Enhancements of congestion controls for unicast and multicast over wireless networks

Zhang, Ke

2009

Zhang, K. (2009). Enhancements of congestion controls for unicast and multicast over wireless networks. Doctoral thesis, Nanyang Technological University, Singapore.

<https://hdl.handle.net/10356/42242>

<https://doi.org/10.32657/10356/42242>

Enhancements of Congestion Controls for Unicast and Multicast over Wireless Networks

ZHANG KE

SCHOOL OF COMPUTER ENGINEERING

A thesis submitted to the Nanyang Technological University
in fulfillment of the requirement for the degree of
DOCTOR OF PHILOSOPHY

2009

Abstract

Transmission Control Protocol (TCP) is a connection-oriented, reliable and in-order transport layer protocol that serves most of the data transfers in the current Internet. However, running TCP over wireless networks is challenged by the problem of high bit error rate. The current legacy TCP, namely TCP Reno, is originally designed for the wired networks where packet loss is an indication of network congestion. TCP Reno slows down the sending rate in response to the packet loss in order to alleviate network congestion. But in the wireless networks, packet loss is often induced by high bit error rate (referred to as random loss) rather than congestion. Misinterpretation of random losses as congestion causes TCP Reno to reduce its sending rate unnecessarily, resulting in significant performance degradation.

Such problem also exists in TCP-friendly multicast congestion control protocols, such as TCP-Friendly Multicast Congestion Control (TFMCC). TFMCC is a single-rate multicast protocol that makes use of TCP Reno's throughput equation to adjust its sending rate, and provide TCP-like long-term throughput for achieving TCP-friendliness. Moreover, TFMCC's sending rate is shaped to be smooth and responsive to the changes in network conditions, so that TFMCC is suitable for serving streaming multimedia applications such as IP TV. However, due to the nature of TCP Reno in lossy environment, TFMCC, equipped with the Reno equation, will inevitably suffer performance degradation from wireless links.

This thesis studies the fundamental issue - congestion control - which is embedded in TCP and multicast. There are two parts; in the first part, we study TCP congestion control over wireless networks by focusing on TCP Veno – a novel TCP enhancement for eliminating TCP suffering in the wireless networks. Specifically, 1) we study TCP Veno's loss-distinguishing accuracy between random and congestion losses under different

network conditions, and how Veno's performance is affected by such loss-distinguishing accuracy; 2) we study the dynamics of TCP Veno with RED queue management by deriving a math model, and then provide theoretical guidelines of tuning RED parameters to stabilize router queue oscillation; 3) we also investigate the limitations of traditional slow-start, and propose an enhanced startup algorithm to improve Veno's startup performance; 4) we propose and evaluate a new scheme of adding Forward ACK into Veno to improve its recovery efficiency in heavy loss environment.

In the second part, we propose to enhance TFMCC by making use of TCP Veno's throughput formula. This formula is derived from TCP Veno's window evolution. We use it to replace TCP Reno's formula in TFMCC to control the sending rate. Extensive simulations results validate that the new TFMCC scheme achieves significant performance improvement in the wireless networks. Meanwhile, the performance improvement is obtained by making better use of the bandwidth wasted by other connections rather than aggressiveness, so that the nice features in original TFMCC, namely, TCP-friendliness, smoothness, responsiveness, are fully inherited. Moreover, our scheme only needs modification on the end-node without any changes of intermediate router stack, making it easy to deploy in the current Internet.

Acknowledgments

First of all, I would like to send my best appreciations to Dr. Fu Cheng Peng, Franklin, my supervisor, for taking me as his student and leading me to the area of research. I would like to thank for his guidance and patience for showing me the way of doing research, also the weekly meeting where we practice writing skills, presenting skills, and discuss research ideas. I am so lucky to have Dr. Fu as my supervisor. His knowledge, idea, and patient support in every area help me and inspire me through the study. I have greatly benefited from him. Thank you very much! Dr. Fu!

I am also very grateful to Dr. Foh Chuan Heng. His excellent knowledge in mathematics and invaluable suggestions help me a lot in the study. Without his help, I cannot finish the math model derivations in this thesis.

I would like to thank my group partners Zhou Bin, Yu Kuan, Zhang Chunlei, and LV Qian. They gave me a lot of help. I benefit much from their discussions and novel ideas. I would also like to thank the lab colleagues and technicians, for their assistant in research facilities and experiments.

This thesis is dedicated to my family. At this moment, words cannot express my deepest thanks to you: my parents, and my grandmother. I know that these years is a long long way that I cannot stay with you. But you are always keeping close to my heart: every minute and every second. What I want to tell you is that, every success and progress I made in my life, it is for you!

Contents

Abstract	i
Acknowledgments	iii
List of Figures	viii
List of Tables	xii
1 Introduction	1
1.1 Internet and TCP/IP	1
1.2 Problem Statement and Motivation	5
1.2.1 Challenges for TCP Congestion Control	5
1.2.2 Challenges for TCP-Friendly Multicast Congestion Control	7
1.3 Summary of Contributions	7
1.3.1 Contributions for Unicast Congestion Control	8
1.3.2 Contributions for Multicast Congestion Control	9
1.4 Structure of the Thesis	10
2 Literature Review	12
2.1 TCP Congestion Control Schemes	12
2.1.1 TCP Enhancements for Congestion Avoidance Schemes	13
2.1.2 TCP Enhancements for Startup Schemes	24
2.1.3 TCP Enhancements for Recovery Schemes	27
2.2 TCP-Friendly Multicast Congestion Control Schemes	29
2.2.1 Window-based Congestion Control Schemes	29
2.2.2 Rate-based Congestion Control Schemes	30

- 3 A Sender-side Enhancement for TCP Startup 32**
 - 3.1 Introduction 32
 - 3.2 Traditional slow-start limitations 33
 - 3.3 Mechanism of Enhanced startup algorithm 34
 - 3.4 Performance evaluation 37
 - 3.5 Summary 39
- 4 An Enhancement of TCP Veno with Forward Acknowledgement 40**
 - 4.1 Introduction 40
 - 4.2 Mechanism of TCP Veno-F 41
 - 4.3 Performance evaluation in different network scenarios 46
 - 4.3.1 Wireless scenario 46
 - 4.3.2 Cross traffic scenario 48
 - 4.3.3 Reverse traffic and multi-hop scenario 49
 - 4.3.4 A comprehensive scenario 50
 - 4.4 Fairness and compatibility 51
 - 4.4.1 Intra-protocol fairness 51
 - 4.4.2 Inter-protocol fairness 53
 - 4.5 Summary 55
- 5 Loss Distinguishing Accuracy in TCP Veno and its Performance Influence 57**
 - 5.1 Introduction 57
 - 5.2 Loss distinguishing accuracy 58
 - 5.2.1 Simulation topology 60
 - 5.2.2 Impact of random loss rate. 61
 - 5.2.3 Impact of cross traffic. 62
 - 5.2.4 Impact of buffer size. 63
 - 5.2.5 Impact of delay. 64
 - 5.2.6 Impact of reverse traffic and β 65
 - 5.3 The performance influence 66
 - 5.3.1 Wireless scenario. 67

5.3.2	Cross and reverse traffic scenario	69
5.3.3	Compatibility with NewReno.	71
5.4	Summary	73
6	Theoretic Analysis of TCP Veno with RED	75
6.1	Introduction	75
6.2	Fluid-flow model of TCP Veno with RED	75
6.2.1	Network and RED queue management model	76
6.2.2	Modeling of TCP Veno	78
6.3	Theoretic analysis of the fluid-flow model	80
6.3.1	Linearization and analysis	81
6.3.2	Analysis of relative stability	84
6.4	Simulation and validation	86
6.4.1	Queue dynamics of TCP Veno with RED	87
6.4.2	Effect of RED parameters	89
6.4.3	Effect of network parameters	91
6.5	Summary	93
7	An Enhancement of TCP-Friendly Multicast Congestion Control over Wireless Networks	94
7.1	Throughput Model of TCP Veno	95
7.1.1	TCP Reno model	95
7.1.2	Modeling of TCP Veno	97
7.1.3	Modeling of TCP Veno with timeouts	100
7.1.4	TCP Veno and TCP Reno	100
7.1.5	Model validation	101
7.1.6	Real network experiments	103
7.2	TFMCC Veno and its Performance Evaluation	106
7.2.1	Mechanism of TFMCC Veno	106
7.2.2	Performance evaluation of TFMCC Veno	107
7.3	Summary	114

8	Conclusions and Future Work	115
8.1	Conclusions	115
8.2	Future work	119
	References	122
	Publications	130

List of Figures

1.1	Transitions between different states in TCP Reno.	4
2.1	Difference between expected and actual throughput.	13
2.2	Biaz loss distinguishing scheme.	21
2.3	Packet marking probability in CA.	23
3.1	Network simulation topology.	33
3.2	Comparison of slow-start and Enhanced Start <i>cwnd</i> evolutions.	34
3.3	Enhanced Start <i>cwnd</i> evolution under different values of β	35
3.4	Enhanced Start <i>cwnd</i> evolution under burst UDP cross-traffic (10Mbps starts at 1 sec and stops at 5 sec).	35
3.5	NewReno (NR) throughput versus bottleneck bandwidth (first 20 s).	36
3.6	NewReno (NR) throughput versus delay (first 20 s).	37
3.7	NewReno (NR) throughput versus buffer size (first 20 s).	38
3.8	Co-existence of multiple Enhanced Start and slow-start connections.	38
4.1	Simulation topology: the single-bottleneck topology	41
4.2	Simulation topology: the multi-bottleneck topology	44
4.3	TCP Veno-F in wired network.	44
4.4	TCP Veno-F in wireless network.	44
4.5	TCP Veno-F recovery phase.	45
4.6	Fack TCP recovery phase.	45
4.7	Impact of random loss rate	47
4.8	Impact of cross traffic	49
4.9	Impact of reverse traffic and multi-hop	50

4.10	The goodput comparison in the comprehensive scenario.	51
4.11	Low bottleneck bandwidth.	52
4.12	High bottleneck bandwidth.	52
4.13	Background traffics: Veno-F.	53
4.14	Background traffics: Fack TCP.	53
4.15	HyblaS with Fack TCP.	55
4.16	Veno-F with Fack TCP.	55
5.1	Simulation topology.	60
5.2	Impact of random loss rate.	61
5.3	Impact of cross traffic.	61
5.4	Impact of buffer size.	63
5.5	Impact of propagation delay.	63
5.6	Impact of Reverse traffic.	65
5.7	Impact of β	65
5.8	Goodputs in wireless scenario.	67
5.9	Window evolution illustration.	67
5.10	Multiple flows goodputs.	69
5.11	Multiple flows window.	69
5.12	NewVeno, NewReno and Westwood+ in cross and reverse traffic scenario.	70
5.13	NewVeno and NewReno goodputs in the compatibility scenario.	71
5.14	NewVeno and NewReno window evolution when the cross traffic is 80%.	71
5.15	Multiple connections in the wired scenario.	72
5.16	Multiple connections' window evolution when the cross traffic is 80%.	72
6.1	The single bottleneck topology.	79
6.2	Block-diagram of TCP Veno/RED feedback system.	83
6.3	Instantaneous queue of TCP Veno with RED (loss 1% and $n = 60$), case 1.	87
6.4	Instantaneous queue of TCP Veno with RED, case 2.	88
6.5	Instantaneous queue of TCP Veno with RED, case 3.	89
6.6	CoV of TFRC in three cases.	89
6.7	Throughput of TFRC in three cases.	90

6.8 Effect of increasing random loss rate. 91

6.9 Effect of increasing delay. 91

6.10 Effect of increasing bandwidth. 92

6.11 Effect of decreasing flow number. 92

7.1 Evolution of $N(t)$ 98

7.2 Settings of m and n for VenO for equal bandwidth sharing between VenO and Reno. 101

7.3 The experiment network topology with different values of delay. 102

7.4 Single flow varying delay. 102

7.5 Single flow varying background traffic. 102

7.6 Single flow varying the values of m and n 103

7.7 Mixed multiple flows. 103

7.8 Real network topology. 104

7.9 Results on the wired network. 105

7.10 Dummynet case. 105

7.11 Wireless LAN. 105

7.12 TCP VenO co-existing with TCP Reno flows. 105

7.13 TFMCC VenO mechanism illustration. 107

7.14 Throughput improvement of TFMCC VenO over TFMCC in the wireless network. 108

7.15 Performance evaluation of TFMCC and TFMCC VenO over two-state distribution loss model. 108

7.16 Coefficient of Variation of TFMCC and TFMCC flows in the wireless network. 109

7.17 Coefficient of Variation of TFMCC and TFMCC VenO flows with different timescale. 109

7.18 Sequence number against time, one TFMCC VenO flow co-exists with four SACK TCP flows, loss rate 0.01. 110

7.19 Sequence number against time, five SACK TCP flows, loss rate 0.01. . . 110

7.20 Equivalence ratios of TFMCC and TFMCC VenO with SACK TCP, wired network. 111

7.21 Equivalence ratios of TFMCC and TFMCC Veno under different timescale.112

7.22 Responsiveness of TFMCC Veno’s transmission rate, heterogeneous delay
and SACK TCP flows join at different time. 113

7.23 Responsiveness of TFMCC Veno’s transmission rate, heterogeneous delay
and TFMCC Veno joins at different time. 113

8.1 Architecture of the Uniform transport platform. 121

List of Tables

5.1	Data collection in the wireless scenario	67
5.2	Data collection in cross and reverse traffic scenario	70
5.3	Data collection in the compatibility scenario	71
8.1	Summary of TCP-friendly multicast congestion control schemes	118

Chapter 1

Introduction

1.1 Internet and TCP/IP

Transmission Control Protocol/Internet Protocol (TCP/IP) stack is the foundation of the modern Internet. The first reference to TCP was in a 1973 review note, entitled “A Partial Specification of an Internet Transmission Protocol” by Dr. Vinton G. Cerf to a few colleagues. The concept of layering was then not very clear. Most of the networking tasks were handled by Network Control Protocol (NCP). These were the early days of networking.

In 1974, Dr. Vinton G. Cerf and Dr. Robert Kahn published a paper entitled “A Protocol for Packet Network Interconnection” [1]. They provided a detailed discussion of several choices in TCP design. Afterwards, the first official specification of TCP was written by Vint Cerf, Yogen Dalal, and Carl Sunshine in [2]. The second version of TCP specification was written by Cerf in March 1977 [3]. Two months later, Jon Postel observed that Internetwork communication should be viewed as having two components: the hop-by-hop relaying of messages and the end-to-end control of the conversation [4]. In January of 1978, Cerf and Postel split TCP into two protocols: TCP and IP [5]. IP specified routing packets while TCP handled packetization, error control, retransmissions, and reassembly. They argued that simplicity of IP would allow fast and inexpensive gateways

CHAPTER 1. INTRODUCTION

to be built. Postel then wrote a new specification for IP and TCP. The specification was discussed in several IEN meetings and finally published in the reference [6].

In February of 1980, the U.S. Department of Defense (DoD) adopted TCP/IP as the preferred protocol. DoD required that every site connected to Advanced Research Projects Agency (ARPA) net should switch to TCP/IP by 1983. In 1984, the network of networks so connected was later split into MILNET for military-related sites and the regular Internet for other sites.

In the 1980s, there were many other networking protocol stacks that competed with TCP/IP. For example, Digital Equipment Corporation (DEC) net, Systems Network Architecture (SNA), AppleTalk, Xerox Network Systems (XNS). Most of these protocols were vendor-specific and proprietary. It was hoped that someday they would all be replaced by a single standard protocol designed by the International Standards Organization (ISO). However, it took too long and became too complicated. By this time, TCP/IP had been widely adopted because of its simplicity. Not only were its protocol specifications openly available, but the source code was available in open way. Meanwhile, the implementation of TCP/IP in Berkeley UNIX 4.2 BSD distribution in 1983 also speed up the wide adoption of TCP/IP.

However, in 1988, Internet suffered the problem of congestion collapse. Jacobson observed this phenomenon and proposed several innovative congestion control mechanisms to enhance TCP [7]. This TCP version was known as TCP Tahoe. Two years later, a modified version, called TCP Reno, was proposed [8]. TCP Reno is currently the dominant TCP version deployed in the Internet. Roughly speaking, TCP Reno is mainly composed of three parts [9, 10, 11, 12, 13]:

- **Slow-start.** TCP Reno maintains two variables: *ssthresh* and *cwnd*. *ssthresh* is slow-start threshold, while *cwnd* (congestion window) indicates the number of segments allowed to transmit by the sender. In slow-start, the sender starts by trans-

CHAPTER 1. INTRODUCTION

mitting one segment and waits until ACK is received. Then, *cwnd* is incremented to two, indicating two segments can be sent. After both of them are acknowledged, *cwnd* is increased to four. This provides an exponential growth of *cwnd*, until the value of *ssthresh* is reached. Thereafter, the state changes to congestion avoidance. The pseudo-code of slow-start is as follows.

Listing 1.1: Slow-start algorithm

```
for every new ACK received,
    cwnd = cwnd + 1. //It also means that cwnd is doubled
    every RTT
```

- **Congestion avoidance.** In congestion avoidance phase, TCP Reno performs in a conservative manner to probe available bandwidth, while prevent severe congestions. Specifically, on receipt of every new ACK, *cwnd* is increased by $1/cwnd$, while on receipt of three duplicate ACKs, *cwnd* is dropped by an amount of $1/2$ and the state changes to recovery phase at once. This window evolution pattern is called Additive Increase and Multiplicative Decrease (AIMD) [14]. The pseudo-code is as follows.

Listing 1.2: Additive Increase and Multiplicative Decrease algorithm

```
for every new ACK received,
    cwnd = cwnd + 1/cwnd,
otherwise, if loss is detected by three duplicate ACKs,
    ssthresh = cwnd/2,
    cwnd = ssthresh.
```

- **Fast retransmit and Fast recovery.** When three duplicate ACKs are received, the lost segment is deduced and retransmission is carried out immediately. Thereafter, *cwnd* is set to *ssthresh* + 3, and for every following duplicate ACK received, *cwnd* is increased by one. New segment is transmitted whenever *cwnd* is bigger than the number of unacknowledged segments. When the next ACK arrives that

Figure 1.1: Transitions between different states in TCP Reno.

acknowledges new data, $cwnd$ is reduced to $ssthresh$, and the state returns to congestion avoidance. The pseudo-code is as follows.

Listing 1.3: Fast retransmit and fast recovery algorithm

```
for three dup ACKs received,
    cwnd = ssthresh + 3,
for every following dup ACK received,
    cwnd = cwnd + 1,
for ACK of new data received,
    cwnd = ssthresh.
```

Fig. 1.1 shows the transitions between different states in TCP Reno. Note that, ISS means the initial slow-start phase. Besides packet loss retransmissions, TCP Reno also employs a mechanism for timeout occurrence. If packet loss is identified by timeout, TCP Reno sets $ssthresh = cwnd/2$ and $cwnd = 1$, and slow-start phase takes over.

TCP Reno serves most of the bulk data transfers such as FTP, Email [15]. But, TCP is not suitable for streaming applications like IP TV, Video Conference, Voice, and Internet games. TCP-friendly multicast congestion control protocols are more suitable for these applications. A number of works [16, 17, 18, 19, 20] have been proposed to

tackle this issue. For instance, TFMCC[17], one rate-based multicast congestion control scheme, makes use of TCP Reno's throughput equation to adjust its sending rate, so as to provide TCP-like long-term throughput for achieving TCP-friendliness. Moreover, TFMCC's sending rate is shaped to be smooth and responsive to changes in network conditions, so that TFMCC is suitable for streaming multimedia applications.

1.2 Problem Statement and Motivation

In recent years, new communication technologies and infrastructures are incorporated into Internet. For instance, wireless communication technology has been making significant progress in the recent past and will be playing a more and more important role in future Internet. Meanwhile, new communication technologies are also bringing new challenges for Internet congestion controls, as such in unicast and multicast traffic.

1.2.1 Challenges for TCP Congestion Control

The challenges for TCP congestion control can be summarized in three aspects.

- **Congestion avoidance.** The current legacy TCP, namely TCP Reno, is originally designed for the wired networks. In the wired networks, packet loss is often caused by router buffer overflow. Therefore, TCP Reno takes packet loss as the indication of network congestion. In response to the packet loss, TCP Reno's AIMD algorithm halves its congestion window for slowing down the sending rate, in order to alleviate network congestions and avoid congestion collapse. TCP Reno works efficiently in the wired networks. However, in the wireless networks, packet loss is often induced by high bit error rate or noise on wireless links, rather than congestion. We refer to such packet loss as random loss. TCP Reno's AIMD algorithm misinterprets the random loss as congestion signal, and thus reduces its sending rate unnecessarily, resulting in severe performance degradation.

CHAPTER 1. INTRODUCTION

- **Slow-start.** In traditional slow-start, sender increases congestion window by one segment upon receiving every new ACK, until congestion window reaches initial slow-start threshold. However, initial *ssthresh* is set before the TCP connection starts, and its value is arbitrary, ranging from 4KB to extremely high. Due to blind initial *ssthresh* setting, TCP suffers low startup performance, especially when the network has high values in bandwidth and delay. There are two typical cases where initial *ssthresh* mismatches the network bandwidth-delay product(BDP). The first case is that *ssthresh* is set much higher than the value of BDP. In this case, TCP connection suffers many packets losses and long recovery time. The second case is that *ssthresh* is set much lower than the value of BDP. In this case, TCP connection exits slow-start and switches to congestion avoidance phase prematurely. Both cases cause quite low utilization in link capacity.
- **Fast retransmit and fast recovery.** In traditional fast retransmit and fast recovery algorithm, the recovery phase is invoked by three duplicated ACKs, and terminated by the ACK for new segments. This means each invocation of the recovery phase leads to the retransmission of only a single data packet. Problems can arise, therefore, when multiple packets have been dropped from a single window of data, which may happen in burst loss and heavy random loss environment. In such case, recovery phase will be invoked multiple times, and thus causes congestion window to be reduced multiple times or even time out. This also causes performance degradation for TCP Reno.

A number of enhanced TCP schemes have been proposed to improve TCP Reno's performance. For instance, TCP Vegas [21], TCP Veno [22], TCP Westwood [23], High Speed TCP (HS TCP) [24], Jitter-Based TCP (JTCP) [35], Biaz scheme [36], and TCP Hybla [37] use end-to-end mechanisms to enhance Reno's AIMD algorithm, while Ad Hoc

CHAPTER 1. INTRODUCTION

TCP (ATCP) [25], TCP Jersey [26] use router-support information to distinguish random loss from congestion loss. TCP NewReno [27], Selective Acknowledgement (SACK) TCP [28] and Forward Acknowledgement (FACK) TCP [29] aim to improve TCP's recovery efficiency, while Hoe's modification [38], Agile Probing [30], Limited slow-start [31], and TCP Peach [39] aim to improve TCP's startup performance.

1.2.2 Challenges for TCP-Friendly Multicast Congestion Control

Similar to TCP, TCP-friendly multicast congestion control schemes also suffer performance degradation in the wireless networks. For instance, TFMCC makes use of TCP Reno's throughput formula to control its sending rate, and provide TCP-like long-term throughput to achieve TCP friendliness. However, due to the nature of TCP Reno in lossy environment, TFMCC, equipped with the Reno equation, will inevitably suffer performance degradation from wireless links.

Different from TFMCC's rate-based mechanism, some other TCP-friendly multicast congestion controls are based on TCP-like AIMD window evolutions. Multicast TCP (MTCP) [19] directly extends TCP congestion control mechanism to multicast mode, by employing intermediate routers for duplicating data packets from one sender to multiple receivers. Pragmatic General Multicast Congestion Control (PGMCC) [20] is an end-to-end multicast scheme that also makes use of TCP-like window evolution mechanism. However, it separates reliable transmission from congestion control mechanisms. These performance suffering in the wireless networks is also severe as such in TCP, which cannot distinguish random loss from congestion loss.

1.3 Summary of Contributions

The contributions of this thesis are summarized in two parts: one is for unicast congestion control and another is for multicast congestion control.

1.3.1 Contributions for Unicast Congestion Control

- **Loss Distinguishing Accuracy in TCP Veno and its Performance Influence.** TCP Veno makes use of a sender-side congestion monitoring scheme to distinguishing random loss from congestion loss. We study TCP Veno's loss distinguishing accuracy with different loss rate, different link delay as well as different cross traffic. Based on the analytical and experimental results, we observe that in most situations, such loss distinguishing accuracy is quite high, but in some specific situations, it is very low, especially, when there is heavy cross traffic competing with TCP Veno. The further study points out that the heavy cross traffic leads to more timeout events for a TCP connection and it is this increase of timeout events that make TCP Veno perform harmoniously with existing legacy TCP, different from our intuition that TCP Veno could have become quite aggressive due to its low loss distinguishing accuracy.
- **Theoretic Analysis of TCP Veno with RED.** We study the dynamics of TCP Veno with the queue management of RED (Random Early Detection) [32]. We develop a fluid-flow model to describe the behaviors of TCP Veno in wired/wireless networks. This model is further linearized to study TCP Veno's stability issue through the linear feedback control theory. The analysis points out how the RED queue oscillates under different network parameters such as link capacity, round-trip time. Simulations validate our theoretical analysis. Furthermore, these analysis results can provide guidelines on how to tune RED parameters and improve the co-existence between TCP Veno and TFRC (TCP-friendly Rate Control) [33, 34] flows.
- **A Sender-side Enhancement for TCP Startup.** TCP Veno only modifies TCP Reno's AIMD algorithm, but traditional slow-start remains unchanged. We

CHAPTER 1. INTRODUCTION

demonstrate that, TCP slow-start algorithm suffers low performance in high-speed long-delay networks. We present a sender-side enhancement that is simple and efficient to improve TCP startup performance in high-speed long-delay networks. The key idea is to make use of TCP Vegas congestion-monitoring scheme to track the router queue, and help slow-start window evolution to quickly reach the proper window size. Simulation results demonstrate that, this idea can significantly improve link utilization without affecting co-existing TCP connections adversely.

- **An Enhancement of TCP Veno with Forward Acknowledgement.** TCP Reno's fast retransmit and fast recovery algorithm also remains unchanged in TCP Veno. Therefore, TCP Veno suffers low performance in heavy loss environment. We study the enhancement of TCP Veno with Forward ACK (FACK), called TCP Veno-F. TCP Veno-F aims to improve Veno's performance through better recovery efficiency. We take Fack TCP as a legacy TCP, and compare TCP Veno-F, Fack TCP, and other TCP enhancements over a variety of network conditions. Our network simulation demonstrates that, 1) TCP Veno-F improves TCP performance by better utilization of link bandwidth; 2) TCP Veno-F can co-exist harmoniously with legacy TCP in the wired networks; 3) TCP Veno-F deals with a large range of network scenarios, i.e., heterogeneous networks, RED queue management.

1.3.2 Contributions for Multicast Congestion Control

- **An Enhancement of TFMCC over Wireless Networks.** We propose and study the idea of enhancing TFMCC with an advanced throughput formula equipped. This work is mainly divided into two steps. Firstly, a generalized formula is derived from Veno's behaviors, considering its window evolution parameters, loss rate and round-trip time. Simulation experiments and real network measurements are conducted to validate the accuracy of this model. Secondly, we use TCP Veno's

throughput formula to replace TCP Reno's formula in TFMCC. This enhanced version is referred to as TFMCC VenO. Extensive simulations are carried to evaluate its performance over a large range of network environments. The results demonstrate that, 1) TFMCC VenO achieves significant performance improvement in the wireless networks. Specifically, up to 350% performance enhancement, compared with original TFMCC, is obtained in the environment of 10% random loss rate. 2) The performance improvement is obtained by making better use of the bandwidth, so that the nice features in original TFMCC, namely, TCP-friendliness, smoothness, responsiveness, are fully inherited. Moreover, our scheme only needs modification on the end-node without any changes of intermediate router stack, making it easy to deploy in the current Internet.

1.4 Structure of the Thesis

The structure of this thesis is as follows. In Chapter 2, we review the related works on TCP and TCP-friendly multicast congestion control schemes. In the part of TCP congestion controls, we have three sub-parts: TCP enhancements for congestion avoidance schemes, TCP enhancements for recovery schemes, and TCP enhancements for startup schemes. In the part of TCP-friendly multicast congestion controls, window-based schemes and rate-based schemes are discussed.

In Chapter 3, we study the enhancement of TCP startup algorithm. Chapter 4 studies the enhancement of TCP VenO with Forward Acknowledgement. Chapter 5 studies the loss distinguishing accuracy in TCP VenO and its performance influence. Chapter 6 studies dynamics of TCP VenO with RED.

In Chapter 7, we study the enhancement of TFMCC over wireless networks. This chapter has two sections. A long-term throughput formula for TCP VenO is derived

CHAPTER 1. INTRODUCTION

in Section 7.1, and validation is also given. Section 7.2 introduces the mechanism of TFMCC Veno, and evaluates its performance under different network conditions.

Chapter 8 concludes this thesis and discusses some works in future.

Chapter 2

Literature Review

In this chapter, we review the related works on TCP and TCP-friendly multicast congestion control schemes. For TCP congestion controls, we divide them into three categories: TCP enhancements for congestion avoidance schemes, TCP enhancements for recovery schemes, and TCP enhancements for startup schemes. For TCP-friendly multicast congestion controls, we divide them into two categories: window-based schemes and rate-based schemes.

2.1 TCP Congestion Control Schemes

TCP is the dominant transport protocol in current Internet. The modern TCP congestion control mechanism is based on TCP Reno, which is mainly composed of three components: slow-start, congestion avoidance, and retransmit and recovery. In recent years, many schemes are proposed for enhancing TCP Reno's performance over various network environments, such as TCP Vegas [21], TCP Veno [22], TCP Westwood [23], JTCP [35], HS TCP [24], ATCP [25], Biaz scheme [36], TCP Jersey [26], TCP Hybla [37], Hoe's modification [38], Agile Probing [30], Limited Slow-Start [31], TCP Peach [39], TCP NewReno [27], TCP SACK [28], TCP FACK [29]. These schemes are divided into three categories.

CHAPTER 2. LITERATURE REVIEW

Figure 2.1: Difference between expected and actual throughput.

2.1.1 TCP Enhancements for Congestion Avoidance Schemes

- TCP Vegas

TCP Vegas [21] is an end-to-end TCP enhancement. Unlike TCP Reno that is based on packet loss, TCP Vegas monitors network state by tracking round-trip time. Specifically, TCP Vegas calculates the difference between expected throughput and actual throughput:

$$Diff = (Expected - Actual)$$

where, $Expected = cwnd/BaseRTT$ and $Actual = cwnd/RTT$. $BaseRTT$ is the minimum of the round-trip time, and RTT is the actual round-trip time of a tagged packet. $cwnd$ is TCP congestion window.

Intuitively, as TCP's *cwnd* increases, its throughput should also increase. When the network capacity has not been fully utilized yet, the actual throughput should be close to the expected throughput; otherwise, the actual throughput will be smaller than the expected throughput due to buffer queue buildup in intermediate routers. This difference is shown in Fig. 2.1. As a result, Vegas makes use of the difference in throughput as an information for adjusting *cwnd*. If the difference exceeds the up bound (β), the network is deemed as in congestive state where network capacity has been fully utilized.

CHAPTER 2. LITERATURE REVIEW

Vegas decreases $cwnd$ to avoid congestion. If the difference is lower than the low bound (α), the network is deemed as in non-congestive state where network capacity is underutilized. Vegas increases $cwnd$ to probe available bandwidth. Vegas congestion avoidance algorithm is shown below.

Listing 2.1: TCP Vegas congestion avoidance algorithm

```

if ( Diff <  $\alpha$  )
     $cwnd = cwnd + 1$ 
    /* increase congestion window size by one */
else if ( Diff >  $\beta$  )
     $cwnd = cwnd - 1$ 
    /*decrease congestion window size by one */
else  $cwnd = cwnd$ 
    /* congestion window remains unchanged */

```

- TCP Veno

TCP Veno [22] is a sender-side enhancement to eliminate TCP performance suffering over wireless networks, and adopted by Linux Kernel since version 2.6.18 [40]. TCP Veno makes use of the idea of congestion-monitoring scheme from TCP Vegas, and integrates it into Reno's congestion avoidance phase. In Vegas, the monitoring scheme is calculated by the difference between the actual throughput and the expected throughput, namely,

$$\begin{aligned}
 Diff &= (Expected - Actual) \\
 &= \left(\frac{cwnd}{BaseRTT} - \frac{cwnd}{RTT} \right)
 \end{aligned}$$

where $BaseRTT$ is the minimum of all measured RTT (round-trip time) and is reset when packet loss is detected, and RTT is the actual round-trip time of a tagged packet.

Denote the backlog at the router queue by N , then,

$$RTT = BaseRTT + \frac{N}{Actual}$$

Rearrange this equation,

$$\begin{aligned}
 N &= Actual \times (RTT - BaseRTT) \\
 &= \left(\frac{cwnd}{BaseRTT} - \frac{cwnd}{RTT} \right) \times BaseRTT
 \end{aligned}$$

CHAPTER 2. LITERATURE REVIEW

The main idea of VenO is to use the measurement of N as an indication of whether the network is in congestive state or non-congestive state. If $N \geq \beta$, the network is said to have evolved into congestive state. Packet loss in the congestive state is deemed as congestion loss. β is set to 3 in current VenO implementation. If $N < \beta$, it is in the non-congestive state. Packet loss in the non-congestive state is deemed as random loss. Based on the differentiated state, VenO modifies Reno's Additive Increase Multiplicative Decrease algorithm to make $cwnd$ evolution more efficient:

Listing 2.2: TCP VenO congestion avoidance algorithm

```

Multiplicative Decrease (MD):
If ( $N < \beta$ )
     $cwnd = cwnd \cdot (n - 1)/n$ ;
Else
     $cwnd = cwnd/2$ ;

Additive Increase (AI):
If ( $N < \beta$ )
     $cwnd+ = 1/cwnd$  for every new ACK received
Else
     $cwnd+ = 1/cwnd$  for every  $m$  new ACK received

```

where, n is congestion window drop factor and m is delay increase factor. In VenO's AIMD, $n = 5$ and $m = 2$ while in Reno's AIMD, $n = 2$ and $m = 1$. Other components, i.e. slow-start, fast retransmit and fast recovery, remain unchanged.

- **TCP Westwood**

TCP Westwood [23] is also a send-side TCP enhancement. TCP Westwood proposes a novel bandwidth-estimation mechanism at the sender, so as to adapt the congestion window to the available bandwidth and hence eliminate the negative effect of random losses. TCP Westwood estimates the available bandwidth as follows.

$$b_k = \frac{d_k}{t_k - t_{k-1}}$$

where b_k is the estimated bandwidth, t_k and t_{k-1} are the time when k^{th} and $k-1^{th}$ ACKs are received, d_k is the amount of data received by the TCP receiver. In order to eliminate

CHAPTER 2. LITERATURE REVIEW

the noises in the measurement, TCP Westwood employs a low-pass filter to average the sampled measurements [23].

TCP Westwood then makes use of the estimated bandwidth to guide the sender's behavior when packet loss is detected. Specifically, if packet loss is detected by three duplicate ACKs, TCP Westwood reduces *ssthresh* and *cwnd* to the estimated available bandwidth in number of segments, rather than blindly halving *cwnd* by TCP Reno. *ssthresh* is set as:

$$ssthresh = \frac{BWE * RTTmin}{seg_size};$$

where *BWE* is available bandwidth, *RTTmin* is the smallest round-trip time observed over the duration of the transmission, and *seg_size* is the size of TCP segments.

When loss is indicated by timer expiration, Westwood also sets *ssthresh* to $(BWE * RTTmin)/seg_size$, but *cwnd* is set to one segment. Then, slow-start is invoked. Other components in Reno, such as Additive Increase, fast retransmit, slow-start, are not modified.

• Jitter-based TCP

Jitter-based TCP (JTCP) [35] is an enhancement for TCP over wireless networks. The inter-arrival jitter is normally defined as the value of packet spacing at the receiver compared with packet spacing at the sender for a pair of packets. Specifically, for two consecutive packets *i* and *j*, inter-arrival jitter $D(i, j)$ is expressed as:

$$D(i, j) = (R_j - R_i) - (S_j - S_i) = (R_j - S_j) - (R_i - S_i)$$

where S_i and R_i denote the sending time and receiving time respectively of packet *i*. If no delay happens in two consecutive packets transmission, the inter-arrival jitter should be zero; otherwise, the inter-arrival jitter should be bigger than zero, as the transmission time of the latter packet is longer than the previous one due to router queue delay. As

CHAPTER 2. LITERATURE REVIEW

a result, the value of jitter reflects the router queue length. JTCP catches this point to distinguish the types of packet losses by continuously monitoring the sending and receiving time of every packet. JTCP defines a Jitter Ratio, or J_r , to evaluate the increasing of the router queue.

$$J_r = \frac{D}{t_R(i) - t_R(i-1)}$$

where D is the value of jitter of consecutive packets, and $t_R(i)$ is the receiving time of packet i . J_r is an indication for whether the packet is queued or not. The bigger value of J_r , the more likelihood that the packet is queued. With the guidance of J_r , a packet loss is deemed as congestion loss if 1) the sender receives three DUPACKs (Duplicate Acknowledgement), 2) the ACKs are received at least in the next round-trip time, and 3) the value of J_r is bigger than the threshold $1/cwnd$. Otherwise, the loss is deemed as a random loss. If the packet loss is deemed as congestion loss, JTCP halves the congestion window like that in Reno. If the packet loss is deemed as random loss, on the other hand, JTCP reduces $cwnd$ to $D * cwnd$. The experiment results show that JTCP gets the best performance when the value of D is 1. As a result, it is better for JTCP not to reduce the sending rate in response to random loss.

JTCP also modifies the TCP action after timer expires, aiming for handling the case of timeout caused by multiple random losses occurring in a single window of data. When this happens, JTCP compares the value of J_r with the threshold $1/cwnd$. If J_r is bigger than threshold $1/cwnd$, JTCP sets $cwnd$ to 1 and invokes slow-start like that in Reno. If J_r is smaller than $1/cwnd$, otherwise, JTCP just halves the $cwnd$, rather than invoking slow-start. JTCP considers that the timeout is induced by multiple random losses. Halving the congestion window would avoid unnecessary performance degradation.

• High Speed TCP

CHAPTER 2. LITERATURE REVIEW

High Speed TCP (HS TCP) [24] is designed to solve the problem of Reno's performance degradation in large bandwidth-delay product networks where link BDP is often ten times as large as buffer size. Reno wastes the available link capacity because the AIMD algorithm increases $cwnd$ too slowly and cuts too severely. TCP congestion avoidance can be expressed in a general form:

Listing 2.3: General form of TCP congestion avoidance

```
When each ACK received,
    cwnd = cwnd + a/cwnd;
When packet loss detected,
    cwnd = cwnd - b * cwnd.
```

For Reno's AIMD, a is 1 and b is 0.5. But in the large-BDP network to maintain a high value of average $cwnd$, it is necessary that $cwnd$ increases faster (bigger value for a) and cuts less (smaller value for b). HS TCP dynamically adjusts the parameters a and b as the function of $cwnd$. First, for the purpose of compatibility with existing Reno connection in normal networks, HS TCP adopts traditional AIMD algorithm when $cwnd$ is at most 31. In this scenario, the steady state loss rate (average loss rate) is at least 0.15%.

Second, for the extreme scenario of large-BDP network of 10 Gbps where the $cwnd$ reaches about 83,000 in segment, HS TCP sets $a(w) = 72$, $b(w) = 0.1$, which means a decrease of 10% after a drop and an increase of 72 segments per round-trip time in the absence of loss. This choice of a and b allows HS TCP to improve the steady state loss rate from 10^{-10} to 10^{-7} by Reno.

Third, consider the scenario of $(cwnd, \text{loss rate})$ between $(31, 0.0015)$ and $(83,000, 10^{-7})$, HS TCP specifies $cwnd$ and loss rate to meet linear manner on a log-log scale,

$$\log cwnd - \log 31 = s(\log p - \log 0.0015)$$

CHAPTER 2. LITERATURE REVIEW

where s is the slope for $s = -0.82$ by solving this equation. Therefore, the flow-level relationship between $cwnd$ and steady state loss rate p is:

$$cwnd = \frac{0.15}{p^{0.82}}.$$

It is the response function of HS TCP. This flow-level equation should be converted to packet-level parameters of $a(w)$ and $b(w)$ for congestion control behaviors.

$$cwnd = \frac{\sqrt{\frac{a(w)(2-b(w))}{2b(w)}}}{\sqrt{p}} ppr$$

There are many solutions of $a(w)$ and $b(w)$ pairs for above function. HS TCP lets $b(w)$ to vary linearly as the log of $cwnd$.

$$b(w) = (B - 0.5) \frac{\log cwnd - \log W}{\log W_1 - \log W} + 0.5$$

where W is the window size of 31 in segment, and W_1 is the window size of 83,000 in segment. B is $b(W_1) = 0.1$. Then the increase parameter $a(w)$ can be computed as,

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

The values of $a(w)$ and $b(w)$ could be chosen in other ways. Different ways are associated with different tradeoffs between severe or moderate increases and decreases. The most severe extreme is $b(w) = 0.5$ for any value of $cwnd$, while the most moderate extreme is $a(w) = 1$ for any value of $cwnd$. HS TCP chooses the values of $a(w)$ and $b(w)$ to achieve an acceptable compromise between moderate increase and timely decrease.

• Ad Hoc TCP

Ad Hoc TCP (ATCP) [25] is a TCP enhancement for mobile Ad Hoc networks, wherein traditional TCP suffers performance degradation by a major problem of random

CHAPTER 2. LITERATURE REVIEW

loss, along with other problems such as route re-computations and network partitions. ATCP is implemented as a thin layer inserted into standard TCP and IP layers, without any modification of TCP Reno congestion control algorithm. ATCP relies on ECN (Explicit Congestion Notification) [41] to distinguish the types of packet loss. Specifically, if packet loss is preceded by an ECN notification, then the loss is deemed as congestion loss. The sender invokes congestion control, halves the *cwnd*, and retransmits the lost packet like that in Reno. ATCP doesn't involve in. On the other hand, if the packet loss is detected by three duplicate ACKs without ECN notification, ATCP deems it as random loss. In this case, ATCP sub-layer puts TCP connection into persist state by intercepting the third DUPACK, while ATCP on its own retransmits the lost segment from TCP's send buffer. As a result, the TCP connection does not know the packet lost at all. Eventually, ATCP brings the TCP connection out of the persist state on the arrival of the fresh ACK.

Secondly, ATCP relies on the ICMP (Internet Control Message Protocol) Destination Unreachable message to handle the problems of route re-computation and network partition. When the sender continuously receives ICMP Destination Unreachable messages, the ATCP sub-layer puts the TCP connection into persist state so that it does not needlessly transmit and retransmit packets. When the period of disconnection is finished and destination is reachable again, ATCP moves the TCP connection out of the persist state. ATCP sets the *cwnd* value to one segment because the *cwnd* size computed for the stale route may be not suitable for the new route.

- **Biaz loss distinguishing scheme**

The Biaz scheme [36] explores an innovative way of distinguishing loss type at the receiver by making use of the packets inter-arrival time. Distinguishing loss type at the receiver enjoys some benefits than other schemes that distinguish loss type at the sender.

CHAPTER 2. LITERATURE REVIEW

Figure 2.2: Bias loss distinguishing scheme.

For instance, with the cumulative acknowledgement switched on, the sender may not know exactly which packet is lost, but the receiver has full knowledge of it. Moreover, the inter-arrival time in the data sequence may be further spaced out in the reverse path, due to ACK queuing in the routers. This negatively affects the sender's judgment of the loss type, resulting in low accuracy of the loss distinguishing schemes.

The Biaz scheme focuses solely on the topology of last-hop wireless networks, wherein only the last hop is wireless link and it is the bottleneck of the transmission path. Assuming that all the packets are of the same size, let T_{min} denote the minimum inter-arrival time observed so far by the receiver during the transmission. Let P_i denote the last in-sequence packet and P_{i+n+1} denote the first out-of-sequence packet received after the loss, where n is the number of packets lost. The gap of the arrival time between P_i and P_{i+n+1} is T_g . Then, if $(n+1)T_{min} \leq T_g < (n+2)T_{min}$, the n lost packets are assessed as random losses, otherwise, they are assessed as congestion losses.

The rational behind the Biaz scheme is as Fig. 2.2. Suppose packets a, b, c are in-sequence data packets transmitted from sender to receiver. In the absence of packet loss, the inter-arrival time between a and b, b and c are both T_{min} , or equivalently, the

CHAPTER 2. LITERATURE REVIEW

processing time of the segment at the bottleneck router. If packet b is lost due to link error, then the inter-arrival time gap remains there, that is, the inter-arrival time between a and c is $2 * T_{min}$ because the wireless link only exists in the last hop. But if packet b is lost due to bottleneck router buffer overflow (congestion loss), the inter-arrival time between a and c may be compressed due to the router queuing packet c directly after packet a , regarding packet b is discarded.

- **TCP Jersey**

TCP Jersey [26] takes a joint design of both sender and router to distinguish random loss from congestion loss, and reacts accordingly. TCP Jersey introduces two novel components: available bandwidth estimation (ABE) algorithm, which is located at the sender, and congestion warning (CA) configuration, which is located at the router. In addition, ECN is needed to implement CA.

ABE at the sender continuously estimates the bandwidth available for the connection, and guides the sender to adjust its sending rate when the network becomes congested. The ABE estimator is,

$$R_n = \frac{RTT * R_{n-1} + L_n}{(t_n - t_{n-1}) + RTT}$$

where R_n is the estimated bandwidth when the n^{th} ACK arrives at time t_n , t_{n-1} is the previous ACK arrival time, L_n is the size of data and RTT is round-trip time of the latest received ACK. Based on the estimated bandwidth, ABE calculates the optimal window size as follows,

$$ownd_n = \frac{RTT * R_n}{seg_size}$$

where the *seg_size* is the segment size. The network simulation results show that the accuracy of Jersey's ABE algorithm is quite good.

CA at the router alerts the sender by marking packet when there is a sign of incipient congestion. CA functions are similar to ECN. Jersey considers that using ECN to

CHAPTER 2. LITERATURE REVIEW

Figure 2.3: Packet marking probability in CA.

notify the sender of the congestion is not enough, as ECN may not be timely and, more importantly, ECN is so sensitive to parameter settings that improper parameter settings may lead to unsatisfactory TCP performance. CA is designed as a simple modified version of ECN with a new marking policy and a new method of computing average queue length. In details, CA allows the average queue length to closely track the instantaneous queue length with less historical consideration than traditional ECN. Then if the queue length exceeds a threshold, CA marks every packet with probability of 1 as in Fig. 2.3, otherwise, the marking probability is zero.

TCP Jersey uses ABE and CA to modify the sender behaviors of traditional TCP as follows. When the sender receives an ACK or duplicate ACK, Jersey invokes the ABE algorithm to probe the available bandwidth and accordingly computes the optimal size of *cwnd*. Then, 1) if this is an ACK without CA mark, Jersey performs the same as Reno. 2) If this is an ACK, or DUPACK, with CA mark, Jersey sets the *ssthresh* and *cwnd* to the size of the optimal *cwnd*, and invokes the congestion avoidance phase. Besides, Jersey retransmits the lost segment on the third DUPACK. 3) If this is the third DUPACK without CA mark, the loss is deemed as random loss. So Jersey retransmits

the lost segment without reduction of the *cwnd*.

- **TCP Hybla**

TCP Hybla [37] aims to solve the bias in performance of TCP connections with long round-trip time. TCP Hybla chooses a reference connection with reference round-trip time RTT_0 , the AIMD algorithm is then adjusted to take into account of the real value of RTT so that all the co-existing Hybla connections have similar performance to the reference connection. TCP Hybla exhibits significant performance improvement in networks with extreme long RTT, such as satellite. However, it is difficult to dynamically estimate the efficient reference round-trip time in the wired-wireless network.

2.1.2 TCP Enhancements for Startup Schemes

- **TCP Vegas startup algorithm**

TCP Vegas startup algorithm makes use of Vegas congestion-monitoring scheme to detect incipient congestion. Namely, the sender records the packet round-trip time and then calculates the difference between *expected* throughput and *actual* throughput. Based on this difference, Vegas slow-start doubles *cwnd* every other round, and is exited when the difference exceeds a certain threshold. However, Vegas slow-start may be exited prematurely before reaching the eligible window size due to temporary queue buildup in intermediate routers [30].

- **Hoe's modification and Agile Probing**

In [38][42], Hoe proposed to set initial *ssthresh* to the estimated value of BDP obtained by using packet pair measurements in the early stage of slow-start. This method can avoid traditional slow-start limitations caused by blind *ssthresh* setting. However, due to *cwnd* increasing too fast towards the estimated BDP, Hoe's change may suffer temporary queue

CHAPTER 2. LITERATURE REVIEW

overflow if the buffer size is not large enough with respect to the BDP, which can easily happen in high-speed long-delay networks. Agile Probing [30] also aims to estimate the BDP during startup phase, but it repeatedly estimates the value and resets *ssthresh*, allowing *cwnd* to adaptively converge to the BDP. It can prevent the temporary queue from overflowing a small buffer.

- **Limited slow-start**

Limited Slow-Start [31] introduces an additional threshold *max_ssthresh* and modifies the manner in which *cwnd* increases. Namely, when $cwnd \leq max_ssthresh$, *cwnd* doubles per RTT, the same as slow-start. When $max_ssthresh < cwnd \leq ssthresh$, *cwnd* is increased by a fixed amount of $max_ssthresh/2$ segments per RTT. This method can reduce the number of drops during slow-start. However, *max_ssthresh* is statically set before the TCP connection starts. The recommended value is 100 packets.

- **TCP Peach**

TCP Peach [39] is designed for the satellite network in which current TCP suffers performance degradation due to long propagation delays and high link error rate. TCP Peach introduces two new algorithms, namely Sudden Start and Rapid Recovery. These new algorithms are based on the novel concept of using dummy segments to detect the network states.

In contrast with normal TCP data segment, the dummy segment is a type of low priority segment that carries a copy of latest transmitted data segment. The sender transmits the data segments and the dummy segments if allowed in the Sudden Start and Rapid Recovery phases. Since dummy segments are assigned lower priority in the router buffer queuing, the router will discard redundant dummy segments that exceed the network capacity to guarantee the transmission of the data segments. On receipt

CHAPTER 2. LITERATURE REVIEW

of these segments, the receiver acknowledges the data segments as well as the dummy segments. As a result, the sender could easily detect the path capacity by counting the number of ACKs for both data segments and dummy segments.

TCP Peach uses dummy segments to redesign the slow-start phase and the recovery phase, while leaving congestion avoidance phase and retransmit phase intact. Recall that, the TCP connection starts slow-start in the case of 1) first initialized and 2) after a timeout. The TCP connection starts a recovery phase in the case of packet loss indicated by duplicate ACKs. In these cases, the TCP connection has least information about network capacity, so that employing dummy segments in these phases benefits the most. The modified slow-start is called “sudden start”. The mechanism is as follows. In the first round, the *cwnd* is set to 1; the sender transmits 1 data segment and up to $(rwnd - 1)$ dummy segments, where the *rwnd* is the maximum window size specified by the receiver. In the second round, the sender counts the returned ACKs, and immediately jumps to the calculated value of the network capacity, then switching to the congestion avoidance phase. As a result, TCP Peach “suddenly” inflates the *cwnd* to the optimal size in the second round, rather than Reno that doubles *cwnd* every round.

TCP Peach also modifies the recovery phase called “rapid recovery”. For rapid recovery, Peach defines a variable *adsn* (allowed dummy segments number). When loss is detected, the sender performs Fast Retransmit as that in Reno. Then, the sender enters the rapid recovery phase. Specifically, 1) TCP Peach halves the *cwnd*. 2) If the *cwnd* is smaller than the number of unacknowledged data segments and $adsn > 0$, then at each time the sender receives an ACK, it transmits 2 dummy segments, increases *cwnd* by 1 and decreases the *adsn* by two. 3) The sender transmits 1 data segment and increases *cwnd* by 1 each time it receives an ACK. 4) Such transmission continues until the lost segment is acknowledged. *adsn* is set to 1. When the ACK for the lost segment is received, the sender sets *cwnd* and *adsn* to the original halved values. Then, the sender exits the rapid recovery and enters the congestion avoidance phase.

2.1.3 TCP Enhancements for Recovery Schemes

- **TCP NewReno**

TCP NewReno [27] introduces a slight but significant modification in Reno's FF (Fast retransmit and Fast recovery) for Partial ACK that acknowledges some but not all of the unacknowledged segments. Specifically, when three duplicate ACKs are received, the lost segment is retransmitted immediately. The highest sequence number of the transmitted segments is recorded in the variable "recover". Thereafter, *ssthresh* is set to half of the window size, and *cwnd* is set to *ssthresh* + 3. Then, for every following duplicate ACK received, *cwnd* is increased by one. New segment is transmitted whenever *cwnd* is bigger than the number of unacknowledged segments. When an ACK that acknowledges new data is received, 1) if this is a partial ACK, NewReno immediately retransmits the next unacknowledged segment, deflates the *cwnd* by the amount of new data acknowledged, and sends new segments if permitted by the new value of *cwnd* but does not exit the recovery procedure. Otherwise, 2) if this ACK acknowledges all of the data up to and including "recover", NewReno sets *cwnd* to *ssthresh* and exists the recovery phase.

- **Selective Acknowledgement TCP**

Selective Acknowledgement (SACK) TCP [28] makes use of the SACK option returned from receiver to report the lost segments during transmission. Meanwhile, SACK scheme requires the sender to keep a "pipe" variable during recovery phase to estimate the amount of outstanding segments. The initial value of pipe is (*cwnd* - 3), and thereafter, receiving a duplicate ACK results in pipe minus 1 and receiving a partial ACK results in pipe minus 2. When the pipe reduces to less than half of the *cwnd*, the sender can transmit lost or new segments. Transmitting one segment results in pipe plus one. Then the sender could intelligently retransmit the lost segments as soon as possible.

- **Forward Acknowledgement TCP**

Forward Acknowledgement (FACK) TCP [29] scheme is based on the SACK option, and then performs a precise control of the outstanding segments. FACK makes use of *snd.fack* to represent the correctly received data with the highest sequence number, *snd.una* to represent the first unacknowledged segment, *snd.nxt* to represent the next new segment to be sent, and *retran_data* to represent the number of the retransmitted segments in the network.

In the congestion avoidance phase where no packet loss occurs, the *snd.una* must be equal to *snd.fack*, and smaller than *snd.nxt*. But if packet loss occurs, *snd.una* will be smaller than *snd.fack*, because the SACK option from receiver reports non-contiguous blocks. Therefore, recovery phase is triggered by three DUPACK or $snd.fack - snd.una > 3$. FACK uses *awnd* to represent sender's estimate of the outstanding segments in the network. The *awnd* is calculated as follows,

$$awnd = snd.nxt - snd.fack + retran_data \quad (2.1)$$

where *retran_data* is the amount of the retransmitted segments. When a segment is retransmitted, the sender increases the *retran_data* by one segment. On the other hand, when a retransmitted segment is determined to have left the network, the sender decreases the *retran_data* by one segment.

In the recovery phase, if the *awnd* is smaller than the *cwnd*, the sender can inject segments into the network. The sender could choose to send new data, or to retransmit lost segments reported in the SACK option. If new data is sent, FACK advances *snd.nxt*; if lost segment is retransmitted, FACK increases *retran_data*. Consequently, ACK from receiver will either decrease *retran_data* or increase *snd.fack*. Since FACK keeps an accurate estimate of the amount of outstanding segments in the recovery phase, FACK is more efficient when multiple packets are lost from one window of data.

2.2 TCP-Friendly Multicast Congestion Control Schemes

TCP serves most of the bulk data transfers in the Internet. However, for multimedia applications such as IP TV, video conference, TCP-friendly multicast congestion control protocols are more suitable. In this thesis, we only focus on single-rate multicast protocols [43]. In the past years, a number of schemes are proposed for single-rate TCP-friendly multicast congestion controls, such as MTCP [19], PGMCC [20], EPGMCC [44, 45], TFMCC [17]. In the single-rate scheme, all the receivers get the same transmission rate that is allowed by the slowest receiver. Moreover, the transmission rate is adjusted by TCP-friendly congestion control mechanisms. Roughly speaking, they can be divided into two categories: window-based and rate-based.

2.2.1 Window-based Congestion Control Schemes

- **Multicast TCP**

The window-based protocols make use of congestion window information, as such in TCP, to perform multicast congestion control. For example, MTCP [19] employs TCP Vegas-like congestion control mechanism to achieve its data transmission. Specifically, in MTCP, sender is regarded as the root and receivers are the leaves with sender's agents (SAs) in between. The sender and each SA have a congestion window (cwnd) that indicates the congestion level of the network and a transmit window (twnd) that indicates the number of outstanding packets. The summaries of congestion information are reported upstream from leaves to sender through intermediate SAs.

- **Pragmatic General Multicast Congestion Control**

PGMCC [20] is another example of window-based multicast. In PGMCC scheme, the multicast receivers monitor the round-trip time and loss rate, and report these information to the sender. With the feedback information, an acker - the representative

CHAPTER 2. LITERATURE REVIEW

of receivers, is dynamically selected during the transmission. A TCP-like window-based mechanism is then performed between the sender and the receivers.

- **Enhanced Pragmatic General Multicast Congestion Control**

EPGMCC [44, 45] is an enhancement of PGMCC for wireless networks. In original PGMCC's acker selection, there is no loss-distinguishing mechanism in the loss rate measurement scheme. As a result, in the wired-wireless heterogeneous networks, the measured loss rate reflects the sum of the congestion loss rate and random loss rate. This erroneous loss rate measurement can result in erroneous selection of the acker, and thus degrade the performance of the whole session. EPGMCC introduces a new loss-distinguishing mechanism based on forward-delay variations to eliminate random loss from loss rate measurement. In addition, PGMCC's window-based controller is modified so that it suffers smaller amount of window reduction for the random loss. Experimental results demonstrate that throughput improvement of 40% is achieved in the environment of 0.01 random loss rate; meanwhile, the TCP-friendliness in PGMCC is also inherited.

2.2.2 Rate-based Congestion Control Schemes

- **TCP-Friend Multicast Congestion Control**

TFMCC [17] is a rate-based multicast congestion control protocol. In TFMCC scheme, each receiver tracks the loss rate and round-trip time, and then makes use of TCP Reno's throughput equation [46] to calculate the expected data rate:

$$B = \frac{s}{RTT(\sqrt{\frac{2p}{3}} + (12\sqrt{\frac{3p}{8}})p(1 + 32p^2))} \quad (2.2)$$

where B is TCP's throughput in bytes/second, s is the packet size in bytes, RTT is the round-trip time, and p is the steady-state loss-event rate. This rate is then fed back to the sender. The feedback mechanism is designed to make sure that the feedback from

CHAPTER 2. LITERATURE REVIEW

Current Limiting Receiver—the receiver with lowest expected rate in the group—reaches the sender meanwhile the feedback implosion is prevented. The sender then adjusts its sending rate according to the received values.

Generally speaking, window-based and rate-based schemes aim at different types of applications. The window-based schemes are more suitable for applications that can tolerate sending rate variations while applications such as IPTV, VoIP, prefer rate-based schemes in order to achieve smooth changes in the sending rate.

Chapter 3

A Sender-side Enhancement for TCP Startup

3.1 Introduction

The current legacy TCP, namely TCP Reno, and its enhancements such as NewReno use slow-start during startup phase to probe network capacity. However, traditional slow-start algorithm suffers performance degradation in high-speed long-delay networks due to blind initial *ssthresh* (slow-start threshold) setting. Slow-start sets initial *ssthresh* before a TCP connection starts, and the value is arbitrary depending on different operating system implementations. As a result, 1) if *ssthresh* is set too high with respect to the bandwidth-delay product (BDP), multiple packet losses and timeout may occur. 2) If *ssthresh* is set too low, the TCP connection will exit slow-start and switch to congestion avoidance phase prematurely. Both cases cause low utilization in link capacity.

This Chapter presents a sender-side enhancement that is simple and efficient to improve TCP startup performance in high-speed long-delay networks. The key idea is that it makes use of TCP Vegas congestion-monitoring scheme to monitor the router queue, and accordingly refines slow-start window evolution to quickly reach the eligible window size, meanwhile avoid multiple packet losses. Simulation results demonstrate that, compared with traditional slow-start and many other enhancements, this idea can significantly improve link utilization without adversely affecting co-existing TCP connections.

Figure 3.1: Network simulation topology.

3.2 Traditional slow-start limitations

We use network simulator NS-2 [63][64] to simulate TCP startup performance in high-speed long-delay networks. Fig.3.1 shows the simulation topology. Side links are 500 Mbps and 0.1 ms one-way delay, while bottleneck link is 40 Mbps and 50 ms one-way delay. The packet size is 1000 bytes and ACK is 40 bytes, so the BDP value is 500 packets. The bottleneck router is Drop-Tail with 250 packets buffer size ($BDP/2$). TCP sources run NewReno with traditional slow-start, and the max burst size is 5.

In traditional slow-start, sender increases *cwnd* (congestion window) by one packet upon receiving every new ACK, until *cwnd* reaches initial *ssthresh*. However, initial *ssthresh* is set before the TCP connection starts, and its value is arbitrary, ranging from 4KB to extremely high. Due to blind initial *ssthresh* setting, TCP suffers low startup performance in high-speed long-delay networks. Fig.3.2 shows two typical cases where initial *ssthresh* mismatches the BDP. In the first case, *ssthresh* is set to 800 packets, which is higher than the BDP. We observe that the TCP connection suffers many packets losses and long recovery time. In the second case, *ssthresh* is set to 32 packets, which is below the BDP. We observe that the TCP connection exits slow-start and switches to congestion avoidance phase prematurely.

Figure 3.2: Comparison of slow-start and Enhanced Start *cwnd* evolutions.

3.3 Mechanism of Enhanced startup algorithm

Our enhancement makes use of TCP Vegas congestion-monitoring scheme to monitor the router queue. In Vegas, the throughput difference is calculated by:

$$Diff = (Expected - Actual) = \left(\frac{cwnd}{BaseRTT} - \frac{cwnd}{RTT} \right)$$

where $BaseRTT$ is the minimum of all measured RTT , and RTT is the actual round-trip time of a tagged packet. Denote the backlog at router queue by N , we have,

$$RTT = BaseRTT + N/Actual$$

rearrange this equation, we deduce that,

$$N = \left(\frac{cwnd}{BaseRTT} - \frac{cwnd}{RTT} \right) \times BaseRTT \quad (3.1)$$

therefore, during startup phase we can use equation (3.1) to estimate the backlog at router queue. If N exceeds a threshold of β packets, we can imply that the router queue

CHAPTER 3. A SENDER-SIDE ENHANCEMENT FOR TCP STARTUP

Figure 3.3: Enhanced Start *cwnd* evolution under different values of β .

Figure 3.4: Enhanced Start *cwnd* evolution under burst UDP cross-traffic (10Mbps starts at 1 sec and stops at 5 sec).

is building up and we call it a *congestive event*. β is set to 3 packets. Then, slow-start is modified as follows.

When a TCP connection starts, the sender increases *cwnd* by one packet for every ACK received, the same as traditional slow-start, until router queue buildup and congestive event is detected. The sender can imply that the network capacity has been fully utilized, and thus switches to linear increase manner to avoid buffer overflow and multiple losses.

The congestive event, however, may be temporary queue buildup due to *cwnd* increasing too fast. In such a case, by switching to linear increase manner, the sender can quickly drain the temporary queue, and then senses that the router queue turns empty. As a

Figure 3.5: NewReno (NR) throughput versus bottleneck bandwidth (first 20 s).

result, the sender can speed up again to probe the available bandwidth, but the increase speed reduces to half of that before the congestive event because the spare bandwidth is less than before. The startup phase is exited when packet loss occurs. Then $ssthresh$ is set to $cwnd/2$. We simply call our scheme as Enhanced Start. The pseudo-code is:

Listing 3.1: Enhanced Start algorithm

```

If (DUPACKs are received) /*startup phase is exited*/
    ssthresh=cwnd/2; switch to congestion avoidance phase
else /*in startup phase*/
    If ( $N \geq \beta$ )
         $cwnd += \frac{1}{cwnd}$  for every ack (linear increase manner)
    else
         $cwnd += \max(cwnd^{-1}, 2^{-(Congestive\ Event\ Number)})$ 

```

The Congestive Event Number is the number of Congestive events that the sender experiences during the startup phase. Fig. 3.2 shows the queue estimation and $cwnd$ evolution of Enhanced Start. We see that, the estimated queue at sender and the actual queue at router match quite well. By correctly estimating the router queue, the sender increases $cwnd$ in exponential-linear cycles, allowing $cwnd$ to adaptively converge to the BDP in a timely manner. Meanwhile, it prevents the temporary queue from overflowing a small buffer. In Fig. 3.3, we vary the value of β to study the sensitivity of our algorithm to it. We see that, it does not cause much difference.

Fig. 3.4 shows Enhanced Start queue estimation and $cwnd$ evolution under burst

Figure 3.6: NewReno (NR) throughput versus delay (first 20 s).

UDP cross-traffic. The UDP traffic is 10Mbps starting at 1 second and stopping at 5 second. We see that, the queue estimation is still quite accurate. Enhanced Start can intelligently avoid buffer overflow and multiple losses by switching to linear increase manner, and speed up again to quickly reach the BDP after the UDP traffic is off.

3.4 Performance evaluation

Fig. 3.5 shows NewReno throughput with different startup algorithms under bottleneck bandwidth varying from 10 Mbps to 150 Mbps. We fix the bottleneck one-way delay to 50 ms and buffer size to $BDP/2$. We compare NewReno with Enhanced Start(ES), Hoe's Change(HC), Limited Slow-Start(LSS), slow-start with small *ssthresh*(32 packets), slow-start with large *ssthresh*(extremely high), and Vegas TCP. To focus on the startup performance, we only calculate the throughput in the first 20 seconds. We see that, NewReno with Enhanced Start and Hoe's Change can scale well with bandwidth. Other algorithms lack the ability to grab network bandwidth.

Fig. 3.6 shows the throughput comparison under bottleneck one-way delay varying from 10 ms to 100 ms. We fix the bandwidth to 40 Mbps and buffer size to $BDP/2$. NewReno with Enhanced Start and Hoe's Change can scale well with delay, while other startup algorithms suffer performance degradation as delay increase. In Fig. 3.7, we

Figure 3.7: NewReno (NR) throughput versus buffer size (first 20 s).

Figure 3.8: Co-existence of multiple Enhanced Start and slow-start connections.

fix the bandwidth to 40 Mbps and delay to 50 ms, but vary the buffer size from 100 packets to 300 packets. We see that, when the buffer size is small, Hoe’s Change suffers performance degradation, but Enhanced Start still keeps high throughput.

Fig. 3.8 shows the co-existence of multiple Enhanced Start and slow-start connections. We consider five NewReno connections, in which connections 1 and 2 are NewReno with slow-start (*ssthresh* 32 packets) and connections 3,4,5 are NewReno with Enhanced Start. Connections 1,2,3,4 start at 0 second to investigate the effect of Enhanced Start and slow-start startup at the same time. Connection 5 starts at 30 second to investigate the effect of Enhanced Start on existing TCP connections. We see that, Enhanced Start starts up quickly by making better use of the network bandwidth that is left unused by other connections. It does not adversely affect co-existing slow-start and existing TCP

CHAPTER 3. A SENDER-SIDE ENHANCEMENT FOR TCP STARTUP

connections. Five connections quickly converge to the same window size.

3.5 Summary

In this Chapter, we presented a sender-side enhancement to improve TCP startup performance in high-speed long-delay networks. It makes use of TCP Vegas congestion-monitoring scheme to monitor the router queue, and refines slow-start window evolution to quickly reach the eligible window size, meanwhile avoiding multiple packet losses. Simulation results demonstrated that it can significantly improve TCP startup performance without adversely affecting co-existing TCP connections.

Chapter 4

An Enhancement of TCP Veno with Forward Acknowledgement

4.1 Introduction

TCP Veno exhibits significant performance improvement in the wireless networks and its harmonious co-existence with TCP Reno in the wired networks. However, TCP Veno only refined Reno's Additive Increase Multiplicative Decrease (AIMD) algorithm, leaving traditional Fast retransmit and Fast recovery (FF) algorithm intact. It is well known that FF will cut congestion window several times if multiple packets are lost within a window of data [47], and drive the TCP connection into timeout. This causes TCP performance degradation.

In this Chapter, we study the enhancement of TCP Veno with Forward ACK (FACK) [29], called TCP Veno-F. TCP Veno-F aims to improve Veno's performance through better recovery efficiency. We'll take Fack TCP as a legacy TCP, and comprehensively compare TCP Veno-F, Fack TCP, and other TCP enhancements over a variety of network settings. Our network simulation results demonstrate that, 1) TCP Veno-F improves TCP performance by better utilization of link bandwidth; 2) TCP Veno-F co-exists with legacy TCP harmoniously in the wired networks; 3) TCP Veno-F deals with a large range of network scenarios such as heterogeneous network, RED queue management.

Figure 4.1: Simulation topology: the single-bottleneck topology

4.2 Mechanism of TCP Veno-F

TCP Veno-F mechanism comprises of two parts, one is Veno Congestion Avoidance algorithm (Veno CA), and another is Forward Acknowledgement scheme (FACK). Veno congestion avoidance algorithm makes use of the idea of congestion monitoring scheme in TCP Vegas and intelligently integrates it into Reno's window evolution. Like Vegas, the monitoring scheme is calculated by the difference between the measured and expected throughput, namely,

$$Diff = (Expected - Actual)$$

where *Expected* is calculated by $cwnd/BaseRTT$. The *BaseRTT* is the minimum of all measured RTT (round-trip time) and reset after packet loss is detected. *Actual* is the measured throughput at the sender given by $cwnd/RTT$, where *RTT* is the actual round-trip time of a tagged packet.

In Veno Congestion Avoidance algorithm, $N = Diff * BaseRTT$ is used to estimate the number of packets accumulated at the bottleneck buffer. If there is more than an upper threshold (β) of packets queuing for processing (in current Veno implementation β is set to 3), the network is said to have evolved into congestive state. Otherwise, it is in the non-congestive state. As in TCP Reno, packet loss in the congestive state (congestive drop) will cause the *cwnd* (congestion window) to be halved. However, packet loss in

CHAPTER 4. AN ENHANCEMENT OF TCP VENO WITH FORWARD ACKNOWLEDGEMENT

the non-congestive state (non-congestive drop) will only cause the window size to be decreased by a factor of $1/5$. Moreover, Veno refines the Additive Increase phase of Reno by forcing the TCP connection to stay longer at the operating region. The whole algorithm is shown below. Other components, i.e. slow-start, Fast retransmit and Fast recovery, remain unchanged in Veno.

Multiplicative Decrease:

If $(N < \beta)$ //most likely it is a random loss

$$ssthresh = cwnd \cdot (4/5);$$

Else //most likely it is a congestion loss

$$ssthresh = cwnd/2;$$

Additive Increase:

If $(N < \beta)$ //available bandwidth under-utilized

$$cwnd+ = 1/cwnd \text{ when every new ACK received}$$

Else //available bandwidth fully utilized

$$cwnd+ = 1/cwnd \text{ when every other new ACK received}$$

Forward Acknowledgement is evoked when packet loss is detected. FACK uses variables *snd.fack* representing the correctly received data with the highest sequence number, *snd.una* representing the first unacknowledged segment, *snd.nxt* representing the next new segment to be sent, and the *retran_data* representing the number of the retransmitted segments in the network. In FACK scheme, recovery phase is triggered by three DUPACK or $snd.fack - snd.una > 3$. Then, FACK uses *awnd* to represent sender's estimate of the outstanding segments in the network. The *awnd* is calculated as follows,

$$awnd = snd.nxt - snd.fack + retran_data$$

CHAPTER 4. AN ENHANCEMENT OF TCP VENO WITH FORWARD ACKNOWLEDGEMENT

where *retran_data* is the amount of the retransmitted segments. When a segment is retransmitted, the sender increases the *retran_data* by one segment. On the other hand, when a retransmitted segment is determined to have left the network, the sender decreases the *retran_data* by one segment. If the *awnd* is smaller than the *cwnd*, the sender can transmit the lost or new segments into the network.

We use network simulator NS-2 to study TCP Veno-F. Two network topologies are employed: the single-bottleneck scenario shown in Fig. 4.1 and the multi-bottleneck scenario shown in Fig. 4.2. The single-bottleneck scenario is suited to study TCP's sensitivity on bottleneck parameters. Each link connecting node and according router is 10 Mbps with 0.1 ms propagation delay. The link between two routers is the bottleneck using a drop-tail queue, or RED queue if otherwise specified. The bottleneck link parameters are: bandwidth μ packet/s, one-way delay τ second, and buffer size for both directions B packets. The packet size is 1000 Bytes and ACK size is 40 Bytes. The links connecting TCP sinks are last-hop wireless links if random loss is introduced. If the UDP sender is switched on, it continuously sends UDP packets to the corresponding receiver. The UDP cross traffic follows the Pareto distribution with shape parameter of $1.7(< 2)$ producing Long Range Dependent (LRD) traffic [48]. Assuming all sources have infinite data to send, and the receivers advertised windows have been set large enough.

The multi-bottleneck scenario is suited to investigate the compatibility between TCP connections [49]. The *Src0* wants to send data to *Dst0* across the network. But the network resources are taken up by many cross traffics and reverse traffics. The *Src0* need to grab and fairly share the resources with other TCP connections. *Src1, Src3, \dots, Src(2n-1)* are forward cross traffics, and *Src2, Src4, \dots, Src2n* are reverse traffics, where n is the number of hops. The links between nodes and routers are 10 Mbps and 0.1 ms one-way delay. The links between routers are 4.0 Mbps and 10 ms one-way delay. Other parameters are the same as the single-bottleneck scenario.

Figure 4.2: Simulation topology: the multi-bottleneck topology

Figure 4.3: TCP Veno-F in wired network. Figure 4.4: TCP Veno-F in wireless network.

We first use the single-bottleneck topology to investigate where the performance improvement of TCP Veno-F is gained from. We set the bottleneck parameters: bandwidth 1.2 Mbps, 60 ms one-way delay, and 16 packets buffer size for both directions. So, the bandwidth delay product, or BDP, is 18 packets. Fig. 4.3 shows the typical window evolution and backlog estimation of TCP Veno-F in the wired networks. We observe that, by the effort of Veno congestion avoidance algorithm, TCP Veno-F intelligently senses the network congestive state ($N \geq \beta$) and slows down the window increase speed. This allows TCP Veno-F to stay longer at the high throughput region, and experience fewer window oscillations than TCP Reno and Reno variants.

Fig. 4.4 shows the window evolution and backlog estimation of TCP Veno-F in the wireless network environment. We introduce 1% random loss rate into the last-hop link

CHAPTER 4. AN ENHANCEMENT OF TCP VENO WITH FORWARD ACKNOWLEDGEMENT

Figure 4.5: TCP Veno-F recovery phase.

Figure 4.6: Fack TCP recovery phase.

while other network parameters are retained the same. We observe that, when the packet loss occurs conditioned on $N \geq \beta$, TCP Veno-F cuts half of the *cwnd* like TCP Reno. But when the packet loss occurs conditioned on $N < \beta$, TCP Veno-F qualifies it as random loss and cuts 1/5 of the window. As a result, TCP Veno-F maintains higher average *cwnd* and hence achieves higher throughput than TCP Reno and Reno variants in the wireless networks.

To understand the different behaviors in the recovery phase, for Veno-F and Fack TCP respectively, we investigate the scenario that *cwnd* is 16 with 4 losses in the single window. Since the value of *cwnd* is smaller than BDP (18), the network is far from congestion. Fig. 4.5 and Fig. 4.6 show the detailed behaviors of Veno-F and Fack TCP.

For Veno-F, segments 2851, 2853, 2855, and 2857 are lost. When sender receives the second DUPACK, the condition $snd.fack - snd.una > 3$ is satisfied, so the sender triggers recovery phase and immediately retransmits segment 2851 instead of waiting for the third DUPACK. Veno-F senses loss happened in the non-congestive state, and hence cuts 1/5 of *cwnd*, resulting in *cwnd* becoming 12. This is what we expected. On receipt of next DUPACK, the value of *awnd* reduced to 11 ($< cwnd$), so Veno-F retransmits the second segment. Thereafter, more DUPACKs and partial ACKs allow Veno-F to transmit third and fourth lost segments and 12 new segments.

For Fack TCP, segments 2783, 2785, 2787, and 2789 are lost. When the sender receives the second DUPACK, recovery phase is triggered like that in Veno-F. But unlike Veno-F intelligently sensing the non-congestive state, Fack TCP blindly halves the *cwnd*, resulting in *cwnd* becoming 8. The direct consequence is that the sender must wait for 6 DUPACKs to deflate *awnd* to 7 ($< cwnd$) for retransmitting the second segment. Thereafter, more DUPACKs and partial ACKs allow Fack TCP to transmit third and fourth lost segments and 7 new segments.

Compare Veno-F and Fack TCP in the recovery phase, Veno-F transmits 12 new segments and after recovery phase the *cwnd* is 12, but Fack TCP transmits 7 new segments and after recovery phase the *cwnd* is 8. Therefore, Veno-F is more efficient than Fack TCP in the recovery phase.

4.3 Performance evaluation in different network scenarios

We investigate the performance of TCP Veno-F in different network scenarios. Three scenarios are to be considered: wireless, cross traffic, multi-hop and reverse traffic. We aim to compare TCP Veno-F with Fack TCP and other TCP enhancements. For each simulation, we run it for 1000 seconds and compute the goodput. The goodput is defined as follows[49], it is the effective amount of data delivered through the network.

$$Goodput = \frac{sent_data - retx_data}{transfer_time} \quad (4.1)$$

Meanwhile, for each simulation, we change the seed to repeat 20 times, and report the average results as well as 95% confidence interval.

4.3.1 Wireless scenario

The first scenario we consider is the wired-wireless heterogeneous network. The single-bottleneck simulation topology is used. The last-hop is wireless link. The bottleneck link

CHAPTER 4. AN ENHANCEMENT OF TCP VENO WITH FORWARD ACKNOWLEDGEMENT

Figure 4.7: Impact of random loss rate

bandwidth is 4.0Mbps, one-way delay is 60ms and buffer size is 42 packets. To consider the effects of burst losses on TCP performance, we use the two-state Markov process to model the random loss [50]. Particularly, the wireless channel may be in either Good state, or Bad state. In the Good state random loss rate is L_g by exponential distribution; in the bad state the random loss rate is L_b also by exponential distribution. In our model, L_g is set 10^{-7} and L_b is set 10^{-1} . The transition probabilities between Good state and Bad state are both 0.5. The duration of the Good state is deterministic and equal to 1 second, whereas the duration of the Bad state is also deterministic but varies from 10^{-4} second to 10^{-1} second. Fig.4.7 compares the goodput of Veno-F, Veno, Fack TCP, Newreno, and WestwoodNR [51] under different values of Bad state duration time.

As the duration of the Bad state increase, we observe that, Veno-F and WestwoodNR keep the highest goodput; Veno is better than Newreno and Fack TCP. In details, when the loss rate is low—Bad state duration time from 10^{-4} to 10^{-3} second, Veno-F has 8.7% goodput improvement over Fack TCP. But as the Bad state duration increase to 0.01, the loss rate increases and the wireless losses dominate the total packet losses. Veno-F experiences less window cut by the effort of Veno congestion avoidance algorithm

distinguishing random loss, and recovers quickly from packet loss by the effort of Forward ACK scheme. As a result, Veno-F exhibits 58.13% goodput improvement over Fack TCP. When the Bad state duration increase to 0.1, the loss rate is heavy. Neither Veno congestion avoidance algorithm nor FACK scheme could solely handle so many packet losses, so the goodputs of Veno and Fack TCP degrade drastically. But Veno-F still keeps high goodput.

4.3.2 Cross traffic scenario

Now the random loss rate is switched off, instead, we introduce UDP cross traffic varying from 0% to 80% of the bottleneck bandwidth. Other parameters are kept unchanged. Fig. 4.8 shows the results. When the cross traffic is 0%, it is the common case of no cross traffic, no random loss and no reverse traffic, normalized buffer size¹ is 0.7. In this case Veno-F, WestwoodNR, Veno, and Newreno have the same goodputs as Fack TCP (3.7 Mbps). This means Veno-F and other TCP enhancements all can efficiently make use of the link bandwidth as Fack TCP.

As the cross traffic increases, however, two key observations are presented in Fig. 4.8. First, Veno-F and Fack TCP both have the highest goodputs in the face of any cross traffic rate. Their goodputs decrease as the cross traffic rate increases. This observation tells us that Veno CA (congestion avoidance) has similar behavior as Reno CA in the wired network. The Veno CA could intelligently distinguish congestion losses and cut half of the *cwnd*. So, Veno-F and Fack TCP have the same goodputs. The second observation is that Veno has the lowest goodput. This observation tells us that, since Veno CA and Reno CA don't have much difference in the wired network, then the recovery phase plays the main role in TCP goodput. Veno goodput is lower than Veno-F because Forward ACK

¹Normalized buffer is defined as buffer size over BDP, it is usually used to differentiate LAN or WAN. For LAN, the normalized buffer is greater than 1, while for WAN, the normalized buffer is smaller than 1

Figure 4.8: Impact of cross traffic

is more efficient than traditional Fast retransmit and Fast recovery algorithm. Similarly, we observe that Veno goodput is below WestwoodNR and Newreno TCP, which are based on Newreno Partial ACK recovery scheme.

4.3.3 Reverse traffic and multi-hop scenario

We use multi-bottleneck topology in Fig. 4.2 to investigate the multi-hop and reverse traffic scenario. The reverse traffics $Src2$, $Src4$, $\dots$, $Src2n$ are switched on, while forward traffics $Src1$, $Src3$, $\dots$, $Src(2n-1)$ are switched off. The reverse traffics are controlled by Fack TCP. We vary the number of hops n from 1 to 7, and report the results in Fig. 4.9.

As the number of hops increases, Veno-F goodput decreases and is roughly the same as Fack TCP. This observation tells us that Veno-F is not negatively affected by reverse traffic. Besides, it provides more evidence to support our view that Veno-F has the same behavior as Fack TCP in the wired networks. We again observe that Veno goodput is below Veno-F and Fack TCP, due to the lack of Forward ACK scheme. Another important observation is that, WestwoodNR has the lowest goodput. The reason is as follows. TCP Westwood measures the rate of returning ACK to estimate the available

Figure 4.9: Impact of reverse traffic and multi-hop

bandwidth. If the reverse traffic is switched on, however, the intervals of returning ACKs are enlarged by the reverse traffic packets. This makes Westwood to underestimate the available bandwidth even though the forward path is congestion free. As a result, we observe that WestwoodNR suffers significant performance degradation in the reverse traffic scenario. More details of TCP Westwood under different network environments are shown in papers [52, 53, 54, 55, 56, 57, 58, 59, 60].

4.3.4 A comprehensive scenario

Previous subsections investigated TCP Veno-F performance in different network scenarios. Now an interesting question is: how about in a more comprehensive scenario with random loss, cross traffic and reverse traffic? Does TCP Veno-F still perform well? So we need to compare their performance in a comprehensive scenario. The single-bottleneck topology is used. The UDP cross traffic is set to 20% of the bottleneck bandwidth. The reverse traffic is controlled by Fack TCP. The last-hop link random loss model is the same as the wireless scenario. The bad state period varies from 10^{-4} second to 10^{-1} second. Bottleneck delay is 30 ms.

Figure 4.10: The goodput comparison in the comprehensive scenario.

Fig. 4.10 shows the goodputs of Veno-F, WestwoodNR, Veno, Newreno, and Fack TCP. We observe that, in the comprehensive environment including random packet loss, cross traffic, and reverse traffic, TCP Veno-F has the best performance. Compared with other TCP enhancements that Newreno and Fack TCP suffer from wireless link random losses, Veno TCP suffers from low recovery efficiency, WestwoodNR suffers from reverse traffic, TCP Veno-F is robust and flexible. TCP Veno-F is able to improve TCP performance by making better use of the link bandwidth, and its performance improvement is not negatively affected by the cross traffic and reverse traffic.

4.4 Fairness and compatibility

4.4.1 Intra-protocol fairness

We use the single-bottleneck topology to study the intra-protocol fairness. There are N TCP sources switched on to run the same TCP version. TCP $Src1$ RTT is fixed at 40ms, and TCP $SrcN$ RTT is fixed at 300ms. For the other TCP connections, the RTT of TCP $Srci$ is $(40 + i \cdot \frac{260}{N})$. This means N TCP connections have different values of RTT and their RTT are uniformly distributed in the range of [40, 300]ms. This scenario

CHAPTER 4. AN ENHANCEMENT OF TCP VENO WITH FORWARD ACKNOWLEDGEMENT

Figure 4.11: Low bottleneck bandwidth.

Figure 4.12: High bottleneck bandwidth.

is particularly suited to evaluate TCP intra-protocol fairness issue. The metric of TCP intra-fairness is the Jain's fairness index [14]:

$$J = \frac{(\sum_{i=1}^N x_i)^2}{N \cdot \sum_{i=1}^N x_i^2} \quad (4.2)$$

where x_i is the goodput of i th connection and N is the number of connections. The fairness index is bounded between 0 and 1. J increases as fairness increase.

In our experiment, N varies from 10 to 100 connections. The bottleneck router is RED (Random Early Detection) [32] with buffer size 200. RED monitors the average queue length and drops packets in a probabilistic manner. It uses a low-pass filter with an exponentially weighted moving average to calculate the average queue size:

$$avg \leftarrow (1 - w_q) \cdot avg + w_q \cdot q$$

According to the value of avg , the dropping policy is as follows:

- If $avg < min_{th}$, all arriving packets are accepted.
- If $min_{th} < avg < max_{th}$, arriving packets are dropped with probability P ,

$$P = P_{max} \cdot \frac{avg - min_{th}}{max_{th} - min_{th}}$$

- If $avg > max_{th}$, all arriving packets are dropped.

CHAPTER 4. AN ENHANCEMENT OF TCP VENO WITH FORWARD ACKNOWLEDGEMENT

Figure 4.13: Background traffics: Veno-F. Figure 4.14: Background traffics: Fack TCP.

The parameters for RED router are, $min_{th} = 100$, $max_{th} = 160$, $max_p = 0.1$, and $w = 0.005$. Fig. 4.11 shows the fairness index of Veno-F, Veno and Fack TCP when bottleneck bandwidth is 20 Mbps; and Fig. 4.12 shows the fairness index when bottleneck bandwidth is 100 Mbps. We observe that Veno-F and Veno work quite well with RED router; their fairness indexes are better than Fack TCP in most cases. Roughly, when the bottleneck bandwidth is relatively small (20 Mbps), three TCP versions have salient fairness indexes that are greater than 0.8. As the number of connections increase, the fairness indexes also increase. When $N \geq 70$, Veno-F index increases to 0.9. But in the case of large bottleneck bandwidth (100 Mbps), we see that Fack TCP index decreases to 0.7. But Veno-F and Veno indexes are still higher than 0.8. Therefore, we claim that Veno-F has better intra-protocol fairness than Fack TCP in most cases when RED router is deployed.

4.4.2 Inter-protocol fairness

We use the multi-bottleneck topology to study the inter-protocol fairness issue of TCP Veno-F. The number of hops N varies from 1 to 7. Forward traffics $Src1, \dots, Src(2n-1)$ and reverse traffics $Src2, \dots, Src2n$ are switched on. $Src0$ waits a random time to join for sending data to $Dst0$.

Fig. 4.13 shows the goodput of *Src0* as a function of number of hops N , when the forward and reverse traffics are controlled by Veno-F. This scenario studies Veno-F compatibility with Veno and Fack TCP. *Src0* runs Veno-F, Veno, and Fack TCP alternatively. We see that, Veno-F and Fack TCP roughly have the same goodput, while Veno lacks the ability to grab full bandwidth. Pay attention to the case of $N = 1$, Veno-F and Fack TCP are around 1.8 Mbps, but Veno is 1.26 Mbps due to the lack of Forward ACK scheme. Ideally, the fair share of the bottleneck link should be 2.0 Mbps of *Src0*. However, since *Src0* has the longest RTT among all the TCP connections, and it may suffer packet loss at each router, this scenario is what we designed a tough case for *Src0*. So, it is reasonable that Veno-F and Fack TCP goodputs are around 1.8 Mbps, lower than the expected 2.0 Mbps. As the number of hops increase, Veno-F, Veno and Fack TCP goodputs monotonically decrease.

Fig. 4.14 shows the case when forward and reverse traffics are controlled by Fack TCP. We consider Veno-F, Veno, Fack TCP, and HyblaS [61] running on *Src0* alternatively. This scenario studies Fack TCP compatibility with these TCP variants. We again observe that Veno-F and Fack TCP roughly have the same goodputs, while Veno lacks the ability to grab full bandwidth.

In contrast, we see HyblaS reaches more than 2.0 Mbps (the fairness line), and much higher goodput than Fack TCP. This means HyblaS may grab more bandwidth from co-existing Fack TCP. To further investigate the compatibility issue between HyblaS with Fack TCP, in Fig. 4.15, we show the case of HyblaS and Fack TCP co-existing in the single-bottleneck network with bottleneck bandwidth 4.0 Mbps. We see that, as the bottleneck delay increases, HyblaS goodput gradually increases. When the bottleneck delay is 250 ms (round-trip delay is 500 ms), HyblaS goodput increases to 2.28 Mbps while Fack TCP decreases to 1.13 Mbps. Now we use TCP Veno-F to replace HyblaS while other experiment parameters are retained unchanged. The results are shown in Fig. 4.16.

CHAPTER 4. AN ENHANCEMENT OF TCP VENO WITH FORWARD ACKNOWLEDGEMENT

Figure 4.15: HyblaS with Fack TCP.

Figure 4.16: Veno-F with Fack TCP.

Obviously, Veno-F harmoniously co-exists with Fack TCP. When the bottleneck delay is 250 ms, the goodputs of Veno-F and Fack TCP are both around 1.7 Mbps. Based on the results we obtained in these scenarios, we arrive at the conclusion that Veno-F harmoniously co-exists with Fack TCP. Veno-F performance improvement is gained by better use of the available bandwidth rather than the aggressiveness.

4.5 Summary

In this Chapter, we studied the performance of TCP Veno with Forward Acknowledgment, which is called TCP Veno-F. Our motivation is that, although TCP Veno has achieved remarkable success in the wireless network, its potential has not been fully utilized because the traditional Fast retransmit and Fast recovery algorithm has low efficiency in the recovery phase. As a result, TCP Veno will suffer performance degradation in the heavy loss environment. This point is validated in our experiments. We propose the idea of adding Forward Acknowledgement into TCP Veno in order to improve its performance in the heavy loss environment. Simulation scenarios are carefully designed to investigate TCP Veno-F performance. We see that Veno-F performance improvement is gained from two aspects: 1) maintaining a higher average *cwnd* by correctly distinguishing random losses; 2) getting higher efficiency in the recovery phase. Then, we show that Veno-F achieves highest goodput compared with Fack TCP and other TCP

CHAPTER 4. AN ENHANCEMENT OF TCP VENO WITH FORWARD ACKNOWLEDGEMENT

enhancements in a large range of network environments. Moreover, we see that Veno-F has salient intra-protocol fairness and inter-protocol fairness with Fack TCP.

Chapter 5

Loss Distinguishing Accuracy in TCP Veno and its Performance Influence

5.1 Introduction

In this Chapter, we study TCP Veno's loss distinguishing accuracy in networks with different loss rate, different link delay as well as different cross traffic. We give a heuristic formula. Based on the analytical and experimental results, we observe that in most situations, such loss distinguishing accuracy is quite high, but in some other situations, it is very low, especially, when there is heavy cross traffic competing with TCP Veno. Thereafter, we study its performance influence by comparing NewVeno and NewReno in typical cases. NewReno is considered because it is supported by most of the TCP implementations in Internet. For a fair comparison, we also add the Partial ACK into Veno and call it NewVeno. This modification doesn't affect the loss distinguishing accuracy.

The first case is the wireless scenario, in which the loss distinguishing accuracy is extremely high and the number of TO (timeout) is negligible. NewVeno achieves significant performance improvement over NewReno due to high loss distinguishing accuracy. The second case is NewVeno and NewReno co-existing in the wired network where the cross traffic varies in a large range. In this scenario, Veno's loss distinguishing accuracy

can be extremely low but we observe that NewVeno still harmoniously co-exists with NewReno without any negative impact. The detailed analysis shows that TO takes over TD and thus plays the key role in this case. That is the reason why NewVeno co-exists harmoniously with NewReno even if the loss distinguishing accuracy is low.

5.2 Loss distinguishing accuracy

To better describe how accurate packet loss distinguishing is, as seen in [62], we define the following probabilities. Note that, N is the estimate of router backlog as in equation (5.1), and β is Veno's threshold that judges network's congestive state or not.

- $P(N < \beta)$, indicating the probability of estimated non-congestive drops among all the packet losses. Specifically, denote the number of non-congestive drops by a , the number of congestive drops by b , and the number of total packet losses by l . Then, $P(N < \beta) = a/l$.
- $P(N \geq \beta)$, indicating the probability of estimated congestive drops among all the packet losses. Specifically, $P(N \geq \beta) = b/l$.
- Congestion loss distinguishing Accuracy (CA): CA is defined as the conditional probability of identifying a packet loss as congestive drop given that this loss is due to congestion indeed, i.e., $P(N \geq \beta|c)$.
- Random loss distinguishing Accuracy (RA): RA is defined as the conditional probability of identifying a packet loss as non-congestive drop given that this loss is random loss indeed, i.e., $P(N < \beta|r)$.

In TCP Veno, the judgment of whether a packet loss is due to network congestion or random loss relies on the estimate of backlog packets N . The expression of N can be

CHAPTER 5. LOSS DISTINGUISHING ACCURACY IN TCP VENO AND ITS PERFORMANCE INFLUENCE

rewritten as

$$\begin{aligned}
 N &= Diff \cdot BaseRTT \\
 &= cwnd \cdot \left(\frac{RTT - BaseRTT}{RTT} \right) \\
 &= cwnd \cdot \left(\frac{\Delta RTT}{BaseRTT + \Delta RTT} \right)
 \end{aligned} \tag{5.1}$$

where, $\Delta RTT = RTT - BaseRTT$. It is the difference between the sampled round-trip time of the tagged packet and the smallest measured round-trip time.

Heuristically, the $BaseRTT$, measured by Veno, is a function of the propagation delay τ and cross traffic C . In fact, if τ increases, then the $BaseRTT$ increases. Cross traffic C represents the percentage of the bandwidth used by background UDP traffic. For example, given the bandwidth is 4.0 Mbps, the cross traffic C is 50%, if the UDP traffic is 2.0 Mbps on average. Likewise, if the C increases, the $BaseRTT$ increases accordingly because the network has already evolved into congestive state. We can write $BaseRTT$ as follows,

$$BaseRTT = BaseRTT(\tau, C)$$

Similarly, the $cwnd$ is a function of the random loss rate P , buffer size B and cross traffic C . If P increases, more random packet losses will be induced, resulting in smaller $cwnd$. If B increases, less congestion will happen, so that $cwnd$ can be larger. If C increases, $cwnd$ decreases because the available bandwidth reduces. We write the $cwnd$ as follows,

$$cwnd = cwnd(C, P, B)$$

Therefore, by plugging the above two equations into equation (5.1), we can conclude the following equation (5.2),

$$N(B, \tau, C, P) = cwnd(C, B, P) \left(\frac{\Delta RTT}{BaseRTT(\tau, C) + \Delta RTT} \right) \tag{5.2}$$

Figure 5.1: Simulation topology.

We can easily see that N is a function of τ, B, P, C . With the calculus, we can derive the sensitivity of N to τ, B, P, C . So we can use equation (5.2) to study the relationship between the network parameters and Veno's loss distinguishing accuracy. Similar to [62], to simplify the analysis, we assume that ΔRTT is a random variable that is only the function of transmission time, so that it is not affected by other network parameters.

We use the simulation results to validate our analysis. Unlike [62] that keeps the congestion loss rate constant, our experiment varies one of τ, B, P, C , while keeps other three parameters fixed. One TCP Veno connection is switched on to run on the bottleneck. The simulation time is 1000 seconds. For every considered case, we change the seed to run 30 simulations, and report the average value as well as the 95% confidence interval.

5.2.1 Simulation topology

We use network simulator (NS-2) [63][64] from Lawrence Berkeley National Laboratory to study TCP. Fig. 5.1 depicts the experiment topology. The single-bottleneck scenario is suited to study TCP's sensitivity on bottleneck parameters. Each link connecting node and according router is 10 Mbps with 0.1 ms propagation delay. The link between two routers is the bottleneck using drop-tail queue, and parameters are: bandwidth μ packet/s, one-way delay τ second, and buffer size for both directions B packets. The packet size is 1000 Bytes and ACK size is 40 Bytes. The links connecting TCP sinks are last-hop wireless links if random loss is introduced. If the UDP sender is switched on,

CHAPTER 5. LOSS DISTINGUISHING ACCURACY IN TCP VENO AND ITS PERFORMANCE INFLUENCE

Figure 5.2: Impact of random loss rate.

Figure 5.3: Impact of cross traffic.

it continuously transmits UDP packets to the corresponding receiver. The UDP cross traffic follows the Pareto distribution with shape parameter of $1.7 (< 2)$ producing Long Range Dependent (LRD) traffic [48]. We assume that all sources have infinite data to send and the receivers' advertised windows have been set large enough.

5.2.2 Impact of random loss rate.

Using equation (5.2), we get,

$$\begin{aligned} \frac{\partial N(B, \tau, P, C)}{\partial P} &= \frac{\partial N(B, \tau, P, C)}{\partial cwnd} \cdot \frac{\partial cwnd}{\partial P} \\ &= \frac{\Delta RTT}{BaseRTT + \Delta RTT} \cdot \frac{\partial cwnd}{\partial P} \end{aligned} \quad (5.3)$$

As P increases, $cwnd$ decreases, so that,

$$\frac{\partial cwnd}{\partial P} < 0$$

Note that,

$$\frac{\Delta RTT}{BaseRTT + \Delta RTT} > 0$$

Therefore, we deduce that,

$$\frac{\partial N(B, \tau, P, C)}{\partial P} < 0$$

CHAPTER 5. LOSS DISTINGUISHING ACCURACY IN TCP VENO AND ITS PERFORMANCE INFLUENCE

This means increase of P results in decrease of N . Considering that β is a constant, the probability $P(N \geq \beta)$ is expected to decrease. On the other hand, $P(N < \beta) = 1 - P(N \geq \beta)$, increases accordingly.

We vary the last-hop random loss rate from 10^{-5} to 10^{-1} . Other parameters fixed as: bottleneck buffer size 42 packets, bandwidth 4.0 Mbps, 60ms one-way delay, and the cross traffic 40% of the bottleneck bandwidth. Fig. 5.2 shows the experiment results. We observe that as the loss rate increases, $P(N \geq \beta)$ monotonically decreases (while $P(N < \beta)$ increases). The experiment results comply with our analysis. Besides, we observe that the curves of CA and RA follow the similar patterns as $P(N \geq \beta)$ and $P(N < \beta)$, i.e., CA decreases but RA increases.

5.2.3 Impact of cross traffic.

Using equation (5.2), we get,

$$\begin{aligned} \frac{\partial N(B, \tau, P, C)}{\partial C} &= \frac{\partial N}{\partial BRTT} \cdot \frac{\partial BRTT}{\partial C} + \frac{\partial N}{\partial cwnd} \cdot \frac{\partial cwnd}{\partial C} \\ &= \frac{(-1) \cdot cwnd \cdot \Delta RTT}{(BRTT + \Delta RTT)^2} \cdot \frac{\partial BRTT}{\partial C} + \\ &\quad \frac{\Delta RTT}{BaseRTT + \Delta RTT} \cdot \frac{\partial cwnd}{\partial C} \end{aligned} \quad (5.4)$$

For the first part of equation (5.4), we have,

$$\frac{(-1) \cdot cwnd \cdot \Delta RTT}{(BRTT + \Delta RTT)^2} < 0$$

and

$$\frac{\partial BaseRTT}{\partial C} > 0$$

For the second part of equation (5.4), we have,

$$\frac{\Delta RTT}{BaseRTT + \Delta RTT} > 0$$

CHAPTER 5. LOSS DISTINGUISHING ACCURACY IN TCP VENO AND ITS PERFORMANCE INFLUENCE

Figure 5.4: Impact of buffer size.

Figure 5.5: Impact of propagation delay.

and

$$\frac{\partial cwnd}{\partial C} < 0$$

So we deduce that,

$$\frac{\partial N(B, \tau, P, C)}{\partial C} < 0$$

This means increase of C results in decrease of N . So that $P(N \geq \beta)$ is expected to decrease, while $P(N < \beta) = 1 - P(N \geq \beta)$, increases accordingly. To validate this analysis, we vary the cross traffic from 10% to 80% of the bottleneck bandwidth. Other parameters are fixed as: bottleneck buffer size 42 packets, bottleneck bandwidth 4.0Mbps, one-way delay 60ms, and last-hop loss rate 0.001. We observe in Fig. 5.3 that $P(N \geq \beta)$ decreases (while $P(N < \beta)$ increases) as we analyzed. CA and RA follow the similar patterns as $P(N \geq \beta)$ and $P(N < \beta)$, i.e., CA decreases while RA increases.

5.2.4 Impact of buffer size.

Using equation (5.2), we get,

$$\begin{aligned} \frac{\partial N(B, \tau, P, C)}{\partial B} &= \frac{\partial N(B, \tau, P, C)}{\partial cwnd} \cdot \frac{\partial cwnd}{\partial B} \\ &= \frac{\Delta RTT}{BaseRTT + \Delta RTT} \cdot \frac{\partial cwnd}{\partial B} \end{aligned} \quad (5.5)$$

Since,

$$\frac{\partial cwnd}{\partial B} > 0$$

and

$$\frac{\Delta RTT}{BaseRTT + \Delta RTT} > 0$$

So we deduce that,

$$\frac{\partial N(B, \tau, P, C)}{\partial B} > 0$$

This means increase of B causes increase of N , resulting in the increase of $P(N \geq \beta)$ because of the constant β . On the other hand, $P(N < \beta) = 1 - P(N \geq \beta)$ decreases accordingly. Fig. 5.4 shows the experiment results by varying the bottleneck router buffer from 15 packets to 120 packets. We keep other parameters unchanged. The results demonstrate that, similar to our analysis, as the buffer size increases, $P(N \geq \beta)$ monotonically increases (while $P(N < \beta)$ monotonically decreases). The curves of CA and RA follow the similar patterns as $P(N \geq \beta)$ and $P(N < \beta)$, i.e., CA increases but RA decreases.

5.2.5 Impact of delay.

Using equation (5.2), we get,

$$\begin{aligned} \frac{\partial N(B, \tau, P, C)}{\partial \tau} &= \frac{\partial N(B, \tau, P, C)}{\partial BRTT} \cdot \frac{\partial BRTT}{\partial \tau} \\ &= \frac{(-1) \cdot cwnd \cdot \Delta RTT}{(BRTT + \Delta RTT)^2} \cdot \frac{\partial BRTT}{\partial \tau} \end{aligned} \quad (5.6)$$

Since,

$$\frac{\partial BaseRTT}{\partial \tau} > 0$$

and

$$\frac{(-1) \cdot cwnd \cdot \Delta RTT}{(BRTT + \Delta RTT)^2} < 0$$

CHAPTER 5. LOSS DISTINGUISHING ACCURACY IN TCP VENO AND ITS PERFORMANCE INFLUENCE

Figure 5.6: Impact of Reverse traffic.

Figure 5.7: Impact of β .

So, we deduce that,

$$\frac{\partial N(B, \tau, P, C)}{\partial \tau} < 0$$

This means increase of τ results in decrease of N . Therefore, $P(N \geq \beta)$ decreases while $P(N < \beta)$ increases accordingly. Fig.5.5 shows the experiment results by varying the bottleneck one-way delay from $15ms$ to $240ms$. We keep other parameters unchanged. The results demonstrate that, complying with our analysis, $P(N \geq \beta)$ monotonically decreases while $P(N < \beta)$ monotonically increases. CA and RA follow the similar patterns, i.e. CA decreases while RA increases.

5.2.6 Impact of reverse traffic and β .

Some other parameters, such as threshold β and reverse traffic, also affect Veno's loss distinguishing accuracy. We first consider the reverse traffic. Previous studies on TCP Veno's performance [65, 66, 67, 68, 69, 70, 71, 72, 73, 74, 75, 76, 77] have shown that, unlike TCP Vegas that suffers performance degradation from reverse traffic, TCP Veno is robust in the reverse traffic scenario. This characteristic will also be exhibited in the next section of performance evaluation.

As the reverse traffic increases, the reverse path bandwidth and router buffer are taken up by the traffic, so that Veno's returning ACK experiences more queuing delay

and hence the round-trip time is increased. As a result, the impact of reverse traffic is similar to that of delay, specifically, increase of reverse traffic causes $P(N \geq \beta)$ and CA decrease while $P(N < \beta)$ and RA increase. In Fig. 5.6, we vary the reverse traffic from 10% to 80%. The experiment results validate our analysis.

In previous analysis, we take threshold β as a constant, but now, we consider the impact of different values of β . As shown above, β is the threshold for judging random loss or congestion loss. If the value of β is increased, fewer losses will be judged as congestion losses, which results in decrease of $P(N \geq \beta)$ while increase of $P(N < \beta)$. In Fig. 5.7, we vary the value of β from 2 to 20, while other parameters retain unchanged. We can observe that, the experiment results comply with our analysis.

Another observation in the figure is that, the impact of varying the value of β is not significant on TCP Veno's loss distinguishing accuracy. Detailed results show that, when β is chosen as 2, the CA is 0.58, while β is chosen as 20, the CA is 0.53. So in current implementation, Veno chooses $\beta = 3$ for its setting.

5.3 The performance influence

We compare NewVeno, NewReno and Westwood+ [51] goodputs in typical cases to investigate the performance influence of the loss distinguishing accuracy. The goodput is defined as follows [49]; it is the effective amount of data delivered through the network.

$$Goodput = \frac{sent_data - retx_data}{transfer_time} \quad (5.7)$$

NewReno is an enhancement of TCP Reno by Partial ACK to avoid multiple window reductions in the recovery phase. NewReno is considered because it is supported by most of the TCP implementations in Internet. For a fair comparison, we also add the Partial ACK into Veno and call it NewVeno. This modification doesn't affect the loss distinguishing accuracy. For every simulation in this section, the simulation time is 500 seconds. We change the seed to repeat 10 times, and report the average values.

CHAPTER 5. LOSS DISTINGUISHING ACCURACY IN TCP VENO AND ITS PERFORMANCE INFLUENCE

Figure 5.8: Goodputs in wireless scenario. Figure 5.9: Window evolution illustration.

Table 5.1: Data collection in the wireless scenario

Bad state period (second)		0.001	0.005	0.01	0.05	0.1
Loss	CL	6	1	1	1	1
	RL	79	157	200	367	389
V_{TD}	CD	55	50	38	1	1
	NCD	30	108	163	367	389
Random loss distinguishing accuracy		0.38	0.69	0.82	1	1
V_{TO}		1	1	1	3	4

5.3.1 Wireless scenario.

We run NewVeno, NewReno and Westwood+ alternatively and record their goodputs. The last-hop link is wireless link and error model is the two-state Markov model [50]. Particularly, the wireless channel may be in either Good state, or Bad state. In the Good state random loss rate is 10^{-7} by exponential distribution; in the bad state the random loss rate is 10^{-1} also by exponential distribution. The transition probabilities between Good state and Bad state are both 0.5. The duration of the Good state is deterministic and equal to 1 second, whereas the duration of the Bad state is also deterministic but varies from 10^{-4} second to 10^{-1} second. Other parameters are: bottleneck bandwidth 4.0 Mbps, one-way delay 60ms and bottleneck buffer size 42 packets. Fig. 5.8 shows the goodputs of NewVeno, NewReno and Westwood+ as a function of Bad state duration

CHAPTER 5. LOSS DISTINGUISHING ACCURACY IN TCP VENO AND ITS PERFORMANCE INFLUENCE

time.

As the bad state period increases, NewVeno shows the performance improvement over NewReno. To explain the performance improvement, we collect the detailed data in Table 5.1. In this table, CL and RL are the actual numbers of congestion losses and random losses occurred. CD and NCD are the numbers of NewVeno congestive drops and non-congestive drops. Note that $CL+RL = CD+NCD = V_{TD}$, where the V_{TD} is the number of NewVeno losses that are indicated by three duplicate ACKs. We don't count partial ACKs because the partial ACKs don't trigger *cwnd* reduction. V_{TO} is the number of NewVeno timeout.

From Table 5.1, we observe that NewVeno experiences very few of TO compared with TD, so that the impact of TO is negligible. In the total number of TD, the CL is very few, which means that congestion rarely occurs. Therefore, RL and thus the random loss distinguishing accuracy (RA) play the key role in NewVeno performance. Based on the conclusion from previous section, increase of random loss rate causes the increase of RA. We can see, when the bad state period increases from 0.001 second to 0.05 second, the random loss accuracy increases from 38% to 100%, which means that NewVeno correctly judges the RL as NCD, so that NewVeno suffers less window cut compared with NewReno. Fig. 5.9 shows the window evolution of NewVeno and NewReno when the bad state period is 0.01 second. NewVeno maintains higher average window size compared with NewReno, thus achieves higher performance.

Fig. 5.10 compares the goodputs of multiple NewVeno and NewReno connections in the wireless scenario. We run three NewVeno connections and three NewReno connections in the wireless scenario. The network parameters are the same as Fig. 5.8. We can see that, when the bad state period is small, NewVeno and NewReno connections have similar goodputs. But when the bad state period exceeds 0.05 second, the goodputs of three NewReno connections decrease quickly. In contrast, NewVeno connections still

CHAPTER 5. LOSS DISTINGUISHING ACCURACY IN TCP VENO AND ITS PERFORMANCE INFLUENCE

Figure 5.10: Multiple flows goodputs.

Figure 5.11: Multiple flows window.

keep high goodputs in the heavy loss environment. For details, in Fig. 5.11 we show the window evolution of NewVeno and NewReno in the scenario of bad state period 0.1 second. In the heavy loss environment, the packet losses are almost random losses rather than congestion losses. NewReno connections blindly drop the window by the factor of $1/2$, resulting in small average window size. In contrast, NewVeno connections correctly distinguish the random losses and drop the window by a factor of $1/5$, and thus achieve higher goodputs. So, as we have seen, in the wireless environment, the loss distinguishing accuracy has significant performance influence. NewVeno achieves performance improvement over NewReno in the wireless environment due to high loss distinguishing accuracy.

5.3.2 Cross and reverse traffic scenario

Now we consider a complex scenario with cross and reverse traffic in the wired networks. The bottleneck delay is 60 ms, and the bandwidths in forward and reverse path are both 4.0 Mbps. The cross traffic is fixed at 50% of the bottleneck bandwidth, while the reverses traffic varies from 10% to 80% following Pareto distribution with shape parameter 1.7. We run NewReno, NewVeno, and Westwood+ on the source node, and report their goodputs.

CHAPTER 5. LOSS DISTINGUISHING ACCURACY IN TCP VENO AND ITS PERFORMANCE INFLUENCE

Figure 5.12: NewVeno, NewReno and Westwood+ in cross and reverse traffic scenario.

Table 5.2: Data collection in cross and reverse traffic scenario

Reverse traffic		10%	20%	30%	40%	50%	60%	80%
V_{TD}	CD	130	128	126	125	111	93	62
	NCD	37	52	69	71	63	53	35
Loss		167	180	195	196	174	146	97
Congestion loss accuracy		0.78	0.72	0.65	0.64	0.64	0.64	0.64

As seen in Fig. 5.12, the goodputs of NewReno, NewVeno and Westwood+ are almost the same under different values of reverse traffic. This observation show that, TCP Veno is robust and performs as well as Reno variants in the reverse traffic scenario. Unlike TCP Vegas that tries to prevent packet loss from happening, TCP Veno makes uses of Vegas’s congestion monitoring scheme as additional information to distinguish the loss type, and hence avoids under-utilization of the bottleneck bandwidth. Table 5.2 lists the detailed data in the simulation. We see that, as the reverse traffic increases, the congestion loss distinguishing accuracy decreases, but the effect of reverse traffic is not significant. This complies with the results in previous section.

CHAPTER 5. LOSS DISTINGUISHING ACCURACY IN TCP VENO AND ITS PERFORMANCE INFLUENCE

Figure 5.13: NewVeno and NewReno good-

Figure 5.14: NewVeno and NewReno win-
dow evolution when the cross traffic is 80%.

Table 5.3: Data collection in the compatibility scenario

Cross traffic		0	10%	20%	30%	40%	60%	80%
V_{TD}	CD	54	85	116	139	152	37	6
	NCD	2	2	5	12	49	93	73
Loss		56	87	121	151	201	130	79
Congestion loss accuracy		0.96	0.97	0.96	0.92	0.76	0.29	0.08
V_{TO}		0	0.3	0.8	3.3	8.5	75	75
R_{TO}		0	0.4	0.4	3.8	5.7	78	88

5.3.3 Compatibility with NewReno.

Now we run NewVeno and NewReno at the same time to study their compatibility issue in the wired network. The UDP cross traffic varies from 0 to 80% of the bottleneck bandwidth. Other parameters are the same as previous case. Fig. 5.13 shows that NewVeno and NewReno co-exist harmoniously in the wired network. We list the detailed data in Table 5.3.

In the wired network, random loss is absent, so that all the TD losses are CL. We focus on the CA analysis. Based on the conclusion in previous section, increase of cross traffic causes decrease of CA. In Table 5.3, we observe in the extreme case that the cross traffic is 80% of the bottleneck bandwidth, CA is only 0.08. This means most of

Figure 5.15: Multiple connections in the wired scenario. Figure 5.16: Multiple connections' window evolution when the cross traffic is 80%.

the congestion losses are wrongly judged as non-congestive drops, and thus the *cwnd* is only cut $1/5$. One may think that NewVeno must be very aggressive due to low CA, and will grab bandwidth from co-existing NewReno connection. But we observe that NewVeno co-exists harmoniously with NewReno without "stealing" bandwidth from NewReno. Note that, the number of TO increases quickly. When the cross traffic is 80%, NewVeno experiences roughly the same number of TD and TO. As a result, TO plays a more important role in NewVeno performance. Fig. 5.14 shows the window evolution of NewVeno and NewReno when the cross traffic is 80%. NewVeno and NewReno windows have been heavily restricted by the frequent TO. So the difference between congestion window reducing by $1/2$ or $1/5$ is insignificant. The reason of the compatibility between NewVeno and NewReno is that they experience similar numbers of TO, and NewVeno inherits the action for TO from NewReno without modification.

Fig. 5.15 shows the scenario that multiple NewVeno and NewReno connections co-exist in the wired network. We run three NewVeno connections and three NewReno connections together and show their goodputs. Other parameters are not changed. We can see that NewVeno and NewReno connections have similar goodputs. As the cross traffic increases, six connections harmoniously co-exist in the wired network. This is

CHAPTER 5. LOSS DISTINGUISHING ACCURACY IN TCP VENO AND ITS PERFORMANCE INFLUENCE

similar to Fig. 5.13. In Fig. 5.16, we show the window evolution of six connections. As can be seen in the figure, NewReno connections always cut half of the *cwnd* for the congestion losses, but NewVeno connections often mistake the congestion losses as random losses and drop the window by a factor of $1/5$. But this doesn't make NewVeno connections' windows aggressive because frequent timeout events heavily restrict the window size. As a result, the NewVeno and NewReno connections co-exist harmoniously in the wired network.

5.4 Summary

In this Chapter, we studied the TCP Veno loss distinguishing accuracy under different network parameters, and illustrated typical scenarios to investigate its performance influence. Based on the simple mathematical analysis and experiment results, we arrive at the conclusion that increase of buffer size (B) has positive impact on congestion loss distinguishing accuracy but negative impact on random loss distinguishing accuracy; increase of cross traffic (C), random loss rate (P) and delay (τ) have negative impact on congestion loss distinguishing accuracy but positive impact on random loss distinguishing accuracy.

Thereafter, we compared NewVeno and NewReno in typical cases to investigate the performance influence of the loss distinguishing accuracy. In the wireless scenario, NewVeno achieves significant performance improvement over NewReno due to high random loss distinguishing accuracy. In the wired network, NewVeno co-exists harmoniously with NewReno, although the loss distinguishing accuracy is very low. In this case, TO takes over TD, and thus plays the key role in TCP performance. NewVeno is compatible with NewReno because they experience similar numbers of TO and their actions for TO are the same. Actually, the loss distinguishing accuracy is very helpful for understanding TCP Veno's behaviors in different network environment, but when we consider the

CHAPTER 5. LOSS DISTINGUISHING ACCURACY IN TCP VENO AND ITS PERFORMANCE INFLUENCE

performance influence of the loss distinguishing accuracy, the impact of timeout should not be ignored.

Chapter 6

Theoretic Analysis of TCP Veno with RED

6.1 Introduction

In this Chapter, we study the dynamics of TCP Veno with the queue management of RED (Random Early Detection) [32]. We make use of the modeling techniques in papers [78][79][80][81][82] to develop a fluid-flow model of TCP Veno with RED in wired/wireless networks. This model is further linearized around the equilibrium point to study TCP Veno's stability issue. By applying the classical linear feedback control theory, the analytical results reveal how the RED queue oscillation is affected by different network parameters such as link capacity, round-trip time. Simulations are carried to validate our theoretical analysis. Furthermore, based on the analysis results, we are able to provide guidelines for tuning RED parameters to stabilize the router queue, and improve the co-existence between TCP Veno and TFRC (TCP-friendly Rate Control) flows.

6.2 Fluid-flow model of TCP Veno with RED

We develop a fluid-flow model of TCP Veno with RED in heterogeneous networks. Our model is composed of two parts. The first part: network and RED queue management

model is the same as [80][81]. We start with a brief description of them, and then in the second part we model the behaviors of TCP Veno's window evolution.

6.2.1 Network and RED queue management model

The network comprises of L links with capacity $c_l, l \in L$. There are I sources indexed by i . Each source i uses a set of links $L_i \subseteq L$. So we have a $L \times I$ routing matrix:

$$M_{li} = \begin{cases} 1 & \text{if } l \in L_i \\ 0 & \text{otherwise} \end{cases}$$

Denote the round-trip time of source i at time t is $R_i(t)$:

$$R_i(t) = T_i + \sum_l M_{li} \frac{q_l(t)}{c_l}$$

where T_i is the round-trip propagation delay of source i and $q_l(t)$ is the instantaneous queue length in link l at time t . Denote the loss probability of link l at time t by $p_l(t)$. Assume that $p_l(t)$ is small, then the end-to-end loss probability at time t , $v_i(t)$, is approximated as

$$v_i(t) = \sum_l M_{li} p_l(t - R_{li}^b(t))$$

where $R_{li}^b(t)$ is the backward delay from link l to source i . Denote the congestion window size of source i at time t by $w_i(t)$, so the sending rate of source i at time t , $x_i(t)$, is :

$$x_i(t) = \frac{w_i(t)}{R_i(t)}$$

Consider that the last-hop link is wireless link with random loss rate γ_i . Therefore, the aggregate arriving rate of link l at time t is:

$$\begin{aligned} y_l(t) &= \sum_i M_{li} x_i(t - R_{li}^f(t)) (1 - \gamma_i) \\ &= \sum_i M_{li} \frac{w_i(t - R_{li}^f(t)) (1 - \gamma_i)}{R_i(t - R_{li}^f(t))} \end{aligned}$$

CHAPTER 6. THEORETIC ANALYSIS OF TCP VENO WITH RED

where $R_{li}^f(t)$ is the forward delay from source i to link l . Note that,

$$R_i(t) = R_{li}^f(t) + R_{li}^b(t)$$

For all links $l \in L$, given the aggregate arriving rate and link capacity, we can calculate the instantaneous queue length $q_l(t)$ derivative ($q_l(t) > 0$) by:

$$\begin{aligned} \dot{q}_l(t) &= y_l(t) - c_l \\ &= \sum_i M_{li} \frac{w_i(t - R_{li}^f)(1 - \gamma_i)}{R_i(t - R_{li}^f)} - c_l \end{aligned} \quad (6.1)$$

The RED queue management calculates the average queue length $r_l(t)$ by the exponential moving average of instantaneous queue length $q_l(t)$ as follows,

$$r_l(k+1) = (1 - \lambda_l)r_l(k) + \lambda_l q_l(k)$$

where λ_l is the queue averaging weight of link l : $0 < \lambda_l < 1$. Consider that the estimate of the average queue length is based on samples taken every $1/c_l$ second, we have,

$$r_l((k+1)/c_l) = (1 - \lambda_l)r_l(k/c_l) + \lambda_l q_l(k/c_l)$$

It is useful to convert this equation into a differential equation. Its natural candidate is:

$$\dot{r}_l(t) = Ar_l(t) + Bq_l(t)$$

Then, in a sampled data system, $r_l(t_{k+1})$ is given by:

$$r_l(t_{k+1}) = e^{A(t_{k+1}-t_k)}r_l(t_k) + \int_{t_k}^{t_{k+1}} e^{A(t_{k+1}-\tau)} B d\tau q_l(t_k)$$

Comparing the coefficients, we obtain:

$$1 - \lambda_l = e^{A/c_l}$$

or

$$A = c_l \log(1 - \lambda_l) = -B$$

So, the differential equation can be expressed as,

$$\dot{r}_l(t) = c_l \log(1 - \lambda_l) r_l(t) - c_l \log(1 - \lambda_l) q_l(t)$$

Denote $\alpha = c_l \log(1 - \lambda_l)$, then the expression can be simplified as,

$$\dot{r}_l(t) = \alpha(r_l(t) - q_l(t))$$

The drop probability $p_l(t)$ is given by:

$$p_l(t) = \begin{cases} 0 & 0 \leq r_l(t) < min_{th} \\ \rho_l(r_l(t) - min_{th}) & min_{th} \leq r_l(t) \leq max_{th} \\ 1 & r_l(t) > max_{th} \end{cases}$$

where min_{th} , max_{th} , and p_{max} are RED minimum threshold, maximum threshold, and maximum loss probability, respectively. ρ_l is expressed as

$$\rho_l = \frac{p_{max}}{max_{th} - min_{th}}$$

We only consider that the system operates in the region $min_{th} \leq r_l(t) \leq max_{th}$, so that the drop probability is,

$$p_l(t) = \rho_l(r_l(t) - min_{th})$$

Differentiating this equation and using the equation of $\dot{r}_l(t)$, we get,

$$\dot{p}_l(t) = -\alpha(p_l(t) + \rho_l min_{th}) + \alpha \rho_l q_l(t) \quad (6.2)$$

6.2.2 Modeling of TCP Veno

Now we model TCP Veno behavior in the additive-increase and multiplicative-decrease phase. At time t , TCP Veno source i sending rate is $x_i(t)$, and ACK receiving rate is $x_i(t - R_i(t))(1 - \gamma_i)(1 - v_i(t - R_i^b))$. Each ACK increases the congestion window by $1/w_i(t)$ or $1/(2w_i(t))$, depending on the network state. We use a random variable $\xi(t)$ to represent the increase factor. $\xi(t)$ may be either 1 or $\frac{1}{2}$. Therefore, the window $w_i(t)$ increases at the rate of $\xi(t)x_i(t - R_i(t))(1 - \gamma_i)(1 - v_i(t - R_i^b))/w_i(t)$.

CHAPTER 6. THEORETIC ANALYSIS OF TCP VENO WITH RED

Figure 6.1: The single bottleneck topology.

On the other hand, there are $x_i(t - R_i(t))(1 - (1 - \gamma_i)(1 - v_i(t - R_i^b)))$ packet losses. Each loss drops $w_i(t)$ by $w_i(t)/2$ or $w_i(t)/5$, depending on the network state. We introduce another random variable $\eta(t)$ to represent the decrease factor. $\eta(t)$ may be either $\frac{1}{2}$ or $\frac{1}{5}$. Therefore, the window $w_i(t)$ decreases at the rate of $\eta(t)x_i(t - R_i(t))(1 - (1 - \gamma_i)(1 - v_i(t - R_i^b)))w_i(t)$. Putting the increase and decrease rate together and taking the mathematical expectation, we get the window average evolution rate as follows. Note that, both γ_i and v_i are small values, so that we can ignore $\gamma_i v_i(t - R_i^b)$.

$$\begin{aligned} \dot{w}_i(t) = & E[\xi] \frac{x_i(t - R_i(t))(1 - \gamma_i)(1 - v_i(t - R_i^b))}{w_i(t)} \\ & - E[\eta] x_i(t - R_i(t))(\gamma_i + v_i(t - R_i^b))w_i(t) \end{aligned} \quad (6.3)$$

The differential equations (6.1), (6.2), and (6.3) constitute the fluid-flow model. It describes the packet-level behaviors of the closed-loop system of TCP Veno with RED in heterogeneous networks.

Note that, some other simplifying assumptions are implied for the fluid-flow model. 1) the slow-start and fast recovery phases in TCP Veno are ignored because these phases have limited effects on long-run TCP performance. 2) the probability of multiple losses in a round-trip time is assumed to be negligible because the loss probability is very low.

We apply the model to the single bottleneck network. With the reference to Fig. 6.1, there are n TCP Veno flows with round-trip time R , the bottleneck capacity is C . Consider a practical case that the delay of the local access link is often much smaller than the

CHAPTER 6. THEORETIC ANALYSIS OF TCP VENO WITH RED

delay of the bottleneck, so that the delay from source to router is negligible. Therefore, we have $R_i^f \approx 0$ and $R_i \approx R_i^b \approx R$. The random loss rate is $\gamma_i = \gamma$. The RED router drop probability is $p(t)$, which equals to $v_i(t)$. Assume that $p(t)$ is small, thus $1 - p(t) \approx 1$. $w(t) \approx x(t - R)$. Therefore, equations (6.1) and (6.3) can be simplified as follows,

$$\dot{w}(t) = E[\xi] \frac{(1 - \gamma)}{R} - E[\eta] \frac{w(t)^2}{R} (\gamma + p(t - R)) \quad (6.4)$$

$$\dot{q}(t) = \frac{w(t)(1 - \gamma)}{R} n - C \quad (6.5)$$

The parameters $E[\xi]$ and $E[\eta]$ are the average increase and decrease factors that we introduce to describe TCP Veno window evolution in AIMD phase. The values of $E[\xi]$ and $E[\eta]$ can be determined as follows,

$$\begin{aligned} E[\xi] &= 1 \times P(N < \beta) + \frac{1}{2} \times P(N \geq \beta) \\ &= P(N < \beta) + \frac{1}{2} \times (1 - P(N < \beta)) \\ &= \frac{1}{2} + \frac{1}{2} \times P(N < \beta) \end{aligned}$$

Similarly, we have,

$$E[\eta] = \frac{1}{2} - \frac{3}{10} \times P(N < \beta)$$

where $P(N < \beta)$ is the probability that Veno's estimation of N (number of backlog packets) is smaller than the threshold β . The exact value of $P(N < \beta)$ is determined by the network state distinguishing accuracy. We know $P(N < \beta) \in [0, 1]$, so that $E[\xi] \in [\frac{1}{2}, 1]$ and $E[\eta] \in [\frac{1}{5}, \frac{1}{2}]$.

6.3 Theoretic analysis of the fluid-flow model

We use the linear feedback control theory to analyze the fluid-flow model. Since this model is non-linear in nature, we need convert it to a linear model, and then study its stability issue.

CHAPTER 6. THEORETIC ANALYSIS OF TCP VENO WITH RED

6.3.1 Linearization and analysis

Similar to [78], the model is linearized around the equilibrium point. Plug the equilibrium point v_0 , $v_0 = (w_0, q_0, p_0)$, in above equations (6.4) and (6.5), we have $\dot{w}|_{v_0} = 0$, and $\dot{q}|_{v_0} = 0$. Therefore, we can get the equilibrium condition as follows,

$$w_0^2 = \frac{E[\xi] (1 - \gamma)}{E[\eta] (\gamma + p_0)} \quad (6.6)$$

$$w_0 = \frac{R_0 C}{n(1 - \gamma)} \quad (6.7)$$

Here, $R_0 = \frac{q_0}{C} + T$, where T is the propagation delay. According to equations (6.4) and (6.5),

$$\dot{w} = \dot{w}(w, p, q)$$

$$\dot{q} = \dot{q}(w, p, q)$$

We can extend $\dot{w}$ and $\dot{q}$ at the equilibrium point v_0 using Taylor series. By ignoring the high-order powers of small values, we get the following equations,

$$\begin{aligned} \dot{w} &= \dot{w}|_{v_0} + \frac{\partial \dot{w}}{\partial w}|_{v_0} \Delta w + \frac{\partial \dot{w}}{\partial q}|_{v_0} \Delta q + \frac{\partial \dot{w}}{\partial p}|_{v_0} \Delta p \\ \dot{q} &= \dot{q}|_{v_0} + \frac{\partial \dot{q}}{\partial w}|_{v_0} \Delta w + \frac{\partial \dot{q}}{\partial q}|_{v_0} \Delta q + \frac{\partial \dot{q}}{\partial p}|_{v_0} \Delta p \end{aligned} \quad (6.8)$$

Therefore, we have,

$$\begin{aligned} \Delta \dot{w} &= \frac{\partial \dot{w}}{\partial w}|_{v_0} \Delta w + \frac{\partial \dot{w}}{\partial q}|_{v_0} \Delta q + \frac{\partial \dot{w}}{\partial p}|_{v_0} \Delta p \\ \Delta \dot{q} &= \frac{\partial \dot{q}}{\partial w}|_{v_0} \Delta w + \frac{\partial \dot{q}}{\partial q}|_{v_0} \Delta q + \frac{\partial \dot{q}}{\partial p}|_{v_0} \Delta p \end{aligned} \quad (6.9)$$

Using the equilibrium condition:

$$\begin{aligned} w_0^2 &= \frac{E[\xi] (1 - \gamma)}{E[\eta] (\gamma + p_0)} \\ w_0 &= \frac{R_0 C}{n(1 - \gamma)} \end{aligned}$$

CHAPTER 6. THEORETIC ANALYSIS OF TCP VENO WITH RED

Note that, R_0 is a function of q_0 . Specifically, $R_0 = T + q_0/C$. Then, we deduce that,

$$\begin{aligned}
 \frac{\partial \dot{w}}{\partial w}|_{v_0} &= -2E[\xi] \frac{n(1-\gamma)^2}{R_0^2 C} \\
 \frac{\partial \dot{w}}{\partial q}|_{v_0} &= \frac{\partial \frac{1}{R_0}}{\partial q} (E[\xi](1-\gamma) - E[\eta]w^2(\gamma + p_0)) = -\frac{1}{R_0^2 C} (E[\xi](1-\gamma) - E[\eta]w^2(\gamma + p_0)) = 0 \\
 \frac{\partial \dot{w}}{\partial p}|_{v_0} &= -E[\eta] \frac{R_0 C^2}{n^2(1-\gamma)^2} \\
 \frac{\partial \dot{q}}{\partial w}|_{v_0} &= \frac{(1-\gamma)}{R_0} n \\
 \frac{\partial \dot{q}}{\partial q}|_{v_0} &= \frac{\partial \frac{1}{R_0}}{\partial q} w_0(1-\gamma)n = -\frac{1}{R_0^2 C} w_0(1-\gamma)n = -\frac{1}{R_0} \\
 \frac{\partial \dot{q}}{\partial p}|_{v_0} &= 0
 \end{aligned}$$

Therefore, the linearized model is as follows,

$$\begin{aligned}
 \Delta \dot{w} &= -2E[\xi] \frac{n(1-\gamma)^2}{R_0^2 C} \Delta w - E[\eta] \frac{R_0 C^2}{n^2(1-\gamma)^2} \Delta p \\
 \Delta \dot{q} &= \frac{(1-\gamma)}{R_0} n \Delta w - \frac{1}{R_0} \Delta q
 \end{aligned} \tag{6.10}$$

where $(\Delta w, \Delta q, \Delta p)$ is the small perturbation at the equilibrium point (w_0, q_0, p_0) . Perform Laplace transform [83] on the linear model and denote $L(\Delta w(t)) = W(s)$, $L(\Delta q(t)) = Q(s)$ and $L(\Delta p(t)) = P(s)$, we can get the following equations,

$$\begin{aligned}
 \frac{W(s)}{P(s)} &= -E[\eta] \frac{R_0 C^2}{n^2(1-\gamma)^2} \frac{e^{-sR_0}}{s + 2E[\xi] \frac{n(1-\gamma)^2}{R_0^2 C}} \\
 \frac{Q(s)}{W(s)} &= \frac{\frac{n(1-\gamma)}{R_0}}{s + \frac{1}{R_0}}
 \end{aligned}$$

According to [78], for RED we have,

$$\frac{P(s)}{Q(s)} = \frac{\rho}{\frac{s}{\alpha} + 1}$$

where ρ and α are previously defined: $\rho = p_{max}/(max_{th} - min_{th})$ and $\alpha = c_l \log(1 - \lambda_l)$. Therefore, TCP Veno with RED in heterogeneous network could be viewed as a

CHAPTER 6. THEORETIC ANALYSIS OF TCP VENO WITH RED

Figure 6.2: Block-diagram of TCP Veno/RED feedback system.

linear feedback control system. Fig. 6.2 shows the block-diagram of the system, which is composed of queue control block, packet drop control block and Veno window control block.

Seeing the queue control block in Fig. 6.2, window perturbation is input while queue perturbation is output. This block transfer function is:

$$G(s)_{queue} = \frac{\frac{n(1-\gamma)}{R_0}}{s + \frac{1}{R_0}} \quad (6.11)$$

As for the packet drop control block, queue perturbation is input while loss rate perturbation is output. This block transfer function is:

$$G(s)_{RED} = \frac{\rho}{\frac{s}{\alpha} + 1} \quad (6.12)$$

The packet loss is the feedback signal, which causes TCP Veno to reduce the congestion window and hence the sending rate. Packet loss is detected at the sender one RTT later. This induces the delay term e^{-sR_0} . Therefore, the transfer function for the Veno window control block is:

$$G(s)_{Veno} = \frac{E[\eta] \frac{R_0 C^2}{n^2 (1-\gamma)^2}}{s + 2E[\xi] \frac{n(1-\gamma)^2}{R_0^2 C}} \quad (6.13)$$

CHAPTER 6. THEORETIC ANALYSIS OF TCP VENO WITH RED

According to the linear feedback control theory, we can use the system's open-loop transfer function to analyze its stability. Denote the open-loop transfer function by $G(s)$. $G(s)$ is a cascade of the transfer functions of all control blocks and delay terms.

$$G(s) = G(s)_{Veno} \times G(s)_{queue} \times G(s)_{RED} \times e^{-sR_0}$$

Plugging in (6.11)(6.12)(6.13), thus we have,

$$G(s) = \frac{E[\eta](R_0C)^3\rho}{2E[\xi]n^2(1-\gamma)^3} \times \frac{e^{-sR_0}}{\left(\frac{s}{E[\xi]\frac{2n}{R_0^2C}(1-\gamma)^2} + 1\right)\left(\frac{s}{1/R_0} + 1\right)\left(\frac{s}{\alpha} + 1\right)} \quad (6.14)$$

Observing equation (6.14), we can conclude that,

1. Increase of random loss rate (γ) has negative effect on system stability because it would increase the open-loop gain, and thus magnify the system perturbation.
2. Increase of link capacity (C) or decrease of flow number (n) has negative effect on system stability because either way would increase the open-loop gain.
3. Increase of round-trip time (R_0) has negative effect on system stability because it would increase the open-loop gain as well as the delay term e^{-sR_0} .

6.3.2 Analysis of relative stability

The relative stability means that, the linear feedback control system must have enough margins of safety before it is driven into unstable region. There are two classical stability margins to measure the system relative stability: *gain margin* and *phase margin*. Gain margin (GM) is the increase in the system gain when phase is -180° , and phase margin (PM) is the amount of phase shift of the system at unity magnitude. According to the control theory, a stable system should have $GM \geq 2$ and $PM \geq 30^\circ$. Therefore, for the system of TCP Veno with RED in heterogeneous networks, we have the following proposition:

Proposition:

CHAPTER 6. THEORETIC ANALYSIS OF TCP VENO WITH RED

Let ρ and α satisfy:

$$\rho_{veno} \leq \frac{2n^2(1-3\gamma)}{(R_0C)^3} \sqrt{\frac{\omega_g^2}{\alpha^2} + 1} \quad (6.15)$$

where $\omega_g = 0.1 \min\{\frac{2n}{R_0^2C}, \frac{1}{R_0}\}$. Then the system is stable and $GM \geq 5\pi$ and $PM \geq 85^\circ$.

Proof:

The system open-loop transfer function is:

$$G(s) = \frac{E[\eta](R_0C)^3\rho}{2E[\xi]n^2(1-\gamma)^3} \times \frac{e^{-sR_0}}{(\frac{E[\xi]\frac{2n}{R_0^2C}(1-\gamma)^2}{s} + 1)(\frac{s}{1/R_0} + 1)(\frac{s}{\alpha} + 1)}$$

By the frequency analysis method, let $s = j\omega$ and consider the frequency point $\omega = \omega_g$, where $\omega_g = 0.1 \min\{\frac{2n}{R_0^2C}, \frac{1}{R_0}\}$. Note that,

$$\frac{\omega_g}{E[\xi]\frac{2n}{R_0^2C}(1-\gamma)^2}j + 1 \approx 1$$

and

$$\frac{\omega_g}{\frac{1}{R_0}}j + 1 \approx 1$$

So, we have,

$$G(j\omega_g) \approx \frac{E[\eta](R_0C)^3\rho}{2E[\xi]n^2(1-\gamma)^3} \cdot \frac{e^{-j\omega_g R_0}}{\frac{j\omega_g}{\alpha} + 1}$$

Accordingly, the system phase angle is:

$$\angle G(j\omega_g) = \angle e^{-j\omega_g R_0} - \angle(j\frac{\omega_g}{\alpha} + 1)$$

For $e^{-j\omega_g R_0}$, we have $\angle e^{-j\omega_g R_0} = -\omega_g R_0 \geq -\frac{180^\circ}{\pi} R_0 \frac{1}{10R_0} \geq -90^\circ$.

For $j\frac{\omega_g}{\alpha} + 1$, we have $\angle(j\frac{\omega_g}{\alpha} + 1) \leq 90^\circ$.

Therefore, the system phase angle is:

$$\angle G(j\omega_g) = \angle e^{-j\omega_g R_0} - \angle(j\frac{\omega_g}{\alpha} + 1) \geq -180^\circ$$

Consider γ is small, $(1 - \gamma)^3 \approx (1 - 3\gamma)$, so we deduce,

$$\begin{aligned} |G(j\omega_g)| &= \frac{E[\eta](R_0C)^3\rho}{E[\xi]2n^2(1 - 3\gamma)} \frac{1}{\sqrt{\frac{\omega_g^2}{\alpha^2} + 1}} \\ &\leq \frac{E[\eta]}{E[\xi]} \\ &\leq 1 \end{aligned}$$

Therefore, the linear feedback control system is stable, and we have seen that $\angle G(j\omega_g) \approx -95^\circ$, so $PM \geq 85^\circ$. To compute the gain margin, we choose $\omega = \frac{\pi}{2R_0}$. We have,

$$\angle G(j\omega) \geq -90^\circ - \frac{180^\circ}{\pi} \frac{\pi}{2R_0} R_0 \geq -180^\circ$$

Then we deduce that,

$$\begin{aligned} 1/|G(j\omega)| &\geq \frac{E[\xi]\omega}{E[\eta]\omega_g} \\ &\geq \frac{E[\xi]}{E[\eta]} 5\pi \\ &\geq 5\pi \end{aligned}$$

Therefore, we have $GM \geq 5\pi$ and $PM \geq 85^\circ$.

6.4 Simulation and validation

We use network simulator NS-2 [63] to validate our analysis, and furthermore, show how to use the theoretical results to guide the co-existence of TCP Veno with TFRC flows. TFRC [33] is a rate-based congestion control mechanism. It continuously measures the round-trip time and packet loss rate, and then uses TCP response function to calculate its sending rate. The TCP response function is expressed as follows [33],

$$T = \frac{s}{R\sqrt{\frac{2p}{3}} + t_{RTO}(3\sqrt{\frac{3p}{8}})p(1 + 32p^2)}$$

CHAPTER 6. THEORETIC ANALYSIS OF TCP VENO WITH RED

Figure 6.3: Instantaneous queue of TCP Veno with RED (loss 1% and $n = 60$), case 1.

where T is the transmission rate in bytes/second, s is the packet size, R is the round-trip time, p is the steady-state loss event rate, and t_{RTO} is the TCP retransmit timeout value. The main characteristics of TFRC are compatibility and smoothness.

- Compatibility means TFRC and TCP flows should fairly share the network capacity when co-existing with each other. TFRC flows should not steal bandwidth from co-existing TCP flows.
- Smoothness means TFRC throughput should not fluctuate drastically. Smoothness is measured by Coefficient of Variation (CoV)¹ of TFRC throughput sampled every 0.1 second. The greater value of CoV means more fluctuation in TFRC throughput.

If the router queue oscillates drastically, the round-trip time measured by TFRC would vary in a large range, and cause TFRC throughput fluctuate drastically. In this section, we also provide guidelines to stabilize the router queue and hence improve the co-existence of TCP Veno with TFRC.

6.4.1 Queue dynamics of TCP Veno with RED

The network topology is shown in Fig. 6.1. The side links are 100 Mbps with 0.1 ms delay, while the bottleneck link is 15 Mbps with 10ms delay. The bottleneck router is

¹Coefficient of Variation is the standard derivation divided by the mean.

CHAPTER 6. THEORETIC ANALYSIS OF TCP VENO WITH RED

Figure 6.4: Instantaneous queue of TCP Veno with RED, case 2.

RED. There are n flows competing on the bottleneck, in which 90% are TCP flows and 10% are TFRC flows. This is to simulate the Internet traffic that is dominated by TCP while TFRC is a relatively small part. We set n to 60, so that the number of TCP flows is 54 and the number of TFRC flows is 6. The random loss rate in the wireless last-hop link is 1% with exponential distribution. Packet size is 1000 bytes and ACK is 40 bytes. The simulation time is 600 seconds. The TCP flows start at 0 second, while TFRC flows join at 300 second when the queue has settled to the operating area.

The RED parameters are: $max_{th} = 250$, $min_{th} = 150$, $p_{max} = 0.1$, and queue averaging weight $\lambda = 0.0001$. Buffer size is 800. Fig. 6.3 shows the instantaneous queue size. We observe that the queue size oscillates in a large range. We record it as experiment case 1. Such oscillation would have negative effect on co-existing TFRC flows. Due to the RED queue oscillation, the round-trip time measured by TFRC would vary in a large range, and hence cause TFRC throughput change drastically. We vary the number of flows n from 10 to 60, which means the number of TFRC flows varies from 1 to 6. For each number of flows, we compute TFRC throughput CoV, and report the average value as well as the maximum and minimum values. Fig. 6.6 shows the results. We can see that the values of TFRC CoV in case 1 reach 0.45.

CHAPTER 6. THEORETIC ANALYSIS OF TCP VENO WITH RED

Figure 6.5: Instantaneous queue of TCP Veno with RED, case 3.

Figure 6.6: CoV of TFRC in three cases.

6.4.2 Effect of RED parameters

For applications which want smoother changes in the sending rate, the TFRC throughput fluctuation should be avoided. Based on our analysis results, here we can tune RED parameters to stabilize TCP Veno/RED router queue.

Consider equation (6.15), to stabilize TCP Veno/RED system, we need decrease the values of ρ , and/or α^2 . Moreover, ρ and α are determined by RED parameters p_{max} , λ , min_{th} , and max_{th} . Therefore, we can tune RED parameters to decrease the values of ρ , and/or α^2 , and thus to stabilize Veno/RED system. We illustrate two methods. One method is that, we decrease p_{max} to 0.05 from 0.1, and λ to 10^{-6} from 10^{-4} . The

CHAPTER 6. THEORETIC ANALYSIS OF TCP VENO WITH RED

Figure 6.7: Throughput of TFRC in three cases.

total flow number n is still 60. We record it as experiment case 2. Fig. 6.4 shows the instantaneous queue length. We observe that the instantaneous queue is more stable than case 1. However, one disadvantage of this method is that, due to the value of λ too small, the RED queue needs a long time to settle to the operating area.

By the second method, we can retain λ to 10^{-4} and p_{max} to 0.1, but increase the $[min_{th}, max_{th}]$ range to $[50, 750]$ from $[150, 250]$. We record it as experiment case 3. Fig. 6.5 shows the instantaneous queue length. We observe that, the queue is quite stable. Meanwhile, the disadvantage in case 2 is absent. Fig. 6.6 shows the CoV of TFRC in case 2 and case 3 when the flow number n varies from 10 to 60. We see that, in two cases, the values of CoV are around 0.2, which is much smaller than case 1.

Fig. 6.7 shows the average throughput of TFRC in experiments case 1, case 2 and case 3. We plot the average throughput as well as the fairness line. The fairness line depicts the value of throughput that the TFRC flow should have if the bottleneck bandwidth is fairly shared by TCP Veno and TFRC flows. It is calculated by the bottleneck bandwidth divided by the total number of TCP Veno and TFRC flows. We see that, in three experiment cases, TFRC throughput is always close to the fairness line. This means TCP Veno and TFRC always fairly share the bottleneck bandwidth. The queue dynamics of RED does not affect their compatibility.

CHAPTER 6. THEORETIC ANALYSIS OF TCP VENO WITH RED

Figure 6.8: Effect of increasing random loss rate.

Figure 6.9: Effect of increasing delay.

6.4.3 Effect of network parameters

Firstly, we investigate how the queue oscillation is affected by random loss rate. According to our above analysis, increase of random loss rate would increase the system open-loop gain, and hence cause system less stable. We increase the random loss rate to 10% from 1%, while other parameters are retained the same as case 2. Fig. 6.8 agrees with the trend of the theoretical analysis. Compare Fig. 6.8 with Fig. 6.4, we observe that the queue oscillates drastically. Meanwhile, the queue size frequently goes down to zero, which means under-utilization of the bandwidth.

Secondly, we investigate how the queue oscillation is affected by the round-trip time.

CHAPTER 6. THEORETIC ANALYSIS OF TCP VENO WITH RED

Figure 6.10: Effect of increasing bandwidth.

Figure 6.11: Effect of decreasing flow number.

According to our analysis, increase of round-trip time would increase the system open-loop gain and delay term, and hence cause system less stable. We increase the delay to 240ms, while other parameters are retained the same as case 2. Fig. 6.9 agrees the trend of the theoretical analysis. Compare Fig. 6.9 with Fig. 6.4, we observe that the queue size frequently goes down to zero, thus causing low utilization of the bandwidth significantly.

Finally, we investigate the effect of bottleneck bandwidth and flow number. Our above analysis point out that the increase of bottleneck bandwidth or decrease of flow number has negative effect on system stability. To validate this, 1) we increase the bandwidth to 40 Mbps from 15 Mbps, while other parameters are retained the same as

CHAPTER 6. THEORETIC ANALYSIS OF TCP VENO WITH RED

case 2. Fig. 6.10 also agrees our theoretical analysis. 2) We decrease the flow number to 40 from 60, while other parameters are retained the same as case 2. Fig. 6.11 shows that the queue oscillates more drastically in this scenario.

6.5 Summary

This Chapter studies the dynamics of TCP Veno with RED queue management. A fluid-flow model is developed to describe the behaviors of TCP Veno in wired/wireless networks. The model is further linearized to study TCP Veno's stability issue. The subsequent analysis points out how the RED queue oscillation is closely related to network link capacity, round-trip time and other network parameters. The simulation results agree with the theoretical analysis.

Chapter 7

An Enhancement of TCP-Friendly Multicast Congestion Control over Wireless Networks

TFMCC is an equation-based TCP-Friendly Multicast Congestion Control protocol. It is designed to use TCP Reno's throughput equation to adjust its transmission rate and thus provides optimal service for multicast streaming applications in the wired Internet environment. However, in the wireless networks, TFMCC, same as Reno, will inevitably suffer significant performance degradation.

In this Chapter, we explore to enhance TFMCC over wireless networks, using an advanced equation to replace Reno's equation. Firstly, this advanced equation is derived through the modeling of TCP Veno. Simulation and real network experiments verify the effectiveness of this analytical model.

Secondly, we use TCP Veno's throughput equation to replace TCP Reno's equation in TFMCC. This enhanced version is thus referred to as TFMCC Veno. The extensive simulations are conducted to evaluate its performance over a large range of network environments. The results demonstrate that , 1) TFMCC Veno can achieve up to 350% performance enhancement over TFMCC, at the environment of 10% random loss rate; 2) the performance improvement is obtained by making better use of the available band-

CHAPTER 7. AN ENHANCEMENT OF TCP-FRIENDLY MULTICAST CONGESTION CONTROL OVER WIRELESS NETWORKS

width rather than stealing the bandwidth from other existing connections. All nice features in TFMCC, that is to say, TCP-friendliness, smoothness, responsiveness, are fully inherited. Moreover, our scheme only needs modification on the end-node without any changes of intermediate router stack, making it easy to deploy in the current Internet.

7.1 Throughput Model of TCP VenO

In this section, we derive the throughput formula for the bulk transfer of TCP VenO flow. A recent work [84] developed a relationship between VenO throughput and the packet loss rate. However, two main factors are not considered in that model [84], 1) the model didn't take into account of the delay increase factor over the window evolution; 2) for computation, the formula also requires an input value of W_{max} (the maximum congestion window during the TCP VenO evolution), which has to be obtained by empirical measurement in advance. In the following part, we make use of a fluid-based approach to remove above two restrictions and derive a more generalized throughput formula. The extensive simulation and real network experiments are also carried to validate this analytical formula.

The fluid-flow model described this section is different from that in previous Chapter in several points. First, Drop-tail queue management is used in this model instead of RED. Second, we will focus on solving the congestive state probability that was not solved in previous Chapter. Third, some symbols such as round-trip time in this section are different from that in previous Chapter. We will review the fluid-flow model for TCP Reno [78][79][80][81], and then derive the throughput formula for TCP VenO.

7.1.1 TCP Reno model

Consider that the network comprises of L links with capacity $c_l, l \in L$. There are I sources indexed by i . Each source i uses a set of links $L_i \subseteq L$. So we have a $L \times I$ routing

CHAPTER 7. AN ENHANCEMENT OF TCP-FRIENDLY MULTICAST CONGESTION CONTROL OVER WIRELESS NETWORKS

matrix $\{R_{li}\}$, where $R_{li} = 1$ if $l \in L_i$, or 0 otherwise. The round-trip time of source i at time t , $\tau_i(t)$, is

$$\tau_i(t) = d_i + \sum_l R_{li} \frac{q_l(t)}{c_l}$$

where d_i is the round-trip propagation delay of source i and $q_l(t)$ is the instantaneous queue in link l at time t . The loss probability of link l at time t is $u_l(t)$ for the drop-tail queue management scheme. $u_l(t) = 1$ if the queue is full, or 0 otherwise. Let $\gamma_l(t)$ be the random loss rate in link l (if $\gamma_l(t) \equiv 0$ for all $l \in L$, our model depicts the wired networks). Consider that the random loss rate is small, then the end-to-end loss probability detected by source i is approximated as

$$p_i(t) = \min(1, \sum_l R_{li} u_l(t - \tau_{li}^b(t)) + \sum_l R_{li} \gamma_l(t - \tau_{li}^b(t)))$$

where $\tau_{li}^b(t)$ is the backward delay from link l to source i . Denote the congestion window size of source i at time t by $w_i(t)$, then sending rate, $x_i(t) = \frac{w_i(t)}{\tau_i(t)}$. Further denote the aggregate arriving rate of link l at time t by $y_l(t)$. For all links where $l \in L$, given the aggregate arriving rate and link capacity, we can calculate the derivative of the queue length $q_l(t)$ where $q_l(t) > 0$ by

$$\begin{aligned} \dot{q}_l(t) &= y_l(t) - c_l \\ &= \sum_i R_{li} x_i(t - \tau_{li}^f(t)) - c_l \end{aligned} \quad (7.1)$$

where $\tau_{li}^f(t)$ is the forward delay from source i to link l . Let b denote the number of packets that are acknowledged by a received ACK (e.g. the receiver sends a cumulative ACK for two consecutive packets received, then $b = 2$). At the sender, the ACK receiving rate is

$$x_i(t - \tau_i(t)) \frac{1 - p_i(t)}{b}. \quad (7.2)$$

CHAPTER 7. AN ENHANCEMENT OF TCP-FRIENDLY MULTICAST CONGESTION CONTROL OVER WIRELESS NETWORKS

7.1.2 Modeling of TCP Veno

The router backlog $N(t)$ can be expressed as

$$N(t) = w_i(t)(1 - d_i/\tau_i(t)). \quad (7.3)$$

This quantity is important to TCP Veno as Veno uses $N(t)$ to estimate the network states and adjust its AIMD algorithm. If $N(t) < \beta$, each received ACK adds $w_i(t)$ by $\frac{1}{w_i(t)}$. However, each packet loss detection adjusts $w_i(t)$ to $\frac{n-1}{n}$ of its original value, where the packet loss rate is given by $x_i(t - \tau_i(t))p_i(t)$. For the case of $N(t) \geq \beta$, each received ACK adds $w_i(t)$ by $\frac{1}{mw_i(t)}$. Besides, each packet loss detection adjusts $w_i(t)$ to $1/2$ of its original value. Given the above, the window evolution rate can be determined by

$$\dot{w}_i(t) = \begin{cases} x_i(t - \tau_i(t))(1 - p_i(t))/(b \cdot w_i(t)) \\ \quad - x_i(t - \tau_i(t))p_i(t)w_i(t)/n, & N(t) < \beta \\ x_i(t - \tau_i(t))(1 - p_i(t))/(b \cdot mw_i(t)) \\ \quad - x_i(t - \tau_i(t))p_i(t)w_i(t)/2, & N(t) \geq \beta \end{cases} \quad (7.4)$$

Similar to [79][81], we linearize (7.4) to study its performance around the equilibrium.

Linearization on (7.4) and applying

$$\lim_{T \rightarrow \infty} \frac{1}{T} \int_0^T \dot{w}_i(t) dt = 0$$

leads to

$$w_0 = \begin{cases} \sqrt{\frac{n}{bp_0}}, & N(t) < \beta \\ \sqrt{\frac{2}{mbp_0}}, & N(t) \geq \beta \end{cases} \quad (7.5)$$

where w_0 , p_0 , and τ_0 are the quantities of $w_i(t)$, $p_i(t)$, and $\tau_i(t)$, respectively. These values are observed at the equilibrium state for flow i . In the above, we also apply $\frac{1}{p_0} - 1 \approx \frac{1}{p_0}$ as p_0 is small implying $\frac{1}{p_0} \gg 1$.

Let $P(N(t) \geq \beta)$ be the probability that $N(t) \geq \beta$, (7.5) becomes

$$w_0 = \sqrt{\frac{n}{bp_0}} - \left(\sqrt{\frac{n}{bp_0}} - \sqrt{\frac{2}{mbp_0}} \right) P(N(t) \geq \beta). \quad (7.6)$$

CHAPTER 7. AN ENHANCEMENT OF TCP-FRIENDLY MULTICAST CONGESTION CONTROL OVER WIRELESS NETWORKS

Figure 7.1: Evolution of $N(t)$.

We now focus on solving for $P(N(t) \geq \beta)$. Fig. 7.1 depicts the typical evolution of $N(t)$ in a period between two consecutive packet loss events. At the start of the period, a packet loss occurs causing VenO to reset its *BaseRTT* and reduce its $w(t)$ resulting $N(t) = 0$. As time progresses, $w(t)$ increases, and backlog is gradually built up on the links first, then at the router's buffer. When $N(t) \geq \beta$, VenO enters the congestive state with the setting $m = 2$ (i.e. a slower increasing rate). Let n_p and n_c be the duration measured in RTT between two consecutive packet loss events and the duration of the congestive state within, then

$$P(N(t) \geq \beta) = \frac{n_c}{n_p}.$$

With a common assumption that each packet is equally likely to drop with probability p_0 [46], the number of packets transmitted within n_p is geometrically distributed with mean of $\frac{1}{p_0}$. We also know that in the equilibrium state, a source transmits $w_0 n_p$ packets within n_p . Hence we get

$$n_p = \frac{1}{w_0 p_0}. \quad (7.7)$$

We now turn our attention to n_c . The quantity of $N(t)$ in the equilibrium state, denoted N_0 , can be computed by

$$N_0 = \frac{S}{n_p} \quad (7.8)$$

CHAPTER 7. AN ENHANCEMENT OF TCP-FRIENDLY MULTICAST CONGESTION CONTROL OVER WIRELESS NETWORKS

where S is the total number of backlogged packets that the router carries during the n_p period. According to Fig. 7.1,

$$S = \frac{1}{2m}n_c^2 + \frac{1}{2}(2n_c + \beta)\beta \approx \frac{1}{2m}n_c^2. \quad (7.9)$$

Practically, β is chosen to be small for effective operation. Hence, S is dominated by the first term in (7.9) which gives the described approximation. Combining (7.7)-(7.9) yields

$$N_0 = \frac{S}{n_p} = \frac{1}{2m}n_c^2 w_0 p_0. \quad (7.10)$$

Linearizing (7.3) in the equilibrium state gives

$$N_0 = w_0 \left(1 - \frac{\tau_{min}}{\tau_0}\right) \quad (7.11)$$

where τ_{min} is the smallest observed RTT (i.e. *BaseRTT*). Solving (7.10) and (7.11) for n_c gives

$$n_c = \sqrt{\frac{2m}{p_0} \left(1 - \frac{\tau_{min}}{\tau_0}\right)}. \quad (7.12)$$

With (7.7) and (7.12), $P(N(t) \geq \beta)$ can be expressed as

$$P(N(t) \geq \beta) = \frac{n_c}{n_p} = w_0 \sqrt{2mp_0 \left(1 - \frac{\tau_{min}}{\tau_0}\right)}. \quad (7.13)$$

The above formula finally yields

$$w_0 = \frac{1}{\sqrt{b} + (\sqrt{2mn} - 2)\sqrt{1 - \tau_{min}/\tau_0}} \cdot \sqrt{\frac{n}{p_0}} \quad (7.14)$$

and B , the Veno's throughput, as

$$B = \frac{w_0}{\tau_0} = \frac{1}{\tau_0(\sqrt{b} + (\sqrt{2mn} - 2)\sqrt{1 - \tau_{min}/\tau_0})\sqrt{\frac{p_0}{n}}}. \quad (7.15)$$

It is easy to see that when applying Reno's setting (i.e. $m = 1, n = 2$), with $b = 1$, (7.14) is reduced to $w_0 = \sqrt{2/p_0}$ which is identical to the earlier results developed for TCP Reno (i.e. (2) in [78] and (13) in [80]).

CHAPTER 7. AN ENHANCEMENT OF TCP-FRIENDLY MULTICAST CONGESTION CONTROL OVER WIRELESS NETWORKS

It is also interesting to evaluate the throughput when VenO detects a random loss and chooses not to decrease its congestion window. This is done by applying $n \rightarrow \infty$. With this application, VenO's throughput becomes

$$B = \frac{1}{\tau_0 \sqrt{2mp_0(1 - \tau_{min}/\tau_0)}}. \quad (7.16)$$

7.1.3 Modeling of TCP VenO with timeouts

Now we extend our analysis to include the effect of Timeouts (TO). Following the approaches given in [81][46], let $Q(w_0)$ be the probability that a loss is a TO loss given that there is a loss event, and $Q(w_0) \approx \min(1, 3/w_0)$. Hence, a TO event occurs with probability $p_0 Q(w_0)$.

The sender transmitting a window of data now needs a period of τ_0 plus a possible timeout period. Given by Padhye *et al.* [46], the timeout period for one TO loss in the system equilibrium state, denoted T_P here, is expressed as

$$T_P = T_0 \frac{1 + p_0 + 2p_0^2 + 4p_0^3 + 8p_0^4 + 16p_0^5 + 32p_0^6}{1 - p_0}$$

where T_0 is the duration of the waiting period for a sender to retransmit an unacknowledged packet, and each following unsuccessful retransmission doubles the waiting period, until $64T_0$ is reached. Applying this extension, the throughput for VenO can be rewritten as

$$B = \frac{w_0}{\tau_0 + w_0 p_0 Q(w_0) T_P}. \quad (7.17)$$

7.1.4 TCP VenO and TCP Reno

In (7.15), we give the throughput formula for TCP VenO, precisely, $B_{VENO} = B(2, 5)$, where

$$B(m, n) = \frac{1}{\tau_0(\sqrt{b} + (\sqrt{2mn} - 2)\sqrt{1 - \tau_{min}/\tau_0})\sqrt{\frac{p_0}{n}}}.$$

CHAPTER 7. AN ENHANCEMENT OF TCP-FRIENDLY MULTICAST CONGESTION CONTROL OVER WIRELESS NETWORKS

Figure 7.2: Settings of m and n for Veno for equal bandwidth sharing between Veno and Reno.

With the above, the throughput for TCP Reno can be obtained simply by $B_{RENO} = B(1, 2)$. When the two flows operate together sharing a common router, to find appropriate settings for Veno that give similar bandwidth utilization between the two flows, we have the condition that $B(m, n) = B(1, 2)$ where m and n correspond to Veno's settings. Assuming same packet loss rate between the two flows, the condition yields

$$m = \frac{\left(\frac{\sqrt{\frac{n}{2}} - 1}{\sqrt{1 - \frac{\tau_{min}}{\tau_0}}} + 2 \right)^2}{2n}. \quad (7.18)$$

The above result describes the relationship between m and n settings of Veno where Veno will utilize the same bandwidth as that of Reno when two flows operate together. In Fig. 7.2, we plot the above result. As can be seen, for a wide range of n , a practical m is between 1 and 2. Veno chooses $m = 2$ as its setting. This allows Veno to share similar bandwidth with Reno when they operate together.

7.1.5 Model validation

We use network simulation tool NS-2 to validate our developed throughput formula (7.17). Fig. 7.3 depicts the simulated network topology. Each source connects to a router using a 10 Mbps link with 0.1 ms propagation delay. The bottleneck link runs

CHAPTER 7. AN ENHANCEMENT OF TCP-FRIENDLY MULTICAST CONGESTION CONTROL OVER WIRELESS NETWORKS

Figure 7.3: The experiment network topology with different values of delay.

Figure 7.4: Single flow varying delay.

Figure 7.5: Single flow varying background traffic.

at 3 Mbps with 40 ms delay. Drop-Tail router with buffer 60 packets is used at the bottleneck link. Destination connections are wireless links with random loss rate varying from 10^{-5} to 10^{-1} following exponential distribution. Packet size is 1000 bytes and ACK is 40 bytes. Simulation time is 200 seconds.

The first experiment considers a single flow with different values of delay. We vary the one-way delay of the flow from 40ms to 320ms. Fig. 7.4 shows that the model matches the experiment results quite well under different values of delay and random loss rate.

The second experiment considers a single flow with UDP background traffic. The UDP traffic follows Pareto distribution (shape parameter 1.7), with burst and idle periods of 50 ms, and rates varying from 0.5 Mbps to 3 Mbps. The one-way propagation delay of TCP Veno flow is fixed at 80 ms. As shown in Fig. 7.5, a good match between the

CHAPTER 7. AN ENHANCEMENT OF TCP-FRIENDLY MULTICAST CONGESTION CONTROL OVER WIRELESS NETWORKS

Figure 7.6: Single flow varying the values of m and n .

Figure 7.7: Mixed multiple flows.

analytical and simulation results is also observed.

In the third experiment, we measure the performance of VenO under various settings of m and n . Fig. 7.6 demonstrates matching results between the analytical and simulation results. Meanwhile, it is also noticed that VenO's performance can be improved when n is a big value.

At last, multiple flows are conducted. Specifically, we consider the same numbers of VenO TCP and SACK TCP flows co-existing in the network. The bottleneck bandwidth is set to 10 Mbps, and the number of total connections varies from 4 to 32. The values of delay for VenO and SACK flows are both uniformly distributed between 40 ms and 160 ms. Fig. 7.7 gives the analytical and simulation results of the VenO flow of 160 ms delay. Obviously, these experimental results confirm the accuracy of our analytical model.

7.1.6 Real network experiments

The topology of the real network is shown in Fig. 7.8. The server is running Linux 2.6 where TCP VenO is implemented, and it is connected to the gateway via Ethernet. The gateway is FreeBSD 4.9 with Dummynet [85] enabled. The receiver side comprises of two clients. The first client is a FreeBSD 4.9 computer connected to the gateway via wired network. The second client is a Compaq laptop with W200D wireless card in a

CHAPTER 7. AN ENHANCEMENT OF TCP-FRIENDLY MULTICAST CONGESTION CONTROL OVER WIRELESS NETWORKS

Figure 7.8: Real network topology.

wireless LAN. The wireless Access Point is Cisco Linksys WRT54G. The bandwidth of the wireless network is 11 Mbps and the frequency is 2.4 GHz.

To validate the formula, we let the server continuously send data to the client. The packet size is 1460 byte. For every 4 Mbytes transmitted data, we calculate the average throughput and packet loss rate during this interval. The values of average round-trip time and minimum round-trip time are also traced and plugged into Veno's throughput formula to calculate the theoretical results. For each experiment case, the theoretical results and 20 experiment results are reported.

1) *Wired network.* The first experiment we considered is TCP Veno transmission over the wired network. The server sends data to the client in the wired network. Dummynet is used to shape the bandwidth to 2 Mbps and the queue size to 10 packets. The traced values of average round-trip time and minimum round-trip time are 0.214 second and 0.165 second, respectively. We plot the theoretical curve and the experiment results in Fig. 7.9. As can be seen, the theoretical curve accurately predicts the experiment results.

2) *Wireless network.* Next, we consider TCP Veno transmission over the wireless network. Fig. 7.10 shows the case that Dummynet shapes the link with 0.01 random packet loss rate, and Fig. 7.11 shows the case that the client is in the wireless LAN. The

CHAPTER 7. AN ENHANCEMENT OF TCP-FRIENDLY MULTICAST CONGESTION CONTROL OVER WIRELESS NETWORKS

Figure 7.9: Results on the wired network.

Figure 7.10: Dummynet case.

Figure 7.11: Wireless LAN.

Figure 7.12: TCP Veno co-existing with TCP Reno flows.

laptop client moves in the campus so that the random loss rate varies in a larger range. As shown in these figures, the theoretical results are quite accurate in both cases.

3) *Co-existence between TCP Veno and TCP Reno flows.* Finally, we consider the case of TCP Veno and TCP Reno co-existing in the network. The server opens a TCP Reno connection to send data to the client FreeBSD 4.9. Meanwhile, the server opens a TCP Veno connection to send data to the laptop client in the wireless LAN. As shown in Fig. 7.12, the matching results also prove the accuracy of TCP Veno throughput formula.

7.2 TFMCC Veno and its Performance Evaluation

7.2.1 Mechanism of TFMCC Veno

With the reference to [17], building an equation-based multicast congestion control mechanism can be divided into following six steps:

- (i) A TCP throughput formula is chosen to control the data transmission rate. The input parameters of this formula must be measurable.
- (ii) The receivers need a filter in the loss event rate measurement so that the measured rate is stable and responsive for the changes of network conditions.
- (iii) The receivers also measure the round-trip time to the sender.
- (iv) Each receiver uses the throughput formula to calculate its expected transmission rate.
- (v) A feedback scheme is designed to prevent the problem of feedback implosion.
- (vi) The sender determines which feedback should be taken into account and then adjusts the transmission rate accordingly.

TFMCC Veno only needs to modify steps (i) and (iv) while other steps in TFMCC keep intact. In TFMCC Veno scheme, each receiver tracks the loss-event rate, round-trip time as well as minimum of round-trip time, and then uses TCP Veno's throughput formula (7.17) to calculate the expected transmission rate. These values are fed back to the sender and it is the sender that determines which feedback should take into account and then adjusts the transmission rate accordingly. The mechanism is illustrated in Fig. 7.13.

CHAPTER 7. AN ENHANCEMENT OF TCP-FRIENDLY MULTICAST CONGESTION CONTROL OVER WIRELESS NETWORKS

Figure 7.13: TFMCC Veno mechanism illustration.

7.2.2 Performance evaluation of TFMCC Veno

We use network simulator NS-2 to evaluate the performance of TFMCC Veno. Extensive simulations are conducted to study its throughput, TCP-friendliness and responsiveness under a large range of network environments.

7.2.2.1 Throughput improvement

The simulation network topology is shown in Fig. 7.3. Each source connects to a router using a 10 Mbps link with 0.1 ms propagation delay. The bottleneck link runs at 2 Mbps with 40 ms delay. Drop-Tail router with buffer 60 packets is used at the bottleneck link. Destination connections are wireless links with 40 ms delay and random loss rate varying from 10^{-4} to 10^{-1} (with exponential distribution). Packet size is 1000 bytes and ACK is 40 bytes. Simulation time is 200 seconds. Each time, only a single flow, e.g., TFMCC, TFMCC Veno($n=5$), TFMCC Veno($n=20$), is tested. For every experiment, we change the seed to repeat 30 times and report the average value as well as the 95% confidence interval.

Fig. 7.14 gives their respective throughput under different lossy situations. It is seen

CHAPTER 7. AN ENHANCEMENT OF TCP-FRIENDLY MULTICAST CONGESTION CONTROL OVER WIRELESS NETWORKS

Figure 7.14: Throughput improvement of TFMCC Venos over TFMCC in the wireless network.

Figure 7.15: Performance evaluation of TFMCC and TFMCC Venos over two-state distribution loss model.

that when the loss rate is small (from 10^{-4} to 10^{-3}), TFMCC, TFMCC Venos have similar throughputs. When the loss rate is increased above 10^{-3} , TFMCC’s throughput decreases quickly. TFMCC Venos exhibits significant throughput improvement in the heavy loss environment. Particularly, TFMCC Venos ($n = 20$) achieves up to 350% throughput improvement over TFMCC in the environment of 0.1 loss rate.

We also studied TFMCC Venos’s performance under two state loss model. Fig. 7.15 gives the throughput performance for TFMCC, TFMCC Venos($n=5$), TFMCC Venos($n=20$) under two-state (Good or Bad) loss distribution model. In our two loss state model,

CHAPTER 7. AN ENHANCEMENT OF TCP-FRIENDLY MULTICAST CONGESTION CONTROL OVER WIRELESS NETWORKS

Figure 7.16: Coefficient of Variation of TFMCC and TFMCC flows in the wireless network.

Figure 7.17: Coefficient of Variation of TFMCC and TFMCC Veno flows with different timescale.

random loss rates in the Good state and the Bad state are set to be 10^{-7} and 10^{-1} respectively. Noted that no matter the wireless channel is in the Good state or in the Bad state, random loss is exponentially distributed. The transition probability between Good state and Bad state is 0.5. The duration of the Good state is deterministic and equals to 1 second, whereas the duration of the Bad state is also deterministic but varies from 10^{-3} second to 1 second. Comparing Fig. 7.15 and Fig. 7.14, the similar results are achieved.

As above mentioned, streaming applications may demand smooth sending rate. We use the metric CoV (Coefficient of Variation) to measure its instantaneous throughput. CoV is defined as standard deviation divided by the mean. The smaller value of CoV

CHAPTER 7. AN ENHANCEMENT OF TCP-FRIENDLY MULTICAST CONGESTION CONTROL OVER WIRELESS NETWORKS

Figure 7.18: Sequence number against time, one TFMCC Veno flow co-exists with four SACK TCP flows, loss rate 0.01.

Figure 7.19: Sequence number against time, five SACK TCP flows, loss rate 0.01.

indicates the smoother changes in the sending rate. The curves in Fig. 7.16 are the average values that are sampled every 0.1 second, while the curves in Fig. 7.17 are the average values that are measured under different sampling time (varying from 0.0001 to 0.1). Both CoV figures indicate that 1) TFMCC Veno’s sending rate is as smooth as that of TFMCC; 2) choosing different values of n doesn’t cause any negative effects on the smoothness of sending rate.

CHAPTER 7. AN ENHANCEMENT OF TCP-FRIENDLY MULTICAST CONGESTION CONTROL OVER WIRELESS NETWORKS

Figure 7.20: Equivalence ratios of TFMCC and TFMCC VenO with SACK TCP, wired network.

7.2.2.2 TCP-friendliness

TCP-friendliness studies whether TFMCC VenO is aggressive or not when competing with legacy TCP. Fig. 7.18 plots the transmitted sequence number of five co-existing flows, one is TFMCC VenO($n=20$) and other four flows are SACK TCP. This test is conducted under 0.01 random loss rate and bottleneck bandwidth is 10 Mbps. After that, we also test five SACK TCP flows under same conditions. Fig. 7.19 plots their sequence evolution diagrams. Comparing these two figures, we can conclude that TFMCC VenO obtains significant performance improvement through the better use of available bandwidth and meanwhile it does not cause any negative effects on other existing flows.

Furthermore, we make use of *equivalence ratio* [33] to study the TCP-friendliness of TFMCC VenO. Suppose that TFMCC VenO and SACK TCP flows co-exist for a duration of time T . This duration can be divided into time intervals $\{t_0, t_2 \dots t_n\}$. In the i th interval t_i , the average throughputs for SACK TCP and TFMCC VenO flows are th_{SACK}^i and th_{TFMCCV}^i , respectively. The *equivalence* in this interval is defined as:

$$e_i = \min\left(\frac{th_{SACK}^i}{th_{TFMCCV}^i}, \frac{th_{TFMCCV}^i}{th_{SACK}^i}\right).$$

Taking the minimum of the two ratios ensures that the resulting value remains between 0 and 1. The equivalence of SACK TCP and TFMCC VenO during the period

Figure 7.21: Equivalence ratios of TFMCC and TFMCC Ven0 under different timescale.

of time T can be characterized by $\{e_0, e_1 \dots e_n\}$. Then, its average value is called the equivalence ratio. The closer it is to 1, the more “equivalent” SACK TCP and TFMCC Ven0 are.

Fig. 7.20 shows the equivalence ratio for TFMCC and TFMCC Ven0 in the wired networks. The numbers of SACK TCP and multicast group receivers vary from 2 to 16. The sampling interval for instantaneous throughput is 0.1 second. It is observed that TFMCC Ven0($n=5$) and TFMCC Ven0($n=20$) have almost same equivalence ratios as TFMCC, which means all of them behave similarly when they compete with SACK TCP flows. Fig. 7.21 shows how the equivalence ratios of TFMCC and TFMCC Ven0 evolve with different sampling intervals. We vary the sampling interval from 0.0001 second to 1 second, while fix the numbers of multicast receivers and SACK TCP flows at 4. The figure shows that, under various sampling intervals, both TFMCC Ven0($n=5$) and TFMCC Ven0($n=20$) keep the same equivalence ratio as TFMCC does. In other words, choosing different values of n does not affect the TCP-friendliness of TFMCC Ven0.

7.2.2.3 Responsiveness

Responsiveness is another important metric for a multicast congestion control protocol. It indicates how fast the protocol react to the changes of network conditions.

CHAPTER 7. AN ENHANCEMENT OF TCP-FRIENDLY MULTICAST CONGESTION CONTROL OVER WIRELESS NETWORKS

Figure 7.22: Responsiveness of TFMCC Veno’s transmission rate, heterogeneous delay and SACK TCP flows join at different time.

Figure 7.23: Responsiveness of TFMCC Veno’s transmission rate, heterogeneous delay and TFMCC Veno joins at different time.

In this test, we set four SACK TCP flows to have one-way delays 60 ms, while multicast group receivers’ delays are between 20 ms and 80 ms with uniform distribution. TFMCC Veno($n=20$) is initiated first and then four SACK TCP flows sequentially join at 20, 40, 60, 80 second, and leave at 100, 120, 140, 160 second. The throughput curves for TFMCC Veno($n=20$) and four SACK TCP flows are plotted in Fig. 7.22. We see that, TFMCC Veno’s throughput decreases responsively as TCP flows join, and increases quickly as TCP flows leave.

Being different from the above figure, Fig. 7.23 shows that four SACK TCP flows are initiated first. TFMCC Veno($n=20$) joins at 50 second. Thereafter, at 120 second, two of the SACK TCP flows leave. The solid curve in the figure shows that TFMCC Veno can quickly grab the available bandwidth and share with other TCP flows fairly despite that the background traffic is changing from time to time. We also used UDP as background traffic to test and observed similar behavior for TFMCC Veno.

7.3 Summary

In this Chapter, we proposed TFMCC Veno, an end-to-end enhancement scheme of TFMCC, for wireless networks. TFMCC is an equation-based TCP-friendly multicast congestion control designed mainly for streaming applications in wired networks. But in the wireless network, TFMCC suffers performance degradation due to the embedded TCP Reno's equation. TFMCC Veno replaces Reno's throughput formula in TFMCC by an advanced throughput equation derived from a wireless TCP - TCP Veno. Extensive simulations results in this Chapter validate that such enhanced TFMCC can achieve up to 350% throughput improvement in wireless networks. Meanwhile, this performance improvement is obtained by making better use of the available bandwidth rather than stealing from other existing connections. All nice features in original TFMCC, namely, TCP-friendliness, smoothness, responsiveness, are fully inherited by TFMCC Veno.

Chapter 8

Conclusions and Future Work

To conclude this thesis, in section 8.1 we briefly summarize the motivations and contributions of our work. Then in section 8.2, we discuss the directions for future work.

8.1 Conclusions

TCP/IP protocol stack and its congestion control mechanisms have been making great success in today's Internet. However, as Internet continues to grow, especially in the face of big wave of wireless access networks, such as wireless local area networks (WLANs), wireless home networks, and cellular networks, the next round of TCP suffering is just coming round the corner. The reason for TCP performance suffering can be summarized in three points.

- The current legacy TCP, namely TCP Reno, is originally designed for the wired networks, where packet loss is often caused by router buffer overflow. Reno's congestion avoidance algorithm takes packet loss as the indication of network congestion and halves its congestion window in order to alleviate network congestions. However, in the wireless networks, packet loss is often induced by high bit error rate or noise on wireless links, rather than congestion (referred to as random loss). Reno's congestion avoidance algorithm lacks the ability of distinguishing random loss from congestion loss, resulting in severe performance degradation.

CHAPTER 8. CONCLUSIONS AND FUTURE WORK

- In Reno's slow-start, sender increases congestion window by one segment upon receiving every new ACK, until congestion window reaches initial slow-start threshold. However, initial *ssthresh* is set before the TCP connection starts, and its value is arbitrary, ranging from 4KB to extremely high. Due to blind initial *ssthresh* setting, TCP suffers low startup performance especially when the network has high values in bandwidth and delay.
- In Reno's fast retransmit and fast recovery algorithm, each invocation of the recovery phase leads to the retransmission of only a single data packet. Problems can arise when multiple packets have been dropped from a single window of data. In such case, recovery phase will be invoked multiple times, and thus will cause congestion window to be reduced multiple times or even time out. This also causes performance degradation for TCP Reno.

Similar to TCP, the problem of performance degradation over wireless network also exists in TCP-friendly multicast congestion control protocols, such as TFMCC. TFMCC makes use of TCP Reno's throughput formula to control its sending rate, so as to provide TCP-like long-term throughput to achieve TCP friendliness. However, due to the nature of TCP Reno in lossy environment, TFMCC, equipped with the Reno equation, will inevitably suffer performance degradation from wireless links.

In recent past, a number of intelligent ideas have been proposed to tackle above issues. We give a literature review in Chapter 2. Based on their experience and our understanding in TCP congestion control, we present our works and contributions in this area. Our work is based on TCP Veno—a sender-side TCP enhancement for eliminating TCP performance suffering over wireless networks, and divided into two parts: for TCP congestion control and for TCP-friendly multicast congestion control.

Contributions for TCP congestion control:

CHAPTER 8. CONCLUSIONS AND FUTURE WORK

- We study TCP Veno's loss distinguishing accuracy with different loss rate, different link delay as well as different cross traffic. Based on the analytical and experimental results, we observe that in most situations, such loss distinguishing accuracy is quite high, but in some other situations, it is very low, especially, when there is heavy cross traffic competing with TCP Veno. The further study points out that the heavy cross traffic leads to more timeout events for a TCP connection and it is this increase of timeout events that makes TCP Veno perform harmoniously with existing legacy TCP, different from our intuition that TCP Veno could have become quite aggressive due to its low loss distinguishing accuracy.
- We study the dynamics of TCP Veno with the queue management of RED. We develop a fluid-flow model to describe the behaviors of TCP Veno in wired/wireless networks. This model is further linearized to study TCP Veno's stability issue through the linear feedback control theory. The analysis points out how the RED queue oscillates under different network parameters such as link capacity, round-trip time. Simulations are carried to validate our theoretical analysis. Furthermore, based on the analysis results, we are able to provide guidelines for tuning RED parameters to stabilize the router queue, and improve the co-existence between TCP Veno and TFRC flows.
- We present a sender-side enhancement that is simple and efficient to improve TCP startup performance in high-speed long-delay networks. The key idea is that it makes use of TCP Vegas congestion-monitoring scheme to monitor the router queue, and accordingly refines slow-start window evolution to quickly reach the eligible window size, meanwhile avoiding multiple packet losses. Simulation results demonstrate that, compared with traditional slow-start and many other enhancements, this idea can significantly improve link utilization without adversely affecting co-existing TCP connections.

CHAPTER 8. CONCLUSIONS AND FUTURE WORK

Table 8.1: Summary of TCP-friendly multicast congestion control schemes

	Category	End-to-end	Single rate	Wireless
MTCP	Window-based	No	Yes	No
PGMCC	Window-based	Yes	Yes	No
EPGMCC	Window-based	Yes	Yes	Yes
TFMCC	Rate-based	Yes	Yes	No
TFMCC Ven0	Rate-based	Yes	Yes	Yes

- We investigate the enhancement of TCP Ven0 with Forward ACK, called TCP Ven0-F. TCP Ven0-F aims to improve Ven0’s performance through better recovery efficiency. Our network simulation results demonstrate that, 1) TCP Ven0-F improves TCP performance by better utilization of link bandwidth; 2) TCP Ven0-F co-exists with legacy TCP harmoniously in the wired networks; 3) TCP Ven0-F deals with a large range of network scenarios such as heterogeneous network, RED queue management.

Contributions for TCP-friendly multicast congestion control:

- We propose and study the idea of enhancing TFMCC by equipping an advanced throughput formula derived from TCP Ven0 in order to eliminate the performance suffering by Reno’s formula. We develop a generalized formula for Ven0’s throughput, and then use this formula to replace Reno’s formula in TFMCC, called TFMCC Ven0. Extensive simulations results demonstrate that, 1) TFMCC Ven0 achieves significant performance improvement in the wireless networks. Specifically, up to 350% performance enhancement is achieved in the environment of 10% random loss rate. 2) The performance improvement is obtained by making better use of the bandwidth wasted by other connections rather than aggressiveness, so that the nice features in original TFMCC, namely, TCP-friendliness, smoothness, responsiveness, are fully inherited. Moreover, our scheme only needs modification on the

end-node without any changes of intermediate router stack, making it easy to deploy in the current Internet. We conclude the differences between TCP-friendly multicast congestion control schemes in Table. 8.1.

8.2 Future work

There are several interesting avenues of future work which we intend exploring.

- **Time-varying non-linear control theoretical analysis of the stability issue for TCP Veno with RED.**

In this thesis, we have studied the dynamics of TCP Veno with RED using classic linear feedback control theory. This method linearizes the network model around the equilibrium point to study its steady-state stability issue. However, the linear feedback control theory has some limitations. For instance, it cannot depict the instantaneous stability of the network in the startup phase. This issue is important because many Internet applications such as HTTP are short transfers so that the transfer is finished before the system has settled into the steady state.

In the future, we plan to use the modern Time-varying non-linear control theory to give a comprehensive study for the stability issue of TCP Veno with RED. By this method, network parameters such as Veno's congestion window, round-trip time, and router queue, are treated as time-varying functions so that we are able to model the network during the startup phase. Moreover, this method can point out how long it takes that the system settles to the steady state.

- **A theoretical analysis of Veno's window evolution parameters.**

TCP Veno introduces two parameters m —delay increase factor, and n —window drop factor, to adjust its window evolution under different network states. During

the congestive state, Veno's window is increased by $1/cwnd$ for every m ACKs received, while during the non-congestion state, Veno's window is dropped by an amount of $1/n$ only. Normally, m is set as 2 and n is set as 5, based on empirical experiments. These settings also affect TFMCC Veno, which is similar to TCP Veno in long-term throughput. In this thesis, we have shown that increase of n can achieve higher throughput while maintain TCP-friendliness without negative effects. In the future, we plan to give a theoretical analysis for the best choice of m and n .

- **Extending TFMCC Veno to multi-rate multicast congestion control schemes.**

In this thesis, we have proposed and studied TFMCC Veno for single-rate multicast congestion control schemes. In the single-rate scheme, all the receivers get the same transmission rate that is allowed by the slowest receiver. But for the heterogeneous receiver groups that different receivers require different transmission rates, multi-rate schemes are suitable. There are two major approaches for multi-rate transmissions: replication and layered multicast [43].

In recent years, an active research area is to extend TFMCC to multi-rate schemes for providing TCP-friendly multi-rate congestion control for heterogeneous receiver groups. Some work, such as SMCC [86][87] has made remarkable progress in this area. However, there is no work on wireless multi-rate congestion control schemes. In the future, we plan to extend TFMCC Veno to the multi-rate multicast congestion control scheme.

- **Veno II: a uniform transport platform for next generation networks.**

In today's Internet, three obvious evolutions are happening in the network architecture. First, in the application layer, streaming transfers are becoming as important as bulk data transfers. Second, in the data link layer, wireless networks are playing

CHAPTER 8. CONCLUSIONS AND FUTURE WORK

Figure 8.1: Architecture of the Uniform transport platform.

a more and more important role. Third, IPv4 platform is broken into a joint layer of IPv4 and IPv6 as IPv4 cannot meet the need of IP addresses. This architecture evolution is presented in Fig. 8.1.

In the future, our team plans to explore a uniform transport platform in TCP layer. This platform hides network heterogeneity downwards, and provides both data transfer and streaming transfer upwards. Based on our work in TCP Veno and TFMCC Veno, we are considering integrating these schemes together. Moreover, some works in addressing and routing need further research efforts.

References

- [1] V. Cerf and R. E. Kahn, “A Protocol for Packet Network Interconnection,” *IEEE Transactions on Communications*, vol. COM-22, pp. 637–648, 1974.
- [2] V. Cerf, Y. Dalal, and C. Sunshine, “Specification of Internet Transmission Control Program,” *RFC 675*, 1974.
- [3] V. Cerf, “TCP Version 2 Specification,” *IEN 5*, March 1977.
- [4] J. Postel, “Comments on Internet Protocols and TCP,” *IEN 2*, August 1977.
- [5] V. Cerf, “TCP Specification 3,” *IEN 21*, January 1978.
- [6] J. Postel, “DOD Standard Transmission Control Protocol,” *RFC 761*, January 1980.
- [7] V. Jacobson, “Congestion Avoidance and Control,” in *ACM SIGCOMM '88*, pp. 314–329, August.
- [8] V. Jacobson, “Modified TCP Congestion Avoidance Algorithm,” *end2end-interest mailing list*, April 1990.
- [9] J. Postel, “Internet Protocol: DARPA Internet Program Protocol Specification,” *RFC 791*, September 1981.
- [10] J. Nagle, “Congestion Control in IP/TCP Internetworks,” *RFC 896*, January 1984.
- [11] M. Allman, V. Paxson, and W. Stevens, “TCP Congestion Control,” *RFC 2581*, April 1999.
- [12] S. Floyd, “Congestion Control Principles,” *RFC 2914*, September 2000.

REFERENCES

- [13] S. Floyd and K. Fall, "Promoting the Use of End-to-end Congestion Control in the Internet," *IEEE/ACM Transactions on Networking*, vol. 7, pp. 458–472, August 1999.
- [14] D. M. Chiu and R. Jain, "Analysis of the Increase and Decrease Algorithms for Congestion Avoidance in Computer Networks," *Journal of Computer Networks and ISDN*, vol. 17, pp. 1–14, June 1989.
- [15] M. Allman and V. Paxson, "On Estimating End-to-end Network Path Properties," in *ACM SIGCOMM '99*, pp. 263–274, September 1999.
- [16] I. Rhee, V. Ozdemir, and Y. Yi, "TEAR: TCP Emulation at Receivers – Flow Control for Multimedia Streaming," *Technical Report, Department of Computer Science, North Carolina State University*, April 2000.
- [17] J. Widmer and M. Handley, "Extending Equation-based Congestion Control to Multicast Applications," in *ACM SIGCOMM '01*, pp. 275–285, August 2001.
- [18] S. J. Golestani and K. K. Sabnani, "Fundamental Observations on Multicast Congestion Control in the Internet," in *IEEE INFOCOM '99*, pp. 990–1000, March 1999.
- [19] I. Rhee, N. Balaguru, and G. N. Rouskas, "MTCP: Scalable TCP-like Congestion Control for Reliable Multicast," in *IEEE INFOCOM '99*, pp. 1265–1273, March 1999.
- [20] L. Rizzo, "PGMCC: a TCP-friendly single-rate multicast congestion control scheme," in *ACM SIGCOMM '00*, 2000.
- [21] L. Brakmo, S. O'Malley, and L. Peterson, "TCP Vegas: New Techniques for Congestion Detection and Avoidance," in *ACM SIGCOMM '94*, pp. 24–35, August 1994.
- [22] C. P. Fu and S. C. Liew, "TCP Veno: TCP Enhancement for Transmission over Wireless Access Networks," *IEEE Journal on Selected Areas in Communications*, vol. 21, pp. 216–228, February 2003.

REFERENCES

- [23] S. Mascolo, C. Casetti, M. Gerla, M. Y. Sanadidi, and R. Wang, "TCP Westwood: Bandwidth Estimation for Enhanced Transport over Wireless Links," in *ACM MobiCom '01*, pp. 287–297, July 2001.
- [24] S. Floyd, "HighSpeed TCP for Large Congestion Windows," *RFC 3649*, December 2003.
- [25] J. Liu and S. Singh, "ATCP: TCP for Mobile AD Hoc Networks," *IEEE journal On Selected Areas in Communications*, vol. 19, pp. 1300–1315, July 2001.
- [26] K. Xu, Y. Tian, and N. Ansari, "TCP-Jersey for Wireless IP Communications," *IEEE Journal on Selected Areas in Communications*, vol. 22, pp. 747–756, May 2004.
- [27] S. Floyd and T. Henderson, "The NewReno Modification to TCP's Fast Recovery Algorithm," *RFC 2582*, April 1999.
- [28] M. Mathis, J. Mahdavi, S. Floyd, and A. Romanow, "TCP Selective Acknowledgment Options," *RFC 2018*, October 1996.
- [29] M. Mathis and J. Mahdavi, "Forward Acknowledgement: Refining TCP Congestion Control," in *ACM SIGCOMM '96*, pp. 281–291, August 1996.
- [30] R. Wang, K. Yamada, M. Y. Sanadidi, and M. Gerla, "TCP with Sender-side Intelligence to Handle Dynamic, Large, Leaky Pipes," *IEEE Journal on Selected Areas in Communications*, vol. 23, pp. 235–248, February 2005.
- [31] S. Floyd, "Limited Slow-Start for TCP with Large Congestion Windows," *RFC 3742*, March 2004.
- [32] S. Floyd and V. Jacobson, "Random Early Detection Gateways for Congestion Avoidance," *IEEE/ACM Transactions on Networking*, vol. 1, pp. 397–413, August 1993.
- [33] S. Floyd, M. Handley, J. Padhye, and J. Widmer, "Equation-based Congestion Control for Unicast Applications," in *ACM SIGCOMM '00*, pp. 43–56, August 2000.

REFERENCES

- [34] M. Handley, S. Floyd, J. Padhye, and J. Widmer, "TCP Friendly Rate Control (TFRC): Protocol Specification," *RFC 3448*, January 2003.
- [35] E. H. Wu and M. Z. Chen, "JTCP: Jitter-Based TCP for Heterogeneous Wireless Networks," *IEEE Journal on Selected Areas in Communications*, vol. 22, pp. 757 – 766, May 2004.
- [36] S. Biaz and N. Vaidya, "Discriminating congestion losses from wireless losses using interarrival times at the receiver," *IEEE Symp. Application-Specific Systems and Software Engineering and Technology*, pp. 10–17, March 1999.
- [37] C. Caini and R. Firrincieli, "TCP Hybla: a TCP enhancement for heterogeneous networks," *Int. J. Satell. Commun. Network*, vol. 22, no. 547 - 566, 2004.
- [38] J. Hoe, "Startup Dynamics of TCP's Congestion Control and Avoidance Schemes," *Master Thesis of Massachusetts Institute of Technology*, 1995.
- [39] I. F. Akyildiz, G. Morabito, and S. Palazzo, "TCP Peach: A New Congestion Control Scheme for Satellite IP Networks," *IEEE/ACM Transactions on Networking*, vol. 9, pp. 307–321, June 2001.
- [40] Linux Kernel, available at <http://www.kernel.org/>.
- [41] S. Floyd, "TCP and explicit congestion notification," *ACM Comput. Commun. Rev*, vol. 24, pp. 10–23, October 1994.
- [42] J. C. Hoe, "Improving The Start-up Behavior of A Congestion Control Scheme for TCP," in *ACM SIGCOMM '96*, pp. 270–280, August 1996.
- [43] V. K. Li and Z. Zhang, "Internet Multicast Routing and Transport Congestion Control," *Proceedings of the IEEE*, vol. 90, pp. 360 – 391, March 2002.
- [44] Y. Liu, C. P. Fu, Z. K. Yang, and B. S. Lee, "An enhancement of multicast congestion control over hybrid wired/wireless networks," *IEEE WCNC '04*, vol. 4, 2004.
- [45] Y. Liu, C. P. Fu, Z. K. Yang and B. S. Lee, "Multicast congestion control scheme over wireless access links: problem and enhancement," *Int. J. Internet Protocol Technology*, vol. 1, no. 1, pp. 40 – 49, 2005.

REFERENCES

- [46] J. Padhye, V. Firoiu, D. Towsley, and J. Krusoe, "Modeling TCP Throughput: A Simple Model and its Empirical Validation," in *ACM SIGCOMM '98*, pp. 303–314, September 1998.
- [47] K. Fall and S. Floyd, "Simulation-based comparisons of Tahoe, Reno and SACK TCP," *Computer communication review*, July 1996.
- [48] M. S. Taqqu, W. Willinger, and R. Sherman, "Proof of a Fundamental Result in Self-Similar Traffic Modeling," *Computer Communication Review*, pp. 5–23, April 1997.
- [49] L. A. Grieco and S. Mascolo, "Performance Evaluation and Comparison of Westwood+, New Reno, and Vegas TCP Congestion Control," *ACM Computer Communication Review*, vol. 34, pp. 25–38, April 2004.
- [50] C. Barakat and E. Altman, "Bandwidth tradeoff between TCP and link level FEC," *Computer Networks*, pp. 133–150, 2002.
- [51] Westwood homepage: <http://www.cs.ucla.edu/NRL/hpi/tcpw/>.
- [52] R. Wang, G. Pau, K. Yamada, M. Y. Sanadidi and Mario Gerla, "Tcp startup performance in large bandwidth delay networks," *IEEE INFOCOM '04*, March 2004.
- [53] K. Yamada, R. Wang, M.Y. Sanadidi and M. Gerla, "Tcp westwood with agile probing: Dealing with dynamic, large, leaky pipes," *IEEE ICC '04*, 2004.
- [54] M. Gerla, B. K. F. Ng, M. Y. Sanadidi, M. Valla and Ren Wang, "Tcp westwood with adaptive bandwidth estimation to improve efficiency/friendliness tradeoffs," *Computer Communications*, vol. 27, no. 1, pp. 41–58, 2004.
- [55] R. Kapoor, M.Y. Sanadidi and Mario Gerla, "An analysis of bluetooth scatternet topologies," *IEEE ICC '03*, 2003.
- [56] G. Yang, Ren Wang, M. Y. Sanadidi and Mario Gerla, "Performance of tcpw br in next generation wireless and satellite networks," *IEEE ICC '03*, 2003.

REFERENCES

- [57] A. Balk, D. Maggiorini, M. Gerla and M. Y. Sanadidi, "Adaptive mpeg-4 video streaming with bandwidth estimation," *Computer Networks on QoS in Multiservice IP Networks*, vol. 44, no. 4, pp. 415–439, 2004.
- [58] S. Shao, M. Y. Sanadidi and Mario Gerla, "A simulation study of the interoperability of tcpw with red and ecn," *SPECTS*, 2003.
- [59] E. Lengliz, H. Touati, F. Kamoun and M. Y. Sanadidi, "Experimentations towards tcp westwood application," *Adhoc Mobile Networks Med-Hoc Net*, 2003.
- [60] M. Luglio, M. Sanadidi, J. Stepanek and M. Gerla, "The combined use of on-board peps and of efficient schemes to improve tcp performance in a satellite environment," *First International Conference on Advanced Satellite Mobile Systems*, 2003.
- [61] Hybla homepage: <http://didattica.arces.unibo.it/index.php?dbName=rfirrincieli>.
- [62] S. Biaz and H. Vaidya, "Distinguishing Congestion Losses from Wireless Transmission Losses: A Negative Result," *Seventh International Conference on Computer Communications and Networks (IC3N)*, October 1998.
- [63] The Network Simulator, available at <http://www.isi.edu/nsnam/ns/>.
- [64] NS-2 Manual, available at http://www.isi.edu/nsnam/ns/doc/ns_doc.pdf.
- [65] C. P. Fu, W. Lu, and B. S. Lee, "TCP Veno Revisited," in *IEEE GLOBECOM '03*, pp. 3253–3258, December 2003.
- [66] K. Zhang and C. P. Fu, "An Enhancement of TCP Veno with Forward Acknowledgement," *IEEE ICON*, 2006.
- [67] Q. X. Pang, S. C. Liew, C. P. Fu, W. Wang, and V. O. K. Li, "Performance study of TCP Veno over WLAN and RED router," in *IEEE GLOBECOM '03*, pp. 3237–3241, December 2003.
- [68] C. L. Zhang, C. P. Fu, M. T. Yap, C. H. Foh, K. K. Wong, C. T. Lau, and M. K. Lai, "Dynamics Comparison of TCP Veno and Reno," in *IEEE GLOBECOM '04*, pp. 1329–1333, December 2004.

REFERENCES

- [69] Z. X. Zou, B. S. Lee, and C. P. Fu, "Packet loss and congestion state in TCP Reno," in *IEEE ICON '04*, pp. 731–735, November 2004.
- [70] K. Zhang and C. P. Fu, "The performance study of tcp veno over different recovery schemes," *APCC '06*, 2006.
- [71] K. Zhang and C. P. Fu, "Dynamics analysis of tcp veno with red," *Computer Communications*, vol. 30/18, pp. 3778–3786, December 2007.
- [72] K. Zhang and C. P. Fu, "Loss distinguishing accuracy in tcp veno and its performance influence," *IEEE Globecom '06*, 2006.
- [73] K. Zhang and C. P. Fu, "Fluid-based analysis of tcp veno with red," *IEEE Globecom '06*, 2006.
- [74] K. Zhang, C. P. Fu, Z. H. Man and C. H. Foh, "Queue dynamics analysis of tcp veno with red," *IEEE WCNC '07*, 2007.
- [75] B. Zhou, C. P. Fu, and K. Zhang, "An enhancement of tcp veno over light-load wireless networks," *IEEE Communications Letters*, vol. 10, no. 6, 2006.
- [76] Z. X. Zou, C. P. Fu and B. S. Lee, "A refinement to improve tcp veno performance with bursty congestion," *IEEE Globecom '05*, 2005.
- [77] C. B. Fu, J. L. Wang, C. P. Fu and K. Zhang, "Performance study of tcp veno in wireless/asymmetric links," *International conference on Cyberworlds*, 2004.
- [78] C. V. Hollot, V. Misra, D. Towsley, and W.-B. Gong, "A Control theoretic analysis of RED," in *IEEE INFOCOM '01*, pp. 1510 – 1519, 2001.
- [79] S. H. Low, F. Paganini, J. Wang, S. Adlakha, and J. C. Doyle, "Dynamics of TCP/RED and a scalable control," in *IEEE INFOCOM '02*, pp. 239–248, 2002.
- [80] J. Chen, F. Paganini, R. Wang, M. Y. Sanadidi, and M. Gerla, "Fluid-flow analysis of TCP Westwood with RED," in *IEEE Globecom '03*, pp. 4064 – 4068, 2003.

REFERENCES

- [81] V. Misra, W. B. Gong, and D. F. Towsley, "Fluid-based Analysis of a Network of AQM Routers Supporting TCP Flows with an Application to RED," in *ACM SIGCOMM '00*, pp. 151–160, August 2000.
- [82] S. Low, "A Duality Model of TCP and Queue Management Algorithms," *IEEE/ACM Transactions on Networking*, vol. 11, pp. 525–536, August 2003.
- [83] M. Abramowitz and I. A. Stegun, "Laplace Transforms," in *Handbook of Mathematical Functions with Formulas, Graphs, and Mathematical Tables*, pp. 1019–1030, 1972.
- [84] B. Zhou, C. P. Fu, D. H. Chiu, C. T. Lau, and L. H. Ngoh, "A simple throughput model for TCP Venet," in *IEEE ICC '06*, pp. 5395 – 5400, 2006.
- [85] Dummynet, available at http://info.iet.unipi.it/~luigi/ip_dummynet.
- [86] G.-I. Kwon and J. Byers, "Smooth multirate multicast congestion control," *IEEE INFOCOM' 03*, vol. 2, pp. 1022 – 1032, March 2003.
- [87] G.-I. Kwon and J. Byers, "Leveraging single rate schemes in multiple rate multicast congestion control design," *Selected Areas in Communications, IEEE Journal on*, vol. 22, pp. 1975 – 1986, Dec 2004.

Publications

- [1] K. Zhang, C. P. Fu, "An enhancement of TCP Veno with Forward Acknowledgement", Computer Communications (2008).
- [2] K. Zhang, C. P. Fu, "Fluid-based Modeling of TCP Veno", Accepted by IEEE Globecom 2008.
- [3] K. Zhang and C. P. Fu, "Dynamics analysis of TCP Veno with RED" Computer Communications (2007), Vol 30/18 pp 3778-3786; doi: 10.1016/j.comcom.2007.09.004.
- [4] K. Zhang, C. P. Fu, Z. H. Man, and C. H. Foh, "Queue dynamics analysis of TCP Veno with RED", IEEE WCNC 2007, Hong Kong.
- [5] K Zhang and C. P. Fu, "Fluid-based analysis of TCP Veno with RED", IEEE Globecom 2006, San Francisco, USA
- [6] K. Zhang and C. P. Fu, "Loss distinguishing accuracy in TCP Veno and its performance influence" IEEE Globecom 2006, San Francisco, USA.
- [7] K. Zhang and C. P. Fu, "An enhancement of TCP Veno with Forward Acknowledgement", IEEE ICON 2006, Singapore.
- [8] K. Zhang and C. P. Fu, "The performance study of TCP Veno over different recovery schemes", APCC 2006, Pusan, South Korea.
- [9] B. Zhou, C. P. Fu, and K. Zhang, "An Enhancement of TCP Veno over Light-Load Wireless Networks," IEEE Communications Letters Vol. 10 No. 6, 2006"
- [10] C. B. Fu, J. L. Wang, C. P. Fu, and K. Zhang, "Performance Study of TCP Veno in Wireless/Asymmetric Links," International conference on Cyberworlds, 2004