在無線隨建即連網路中利用路由器輔助的 TCP 擁塞控制技術

摘要

隨著網路訊務流量的快速成長和無線網路技術日漸成熟，如何妥善的運用有限的網路資源是一個成功擁塞控制機制要面對的根本問題。TCP 為現行網路上最廣為使用的傳輸層協定，並且有許多的不同版本被提出來改進其效能上的問題，例如 TCP NewReno，TCP SACK 及 TCP Vegas 等。然而由於 TCP 傳送端並未具有網路內部狀態的資訊，如可用頻寬等，大部份的 TCP 擁塞控制機制僅能依賴封包遺失做為觸發擁塞控制的指標。許多研究指出在無線的環境下 TCP 無法有效使用有限的資源並且分辨封包遺失的原因，因而造成整體的效能不佳。本篇研究提出一個藉由路由器輔助的 TCP 擁塞控制協定－TCP Muzha，仰賴路由器提供調速資訊，以幫助傳送端能不依靠封包遺失進行傳輸速度控制，並可更快速的達到最佳的傳輸速度。本研究同時提出模糊化的多層級速率調整方法，藉由動態所獲得的細膩資訊做擁塞避免及因應無線環境下因路由改變或傳輸介質不穩所產生的不必要傳輸速度減低。最後我們在 NS2 模擬器上對所提出的協定做效能評估，實驗結果顯示本協定除了能有效的避免擁塞外，並能減少不必要的降速及重傳封包的次數。

A New TCP Congestion Control Mechanism over Wireless Ad Hoc Networks by Router-Assisted Approach

Abstract

Communication networks have evolved tremendously in the past decades. TCP is the most dominant and deployed end-to-end transport protocol across Internet today and will continue to be in the foresee future. It has numerous enhancing versions for wired network such as TCP Reno, TCP NewReno and TCP Vegas to improve the drawbacks of initial version of TCP. As IEEE 802.11 wireless network technology gains popularity, TCP is very likely to be popular for existing applications so far. However due to unawareness of network conditions, regular TCP is not able to fully control the limited resources and distinguish packet loss from congestion loss and random loss. Based on such implicit assumption, many studies have shown this would results in serious performance degradation in wireless environment. In this paper, we proposed a new TCP congestion control mechanism by router-assisted approach which is inspired by the concept of each wireless node playing the roles of terminal and router simultaneously. Based on the information feedback from routers, sender is able to adjust the sending speed dynamically in order to avoid overshooting problem. We also proposed a multilevel date rate adjustment method to control the date rate more precisely.

Finally we evaluate the performance of our approach by NS2 simulator. Our proposed protocol has 5~10% higher throughput than TCP NewReno and much less number of retransmission. The fairness requirement is also achieved while our proposed protocol coexists with other major TCP variants.

Keyword: TCP, Congestion Control, router, MANET

Acknowledgements

Life is always struggle. I believe many graduate students including me have understood the meaning of it so well that the life of graduate school is never being easily forgotten. Eventually and fortunately, I acquire my master degree. Until now, I still can not believe that such achievement is accomplished by my own hand. Two years ago, I was a student who came from Aletheia University without any knowledge and ability for the oncoming challenge of graduate study. However, my supervisor, Dr. Yao-Nan Lien has fulfilled my weakness and trained me to be a good researcher with well presentation and organization skills. He is the person I have to thank mostly as my teacher, supervisor and friend. I could not be this far without his help, advice and full support.

During the suffering and struggling time of my graduate life, there must be some strength to help me to pass through the dilemma. The strength and support are not only from my supervisor, but also my lab members, Tsung-Ming Lin, Ming-Han Wu, Yi-Fan Yu, and Yung-Chuan Wen. They are always standing by my side and support me to the very end mentally and physically. I never forget the time when we play basketball in the middle of night and try to cheer each other up at the most miserable moment.

Here I also would like to give thanks to my friends, Y-Shin, Allen, Fatty, Kenny, Echo and rest of the OMEGA basketball teams. They play major part of my extracurricular life and always encourage me when I was frustrated and

confused. They are just another family I have in Taipei and they always care and look after me. I really appreciate what they have done for me.

Finally, I would devote my thankfulness to my loved families – my parents, my older sister and Liu Family. Without them, I wouldn't be the person I am today. Without their support, I wouldn't have what I have today. They are always in my heart no matter I am up or low even though we are so far apart. Here I would dedicate this dissertation to all of them: my dear father – Mr. Chi-Lin Hsiao, my dear mother – Mrs. Pi-Hua Tsai, my dear sister- Mrs. Ti-Li Hsiao and Mr. Li-Tun Liu.

Ho-Cheng Hsiao

October 2006

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1

Introduction

With the fast expansion of Internet technologies and applications, some important issues such as network congestion have been raised under the circumstance of traffic burstness. The root cause of congestion is usually the amount of packets generated by end users exceeds the capacity of the network. Network congestion will result in long delay time and high packet loss rate as well as many negative effects in performance.

With respect to a path that connects two end points, congestion usually occurs in a bottleneck node. Network elements including routers and end terminals have to be tightly coupled to prevent the network from being crashed by congestion more efficiently.

In the present, TCP/IP is the de facto and most famous standard to Internet society for data transmission and it offers reliable data transfer as well as flow and congestion control so that its behavior is tightly coupled with the overall Internet performance. Based on the window-adjustment algorithm, sender not only guarantees the successful packet delivery, but also maintains the correct sequence of packets by receiving the frequent acknowledgement from the receiver. The strength of TCP also relies on the nature of its congestion avoidance and control algorithm as well as its retransmission mechanism. Therefore the congestion control within the TCP plays a critical role in adjusting data rate to avoid congestion from happening.

TCP protocol is executed at the terminal nodes and it doesn't have real-time information about the network condition. The indicators of network status to the TCP protocol are packet traveling time as well as success or failure of package delivery. Therefore, most current TCP versions count on these indicators to "guess" (estimate) the available bandwidth over the path connecting the sender to the receiver and to adjust data rate accordingly. The accuracy and the promptness of bandwidth estimation are dependent on many factors such as the stability of network traffic and the length of the path. Not surprisingly, most TCP versions are suffering some performance shortcomings such as congesting network by sending data too fast as well as decreasing data rate unnecessarily due to what so called "slow start".

The initial version of TCP suffers from performance degradation due to network congestion. Therefore some enhancing versions of TCP such as TCP Tahoe[2], TCP Reno[3], TCP NewReno[4], and TCP Vegas[11] had been proposed to improve the performance of TCP in context of wired networks.

However, the performance of these proposed congestion control protocols might be far from optimality due to the insufficient or outdated information about the current network condition. This problem will be more significant in future IP-based networks where the integration of different wired and wireless networks with their specific bandwidth, delay and error characteristics will play an important role. Therefore it might be beneficial in term of improving the performance of TCP end-to-end congestion control by appropriate mechanism based on router-assisted approach.

Since network elements should share the responsibility to respond to congestion, our

philosophy having router provide bandwidth information to sender such that sender can adjust their data rate (window size) more accurately. This concept may not be easy to implement on WAN (Wide Area Network) because upgrading a large number of routers in a WAN is almost a business impossible. However, wireless ad hoc network has no such concern so that it is easy for wireless ad hoc network to embrace this new approach because in wireless ad hoc network, each node plays two roles of end host and routers simultaneously.

The introduction of new wireless technologies and protocols such as IEEE 802.11 are making wireless ad hoc network possible for private and commercial purposes. Because of the unique characteristics of ad hoc network such as unstable transmission medium and frequent route failures, the principle problem of TCP lies in performing congestion control in case of losses that are not induced by network congestion. Nearly all TCP versions assume that packet losses are due to congestion and count on this “indicator” to estimate the available bandwidth along the end-to-end path. When a packet loss is detected, TCP slows down the sending rate by reducing its congestion window. Because of the lack of network status such as available bandwidth, the sending rate can easily overshoot. This would result in serious throughput degradation. In addition, wireless ad hoc network suffers from different types of losses that are not related to network congestion. Any packet loss in wireless ad hoc network is mistaken as congestion by regular TCP, thereby reducing the window size to one segment and then activating the slow-start algorithm again. However, during the slow start phase TCP may face several packet losses due to unstable or breaking link. Therefore connections would spend most of the time in slow start phase due to frequent timeout before they reach the maximum available sending rate.

1.1 Motivation

Such behavior makes TCP not adaptable to wireless environment because of the inability to fully control network resources and distinguish packet losses from congestion and other causes. Therefore conventional TCP which is not familiar with network condition due to its designed nature originally will suffer from serious performance degradation by under-estimation of available bandwidth and frequent trigger of congestion control. Figure.1.1 illustrates such behavior of congestion control.

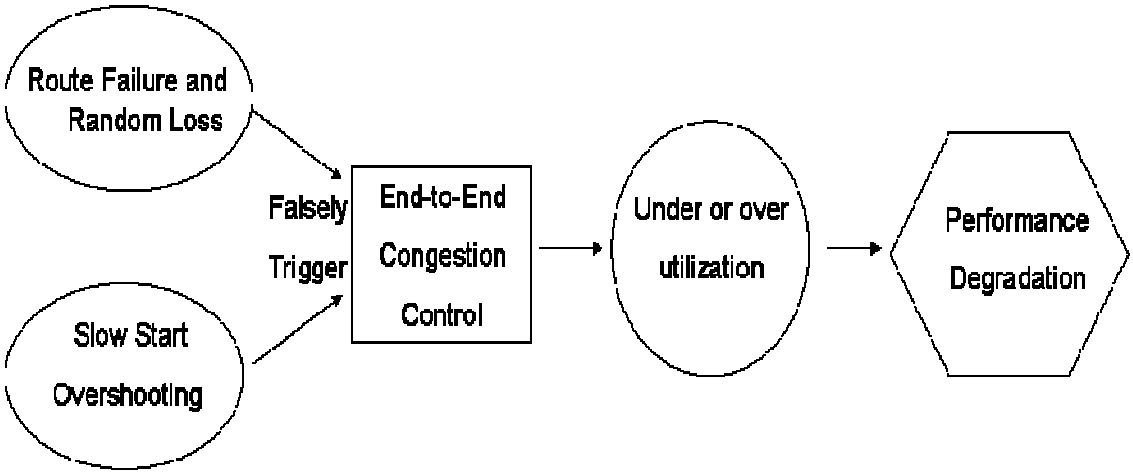

Figure 1.1 The Effect of Mis-triggering Congestion Control

Therefore our objective is to design a new TCP congestion control mechanism named TCP Muzha over wireless ad hoc network. Based on the router-assisted approach, TCP Muzha can dynamically adjust its sending rate in response to the network status more accurately according to the information feedback by routers

1.2 Organization

The rest of this dissertation is organized as follow. In Chapter 2 and 3, we review the relative background and research regarding to TCP and other proposals regarding to performance enhancement in wireless ad hoc network. In Chapter 4, we introduce our proposed congestion control mechanism – TCP Muzha then we evaluate our algorithm with others by simulations in Chapter 5. Finally, we conclude our main contribution of this dissertation and highlight some future work in Chapter 6.

CHAPTER 2

Background

2.1 Transmission Control Protocol

The Transmission Control Protocol (TCP) is a transmission protocol which provides byte-oriented data delivery service for applications over IP networks. It has been tuned to perform well for wired network. It regulates the number of packets it sends by inflating and deflating a window. To do that TCP sender uses the cumulative acknowledgements (ACKs) sent by the receiver. TCP also adapts to problems on congestion which is the main cause of delay. The congestion control scheme in regular (Tahoe) TCP[2] implementation has three major parts: Slow-start, Congestion Avoidance and Fast Retransmit.

Slow-start works as follows: the TCP sender starts with a congestion window (CWND) of size 1. For each received ACK, TCP exponentially increase the window size up to a threshold (ssthresh), then it enters the congestion avoidance phase where it continues to increase its CWND linearly until it reaches the receiver's maximum advertised window.

A TCP sender continually measures the elapse time that acknowledgements take to return to determine whether packet is lost, and provides reliability by retransmitting lost packets. For this purpose, it maintains a running average of this delay (round trip delay) and an estimate of the expected deviation from this average. If the current delay is longer than the average by more than four times the expected deviation (timeout interval), TCP assumes that the packet

was lost. TCP then retransmits the lost packets.

TCP also assumes that the packet was lost if the sender receives a number of duplicate acknowledgements (usually three). This is because the receiver acknowledges the highest in-order sequence number. If it receivers out-of-order packets, it also generates acknowledgements for the same highest in-order sequence number and that results in duplicate acknowledgements. TCP then activates the Fast Retransmit algorithm. The Fast Retransmit algorithm assumes that the missing packets starts with the sequence number that is equal to the number acknowledged by the duplicate ACKs, then thus retransmits it.

2.1.1 TCP Reno

TCP Reno operates almost the same way as TCP Tahoe does. The difference is the introduction of Fast Recovery. After fast retransmit mechanism sends what appears to be the missing segment, congestion avoidance, but not slow start is performed. This is the fast recovery algorithm. It is an improvement that allow high throughput under moderate congestion, especially for large windows.

The reason for not performing slow start in this case is that the receipt of the duplicate ACKs tells TCP more than just a packet has been lost. The fact that three duplicate ACKs can reach the sender indicates that network is not in serious congestion. Although better than Tahoe TCP in dealing with single packet loss, Reno TCP is not much better when multiple packets are lost within a window of data.

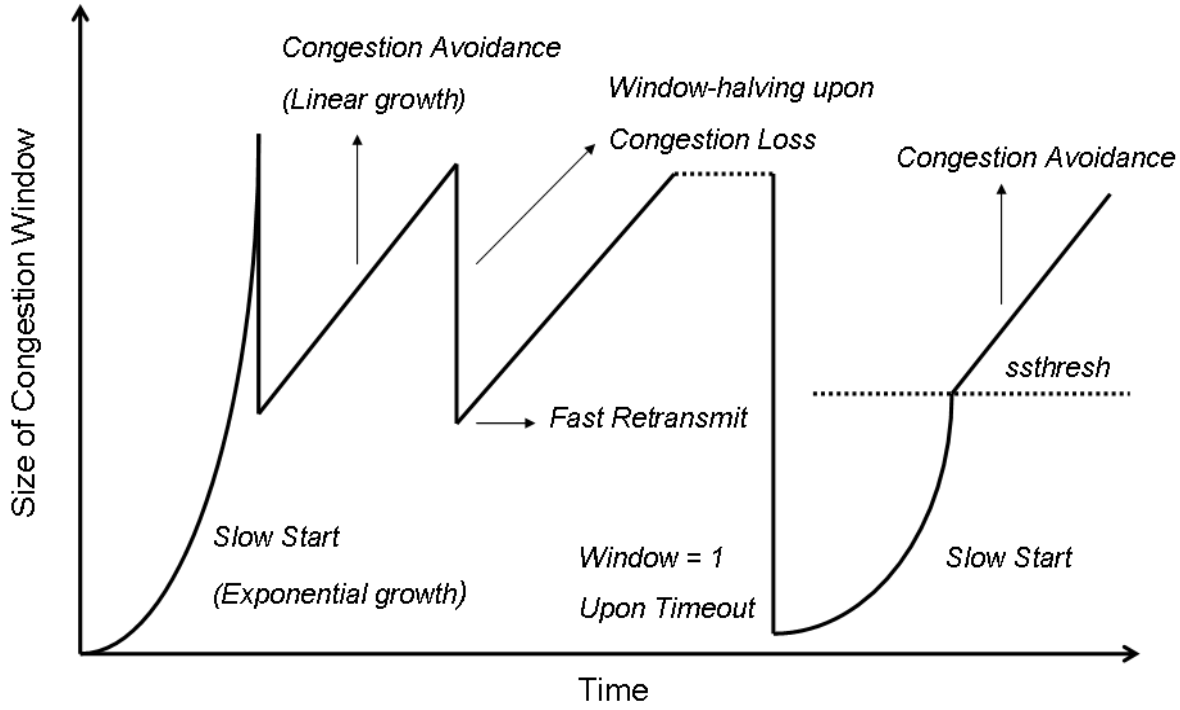

Figure 2.1: Congestion Control in Reno-Style TCP

Figure 2.1 shows the behavior of TCP Reno. The value of CWND increases exponentially during the slow-start phase and increases linearly in the congestion avoidance phase. When TCP finds that a segment is lost, it multiplicatively halves the value of CWND and enters Fast retransmit. This behavior is called the Addictive Increment Multiplicative Decrement (AIMD) [6] for Reno style window-based congestion control.

Although AIMD works fine and results in the robustness and stability of TCP, it is also the key factor that attributes to the poor performance of TCP under wireless environment. We will explain this later.

2.1.2 TCP NewReno and TCP SACK

However TCP Reno is lack of handling multiple packet losses with one transmission window,

which is very likely to happen in wireless links. TCP NewReno and TCP SACK[8] were originally proposed to handle congestion problem in wired network. TCP NewReno modified the fast recovery mechanism of Reno to cope with multiple packet losses from a single window. In TCP NewReno, upon the indication of received partial ACKs, the fast recovery mechanism does not terminate until multiple packet losses from one window are all recovered. On the other hand, TCP SACK is a selective acknowledgement option for TCP, targeting the same problem which New Reno tries to solve. TCP SACK uses information field called SACK blocks to indicate the discontinuous blocks of data which have been received and queued at the receiver buffer. After the sender receives the SACK blocks via the ACK packets from receiver, sender maintains a clear view of buffer status of receiver in order to respond to packet loss. However due to the nature of both schemes which respond to multiple packet losses which can be very likely to occur in wireless network and not able to distinguish the cause of packet loss, these schemes still experience the same performance degradation as TCP Reno does.

2.1.3 TCP Vegas

Unlike most TCP variants, TCP Vegas does not rely on lost packets in order to gauge network capacity, instead using *Round-Trip Time* (RTT) measurements to determine the available network capacity. The congestion control algorithms within TCP Vegas calculate the expected throughput rate and the actual throughput rate once per RTT. The difference between the actual and expected rates is then calculated, effectively indicating the number of packets which are being queued within the network. Once this difference (known as delta) exceeds a certain threshold (gamma, typically set to one packet), slow-start is terminated and congestion-avoidance is activated. Upon exiting slow-start, TCP Vegas decreases the

congestion window by one eighth of its current size in order to ensure that the network does not remain congested. TCP Vegas also has the ability to terminate slow-start before it exceeds the network's available capacity, instead of doubling the congestion window until congestion occurs and packets are dropped by the network. During slow-start, CWND is increased by one segment for every two RTTs, differing from one segment per acknowledgment as used in traditional TCP. When in the congestion-avoidance phase CWND will be increased by 1/CWND, decreased by one segment or left unchanged, with this decision being made once per RTT. The use of RTT measurements results in congestion control algorithms that achieve better throughput and transfer more data for the number of packets transmitted across the network, resulting in increased goodput. TCP Vegas is also more resilient to error prone links and will retransmit packets that have been lost due to corruption far sooner than other variants.

2.2 IEEE 802.11 Standard

The current 802.11[13] protocol covers the MAC and physical layers. The MAC layer defines two different access methods, the distributed coordination function (DCF) and point coordination function (PCF). We now describe the DCF in detail because the PCF cannot be used in ad hoc networks.

DCF is designed to equalize utility and it works as follow. All stations compete for access by using Carrier Sense Multiple Access with Collision Avoidance (CSMA/CA) protocol: they sense the channel before transmitting, either by detecting the carrier of a real transmission, or by deferring to a virtual carrier that is signalled through a Request To Send (RTS) and Clear To Send (CTS) exchange. If no existing transmission is sensed, a station can

transmit one frame, but if an existing transmission is sensed, the station randomly chooses a backoff interval that is uniformly distributed in a range called the Contention Window (CW) and waits for the channel to be idle for intervals that add up to the backoff interval, before it can transmit one frame.

In order to reduce the probability of collision due to stations not hearing each other, the well-known “hidden node problem,” the standard defines a virtual CS mechanism: a station wanting to transmit a packet first transmits a short control packet called request to send (RTS), which includes the source, destination, and duration of the intended packet and ACK transaction. The destination station responds (if the medium is free) with a response control packet called clear to send (CTS), which includes the same duration information. All other stations receiving either the RTS and/or the CTS set their virtual CS indicator, called a network allocation vector (NAV), for the given duration and use this information together with the physical CS when sensing the medium. The physical layer carrier sensing function is called clear channel assessment (CCA). The NAV state is combined with CCA to indicate the busy state of the medium. This mechanism reduces the probability of the receiver area collision caused by a station that is “hidden’’ from the transmitter during RTS transmission, because the station overhears the CTS and “reserves’’ the medium as busy until the end of the transaction.

2.3 Characteristic of Wireless Ad Hoc Network

Wireless ad hoc networks are uniquely characterized by different factors from the traditional wired network. We will explain this briefly in this section.

1. No fixed infrastructure

   Wireless ad hoc network differentiates from the wired network because not only the access medium is wireless, but also each host in wireless ad hoc network plays hybrid roles. Mobile hosts serve as end host and router for connections in the network. Therefore no dedicated router exists in ad hoc networks.

2. Mobility

   Every host in ad hoc network is mobile. For a single connection, end host and routers are not necessarily static. This character has great influence on topology and routing.

3. Shared channel with high BER (Bit Error Rate)

   Due to the nature of wireless ad hoc network, the access medium is highly unstable and the flows have to contend the channel with each other. The contention behavior somehow gets more serious under multihop scenario.

4. Limited resource

   The wireless channel is a very scarce resource and every flow which wishes to use it must contend with each other resulting in multihop flows can only share limited bandwidth of at most a few hundred kilobits per second.

5. Frequent route failure

   The main cause of the route failures are mobility and high BER. The route recovery duration depends on the routing protocol, mobility pattern and traffic characteristics. Every event of route failure has great impact on performance of transport layer protocol such as TCP because the discovering a new route may take a significant amount of time which trigger timeout event of TCP.

2.4 Problem Description

2.4.1 Drawbacks of Slow Start and AIMD

The purpose of growing phase of slow-start is to probe the available bandwidth by increasing the congestion window size exponentially and mostly it is used during the connection initiation and after the timeout event. However slow-start takes several RTT periods before connections actually fully-utilized the available bandwidth. In wired network, connections are expected to spend most of the lifetime in the congestion avoidance phase thus the behavior of slow-start causes no harm to overall performance. However, due to the characteristics of wireless ad hoc network, the frequent route failure and random loss contribute great numbers of timeout. Therefore connections tend to spend a considerable amount of time in slow-start phase which means before connections probe the true available bandwidth, timeout occurs and they re-enter the slow-start phase. Then the overall performance degrades significantly. Also the fairness properties of TCP are very likely to be violated since the connections operate mostly in slow-start phase and can not enter congestion avoidance phase [35].

Another drawback of slow-start is its exponential window growth which causes overshooting problem[29][30].Original TCP tends to take the available resources which is very scarce in wireless ad hoc network as much as possible and this is done by the original design of slow-start. However the routing discovery and maintenance also require and consume part of network resources which is available bandwidth. We describe this problem in detail as follow: Initially TCP sender sends data at a higher rate and the capacity of wireless ad hoc network soon gets overloaded. This leads to contention loss and route failure at MAC layer and forces MAC to trigger route recovery procedure. Meanwhile the TCP connections are interrupted and timeout event occurs. However after the routing is recovered and network

overload is reduced, TCP restarts and soon leads to overload the network again. This phenomenon which named blackout cycle seriously damages the stability and the performance of the whole network because of the overshooting problem.

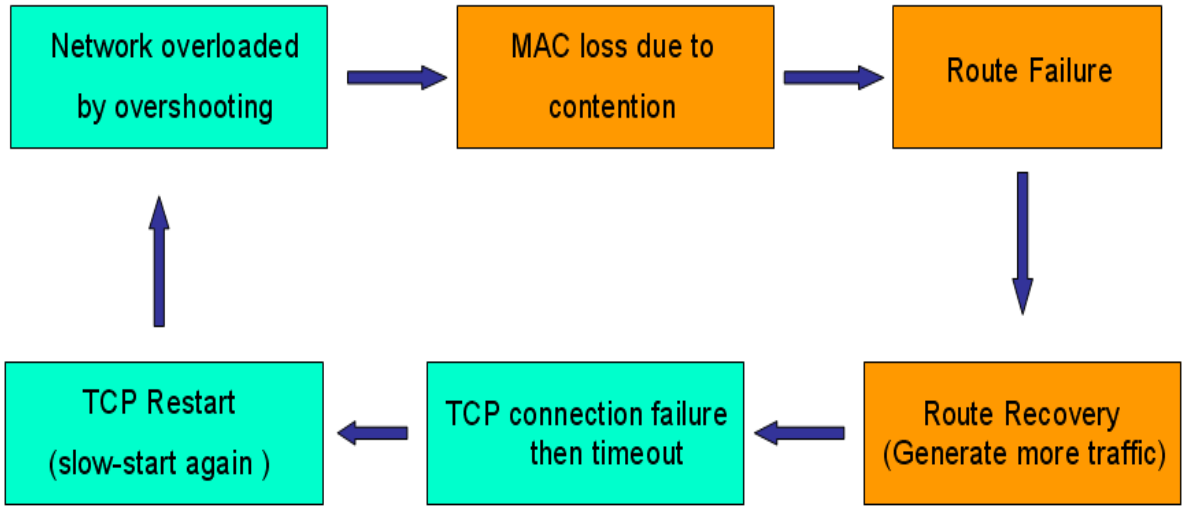

Figure 2.2: Overshooting Problem

AIMD (Additive Increase and Multiplicative Decrease), on the other hand, is not able to perform well in the wireless ad hoc network [35]. The additive increase phase of AIMD has slow convergence to the full available bandwidth and this leads to vulnerability to route failure and random loss. The multiplicative decrease is also the main reason of performance degradation and becomes inappropriate. Since TCP uses packet losses which inferred either receipt of three duplicate ACKs or a timeout to detect congestion, the losses in wireless ad hoc network are not always the symptom of congestion. Losses in wireless ad hoc network can be classified into either link failure induced, or congestion induced. Wireless channel error and mobility are also two primary contributors to losses. Hence treating losses as an indication of congestion seems to be inappropriate because TCP halves its congestion window size while

detecting packet losses. The congestion window size is further reduced to one if a timeout event occurs and connections re-enter the slow-start phase. While the multiplicative decrease is a good reaction to congestion, it is certainly not a good solution to deal with route change or other cause of losses in wireless ad hoc network.

CHAPTER 3

Related Work

In this section, we present several approaches that have been proposed to improve TCP performance in wireless ad hoc networks. These approaches can be classified into two categories: end-to-end approach and router-assisted approach. End-to-end approach requires no network support. The end hosts (sender or receiver) is able to detect the network state by measuring and monitoring the traffic parameters. For instance, large amount of out-of-order delivery indicates route change. The router-assisted approach is able to implicitly or explicitly send network information from routers back to senders therefore senders could responds and react faster to different situations.

3.1 End-to-End Approach

Since resource is scarce in wireless ad hoc network, the ability to accurately probe for the available bandwidth is the key to better performance. Standard TCP scheme such as TCP Reno probes the available bandwidth of the network by continuously increasing the window size until network congestion occurs, and then decreases the window size multiplicatively. The congestion is mostly indicated by packet losses. However, congestion is no longer the only cause of packet loss in wireless ad hoc network. Transmission errors due to high BER (Bit Error Rate), mobility, limited bandwidth and frequent route failure also contribute great amount of packet loss. In such circumstances, Most of the TCP versions with Reno-Style congestion control which treats packet loss as signal of congestion would experience tremendous performance degradation under wireless network.

In [40], a heuristic is employed to distinguish between route failures and congestion without relying on feedback from other network hosts. When timeout occur consecutively, this is taken to be evidence of a route loss. The unacknowledged packet is retransmitted again but the RTO remains fixed until the route is re-established and the retransmitted packet is acknowledged.

TCP-DOOR (Detection of Out-Of-Order and Response)[39] is another pure end-to-end approach to improve TCP performance by detecting and responding to out-of-order (OOO) packet delivery events, which are interpreted as an indication of route failure. The non-decreasing property of ACK sequence numbers makes it simple for the sender to detect OOO delivery of non-duplicate ACK packets. To detect OOO delivery of duplicate ACK packets, they use one-byte TCP option which is incremented with each duplicate ACK.

ADTCP[41] is an end-to-end approach which is based on the use of multi-metric joint identification in order to detect different network states. They introduced four different metrics to be measured. The first metric IDD (inter-packet delay difference) reflects the congestion level along the forwarding delivery path. IDD is unaffected by random channel errors and packet sending behaviors but can be influenced by non-congestion conditions like mobility induced out-of-order packet delivery. The second metric is STT (short-term throughput) which is also used to detect network congestion. STT is less sensitive to short term out-of-order packet delivery than IDD but it is affected by bursty channel errors. Thus they combine IDD and STT to jointly identify network congestion. The other two metrics are used for non-congestion state identification. POR (packet out-of-order delivery ratio) is intended to indicate a route change and PLR (packet loss ratio) is used to measure the

intensity of channel error. Upon each packet arrival at the receiver, it calculates the above four metrics, estimate the network state and send the state information to the sender with every ACK packet so the sender can take the appropriate reaction.

3.2 Router-assisted Approach

On the other hand, an approach based on the translated feedback of network information from routers is proposed by several research groups [17][23][26][28][35]to provide a guideline of adjusting sender behaviors. For instance, the sender is able to respond to different network conditions such as frequent packet loss due to either random loss or congestion according to router feedback and prevents unnecessary decrease of the congestion window. There are various proposals based on such approach with router assistance. Two router-assisted approaches have already standardized: ECN[9] and RED[10]. But these two mechanisms provide only explicit (packet marking) or implicit (packet dropping) single-bit congestion-status information (congestion or no congestion) as feedback to the TCP senders. The lack of more sophisticated information about the router status limits the way in which a TCP sender react on the current router condition adequately. Also, these two simple mechanisms are designed to enable TCP senders to response faster to congestion in router. Their performance gain is limit since ECN and RED are not able to signal information about the available bandwidth to the TCP senders. More appropriate router-assisted approaches for future IP-based network should provide the ability to adapt the sending window of TCP connections both in the case of impending congestion and in the case of available bandwidth.

Other proposal, such as TCP RoVegas[17], is an enhanced version of TCP Vegas by router-assist approach. The main target of this protocol is to solve unreliable reception of

ACK under asymmetric network. In a regular TCP, if an ACK is blocked due to the backward link congestion, TCP senders would determine the forward link congestion occurred and trigger congestion control mechanism to result unnecessary throughput degradation. In RoVegas, the latency that a packet passing through routers are accumulated and marked in IP header so that the sender could use this information to determine whether the lost of ACK is caused by forward or backward path. Thus, it can ignore the congestion on the backward path and keep the data rate on the forward path unchanged.

TCP Jersey[23] is another variant based on router-assisted approach. It develops two key components in its schemes, congestion warning (CW) and available bandwidth estimation (ABE). CW is packet marking scheme that is different from explicit congestion notification (ECN) in the following ways: First, ECN marks packets probabilistically when the average queue length lies between min_{th} and $max_{th,}$ whereas, CW marks all the packets when the average queue length exceeds a threshold. This non-probabilistic marking scheme leaves the TCP sender, which receives the marks, to decide its window adjustment strategy rather than being influenced by the probabilistic marking of the packet in ECN. Second, CW inherits the same information bits used in the original ECN implementation but with simpler parameter setting. So, CW is not as sensitive as ECN to convey a simple image to the bottleneck queue to the sender. TCP Jersey adopts slow start and congestion avoidance from TCP NewReno, but implements the rate-based congestion window control procedure based on ABE.

As mentioned in the pervious chapter, TCP doesn’t have real-time information about network condition so that it has to estimate the available bandwidth. However accuracy and the promptness of bandwidth estimation depend on many factors such as the stability of

network traffic and the length of path. Therefore most TCP versions are suffering performance degradations due to the lack of precise resource control and accurate network information. It is not easy to enhance their performance unless routers can provide assistances. Due to the special characteristics of wireless ad hoc network such as hybrid roles (end host and router) for each node and ease of routers modification, if the routers are able to provide assistance for both end hosts regarding to sending rate and deal with random loss, the unnecessary throughput degradation can be avoided and performance of TCP can be significantly improved over wireless ad hoc network.

.

CHAPTER 4

TCP Muzha

Most of the TCP protocols are not aware of network condition such that they may not be able to control congestion preciously and promptly resulting in unstable bandwidth utilization. Due to the dynamic environment of wireless ad hoc network, each host has even more critical task to control the limited resource such as bandwidth and deal with random loss due to lossy link. However if the routers can share the responsibility of control congestion with end hosts and deal with the unexpected packet loss, the congestion control between two end hosts can be executed more efficiently and preciously.

With respective to a path, congestion usually occurs in the bottleneck point which has the minimum available bandwidth. If the sender can adjust data rate dynamically according to the status of the bottleneck without causing packet loss, the congestion should be avoided or dissolved efficiently. Random loss such as packet loss due to link error or link failure would mis-trigger our proposed congestion control mechanism or under-estimate the available bandwidth. Therefore our proposed protocol has to react to these scenarios and satisfy the objectives as listed below.

4.1 Design Objectives

1. Reducing the occurrence of congestion

   Congestion is always the main target of current existing TCP protocols in both wired and wireless network. Our proposed protocol aims to prevent and dissolve the congestion problem.

2. Maximize throughput and improve overall performance

   The bandwidth in wireless ad hoc network seems more valuable than in wired network. The nature of TCP has weakness in controlling limited resource due to frequent overshoot of congestion window size, thus under-utilization of network bandwidth. Therefore the major goal of our proposed protocol is to maximize throughput in wireless ad hoc network.

3. Dealing with the random loss

   This is a critical part of any TCP protocol because in wireless ad hoc network, random loss is a common phenomenon due to the nature of the air medium. However it is always a major concern while using conventional TCP protocols such as TCP NewReno because TCP tends to decrease the congestion window while facing the event of random loss. In our proposed protocol, a simple approach based on packet marking is proposed to deal with this problem.

4. Provide reasonable fairness for different incoming flows

   When Reno-style TCP and Vegas perform head-to-head, Reno-style TCP generally steals bandwidth from Vegas. Therefore our proposed mechanism must to provide certain level of fairness to ensure the fair sharing of bandwidth while coexisting with other TCP variants.

4.2 Design Issues

The design issues are how to estimate the available bandwidth along the path periodically; how to dynamically adjust the sending rate of senders according the router information in order to better utilize network resources; and how to dealing with random loss.

4.3 Estimation of the Available Bandwidth

The available residual bandwidth in each host depends on many factors such as the length of the queue, queueing time, buffer size and length of the spare queue. In wireless ad hoc network, each host plays hybrid role such as sender/receiver and router. Therefore we assume each host is able to estimate the available bandwidth by itself and then feedback to sender by an index called Data Rate Adjustment Index (DRAI), which will be explained later.

4.4 Use of Available Bandwidth

TCP Muzha defines a new IP option named AVBW-S (Available Bandwidth Status) in IP packet header. The sender of a TCP Muzha connection sets the AVBW-S to a maximum value for every packet it sends. Each node compares its own DARI with this value and replaces it if its value is smaller. The receiver notices the minimum value of DRAI and sends this information back to the sender by acknowledgement (ACK). The sender can then use this information to adjust its data rate, i.e. the size of its congestion window.

Because of the following reason, each node publishes a DRAI value instead of the original available bandwidth. If there is more than one TCP Muzha connection passing through a router, which is very likely, most of them may try to adjust their transmitting data rates up to the level they are informed. Disseminating the original residual bandwidth directly

to all TCP senders will lead to an immediate traffic burst. Therefore, a router must smartly share its residual bandwidth to all the TCP connections that pass through it. Unfortunately, routers are usually not aware of the types of transport protocols that control the packets which they are forwarding. They cannot simply divide their residual bandwidth by the total number of TCP connections. Although a router may be able to peek into the content of packets to determine their controlling transport protocols, we do not take this approach because it may consume a significant part of node capacity. Furthermore, violating protocol independence principle may induce unexpected reliability problems.

4.5 Design of DRAI

Because of the difficulties mentioned above, TCP Muzha takes the following approach: instead of publishing available bandwidth, routers make recommendation to the passing traffic flows to increase or to decrease their data rates. Each node determines a DRAI value, which is a quantified data rate adjustment recommendation, according to its own network status and publishes this information. With respect to each TCP connection, there is a minimum DRAI value called Minimum data Rate Adjustment Index (MRAI). Senders can refer to this value to increase or decrease its data rate (i.e. window size). By this approach, the decision of data rate adjustment is no longer the sole responsibility of senders. Routers which have knowledge of network status can participate in the decision of data rate adjustment. The sender will be able to adjust its data rate according to the MRAI and doesn't rely on the actual occurrence of congestion to trigger congestion control.

4.6 Multi-Level Data Rate Adjustment

The most critical design issue in TCP Muzha is the determination of DRAI. Currently, there doesn't exist any theoretical formula for this. We take empirical approach to design the DRAI formula. Due to a lack of mature knowledge, we choose a coarse grain multi-level quantization formula that defines the data rate adjustment recommendation into levels such as aggressive acceleration / deceleration, moderate acceleration / deceleration, and stabilizing. Further empirical research is needed to find a formula for routers to determine their DRAIs based on their bandwidth utilization.

In fact, ECN is the perfect example of router-assisted and it can be viewed as an extreme case of multi-level DRAI. But this approach is too brief for sender to gain further network status. Therefore, only passive approach such as AIMD is used for congestion avoidance under such scenario. That is why the binary approach of ECN still has drawbacks on controlling flow and date rate. If more information can be provided by routers, the congestion control mechanism is able to work more efficiently and precisely. Therefore we proposed a fuzzy multilevel date adjustment approach as a guideline for routers and end hosts.

4.7 Dealing with Random Loss

Unlike wired links, wireless links that use the air as a transmission medium suffer from high error rates, whether from stationary obstacles, moving objects, interference, weather conditions, or other causes. Bit error rates in wireless communications of over 1 percent are typical, and the errors occur in bursts. This causes the sender to retransmit, timeout, and unnecessary decrease of congestion window which leads to the reduction of throughput even though there is no congestion. Two link layer solutions available that hide random losses are

forward error correction (FEC) and automatic repeat request (ARQ).

FEC sends redundant data so that corrupted packets can be recovered, and introduces a constant delay and bandwidth overhead. However it cannot correct all forms of corruption. ARQ allows resending of corrupted data, but this may lead to incorrect RTT estimated by TCP or timeouts and resending of the same data. These methods seem inadequate, so a TCP level solution may be necessary.

Since our proposal is aimed to solve congestion problem by router-assisted approach, the random loss can also be distinguished by our packet marking scheme which is different from explicit congestion notification (ECN) [9]. When a marked duplicated ACK with a data rate deceleration index is received, the sender notices that the loss was caused by congestion. Otherwise, the loss can be classified into random loss and sender is able to retransmit the loss packets without unnecessary congestion window reduction.

4.8 TCP Muzha Congestion Control Mechanism

Unlike other TCP versions that need to "probe" network bandwidth by increasing their data rates carefully using Slow Start and AIMD, TCP Muzha is able to adjust data rate based on the recommendation given by routers. Thus, TCP Muzha simplifies the three phases of TCP NewReno into two phases; CA (Congestion Avoidance) phase and FF (Fast Recovery & Fast Retransmit) phase.

While TCP session initiated, it directly enters the CA phase. After receiving the new ACK, sender adjusts the CWND size according to the MRAI. After congestion occurs, TCP Muzha inherits most of the congestion control mechanisms from the traditional TCP

NewReno in order to response to congestion immediately. If three marked duplicate ACKs are received by the sender, TCP Muzha enters the FF phase and reduces CWND to one half because the sender treats such indication the sigh of congestion. However if sender receives three unmarked duplicated ACKs, it simply retransmit the loss packets without reduction of congestion window because the loss is treated as random. If transmission timer expires, the sender would reset CWND to 1 and return to CA phase.

Table 4.1: Congestion control mechanism of TCP Muzha

Event	**Status**	**Behavior of TCP Sender**	**Note**
Receive the ACK of the pervious packet	Congestion Avoidance (CA)	Dynamically adjust CWND according to the returning rate adjustment index	Adjust CWND in every RTT
Receiving *marked* three duplicate ACKs	Congestion Avoidance (CA)	(1)CWND = CWND * (1/2) (2) Enter FF phase	Fast respond and half the CWND
Receiving three duplicate ACKs	Congestion Avoidance (CA)	(1) Enter FF phase without change of CWND	Retransmit the loss packets
Timeout	Congestion Avoidance (CA)	(1)CWND = 1 (2)Re-enter CA phase	Re-enter the congestion avoidance phase

CHAPTER 5

Performance Evaluation

In this section, we evaluate and compare the performance of our proposed congestion control mechanism, TCP Muzha with TCP NewReno, TCP SACK and TCP Vegas in different designed network environments by using the network simulator NS2 [25]. We also observe and show the behavior and performance while TCP Muzha coexists with TCP NewReno and its throughput dynamics for different incoming flows.

5.1 Parameters

The results reported in this work are based on NS2 network simulator version 2.29. The link layer of the simulator implements the complete IEEE 802.11 standard MAC protocol DCF in order to accurately model the contention of nodes for the wireless medium. All nodes communicate with half duplex wireless radio with a bandwidth of 2Mbps and a nominal transmission radius of 250m. We also choose the most common parameter for our simulation setup. Each node has a queue (called IFQ) for packets awaiting transmission by the network interface that holds up to 50 packets and is managed in a drop tail fashion. AODV routing protocol is used.

We use two types of network topologies: a chain topology and a cross topology with *h*-hops which *h* is varied from 4 to 32. The chain topology is a good example for multihop connectivity. The distance between any two neighboring nodes is equal to 250m, which allows a node to connect only to its neighboring nodes. Nodes are static because we don’t

consider the link failure problem caused by mobility in this work. Our target network is a wireless multihop network, which is the basis of wireless mobile ad hoc network (MANET).

5.2 Evaluation Metric

The metrics of performance evaluation are as follows.

1. Change of CWND (congestion window size)
2. Throughput under different setting of advertised window (*window_*)
3. Retransmission rate
4. Fairness and throughput dynamics

The general parameters are listed in Table 1 and the DRAI formula used by TCP Muzha is shown in Table 2.

Table 5.1: Simulation Parameters

Parameter	**Range**
Number of Nodes	4~32
Link Bandwidth	2Mbps
Transmission Range	250 m
MAC	802.11
Routing	AODV

Table 5.2 : DRAI Formula

DRAI	Meaning	Change of CWND
5	Aggressive Acceleration	CWND = CWND *2
4	Moderate Acceleration	CWND = CWND+1
3	Stabilizing	CWND = CWND
2	Moderate Deceleration	CWND = CWND -1
1	Aggressive Deceleration	CWND =CWND *1/2

Our simulations are executed to evaluate the overall performance of TCP Muzha over wireless multihop network by several observations as listed below:

1. Observe the change of CWND for TCP Muzha under a chain topology with 4 , 8 and 16 hops respectively.
2. Observe the performance of TCP Muzha under different setting of advertised window size.
3. Observe the behavior regarding to fairness while coexisting with TCP Muzha itself and others.
4. Observe the throughput dynamics of three TCP Muzha flows.

5.3 Simulation 1: Change of Congestion Window Size

In this subsection, we investigate the change of CWND for TCP Muzha under a 4, 8, 16 hop chain topology. An example of the first network topology for the simulation is shown in Figure 5.1. This topology includes source, destination and different numbers of routers only have a single TCP session. The bandwidth between each hop is 2Mbps and the queuing management of routers is in drop tail fashion. The packet size is set to 1460 bytes. We observe the behavior and the CWND change of TCP Muzha and compared it to other TCP variants.

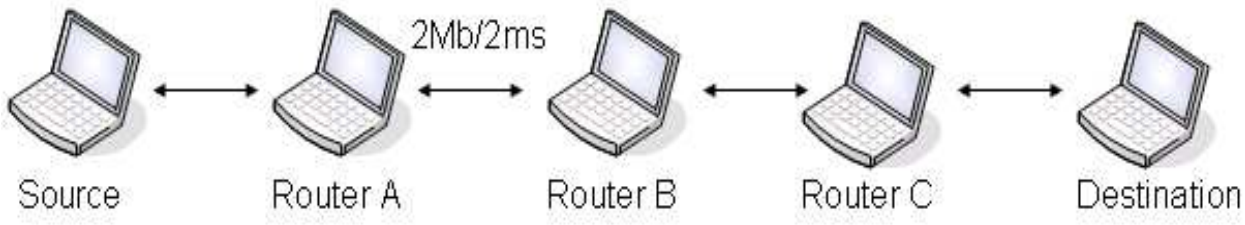

Figure 5.1: 4-hop chain topology with a single flow

The results are shown in Figure 5.2 to 5.7 respectively. TCP Muzha is capable to adjust its CWND size up to the network bandwidth promptly and maintain its CWND while facing the event of random loss. TCP Vegas remains its CWND steadily but due to its conservative nature in congestion control. The CWND size is not able to keep up to the network bandwidth with increase of hop number. TCP NewReno and SACK tend to trigger its congestion control mechanism more frequently due to periodic packet loss and random loss in wireless ad hoc networks.

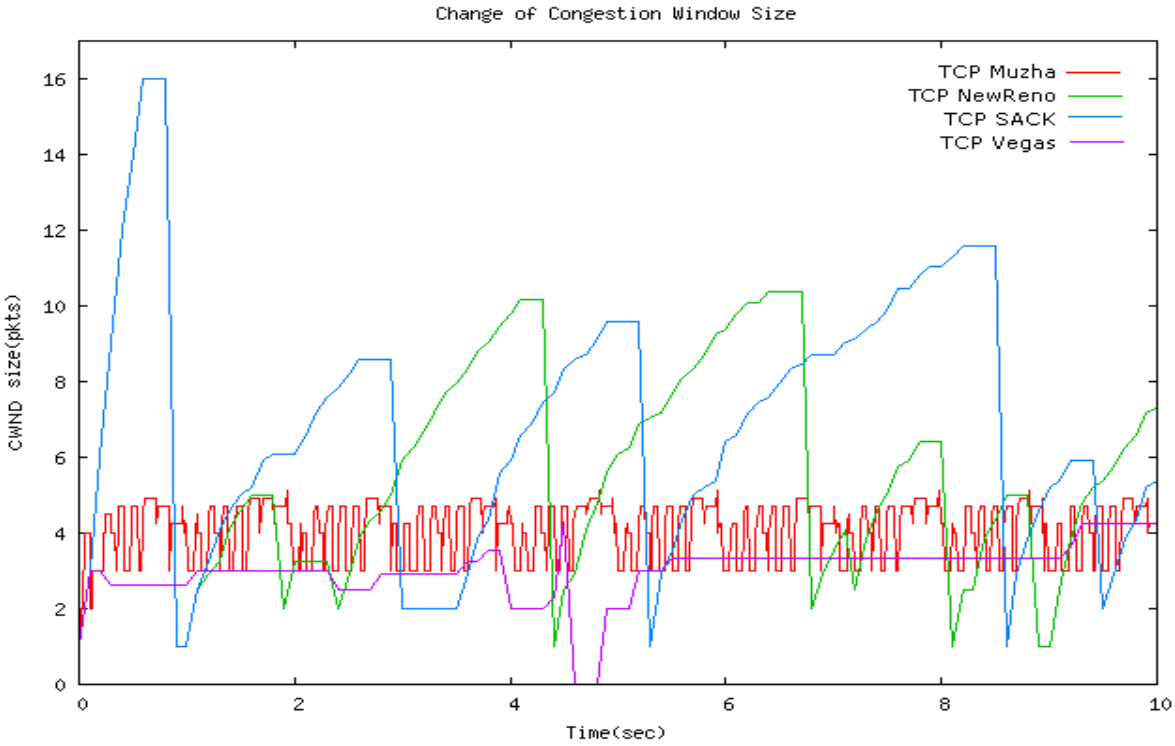

Figure 5.2: Change of Congestion Window Size (4-hop, 0~10 sec)

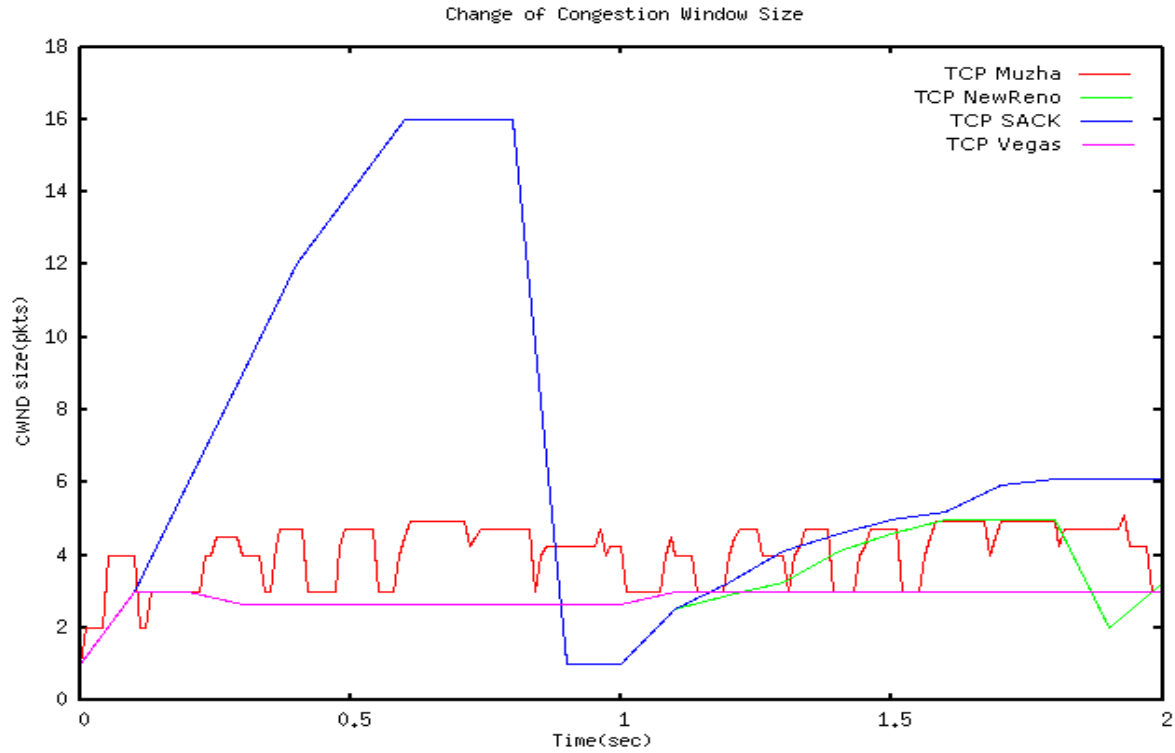

Figure 5.3: Change of Congestion Window Size (4-hop, 0-2 sec)

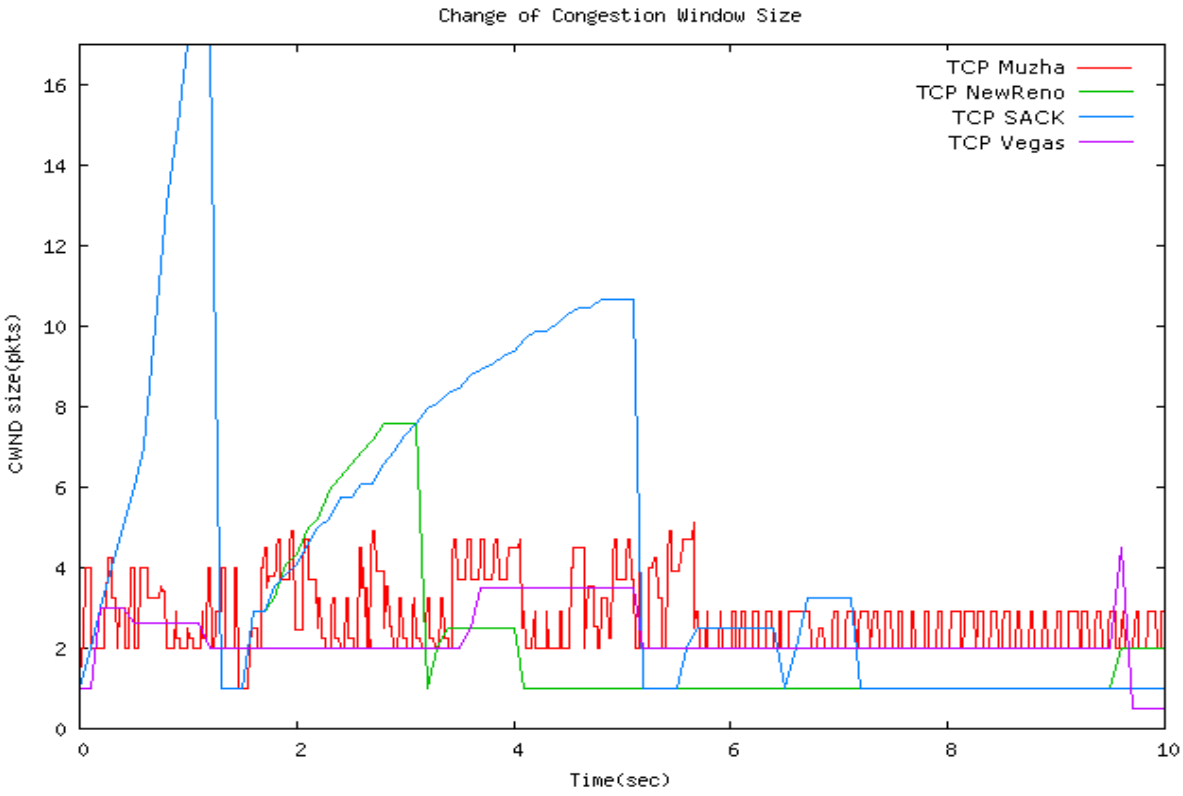

Figure 5.4: Change of Congestion Window Size (8-hop, 0~10 sec)

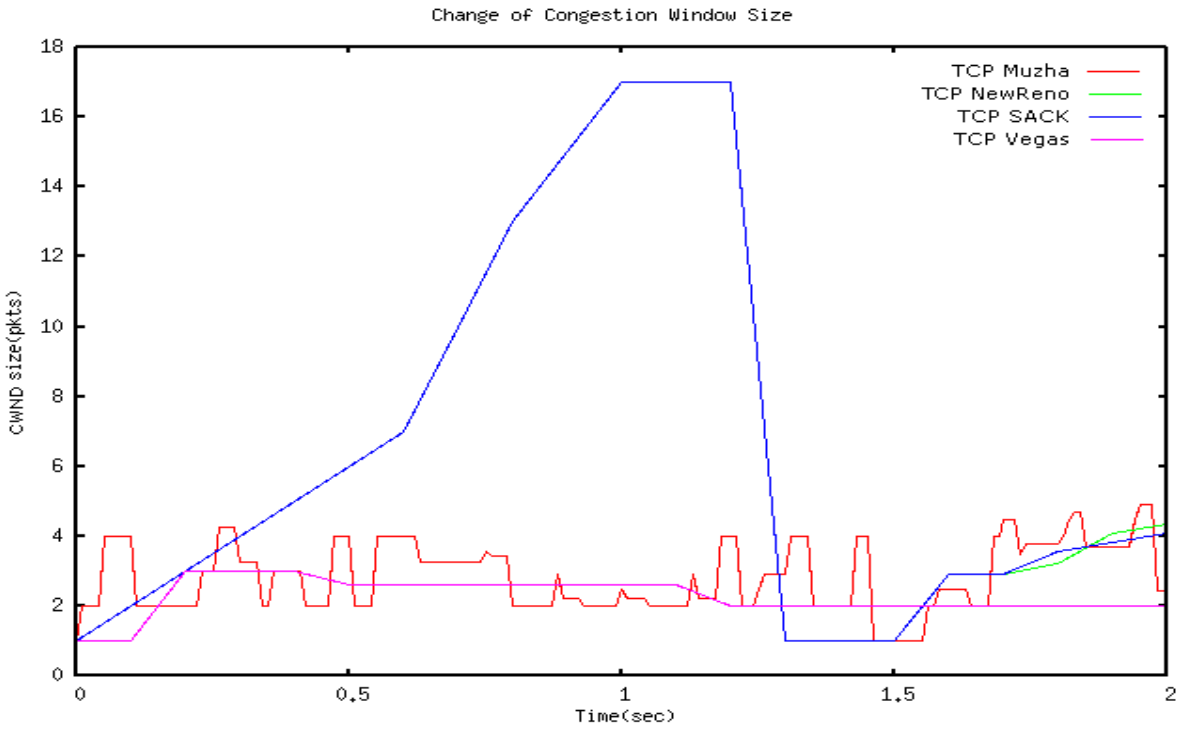

Figure 5.5: Change of Congestion Window Size (8-hop, 0~2 sec)

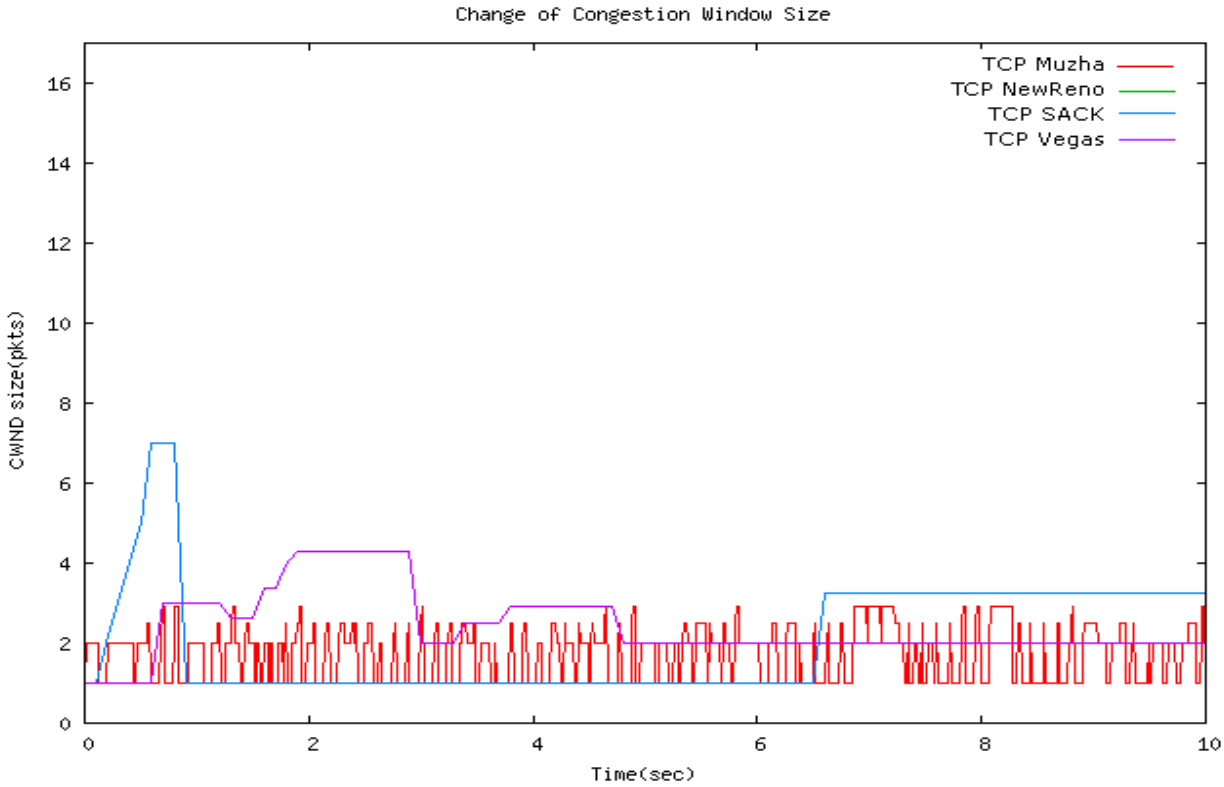

Figure 5.6: Change of Congestion Window Size (16-hop, 0~10 sec)

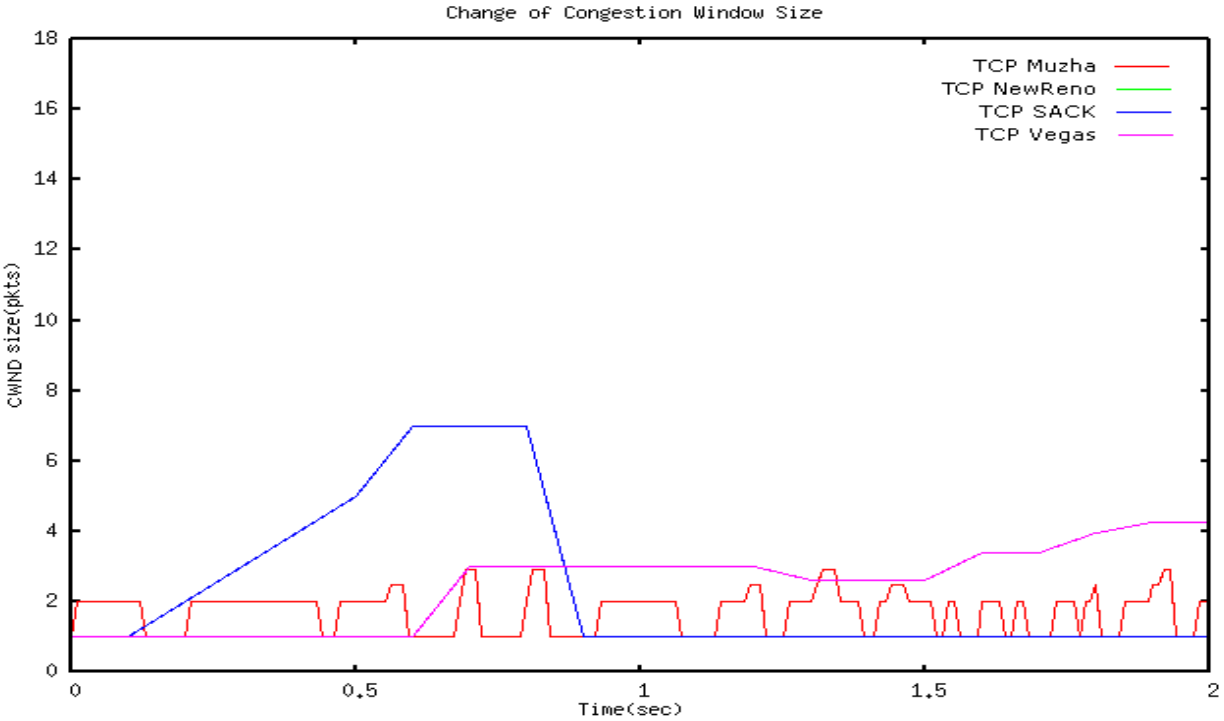

Figure 5.7: Change of Congestion Window Size (16-hop, 0~2 sec)

However when the length of path is increased, the packet loss rate raises tremendously due to frequent contentions and link failures. From Figure 5.4 and Figure 5.6, we found that TCP Muzha can’t avoid the effect of contentions and link failures but the dynamic adjustment mechanism can stabilizes the change of the CWND Immediately. On the other hand, the CWND of TCP NewReno and Sack fluctuate extremely due to the frequent packet loss caused by contention and link failure. TCP Vegas still controls its CWND conservatively.

5.4 Simulation 2: Comparison of Throughput and Retransmission

The network topology and parameters used in the simulation 2 is same as simulation 1 which is considered an equally spaced chain comprising of h+1 nodes (h hops) with a single flow. However we have different setting of advertised window size which is the limit of the real transmission window size in a TCP connection. Because the advertised window size has the influence on TCP instability problem [31], we would like to investigate the influence of this parameter on our proposed protocol and other TCP variants. The overall simulation time is 30 sec. The TCP session is the only traffic in that network; no background traffic exits. Hence there are no network condition changes in the whole life time of this experiment. In this simulation, we consider TCP Muzha, TCP NewReno, TCP SACK and TCP Vegas for h-hop chain with varying hop count. As measures, we consider throughput, number of retransmissions as function of chain length. The bandwidth is kept fixed to 2Mbps. Figures 5.8 to 5.13 show the simulation results.

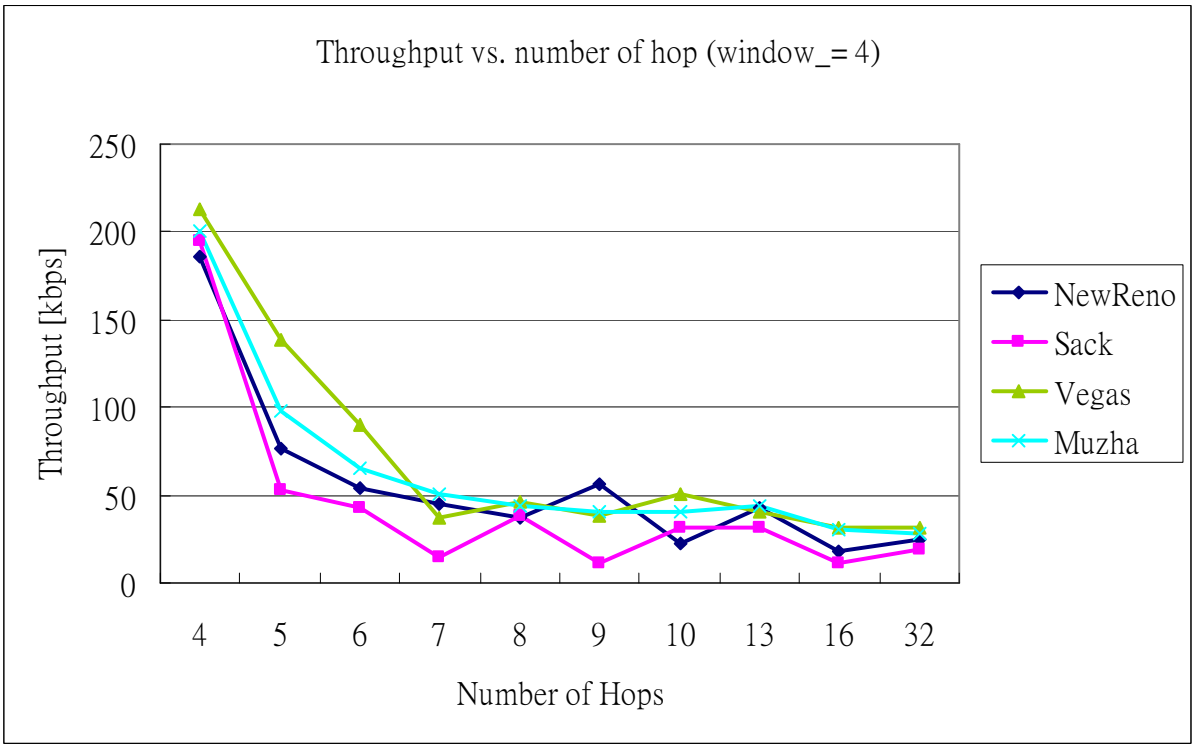

Figure 5.8: Throughout vs. Number of Hops in the h-hop chain (*window_*= 4)

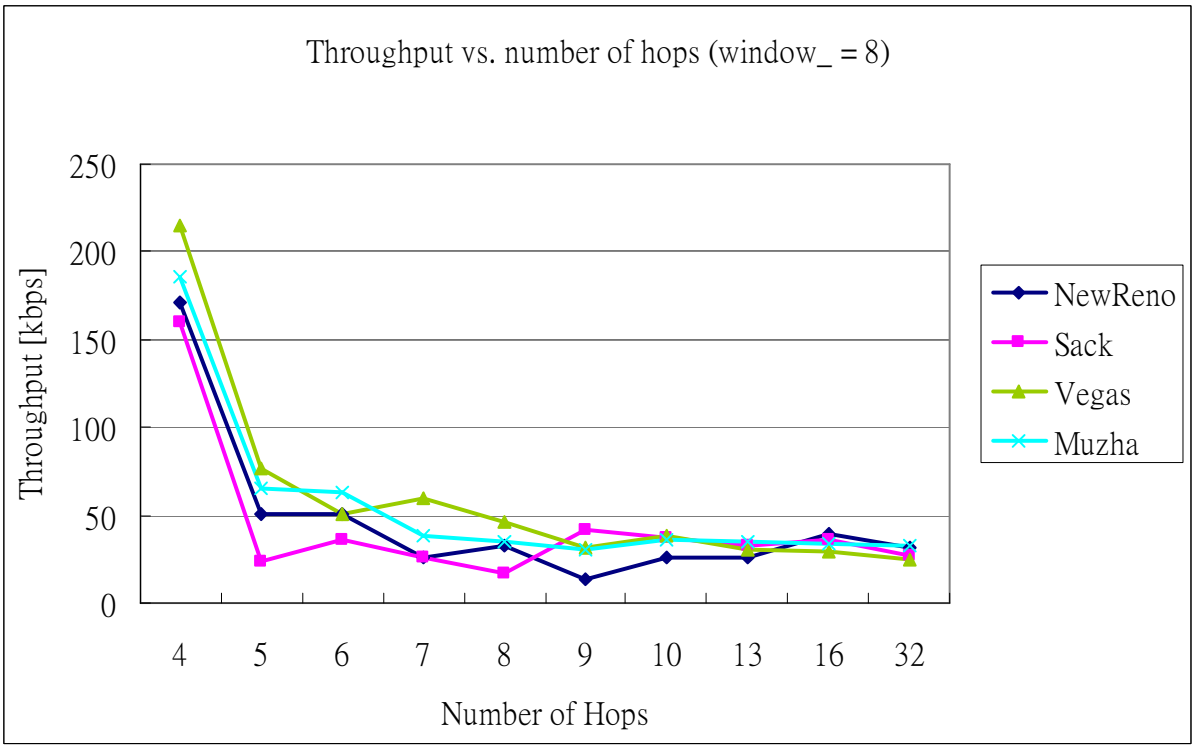

Figure 5.9: Throughout vs. Number of Hops in the h-hop chain (*window_*= 8)

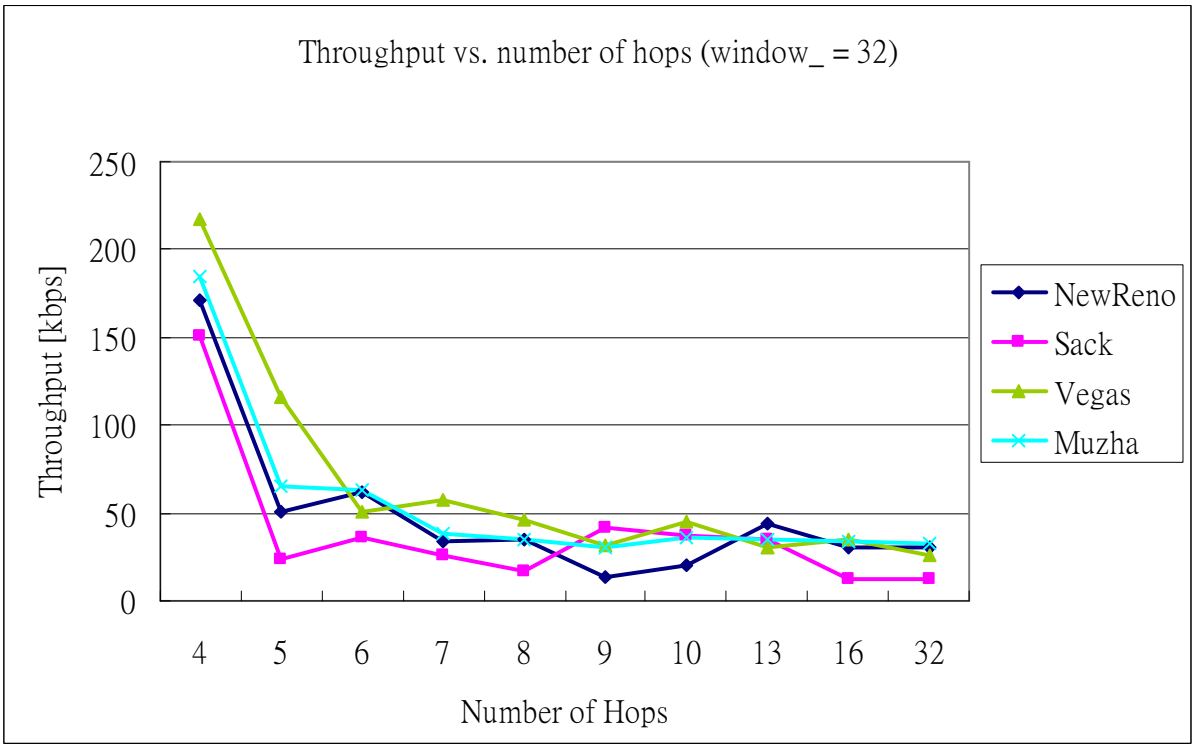

Figure 5.10: Throughout vs. Number of Hops in the h-hop chain (*window_*= 32)

From Figure 5.8 to 5.10, we observe that TCP Vegas has higher throughput than other TCP protocols including TCP Muzha as the hop count is less than 8 due to its fine-controlled congestion mechanism. However TCP Vegas no longer performs well with the longer path because TCP Vegas keeps its congestion window size too small (about 3 packets). On the other hand, TCP Muzha has better performance than TCP NewReno and TCP SACK about 5% ~ 10%. The main reason is that the aggressive window growth of TCP NewReno and TCP SACK cause network overloaded and periodic packet drops which leads to more frequent timeouts in transport layer and more contentions and link failures in MAC layers. TCP Muzha tends to avoid the periodic packet loss in slow-start phase and controls the congestion window size more precisely according to router feedbacks. Furthermore TCP Muzha provides more stable throughput than TCP NewReno and SACK with longer forwarding path.

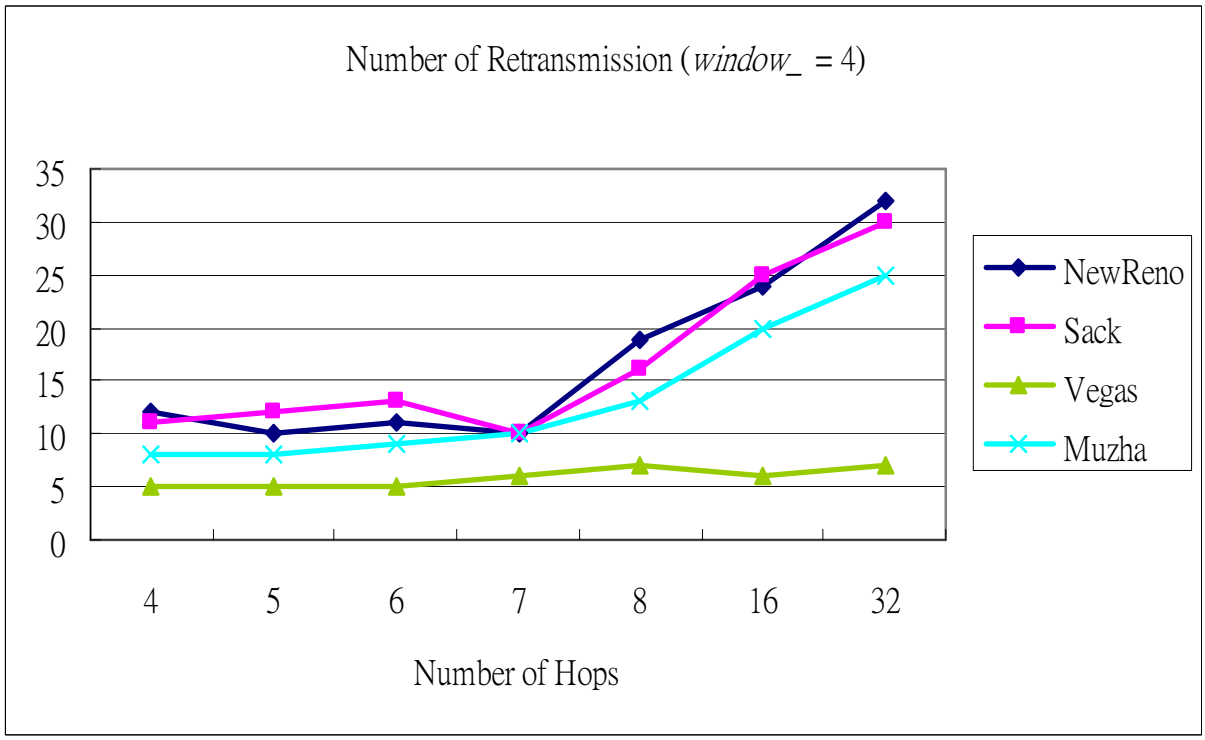

Figure 5.11: Retransmission vs. Number of Hops in the h-hop chain (*window_*= 4)

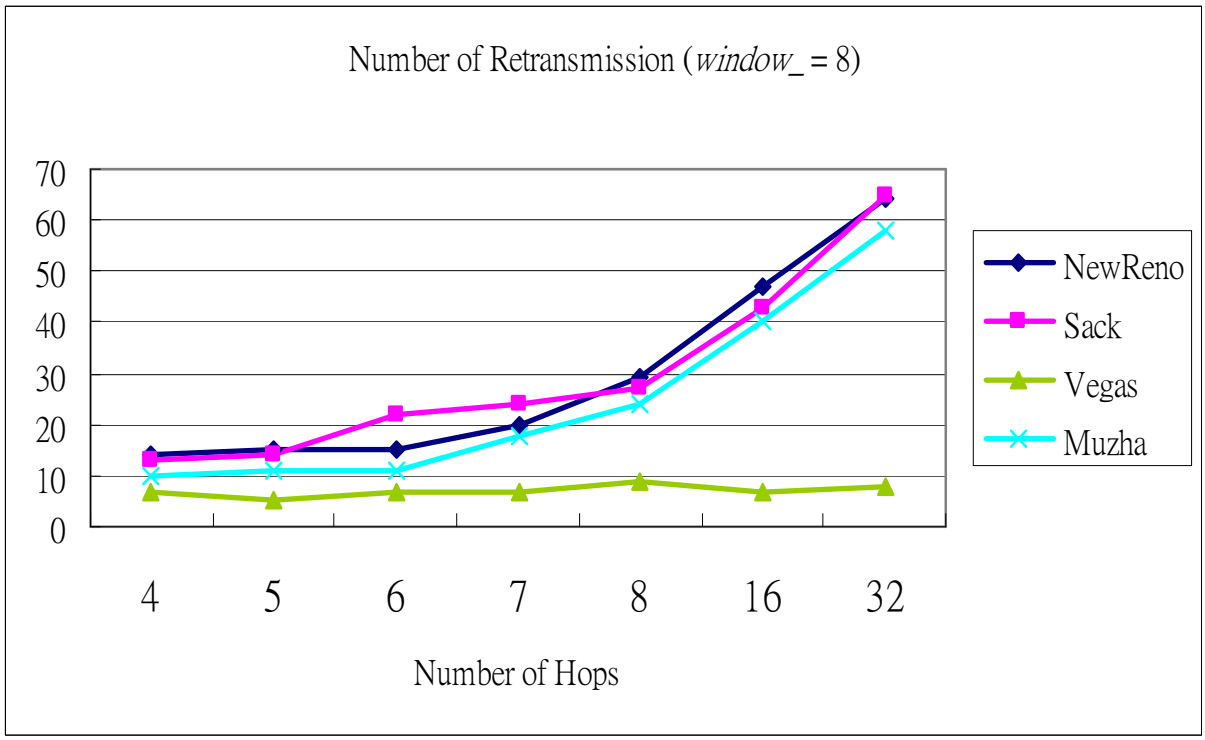

Figure 5.12: Retransmission vs. Number of Hops in the h-hop chain (*window_*= 8)

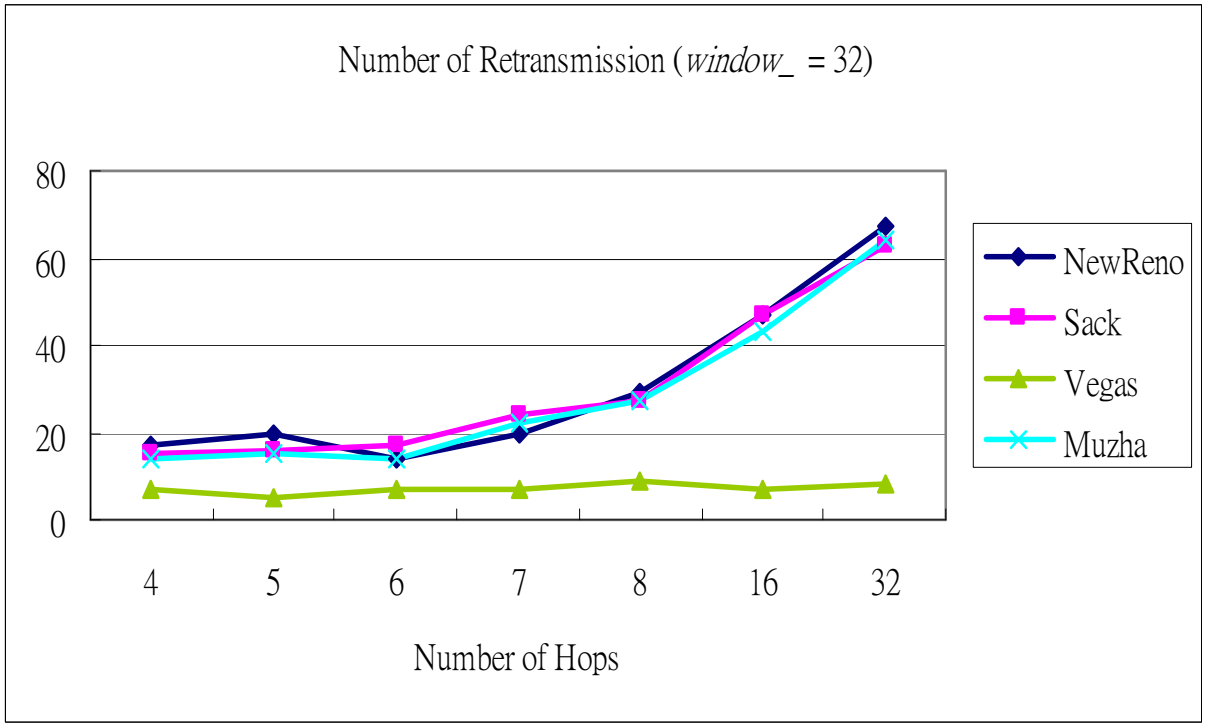

Figure 5.13: Retransmission vs. Number of Hops in the h-hop chain (*window_*= 32)

Figure 5.11 to 5.13 show that TCP Vegas causes much less retransmission than other TCP variants. In fact, the number of retransmission maintains very low for TCP Vegas for any number of hops. Oppositely, the number of retransmission of TCP NewReno and TCP SACK is much greater than TCP Vegas. This is because during the slow start phase, TCP NewReno increases the congestion window size aggressively and we have noticed that TCP NewReno operates during more than 40% of connection in slow start. Such behavior causes periodic packet loss which has to be retransmitted and thus, a great number of retransmission. The inappropriate window growth mechanism of TCP NewReno also results more packet drop on the link layer and thus more route failures. TCP Muzha has less number of retransmission than TCP NewReno and TCP SACK at $h < 8$ while the advertised window size is equal to 4 and 8. This is due to the precise control of congestion window size and furthermore the

avoidance to TCP overshooting problem as we mentioned before. With increase number of hop count, the number of retransmission are all increasing for all three window-based congestion control protocols (NewReno, SACK and Muzha) but TCP Muzha still has the smallest number of retransmission among three of them. However while the advertised window size is greater, the number of retransmission among TCP NewReno, SACK and Muzha are almost the same. The reason for this result is the link layer contention increased with increasing size of advertised window since packets in flight are not able to distribute evenly among nodes, leading to more packet drops and thus, to more number of retransmission.

5.5 Simulation 3: Fairness Test

5.5.1 Simulation 3A: Coexistence with other TCP NewReno

In simulation 3A, we investigate the fairness issue of TCP Muzha while coexisting with TCP NewReno and the performance of TCP Muzha. First, we consider a *h*-hop cross topology with two flows as shown in Figure 5.15. *h* is 4, 6 and 8 respectively. Each flow is a single FTP session using different TCP protocols. One travels vertically and the other travels horizontally. The simulation time is 50 seconds and the bandwidth is fixed to 2Mbps. We run two sets of simulation: TCP Vegas vs. TCP NewReno and TCP Muzha vs. TCP NewReno. The results of throughput and fairness are showed in Figure 5.16 to 5.18. The fairness results are computed using the fairness index as defined in [38] as shown in Figure 5.14.

$$\left(\sum_{i=1}^{n} x_i \right)^2 \Big/ n \sum_{i=1}^{n} x_i^2$$

Figure 5.14: Jain's Fairness Index

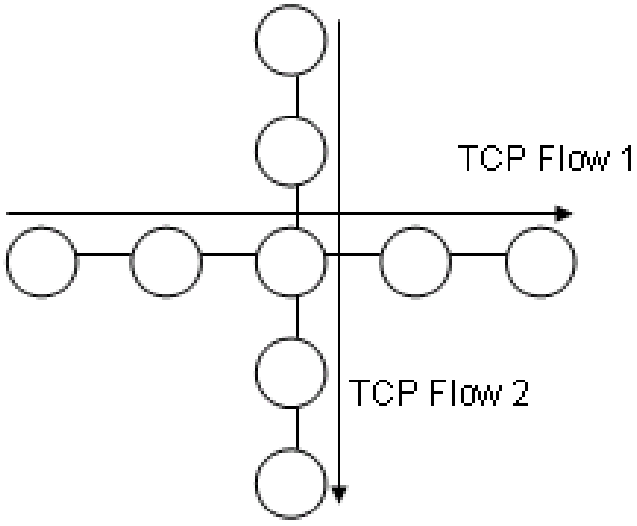

Figure 5.15: 4-hop Cross Topology with 9 Nodes and 2 TCP flows

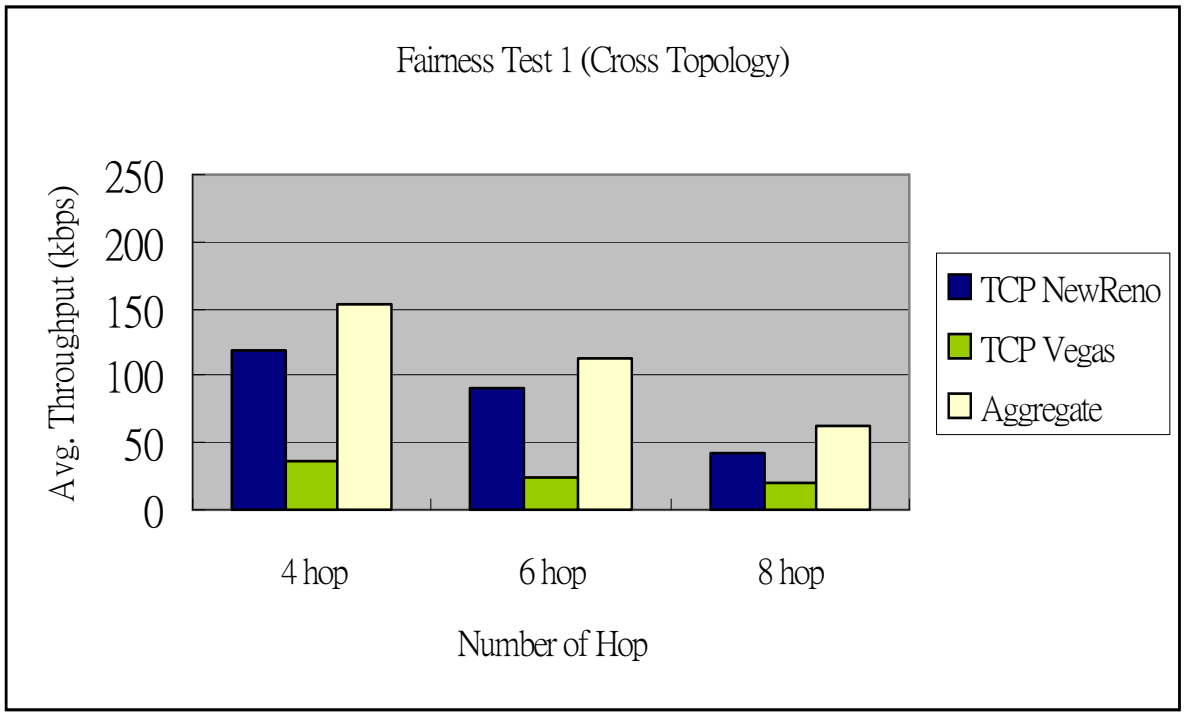

Figure 5.16: Throughput for Coexisting flows of TCP NewReno and Vegas

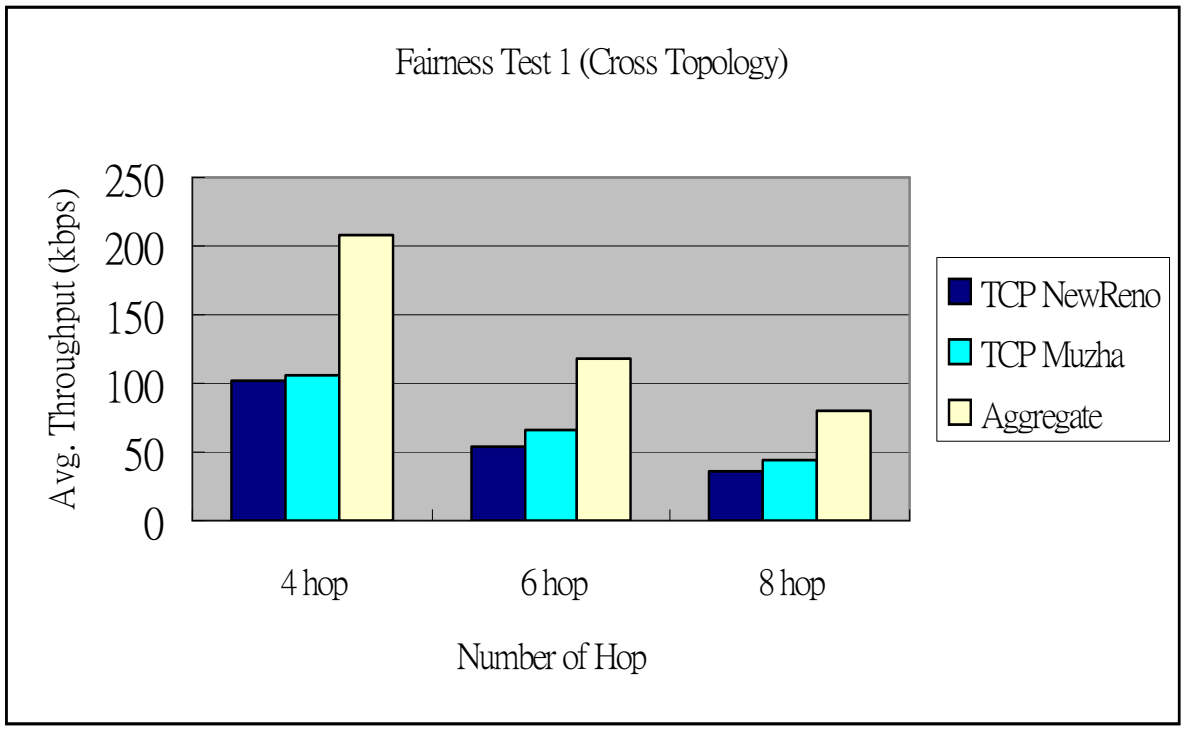

Figure 5.17: Throughput for Coexisting flows of TCP NewReno and Muzha

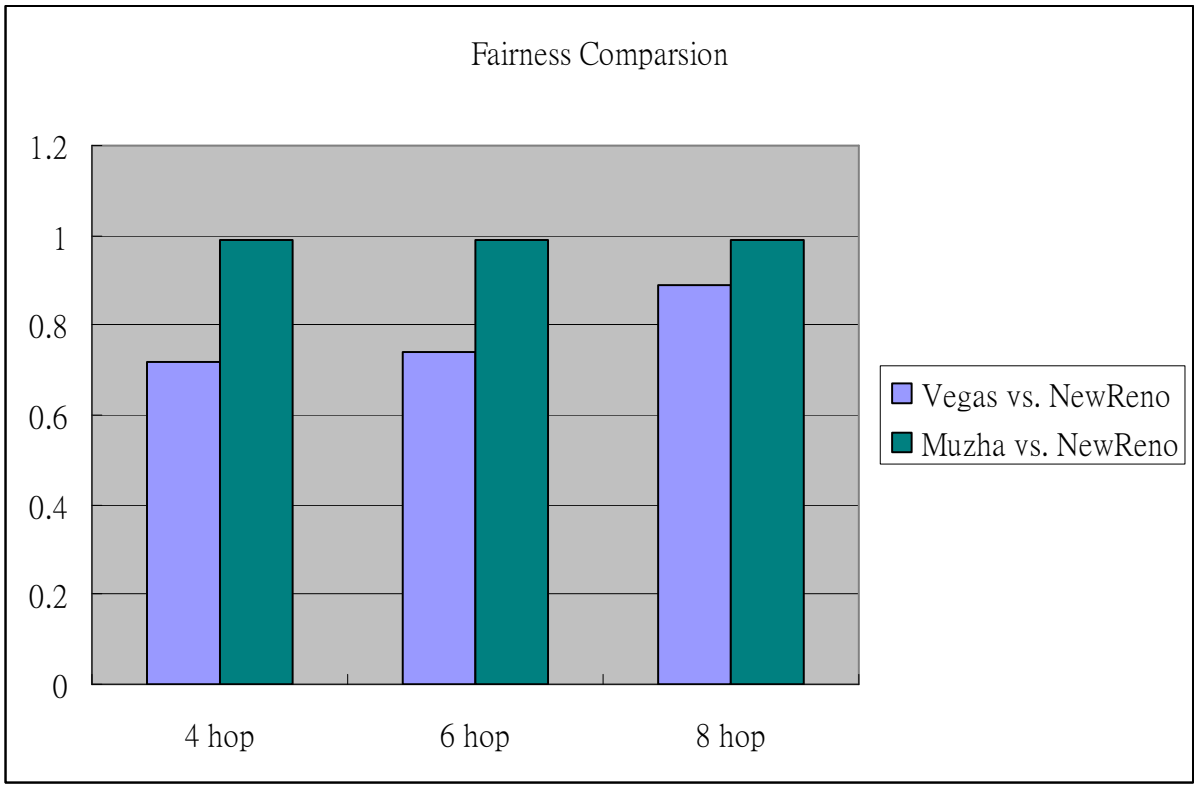

Figure 5.18: Fairness Index for Coexisting flows

TCP Vegas adopts a proactive congestion control avoidance scheme. It reduces its congestion window before an actual packet loss occurs. Reno-style TCP, on the other hand, employs a reactive congestion control mechanism. It keeps increasing its congestion window until a packet loss is detected. Many researchers [42] [43] [44] have found that when Reno-style TCP and Vegas perform head-to-head, NewReno generally steals bandwidth from Vegas. From our simulation results, we have discovered the similar behavior while TCP NewReno and TCP Vegas coexist. Figure 5.16 shows that TCP NewReno consumes most of the bandwidth and results the low throughput of TCP Vegas. While our proposed protocol coexists with TCP NewReno, Figure 5.17 shows the fair bandwidth sharing has been achieved between TCP NewReno and TCP Muzha because TCP Muzha is able to control bandwidth

usage precisely according to periodic router feedback of available bandwidth. This not only guarantees the fairness requirement but also provides higher aggregate throughput. The fairness index from Figure 5.18 also provides evidence of fairness achievement by our proposed protocol.

5.5.2 Simulation 3B: Throughput Dynamics

In simulation 3B, we consider a simple chain topology consisting of a four-hop linear chain with three flows. Each flow enters the network at 0 sec, 10 sec and 20 sec respectively. We investigate the throughput dynamic to observe if our protocol itself is able to achieve fair utilization among different flows. The throughput dynamic for different TCP variants are presented from Figure 5.19 to 5.22. As can be seem, three flows converge the fair utilization of available bandwidth with our proposed protocol. However the convergence of fair bandwidth utilization for three flows by other TCP variants is slow and oscillatory.

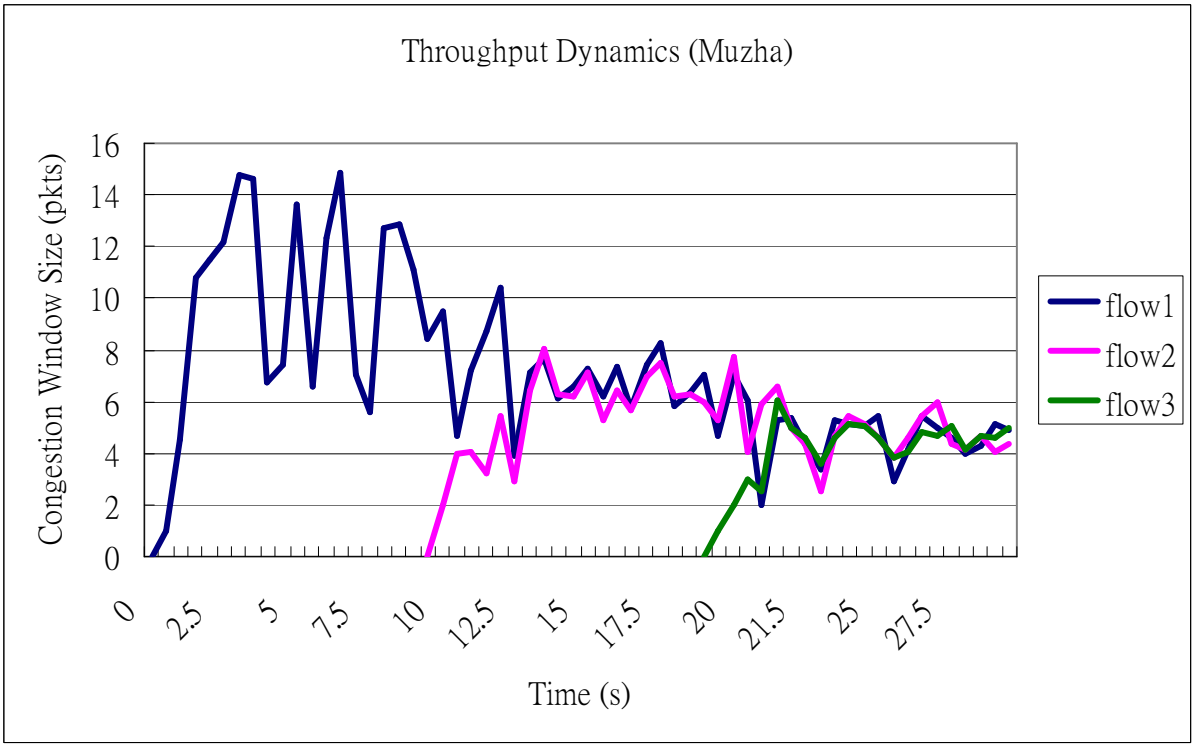

Fig 5.19: Throughout Dynamics [three flows] – TCP Muzha

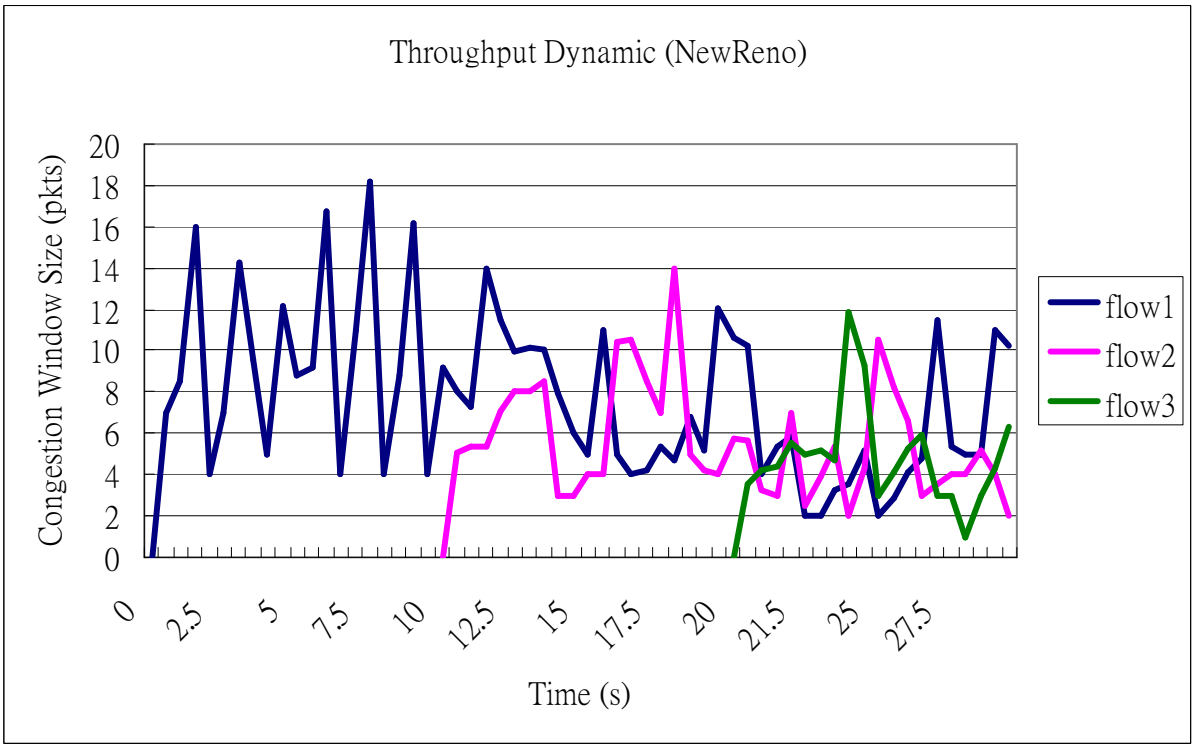

Figure 5.20: Throughout Dynamics [three flows] – TCP NewReno

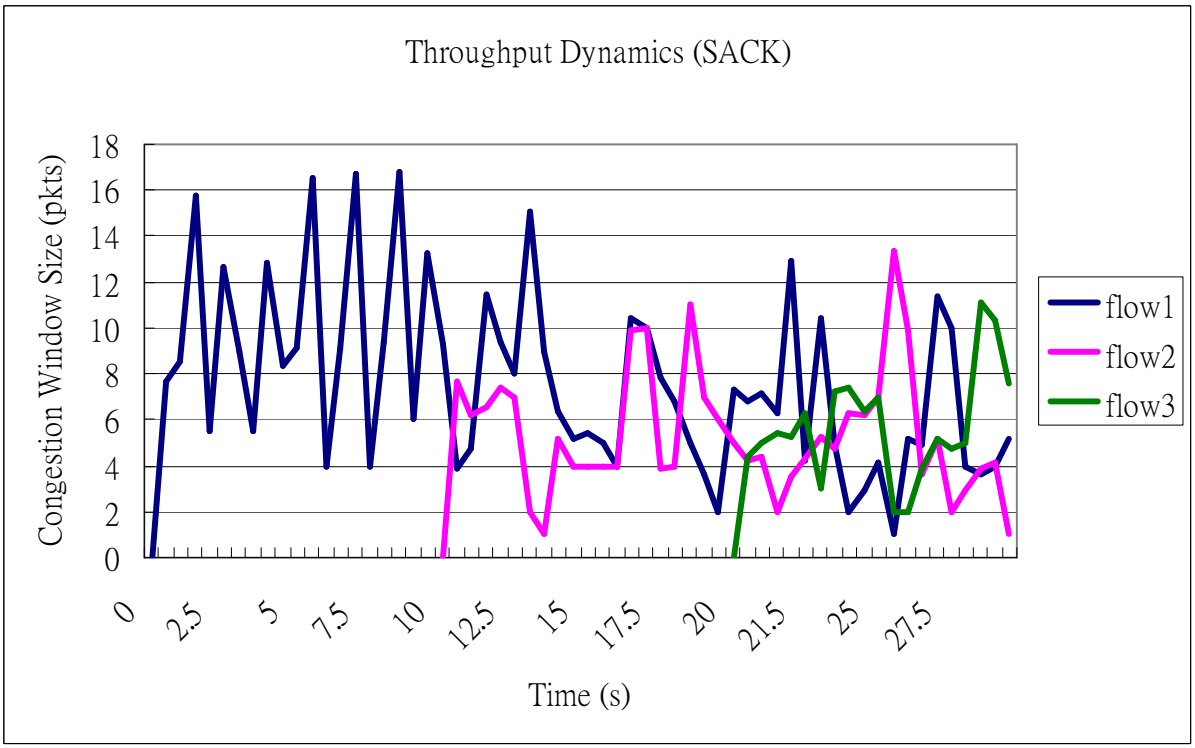

Figure 5.21: Throughout Dynamics [three flows] – TCP SACK

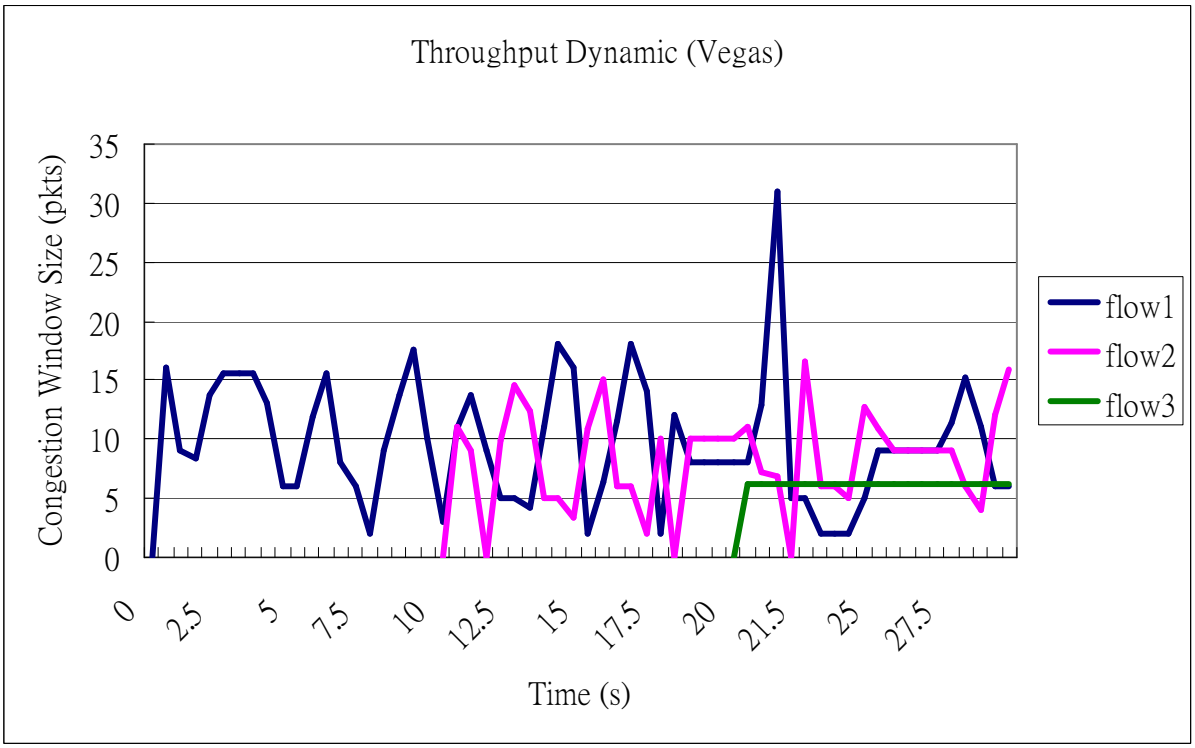

Figure 5.22: Throughout Dynamics [three flows] – TCP Vegas

CHAPTER 6

Conclusions and Future Work

From the results shown from a series of simulations, we conclude that TCP Muzha can resolve congestion efficiently and has higher average throughput than TCP NewReno. TCP Muzha performs better than TCP NewReno and TCP SACK due to its dynamic data rate adjustment mechanism and the ability to deal with random loss. While coexisting with TCP NewReno, TCP Muzha remains stable throughput and fair share of available bandwidth compared with other major TCP variants.

The concept of multi-level data rate adjustment and the details of how to control the size of CWND are still required to be investigated. For instance, improvement of the vibrating behavior of CWND of TCP Muzha, consideration of queue size, RTT as part of DRAI formula and support of mobility are essential. The mechanism of using ACKs as congestion indicator still has drawbacks, especially in the erroneous environment such as wireless network. For example, while different sessions pass through the same link, the sender with large link delay would have weakness while competing bandwidth due to the delayed ACKs.

Fairness is still important problem which needs to pay extra attention on except throughput. TCP Vegas suffers seriously in multi-protocols environment because it can't utilize bandwidth fairly and this drives to difficulties of real-world implementation of TCP Vegas. Therefore how to enhance TCP Muzha to meet such demands will be our future task.

Reference

[1] J. Postel, "Transmission Control Protocol," *IETF RFC 793*, 1981.

[2] V. Jacobson, "Congestion Avoidance and Control," *Proc. of ACM SIGCOMM*, pp. 314-329, Aug. 1988.

[3] V. Jacobson, "Modified TCP Congestion Avoidance Algorithm," Technical report, Apr. 1990.

[4] S. Floyd, T. Henderson, and A. Gurtov, “The NewReno Modification to TCP’s Fast Recovery Algorithm,” RFC 3782, April 2004.

[5] W. Stevens, "TCP Slow Start, Congestion Avoidance, Fast Retransmit, and Fast Recovery Algorithms," *IETF RFC 2001*, 1997.

[6] D. Chiu and R. Jain, "Analysis of the Increase and Decrease Algorithms for Congestion Avoidance in Computer Networks," *Computer Networks and ISDN Systems*, vol.1, pp. 1-14, 1989.

[7] M. Mathis, J. Mahdavi, S. Floyd, and A. Romanow, "TCP Selective Acknowledgement Options," *IETF RFC 2018*, 1996.

[8] K. Fall and S. Floyd, "Simulation-based Comparisons of Tahoe, Reno, and SACK TCP," *ACM Computer Communication Review*, vol. 26, no.3, pp. 5-21, 1996.

[9] Sally Floyd, "TCP and Explicit Congestion Notification," *ACM Computer Communication Review*, 1994.

[10] S. Floyd and V. Jacobson, "Random Early Detection Gateways for Congestion Avoidance," *IEEE/ACM Transaction on Networking*, vol.1, no.4, pp. 397-413, 1993.

[11] L. S. Brakmo, S. W. O'Malley, and Larry L. Peterson. "TCP Vegas: New Techniques for Congestion Detection and Avoidance, " *Proc. Of ACM SIGCOMM*, pp. 24-35, Aug. 1994.

[12] L. S. Brakmo and L. L. Peterson. "TCP Vegas: End to End Congestion Avoidance on a Global Internet," *IEEE Journal on Selected Areas in Communication*, vol.13, no.8, pp. 1465-1480, Oct. 1995.

[13] IEEE Std. 802.11, "Wireless LAN Media Access Control (MAC) and Physical Layer (PHY) Specifications," 1999.

[14] M. Gerla, K. Tang, and R. Bagrodia, "TCP performance in wireless multi-hop networks," in *Proceedings of IEEE WMCSA'99*, February 1999.

[15] C. Barakat, E. Altman, and W. Dabbous, "On TCP Performance in a Heterogeneous Network: A Survey," *IEEE Communications Magazine*, vol.38, no.1, pp. 44-46, Jan. 2000.

[16] R. Denda, A. Banchs and W. Effelsberg, "The Fairness Challenge in Computer Networks," *Proc. of Quality of future Internet Service*, vol. 1922, pp. 208-220, Springer-Verlag Heidelberg, Jun. 2000.

[17] Yi-Cheng Chan, Chia-Tai Chan, and Yaw-Chung Chen, "RoVegas : A Router-based Congestion Avoidance Mechanism for TCP Vegas," *Computer Communications*, vol.27, Issue 16, pp. 1624-1636, Oct. 2004.

[18] Y. Tian, K. Xu, N. Ansari, "TCP in Wireless Environment: Problems and Solution," *IEEE Communications*, vol 43. no.3, 2005.

[19] S. Ryu, C. Rump, C. Qiao, "Advances in Internet Congestion Control," *IEEE Communications Surveys and Tutorials*, vol 5. no.2, 2003.

[20] A. Hanbali, E. Altman, P. Philippe, "A Survey of TCP over Ad Hoc Networks," *IEEE*

Communications Surveys and Tutorials, vol. 7. no.3, 2005.

[21] W. Stevens, "TCP/IP Illustrated, Volume 1: The Protocols," Addison-Wesley,

[22] C.P.Fu, S.C. Liew, "TCP Veno: TCP enhancement for transmission over wireless access networks", *IEEE Journal on Selected Areas in Communications*, vol. 21, no. 2, Feb 2003

[23] K. Xu, Y. Tian, N. Ansari, "TCP-Jersey for wireless IP communications", *IEEE Journal on Selected Areas in Communications*, vol. 22, no. 4, May 2004.

[24] M. Gerla, M. Y. Sanadidi, R. Wang, A. Zanella, C. Casetti, S. Mascolo, "TCP Westwood: Congestion window control using bandwidth estimation", *GLOBECOM 2001 - IEEE Global Telecommunications Conference*, no. 1, Nov 2001.

[25] UCB/LBNL/VINT. The network simulator – ns-2. http://www.isi.edu/nsnam/ns/

[26] G. Holland and N. Vaidya, "Analysis of TCP performance over mobile ad hoc networks," *Proc. ACM Mobicom '99*, Seattle, WA, 1999.

[27] X. Yu, "Improving TCP performance over mobile ad hoc networks by exploiting cross-layer information awareness," in *Proc. ACM MobiCom '04*, Philadelphia, PA, September, 2004.

[28] J. Liu and S. Singh, "ATCP: TCP for mobile ad hoc networks," *IEEE J-SAC*, vol. 19, no. 7, pp. 1300–1315, 2001.

[29] Z. Fu, P. Zerfos, H. Luo, S. Lu, L. Zhang, and M. Gerla, "The impact of multihop wireless channel on TCP throughput and loss," in *Proc. IEEE INFOCOM*, San Francisco, CA, March 2003.

[30] K. Nahm, A. Helmy, and C.-C J. Kuo, "TCP over Multihop 802.11 Networks: Issues and Performance Enhancement," in *Proc. ACM MobiHoc '05*, Urbana-Champaign, Illinois, USA, May 2005

[31] S. Xu and T. Saadawi, “Does the IEEE 802.11 MAC protocol work well in multihop wireless ad hoc networks?” *IEEE Communications Magazine*, pp. 130–137, June 2001.

[32] K. Chen and K. Nahrstedt, “Limitations of equation-based congestion control in mobile ad hoc networks,” in *Proc. IEEE WWAN*, Tokyo, Japan, March 2003.

[33] Z. Fu, X. Meng, and S. Lu, “How bad TCP can perform in mobile ad-hoc networks,” in *Proc. IEEE ICNP ’02*, Paris, France, 2002.

[34] K. Chen, Y. Xue, and K. Nahrstedt, “On setting TCP’s congestion window limit in mobile ad hoc networks,” in *Proc. IEEE ICC 2003*, Anchorage, Alaska, May, 2003.

[35] K. Sundaresan, V. Anantharaman, H. Hsieh, and R. Sivakumar, “ATP: a reliable transport protocol for ad-hoc networks,” in *Proc. ACM MobiHoc ’03*, Annapolis, ML, 2003.

[36] C. Perkins, E. Belding-Royer, and S. Das, “Ad hoc on-demand distance vector (AODV) routing,” IETF Internet Draft, draft-ietf-manet-aodv-13.txt, February 2003.

[37] J. Monks, P. Sinha, and V. Bharghavan, “Limitations of TCP-ELFN for ad hoc networks,” in *Proc. IEEE MOMUC ’00*, Tokyo, Japan, 2000.

[38] R. Jain, D-M. Chiu, W. Hawe, “A Quantitative Measure of Fairness and Discrimination for Resource Allocation in Shared Computer Systems,” Technical Report TR-301, DEC Research Report, Sept. 1984

[39] F. Wang and Y. Zhang, ”Improving TCP Performance over Mobile Ad-Hoc Networks with Out-of-Order Detection and Response,” *Proc. Third ACM Int’l Symp. Mobile Ad Hoc Networking & Computing* , 2002

[40] T. Dyer and R. Boppana, “A comparison of TCP Performance over three routing protocols for mobile ad hoc networks,” in *Proc. ACM MobiHoc ’02*, Lausanne, Switzerland, June, 2002.

[41] Z. Fu and B. Greenstein, “Design and Implementation of a TCP-Friendly Transport

Protocol Ad Hoc Wireless Networks," *IEEE International Conference on Network Protocols (ICNP'02)*, Paris, France, November 2002.

[42] J. Mo, R. J. La, V. Anantharam, and J. Walrand, "Analysis and comparison of TCP Reno and Vegas," in *Proc. IEEE INFOCOM '99*, pp. 1556-1563, March 1999.

[43] Y. Lai, "Improving the performance of TCP Vegas in heterogeneous environment," in *Proc. Int. Conf. Parallel and Distributed Systems*, pp. 581-587, June 2001.

[44] W. Feng and S. Vanichpun, "Enabling compatibility between TCP Reno and TCP Vegas," in Proc. IEEE Symposium on Application and the Internet, pp. 301-308, January 2003.