in different groups during multicast transfer of data. He worked as a Consultant at i2 Technologies India Pvt Ltd, Bangalore, India from May 2002 till August 2003. In August 2005, he received his Master of Science Degree in computer engineering from Texas A&M University. His research at Texas A&M focused on network protocols for high-speed and long-distance networks. He had also worked at LayerN Networks, Austin, Texas as intern during May-August 2004. His can be reached at, 3, Sukhdham Parisar, Civil Lines, Kota, Rajastha, India - 324001. His E-mail address is, saurabh_jain80@hotmail.com

The typist for this thesis was Saurabh Jain.