Path capacity measurement and congestion control in heterogeneous network

Zou, ZiXuan

2006

Zou, Z. X. (2006). Path capacity measurement and congestion control in heterogeneous network. Doctoral thesis, Nanyang Technological University, Singapore.

<https://hdl.handle.net/10356/2596>

<https://doi.org/10.32657/10356/2596>

Nanyang Technological University

Downloaded on 26 Apr 2023 16:52:00 SGT

Nanyang Technological University

Path Capacity Measurement and
Congestion Control in
Heterogeneous Network

Zou ZiXuan

School of Computer Engineering

2006

Path Capacity Measurement and Congestion Control in Heterogeneous Network

Zou ZiXuan

School of Computer Engineering

A thesis submitted to the
Nanyang Technological University
in fulfilment of the requirement for the degree of
Doctor of Philosophy
of
Computer Engineering

April 2006

Abstract

The Internet is an open and heterogeneous architecture that consists of different transmission medium, diverse platforms and applications. Network capacity has increased tremendously over the past decade. However, network applications and users have also grown tremendously. The end-to-end throughput of an application is dependent on the bottleneck link capacity along the path as well as the application transport protocol.

In our research we have firstly focused on how to determine the path capacity (the capacity of bottleneck link). Active measurement techniques like Packet Pair (which derive from the principle of self-clocking in TCP's congestion control) have to be used to determine the path capacity. One of the major issues is maximizing the statistical correctness for measurement result. We proposed a new end-to-end based measurement technique, called Packet Triplet, to achieve a higher probability of finding the path capacity. A new filtering technique is introduced into Packet Triplet to further refine such measurement approach. It has shown better performance in terms of accuracy even in heavily loaded networks.

Next, we investigated the performance of TCP transport protocol. TCP is widely accepted as the transport layer protocol in today's Internet. It makes up a large proportion of the traffic on the Internet. However, with the diversity of last mile network characteristics, e.g. wireless environment, the efficiency and effectiveness of the protocol have been greatly affected. This is especially so in the large and pervasive deployment of wireless access network.

The great success of TCP is largely due to its end-to-end congestion control scheme, which is designed for avoiding congestion collapse and achieving fairness

and friendliness with respect to coexistent TCP flows. TCP's congestion control scheme was originally designed based on a determinate conclusion that packet loss is the reasonable indicator of congestion, which is true in wired network. However, this conclusion becomes uncertain in wireless environment where random losses due to high bit error or other non-congestion reasons tend to be more prevalent than wired network. This results in severe TCP throughput performance degradation.

Our research work studied the cause of packet loss and the estimated queued packets (to estimate current state of network congestion) along the path in TCP's congestion control mechanism. We choose TCP Veno, which is a refinement designed for better TCP performance in heterogeneous environment, to be the object of our study. TCP Veno uses packet loss as well as congestion state, based on queued packet in the path, as its congestion control signals. From our investigation, it was observed that sometimes the estimated queued packets can veritably reflect the correlation between packet losses due to congestion and congestion state. Thus, TCP variants like TCP Veno have used the congestion state estimation for congestion control. However, the accuracy in determining the cause of congestion loss using congestion state estimate is low when the network is heavily loaded. We revealed that the problem is due to congestion state underestimation.

However, this problem has minimal impact on the performance of TCP Veno in term of TCP friendliness. A more important observation is that TCP Veno's overall performance is not widely impacted by such low accuracy of congestion loss identification and thus, continues to work harmoniously and efficiently with other TCP connections even when the network is highly congested.

Another factor that contributes to the inaccuracy is bursty congestion, which can adversely affect TCP Veno's performance in congestion loss identification especially when the network is lightly loaded. The reason is that the state estimation responds too slowly to the occurrence of bursty congestion. Based on our investigation, we have proposed a refined TCP Veno, called TCP Veno enhancement, to give a more accurate classification of packet loss in the presence of bursty congestion. Two parameters (a smoothed round trip time called $VRTT_{inst}$ and number of queued packets called N_{inst}) are used in the classification of packet loss. The accuracy of TCP Veno enhancement's congestion loss classification has been improved by as much as 60%, as compared with that of TCP Veno. This has also resulted in friendliness improvement to other TCP variants, e.g. TCP Reno.

Acknowledgements

There are a number of people whom I would like to take this opportunity to thank for without their great help, support and encouragement, I would not have got this thesis accomplished.

First of all, I would like to express my gratitude to my supervisor, Associate Professor. Lee Bu Sung, for taking me as his student and leading me into the research area of network performance measurement, and for his constant support through my postgraduate education. No matter how busy Dr Lee was, he was always there to offer me insightful suggestions, to help find solutions for problems with his broad knowledge, to correct spelling and stylistic errors in papers, to teach me how to be a good researcher, and finally to help finish my thesis. I am so lucky to have Dr Lee as my supervisor: his great patience, ceaseless criticisms and unflagging support in all important respects have constantly inspired and fueled me through my studies. During these years, I have learned a lot of things from him. Thank you very much! Dr Lee!

I am also deeply indebted to my co-supervisor, Assistant Professor. Fu Cheng Peng for sharing with me his wealth of knowledge in the area of TCP. Dr Fu has always been a great source of inspiration for me. I feel fortunate enough to have his enthusiastic guidance on my Ph.D works. Thanks a lot! Dr Fu! His insights, thought, constructive comments and great suggestion are fundamental and extremely valuable.

Finally, and most importantly, this thesis is dedicated to my family: my dear parents and loving wife, without whose sacrifices and selfless support, this work would not have been possible. They gave me the extra strength, motivation and love necessary to get things done. Any acknowledgment words cannot express my deep gratitude to

them. My parents gives me their warmest encourage and moral support from thousand of miles away. My wife, He Shu, is always at my side, supporting me with her love. For all the times that I never said thank you I say so now, and I love you.

Table of Contents

Title Page	ii
Abstract	iii
Acknowledgments	vi
Table of Contents	viii
List of Figures	xi
List of Tables	xiii
List of Symbols	xiv
List of Abbreviations	xvi
1 Introduction	1
1.1 TCP/IP Architecture	3
1.2 TCP characteristics	5
1.2.1 Connection-Oriented	5
1.2.2 TCP Window	6
1.2.3 End-to-End Congestion Control	6
1.3 IP layer Performance Characteristics	7
1.3.1 Delay and Jitter	7
1.3.2 Congestion and Packet Loss	7
1.3.3 Link capacity	8
1.4 TCP challenges in heterogeneous environment	8
1.4.1 Heterogeneity of Packet Loss	8
1.4.2 Other Challenges	10
1.5 Bandwidth Measurement Challenges	11
1.6 Addressing Challenges and Contributions	12
1.6.1 Packet Triplet: A Path Capacity measurement Technique	12
1.6.2 Packet Loss and Congestion State in TCP Veno	13
1.6.3 TCP Veno enhancement	14
1.7 Thesis outline	15
2 Background and Related Work	16
2.1 Transmission Control Protocol Overview	16
2.2 TCP Reno	18
2.2.1 Slow Start	19

2.2.2	Congestion Avoidance	20
2.2.3	Fast Retransmit and Fast Recovery Algorithms	20
2.2.4	Retransmission Timer Management	22
2.2.5	Summary	23
2.3	TCP Veno	24
2.3.1	Delay-based Congestion State Estimation	25
2.3.2	TCP Veno's Congestion Control mechanism	27
2.3.3	Performance Issues	29
2.4	Other TCP Variants	30
2.4.1	TCP NewReno	30
2.4.2	TCP Vegas	31
2.4.3	TCP Santa Cruz	31
2.4.4	TCP Westwood	32
2.5	Bandwidth Measurement Technique	32
2.5.1	Bandwidth Related Metrics	32
2.5.2	Technique Classification	33
2.5.3	Packet Dispersion and End-to-End Bandwidth Measurement	34
2.5.4	Related Work	36
3	Path Capacity Measurement	38
3.1	Overview	38
3.2	Packet Triplet	39
3.3	Delay Analysis	41
3.4	Associated Filtering Technique	45
3.5	Performance Evaluation	46
3.5.1	Experiment Topology	46
3.5.2	The size of probing packets	47
3.5.3	Experimental Results	50
3.6	Summary	55
4	Packet Loss and Congestion State in TCP Veno	57
4.1	Overview	57
4.2	Performance Metrics	58
4.3	Experiment Settings	60
4.3.1	The network model of Bandwidth sharing between TCP Veno and TCP Reno	60
4.3.2	Multiple TCP Veno connections	62
4.4	Bandwidth sharing between TCP Veno and TCP Reno	63
4.5	Evaluation of Multiple TCP Veno model	71
4.5.1	Effect of Buffer Size	71
4.5.2	Effect of Bottleneck bandwidth	74
4.5.3	Effect of Number of TCP Veno connections	75
4.5.4	Effect of RTT on packet loss identification	75

5	TCP Veno enhancement	79
5.1	Overview	79
5.1.1	Impact of Bursty Congestion	81
	Revisit Congestion Estimation In TCP Veno	81
	The Problem	81
5.2	TCP Veno enhancement Mechanism	82
5.3	Effectiveness of Packet Loss labeling	84
5.4	Experimental Setting	87
5.5	Performance Evaluation	88
5.5.1	Impact of Bottleneck Buffer Size	89
5.5.2	Impact of RTT	90
5.5.3	Impact of Traffic Load	91
5.5.4	Friendliness to TCP Reno	92
5.6	Summary	93
6	Conclusions and Future work	94
6.1	Summary	94
6.2	Research Contributions	95
6.2.1	Packet Triplet	95
6.2.2	Packet Loss and Congestion State in TCP Veno	96
6.2.3	TCP Veno enhancement	98
6.3	Challenges and Future Works	99
6.3.1	Packet Triplet and Bandwidth Measurement	99
6.3.2	Develop a new end-to-end congestion control algorithm	100
	Bibliography	103
	List of Publications	115

List of Figures

1.1	The 4-layer TCP/IP protocol stack	5
2.1	Slow Start	19
2.2	Fast Retransmit algorithm	21
2.3	TCP Reno window evolution [63]	24
2.4	TCP VenO: Expected throughput versus Actual throughput	26
2.5	(a) TCP VenO's cwnd (b) TCP Reno's cwnd	28
2.6	Packet Pair Dispersion	35
3.1	Packet Triplet probe under ideal condition	39
3.2	Distorted Packet Triplet Probe	41
3.3	Packet Triplet Dispersions	43
3.4	Experiment Topology	47
3.5	Packet Triplet Estimation (Mbps) for different probing packet size, (a)100bytes, (b) 1500bytes	48
3.6	Effect of packet size on Packet Triplet path capacity estimate accuracy	50
3.7	Packet Triplet path capacity estimation (Mbps) for cross traffic size in uniform distribution, (a) 20% cross traffic load, (b) 40% cross traffic load, (c) 60% cross traffic load, (d) 80% cross traffic load	51
3.8	Packet Pair path capacity estimation (Mbps), (a)60% cross traffic load, (b) 80% cross traffic load	53
3.9	Packet Triplet Estimation (Mbps) for cross traffic size in triform dis- tribution, (a) 20% cross traffic load, (b) 40% cross traffic load, (c) 60% cross traffic load, (d) 80% cross traffic load	54
4.1	Network Model	61
4.2	Experiment Topology	62
4.3	TCP VenO/TCP Reno congestion window vs bottleneck queue length (no random loss) CL = 70%	69
4.4	TCP VenO and TCP Reno Congestion window evolution (no random loss)	70
4.5	TCP VenO and TCP Reno throughput vs. avg throughput of Two TCP Reno connections	70

4.6	$P_{c c}$ and $P_{w w}$ versus bottleneck buffer size	71
4.7	$P_{N \geq \beta}$ versus bottleneck buffer size	72
4.8	TCP Veno's congestion window Evaluation versus different bottleneck buffer size, (a), $\mu = 10$; (b), $\mu = 30$	73
4.9	$P_{c c}$ and $P_{w w}$ versus bottleneck bandwidth (Mbps)	74
4.10	$P_{N \geq \beta}$ versus bottleneck bandwidth (Mbps)	75
4.11	$P_{c c}$ and $P_{w w}$ versus number of TCP Veno connections	76
4.12	TCP Veno's Congestion Avoidance	77
4.13	$P_{c c}$ and $P_{w w}$ versus RTT	77
4.14	$P_{N \geq \beta}$ versus RTT	78
5.1	The Simulation Topology	88
5.2	The accuracy for different buffer size, RTT=60ms, background traffic load=10%	89
5.3	The accuracy for different round trip delay, $\mu=15$, background traffic load=10%	90
5.4	The accuracy for different background traffic load, $\mu=15$, RTT=60ms	91
5.5	The comparison of throughput and fairness, RTT=30ms	92

List of Tables

3.1	Packet Triplet path capacity estimation	52
3.2	CapProbe Estimation	53
4.1	Experiment results for loss distinguishing in TCP Veno, $CL = 30\%$.	64
4.2	Experiment results for loss distinguishing in TCP Veno, $CL = 50\%$.	65
4.3	Experiment results for loss distinguishing in TCP Veno, $CL = 70\%$.	66

List of Symbols

Chapter 1

N	TCP Venó's backlog
β	the threshold for congestion state identification
$VRTT$	the smoothed round trip time in TCP Venó

Chapter 2

C	the path capacity
C_i	the capacity of link i
U_i	the current utilization of link i
C_a	the available bandwidth of the path
Δ	the dispersion between two consecutive probing packets
L	the probing packet length

Chapter 3

P_1, P_2, P_3	the 1st, the 2nd and the 3rd probing packet
Δ_1	the dispersion between P_1 and P_2
Δ_2	the dispersion between P_2 and P_3
B_1	the path capacity estimation estimated by L/Δ_1
B_2	the path capacity estimation estimated by L/Δ_2
$(B_{j,1}, B_{j,2})$	the j th path capacity estimate pair
m	the number of estimate pair
τ_b	the transmission time for a probing packet at bottleneck link
σ	the latency variation
M	number of links in the path
τ	end to-end transmission delay
τ_i	the transmission time for a probing packet at link i
D_1, D_2, D_3	the end-to-end one way delay for probing packets
d_1, d_2, d_3	the end-to-end cross traffic induced delay for probing packets
lat_1, lat_2, lat_3	the latency for probing packets
δ	the difference between two path capacity estimations
α	the threshold for indicating the identical path capacity estimations

Chapter 4

$P_{c c}$	the accuracy of congestion loss identification
$P_{w w}$	the accuracy of wireless loss identification
$P_{N \geq \beta}$	the probability of $N \geq \beta$
$P_{N < \beta}$	the probability of $N < \beta$
γ	the wireless transmission error rate
B	the bandwidth of the bottleneck link
μ	the buffer size of the bottleneck link
N_{max}	the maximum backlog in a TCP Veno congestion avoidance phase
$cwnd_{max}$	the maximum congestion window according to N_{max}

Chapter 5

v_sumRTT	the sum of RTT in TCP Veno
v_cntRTT	the number of RTT in TCP Veno
$VRTT_{inst}$	the smoothed RTT in TCP Veno enhancement
N_{inst}	the new backlog in TCP Veno enhancement
$w_s(n)$	TCP Veno's congestion window in n th update step
$E[\Delta W_+]$	the expected increment of congestion window per update step
$E[\Delta W_-]$	the expected decrement of congestion window per update step
$E[\Delta W]$	the expected change in the congestion window per update step
v	the interval between update steps
r_V	the steady throughput of TCP Veno
P_c	the congestion loss probability
P_w	the wireless loss probability
$\tilde{N}$	$\max(N, N_{inst})$

List of Abbreviations

<i>ssthresh</i>	TCP Slow Start Threshold
TCP	Transmission Control Protocol
<i>cwnd</i>	congestion window
UDP	User Datagram Protocol
IP	Internet Protocol
RTP	Internet real-time Protocol
QoS	Quality of Service
ISP	Internet Service Provider
RTO	TCP Retransmission Time Out
CL	Cross traffic load
FCFS	First-Come-First-Serve
LRD	Long Range Dependent
RTT	round trip time

Chapter 1

Introduction

The Internet is a global collection of different computers and networks worldwide. It is constructed based on the TCP/IP protocol stack. TCP/IP model has shown great success in transporting Internet traffic. However, the explosive growth of the Internet fuels a lot of new demands that challenge the design of TCP, as well as the traditional service model offered by IP.

IP (Internet Protocol) is the network layer protocol that provides connectionless service to the transport layer, sending all datagrams imposed on the network to the correct destination, simply in the way of store-and-forward packet switching. IP does not provide any load control or resource reservation to the network traffic. The traffic being sincerely served in this way is so-called *best effort* traffic.

Nowadays, it has been widely accepted that the recent indistinctive *best-effort model* of IP network is unable to support the emerging commercial demands of real-time audio and video traffic services. These multimedia services may require very stringent Quality of Service (QoS). The primary goal of QoS is to provide prior-

ity, including dedicated bandwidth, controlled jitter and latency (required by some real-time and interactive traffic), and improved loss characteristics. Bandwidth measurement is considered to be a very important approach for service provider to trace the characteristics of the path and use these information to construct QoS.

TCP (Transmission Control Protocol) is an end-to-end transport layer protocol that uses congestion control to prevent the network from *congestion collapse* and ensures fair sharing of resources between multiple hosts. Besides TCP, some other network congestion control schemes, such as TCP-friendly congestion control schemes [11] [29] [30] [65] are developed for non-TCP (usually UDP) real-time applications, to ensure the network utilization of UDP sources as well as their fairness towards competing TCP connections. However, TCP is still the predominant transport protocol on which most Internet applications (HTTP, FTP, Telnet, SMTP, DNS, SSL, etc.) are built compared to UDP and will not be taken over soon in the near future. New network techniques, such as wireless local network and cellular network, are bringing new challenges to TCP congestion control.

TCP was originally designed for wired network. As wireless communication grow rapidly, more and more mobile devices are connected to the wired network via wireless link. Problems emerge when the applications in these devices use TCP over such heterogeneous Internet, of which the characteristics are palpably different from that of wired environment. For example, over wired links, congestion can be regarded as the only reason for packet loss, while this is not true over wireless links, where many packet losses are caused by transmission error. Moreover, the small bandwidth, the higher latency and unstable topology are also the faults that TCP performance may

suffer from.

Our work consists of two parts: path capacity measurement and TCP performance in wireless network. The research issue that we were addressing is how we can improve the performance of data transfer across the Internet. Data transfer across the Internet is limited by the path capacity, upper limit, or TCP. Our research started from finding the path capacity and we have proposed a new mechanism “Packet Triplet” that has better measurement accuracy and is more resilient to cross traffic.

However, users are becoming more mobile and wireless access network is pervasive [5] [7] [18] [34] [43]. Thus, we felt the need to investigate the performance of TCP over wireless network; in our case we use TCP Veno, a TCP variant designed for better performance in hybrid wired/wireless environment. Our investigations showed that TCP Veno cannot achieve high accuracy of identifying both congestion and wireless packet losses under some conditions. We proposed a TCP Veno enhancement and found that it could have a better prediction of the cause of congestion packet loss by using an enhanced packet loss identification scheme, resulting in better friendliness towards TCP Reno.

1.1 TCP/IP Architecture

The Internet is an open and global network that allows different computers and terminals around the world to communicate with each other. Before the Internet, diverse local and closed network architectures, such as SNA (IBM’s Systems Network Architecture) or DECnet (Digital Equipment Corp’s network), had been independently developed. However, the communication is only confined to a close commu-

nity and it can no longer meet people's demand of data exchanges that can span across the globe. As these stand-alone networks depend on incompatible addressing and transport protocols, it is inconvenient to directly establish connections between them. These issues motivated the design of TCP/IP [63].

IP (Internet Protocol) uses its standard 32-bit address to hide the differences in addressing and enable routing of packets among heterogeneous networks around the world. The intermediate nodes that support IP-address based routing link these networks together. Such worldwide collection of multiple networks forms the Internet. On top of the IP layer, the transport protocols like TCP (Transmission Control Protocol) and UDP (User Datagram Protocol) are developed to provide more sophisticated and efficient data delivery services for the Internet applications, such as Telnet, FTP, e-mail or HTTP. TCP/IP comes from the combination of IP and TCP.

TCP/IP protocol suite is formed as a layered structure, in which different layers contain different protocols. Using the concept of layering in network architecture can ease the complexity in design. Each layer performs a collection of functions (protocols) and provides a set of services to the upper layers through a set of interface. In this way, each layer is shielded from how the offered services are actually implemented in the lower layer. TCP/IP comprises four layers, as shown in Fig. 1.1

The link layer contains the network devices and interfaces that are directly in charge of data transmission over the physical media. The network layer (IP) provides addressing and routing functions. IP services are unreliable and connectionless. Transport layer comprising TCP and UDP offers end-to-end data services up to the hosts in the application layer. Specifically, TCP provides reliable data delivery ser-

Figure 1.1: The 4-layer TCP/IP protocol stack

vices (e.g. FTP, Telnet), while UDP provides unreliable transmission services, such as real-time video traffic. It is the use of such key transport protocols that drives the tremendous growth of the Internet.

1.2 TCP characteristics

In this section we introduce some essential concepts and TCP characteristics related to our work that we will describe in more detail later in this thesis.

1.2.1 Connection-Oriented

TCP is a *connection-oriented* transmission protocol, which means a *virtual* connection must be established and maintained between two end points (sender and receiver) to achieve a reliable end-to-end byte stream.

The connection is managed by a series of control signal exchange between two ends. The direction of a connection is the direction in which application data segment are

sent. Furthermore, multiple connections can be maintained in both direction at the same time. This is so called *full duplex* data transfer.

1.2.2 TCP Window

For a TCP connection, the amount of data that can be injected into the network is controlled by its window. Two factors determine the size of the window. One is the limitation of receiver's capability of receiving the data. Another is the limitation of pipe size between the two end stations. The window size must not exceed either of these two limitations. To fulfill this requirement, TCP sender uses two windows: the window that tells the buffer size granted at the receiver, and the **congestion window** (*cwnd*), for restricting the data offered to the network.

1.2.3 End-to-End Congestion Control

If the applications offer more traffic load than the network can sustain, congestion builds up. TCP adopts end-to-end congestion control mechanism to deal with the congestion, preventing the network from being exhausted and ensuring fair sharing of available bandwidth with other applications and services. Nowadays, congestion control mechanism receives the most significant attentions in the area of TCP research, because it is a major factor in determining TCP throughput and TCP friendliness, to the performance of tremendous TCP-based Internet applications in use as well as to the behavior of the traffic in the Internet.

In this thesis, our focus is on the issue of TCP operating in heterogeneous environment, where the wireless link is involved.

1.3 IP layer Performance Characteristics

Transport layer is not the only layer that affects the bandwidth related performance. As the transport layer protocols must rely on network layer protocol (IP) to have the application data successfully traverse from one end to the other, the IP performance characteristics [81] (e.g. bottleneck bandwidth or congestion) and measurement are also important topics.

1.3.1 Delay and Jitter

Delay is the time taken by a packet to traverse the network [63]. Of the delay, some are due to the inherent factors of the network, such as the time for the end system or intermediate nodes to process the data, or the time for the physical link to transmit the data; some are due to the variable factors, such as network traffic load.

Jitter [89] reflects the difference in the delay between adjacent segments. This delay is mainly attributed to the queueing delay introduced along the multi-hop path. Some services, such as real-time audio and video conference are constrained by end-to-end delay and jitter requirement.

1.3.2 Congestion and Packet Loss

In the Internet, congestion may occur at any intermediate node, where the capacity cannot cope with the input data traffic. This eventually leads to packet loss when the buffer of the congested node is exhausted by the queuing data.

In wired network, packet loss are mainly due to network congestion, while packet loss due to transmission errors are relatively rare. This is the assumption adopted

by conventional TCP congestion control mechanism. However, this does not hold for wireless channels, where the transmission error occurs frequently.

1.3.3 Link capacity

Link capacity [87] is the maximum throughput that the link can offer to a flow, assuming that the link is not utilized by other traffic. This metric, however, can be used in multi-hop path, namely, path capacity, which is the maximum throughput that the path can offer to an application, if the path is not utilized by other traffic.

Path Capacity [80] is more relevant to network applications and end-to-end based transport protocol, compared to the link capacity. For example, given the amount of data to be sent by TCP and the load of the other competing traffic, larger path capacity means larger TCP window, less congestion packet loss and faster transmission. The measurement of path capacity is therefore a topic that we focus on in this thesis.

1.4 TCP challenges in heterogeneous environment

1.4.1 Heterogeneity of Packet Loss

In TCP, congestion control mechanism signals the occurrence of network congestion and reduces its throughput accordingly when a packet loss occurs. This lead to significant throughput deterioration when TCP is used in wireless network, where a large proportion of packet losses are due to transmission bit errors or other non-congestion reasons. Such packet loss is defined as random loss in [45].

To address the problem, many solutions have been proposed in the past years. Some of them attempted to improve TCP by hiding the random loss from TCP sender and to have only congestion loss made visible to the source [4] [5] [18] [59]. The problems due to wireless link are solved somewhere in the network by specific local retransmission and recovery schemes that are irrelevant to the TCP sender. In contrast, some approaches try to make the TCP sender be aware of the wireless link and the packet losses due to the non-congestion reasons [7] [12] [33] [62]. Thus, TCP can be informed with distinguishable signals from the intermediate nodes, thereby taking appropriate control actions. These approaches also need the collaboration of intermediate nodes in the network.

It has been shown that the aforementioned approaches do help in improving TCP throughput performance in lossy environment. However, the deployment of these approaches is limited, because they require the cooperation of intermediate nodes (router or base station) along the network path. Moreover, enabling the enhancement for all the nodes in an existing and stable network infrastructure of wide dimension is not worth the cost.

Network security is another factor that can lead to the failure of these schemes. If the network traffic is encrypted, the intermediate nodes, which need to seek the specific content of TCP header, cannot recognize the TCP data segments that are packed in encrypted IP traffic. Thus, the approaches such as [4] [5] fail to achieve the expected throughput performance improvement if the IP data is encrypted.

These problems do not exist in the end-to-end based solutions, which does not require changes in intermediate nodes. There are different strategies by which various

end-to-end based solutions can handle the random loss problem and improve TCP performance in wireless environment.

Of all the solutions, [33] relies on enhanced TCP receiver, which is responsible for identifying the random or other non-congestive losses. TCP receivers need to notify the sender about how to take the proper control actions based on the causes of the packet loss, and avoid unnecessary degradation in throughput can be avoided.

In contrast to receiver-side modification, some solutions on the sender-side [5] [25] [51] [53] [54]. That is, the enhanced TCP sender takes on the responsibility of packet loss distinguishing and performance improvement, while the receiver is not modified.

In some solutions, the sender and the receiver are both enhanced. The sender and the receiver are all required to participate in eliminating the faults due to random loss [51] [57] [66] [69] [67]. The specific signals and information that are required by improved congestion control mechanism are obtained by the interaction between the sender and the receiver.

1.4.2 Other Challenges

In addition to the random packet loss, there are some other issues concerning TCP performance in wireless link as well as wired link. These issues include multiple packet loss in a window, large round-trip delay, unstable connection due to the mobility of hosts, and low bandwidth of wireless link. To tackle the above issues, there are a number of proposals and solutions in the related areas [22] [27] [31] [37] [46] [51] [64] [73].

1.5 Bandwidth Measurement Challenges

Nowadays, Internet users need to pay Internet Service Providers (ISPs) for Internet access and services. As the quality of Internet access and services, such as delay and throughput, are normally based on bandwidth of Internet connections, bandwidth measurement becomes very important. In such an environment, the users need to verify the available bandwidth resources. Most ISPs also want to measure the bandwidth in order to handle any congestion and if necessary, upgrade the link capacity to satisfy the users demands.

In addition to network resources management, some widely used network protocols, such as TCP, RTP (Internet real-time Protocol), also adopt bandwidth measurement to optimize their performance.

Currently, bandwidth measurement faces the following challenges [81] [92]:

- **Robustness:** Internet is a complicated, dynamic and wide-spread architecture in which different terminals, devices and services are integrated. In order to measure the bandwidth metrics accurately, a measurement technique must be carefully designed so that it is robust to various network uncertainties and characteristics, such as cross traffic, the change of routing, asymmetry of network path, the clock and the processing time of different systems and devices, the queueing discipline employed in the intermediate nodes, etc.
- **Efficiency:** How to measure bandwidth metrics that may change fast? This is especially important with wireless network, where the route may change frequently, as compared with traditional wired environment.

- **Friendliness:** A measurement technique must be friendly, which means the bandwidth can be measured successfully without sacrificing the network resources or adversely affecting other applications.
- **Deployability:** Due to the wide-scale of Internet and security consideration, a measurement technique should be easy to deploy and not conflict with the security restriction.

1.6 Addressing Challenges and Contributions

Our works and contributions focus on two main areas:

- **Packet Triplet** – a robust end-to-end measurement technique – proposed for measuring the path capacity.
- **Packet Loss and Congestion State in TCP Veno** – We investigated the accuracy of TCP Veno’s packet loss distinguishing scheme under different network conditions. We also proposed an improved congestion state measurement and packet loss identification scheme for TCP Veno to handle the problem due to bursty congestion. The mechanism was incorporated into TCP Veno, and it is known as TCP Veno enhancement.

1.6.1 Packet Triplet: A Path Capacity measurement Technique

In this part, we developed an end-to-end path capacity measurement technique called Packet Triplet. Our studies demonstrated that Packet Triplet is robust and

quite efficient in measuring the path capacity under different conditions, even when the path is heavily loaded.

The name “Packet Triplet” comes from the fact that each Packet Triplet probe consists of three probing packets of equal size. Thus, two dispersions induced by the probed path can be obtained from each Packet Triplet probe.

The fundamental innovation in Packet Triplet is the filtering technique, which is based on a simple fact that, of each Packet Triplet probe, the probability that cross traffic interference can result in identical distortion is very small. During the measurement process, the Packet Triplet sender emits a series of probes into the path of interest; the receiver calculate the dispersions based on the arrival time of each packet and uses the filtering rule to discard the distorted samples and eventually gets the correct path capacity estimation.

The “end-to-end” probe strategy makes Packet Triplet easy to deploy. The probing packets are UDP datagram, which does not violate any network security rules and thereby, can traverse the measured path successfully. The sending rate of probing packet is under careful control so Packet Triplet will not interfere with other applications as well as between probes.

1.6.2 Packet Loss and Congestion State in TCP Veno

In this part, we study the effect of congestion state on the accuracy of distinguishing packet loss in TCP Veno in heterogeneous wireless/wired environment. TCP Veno was first proposed in 2003 [25] as a TCP variant for improving TCP performance in heterogeneous network. Later, a International co-operation project called

“Veno II” [26] was brought up based on TCP Veno to develop an universal transport architecture in next generation communications. Our work is a part of this project as wireless network is a vital part in next generation communications. Some performance issues, such as the accuracy of packet loss identification in TCP Veno, were not well studied before our work.

TCP Veno estimates the congestion state of network and attempt to use this information to distinguish between two different types of packet loss: congestion loss and random loss in heterogeneous network. In this way, TCP Veno avoids the unnecessary throughput reduction when the cause of the packet loss is due to non-congestive reasons over wireless links.

Our study demonstrated that in a network loaded with heavy background traffic, TCP Veno shows poor performance in terms of congestion loss identification, nevertheless, the low accuracy of packet loss identification does not result in any negative effect to TCP Veno’s throughput performance.

1.6.3 TCP Veno enhancement

We also studied the effect of bursty congestion on the accuracy of TCP Veno’s packet loss diagnosis. We found that congestion state measurement algorithm in TCP Veno may not detect the fluctuation of bursty congestion. This may lead to poor accuracy of loss diagnosis in TCP Veno. To solve this problem, we proposed an enhanced packet loss distinguishing scheme. TCP Veno with the new scheme embedded (TCP Veno enhancement) can achieve a more accurate congestion loss identification as well as better friendliness to other TCP connections.

1.7 Thesis outline

The rest of this thesis is organized as follows. In Chapter 2, we present the background knowledge on TCP protocol and bandwidth measurement technique, and review the related works in the two areas. Some TCP variations are also introduced in this chapter.

In Chapter 3, we propose Packet Triplet technique. We introduce the principle of Packet Triplet and the filtering technique in Packet Triplet. We analyze the effect of cross traffic on Packet Triplet probes and demonstrate that the filtering technique in Packet Triplet contributes to the efficient and accurate path capacity measurement.

Chapter 4 contains the experimental results of our study on congestion state measurement and packet loss distinguishing in TCP Veno under different network conditions. We observed that TCP Veno exhibits poor performance in terms of congestion loss identification in network loaded with heavy background traffic. In addition, superior erroneous packet loss identification does not cause any negative effect to TCP Veno's performance.

In Chapter 5, we present our investigation into the specific area of TCP Veno in an environment with bursty congestion, which has shown to adversely affect the accuracy of congestion loss identification in TCP Veno. We propose an enhanced loss identification scheme to tackle the problem. The experiments show the efficiency of the new scheme.

In chapter 6, we draw the conclusion of our works and contributions in this thesis. We provide some possible directions for further research.

Chapter 2

Background and Related Work

TCP/IP protocol stack is the *de-facto* protocol of the Internet. In this chapter, we introduce the background knowledge in the relevant area of our works. We will first introduce the most popular TCP variant, TCP Reno. Then we will introduce TCP Veno, especially the packet loss distinguishing scheme and refined congestion control mechanism. We will also present the background information in the area of network bandwidth measurement.

2.1 Transmission Control Protocol Overview

Transmission Control Protocol (TCP) has been the dominant transmission protocol in the Internet. TCP was designed to provide reliable, connection-oriented, end-to-end, and in-order byte stream services [63].

The characteristic of connection-oriented protocol indicates a conceptual connection that must be constructed and maintained by negotiation between two TCP end points. To accomplish reliability, TCP assigns each segment a sequence number.

When TCP receives a segment, it replies with a “cumulative” acknowledgment (ack) to the TCP sender. Each ack contains an acknowledgment sequence number, which tells the TCP senders that the in-order segment has been successfully received. In TCP, when either duplicate (three) acks are received in series or no ack for an outstanding segment is received before timeout it is inferred as packet loss. If a packet loss is detected, the TCP sender retransmits the lost segments immediately.

TCP flow control mechanism ensures that the data emitted by the TCP sender does not exhaust the buffer capability at the TCP receiver. To do so, the TCP receiver must inform the TCP sender about its current available buffer space. In addition to flow control, another important issue that affects TCP performance is path capacity. This is carried out by end-to-end congestion control mechanism (first proposed by Jacobson in 1988 [36]), which has shown its ability in keeping the network from “congestion collapse” as well as guaranteeing the fair allocation of bandwidth resources among multiple TCP connections. Note that eventually the amount of data sent cannot exceed the smaller value between TCP receiver’s available buffer and the network capacity.

Basically, congestion control continuously runs and repeats in the manner of “available bandwidth probe $\rightarrow$ rate diminishing driven by packet loss”. During data transmission, the TCP sender gradually increases its sending rate (which is controlled by the reception of cumulative ack) - either linearly or exponentially - until packet loss occurs, indicating that the maximum available bandwidth for the connection has been reached. The congestion control design of TCP makes the assumption that packet loss is due to network congestion. So the TCP sender reduces its throughput

to avoid congestion collapse, retransmits lost packets and then repeats the bandwidth probing process again.

TCP Tahoe [36] is the first version of TCP to adopt congestion control mechanism. Later, Jacobson introduced Fast Recovery algorithm into TCP. The extended TCP using Fast Recovery in its congestion control process is called TCP Reno [63] - the TCP version that is still dominating the reliable data delivery in the Internet today. Nevertheless, studies on TCP have never stopped.

For years, researchers have explored different aspects of TCP. Some works focus on enhancing the congestions control [3] [4] [9] [20] [25] [37] [39] [53] [57] . The analysis and modeling of TCP performance in different manners is also a topic that has received much attentions [17] [19] [34] [43] [58] [60].

2.2 TCP Reno

TCP Reno [63] is the most widely deployed TCP variant. Roughly speaking, TCP Reno's congestion control comprises the following mechanisms: retransmission timer management, Slow Start, Congestion Avoidance and Fast Retransmit and Fast Recovery. Retransmission timer management is used to adjust the retransmission time-out timer based on current round trip time. The other mechanisms are designed for window management and packet loss recovery, which deal with the network states that the TCP connection experiences. In the following sections, we introduce these mechanisms separately.

2.2.1 Slow Start

Slow Start mechanism is initiated when a connection is first established or after a retransmission timeout. The purpose of the Slow Start phase is to enable the TCP sender to determine the reasonable window size that is available to the connection, which makes sure that the network will not be driven into saturation by the flooding datagrams.

During Slow Start, the window is exponentially increased by the incoming acks. In fact, Slow Start is not slow at all. As Fig. 2.1 illustrates, the window is initially set to one segment. The TCP sender is only allowed to send one segment and waits for a ack before expanding the window and sending the new data. When the first ack comes back, TCP sender increases the window to 2 and sends a burst of two segments. Then, the TCP sender increases the window by 1 for each incoming new ack. Therefore, after two segments are acknowledged, the TCP sender can send four segments. In the same way, after those 4 segments are acknowledged, the window is set to 8.

Figure 2.1: Slow Start

Such exponential growth goes on until the window hit *ssthresh* (slow start threshold) or packet loss occurs (indicating congestion). At that point, the current Slow

Start phase terminates, *ssthresh* is set to half of the current window. If a timeout occurs, for which it cannot tell how serious the network congestion is, the window is reset to 1 and a new Slow Start phase starts again to probe the network condition, until the window hits the *ssthresh*. If packet loss or timeout does not occur, Congestion Avoidance is used instead to probe the network. The reason of not using Slow Start at this moment is that Slow Start is very aggressive and can easily push the system into saturation, although it has been approved to be effective for initializing a TCP connection.

2.2.2 Congestion Avoidance

In the above section, we have introduced the Slow Start process triggered by timeout, which terminates at the point when *ssthresh* is hit. After the Slow Start, the TCP connection enters a new phase. This phase is Congestion Avoidance. In Congestion Avoidance, the window is increased by 1 for each window of packets that are acknowledged, instead of one per segment. That is, when each ack to a new segment is received, the *cwnd* (congestion window) is increased by $1/cwnd$. Thereby, after the incoming of acks to segments of a congestion window, the increment of *cwnd* is 1 per round trip time, which approximately follows the linear increase.

2.2.3 Fast Retransmit and Fast Recovery Algorithms

Fast Retransmit and Fast Recovery is specifically developed to deal with the three duplicate acks. As Fig. 2.2 illustrates, segment 1 and segment 2 are received, but segment 3 is lost in the network. When segment 4 arrives, the TCP receiver recognizes

that this is an out-of-order segment and sends the duplicate ack for segment 2, and again sends two more duplicate acks for the receipt of segment 5 and segment 6. When the TCP sender gets the third duplicate ack, it immediately retransmits segment 3. The ack for segment 3 also acknowledges all the in-order segments that follow segment 3 and are received so far.

Figure 2.2: Fast Retransmit algorithm

Receipt of three duplicate acks is a faithful signal ¹ of packet loss [63]. The TCP receiver can only generate the duplicate ack for another segment that successfully comes into the TCP receiver's buffer, so three duplicate acks indicates lost packet, while still permitting data flow between the two ends. This is different from the situation of retransmission timeout, which definitely indicates major network problem and thus connection return to Slow Start. In other words, three duplicate acks indicates

¹The receipt of one or two duplicate acks is not strong enough to indicate a packet loss indeed. Instead, it could be just a reordering of segment.

a lighter congestion, compared to timeout. Therefore, in this case, it is able to minimize abrupt throughput reduction. Fast Retransmit and Fast Recovery algorithms are designed for this purpose.

Fast Retransmission and Fast Recovery algorithms are usually implemented together:

- In response to the third duplicate ACK, the TCP sender cuts *ssthresh* to one half of current window, retransmits the lost segment and sets *cwnd* to *ssthresh* plus 3.
- After Fast Retransmit, Fast Recovery takes place. The TCP sender “inflates” the *cwnd* by one each time a duplicate is received and then emits a new segment. In this way, the network is maintained to be full and in equilibrium.
- The Fast Recovery terminates as long as the retransmitted segments are acknowledged. *cwnd* is set to the *ssthresh*

2.2.4 Retransmission Timer Management

During the TCP data transmission, a variable, RTO (Retransmission Time Out) is maintained for determining the retransmission timeout. That is, each time a segment is sent, the retransmission timer starts to work. If the ACK to the segment does not come back to the TCP sender before the timer times out, the segment is retransmitted.

As the network condition often change, the value of RTO must be updated on line to follow the changes. To do so, the TCP sender first estimates the RTT (round-trip time) as the time between the sending of current segment and the reception of its

ACK. A smoothed RTT is calculated via exponential average low-pass filter specified in RFC 793 [56]:

$$SRTT = \alpha * SRTT + (1 - \alpha) * MRTT$$

where $SRTT$ denotes the smoothed RTT and $MRTT$ is currently measured RTT. In [56], α is suggested to be 0.9, which gives greater weight to the previous estimate so as to smooth out the possible fluctuation of current RTT.

Then a smoothed deviation, DEV , is computed by:

$$DEV = DEV + h * (|SRTT - MRTT| - DEV)$$

where h is proposed to be 1/4 in [36].

In [36], RTO is suggested to be calculated as:

$$RTO = SRTT + 4 * DEV$$

2.2.5 Summary

Putting the aforementioned Slow Start, Congestion Avoidance, Fast Retransmit and Fast Recovery algorithm together, we can construct a complete picture of congestion control and congestion window (or throughput) evolution in TCP Reno. As Fig. 2.3 illustrates, a TCP Reno connection is initially opened with Slow Start, which expands the congestion window exponentially to probe the network condition. When the capacity of the pipe is hit, packet loss occurs. In response, Fast Retransmit and Fast Recovery take place (assuming the case is for three duplicate acks), slowing down the throughput to half. As long as the lost packet is successfully recovered, TCP steps

Figure 2.3: TCP Reno window evolution [63]

into Congestion Avoidance, the window increase linearly. This may eventually drive the network into congestion and cause the packet loss to occur again. Likewise, if timeout occurs, Slow Start takes place; if three duplicate acks are received, Fast Retransmit and Fast Recovery comes again, and so on. Such cycle repeats during the life of a TCP connection.

2.3 TCP Veno

TCP assumes that all packet losses are caused by congestion. Upon detecting packet losses, TCP drops congestion window to reduce the sending rate. As wireless communication becomes more prevalent, the reason for packet losses may be transmission error rather than congestion. As a consequence, using TCP without any modifications may result in serious performance degradation.

To date, much work has been centered on accurately distinguishing packet losses that are due to congestion from loss due to bit-errors or other non-congestion reasons.

TCP Veno [25] proposes a congestion state measurement technique, which tries to differentiate congestion loss from wireless loss and takes different windows control actions accordingly.

TCP Veno employs a refined congestion control mechanism that is different from the conventional TCP in two aspects: 1) An enhanced Congestion Avoidance mechanism 2) A refined Slow Start threshold (*ssthresh*) adjustment strategy. Both of them are implemented according to the different estimated congestion levels of a connection.

2.3.1 Delay-based Congestion State Estimation

The idea of congestion state measurement technique in TCP Veno is borrowed from TCP Vegas [3]. Two throughput metrics are estimated: *Expected throughput*, which is calculated by assuming that the network is in the state of non-congestion; and *actual throughput* – the real throughput. The basic idea behind TCP Veno’s congestion state measurement is that the difference between the expected and actual throughput reflects the fluctuation of network congestion. If there is no congestion, the expected throughput and actual throughput should be equal; while if the network is congested, the actual throughput should be smaller than the expected throughput. As the traffic load increases, the difference between the actual throughput and expected throughput will be wider, as Fig. 2.4 shows. TCP Veno uses this information to infer the network congestion state.

The expected and actual throughput is calculated as:

$$actual\_thpt = cwnd / BaseRTT \quad (2.1)$$

Figure 2.4: TCP Veno: Expected throughput versus Actual throughput

$$actual\_thpt = cwnd / VRTT \quad (2.2)$$

where $cwnd$ is the current window size represented in number of packets, $BaseRTT$ ² is the most current minimum RTT in the lifetime of a TCP Veno connection, $VRTT$ ³ is smoothed round-trip time.

The difference between $expected\_thpt$ and $actual\_thpt$ is:

$$diff = expected\_thpt - actual\_thpt \quad (2.3)$$

TCP Veno uses the parameter $diff$ to estimate the number of packets accumulated in bottleneck buffer. let N be the backlog at the bottleneck buffer. That is,

$$N = diff * BaseRTT = cwnd * (1 - BaseRTT / VRTT) \quad (2.4)$$

²In TCP Veno, $BaseRTT$ is reset whenever the Slow Start or Fast Retransmit occurs and then is updated as TCP Vegas does

³ $VRTT$ is different from the smoothed RTT specified in RFC793 [56]. The definition is stated in Chapter 5.1.1

TCP Veno sets a threshold, β (in TCP Veno, β is set to 3^4), to be the point of the judgement of non-congestive and congestive state. If $N < \beta$ when a packet loss is detected, TCP Veno identifies the packet loss as random loss, because the network is supposed to be in non-congestive state. If $N \geq \beta$, the network is declared to be congested and the packet loss is identified as congestion loss.

2.3.2 TCP Veno's Congestion Control mechanism

We will first discuss TCP Veno's Congestion Avoidance mechanism, which is implemented as shown in Algorithm 1.

Algorithm 1 TCP Veno's Congestion Avoidance

---

if ($N < \beta$) **then**

$cwnd = cwnd + 1/cwnd$ when each new ACK received

else if ($N \geq \beta$) **then**

$cwnd = cwnd + 1/cwnd$ when every other ACK received

end if

In this mechanism, the increase in window size goes through two sub-phases. With reference to Fig. 2.5(a), when the network is judged to be in non-congestive state, the window is increased by $1/cwnd$ for every new incoming ACK, which is same as TCP Reno. Such growth goes on until the backlog hits the threshold β , for example, at time 32s. From that point on, the network is supposed to be in congestive-state and the window grows by $1/cwnd$ for every **other** new ACK received rather than every new ACK, until packet loss occurs. Slowing down the speed of window increase in the phase when $N \geq \beta$ helps TCP Veno connection to stay longer in the region of network

---

⁴The setting of β is based on the experiments in [25]

(a)

(b)

Figure 2.5: (a) TCP Veno's cwnd (b) TCP Reno's cwnd

probe and large window and postpones the occur of self-induced congestion loss [25], as compared with TCP Reno in Fig. 2.5(b). Accordingly, another effect is that the loss due to congestion is postponed. These advantages can be reflected by comparing the TCP Veno's cwnd (Fig. 2.5(a)) with TCP Reno's cwnd (Fig. 2.5(b)). As Fig. 2.5(a) and Fig. 2.5(b) illustrate, TCP Veno exhibits less window “*increase* $\rightarrow$ *reduce*” cycles and congestion losses than TCP Reno. [25] has demonstrated that it is this subtle window manipulation that contributes to throughput improvement in TCP Veno.

Another enhancement in TCP Veno is its Slow Start threshold *ssthresh* adjustment. When the packet loss is detected by the receipt of three duplicate acks, TCP Veno adjusts *ssthresh* in the following way.

Algorithm 2 TCP Veno's *ssthresh* adjustment

---

if ($N < \beta$) **then**

$$ssthresh = \frac{4}{5}cwnd$$

else if ($N \geq \beta$) **then**

$$ssthresh = \frac{1}{2}cwnd$$

end if

As Algorithm 2 shows, if the network is in non-congestive state (i.e. $N < \beta$), TCP Veno assumes that the packet loss is random loss. Accordingly, TCP Veno reduces the *ssthresh* by a small amount that is less than half the current window. In [25], the suggested reduction factor is $1/5$. If the packet loss is classified as congestion loss, the penalty of *ssthresh* is $1/2$, which is same as TCP Reno. Likewise, the reduction of *cwnd* follows the change of *ssthresh*.

The simulations and experiments in [25] demonstrate that TCP Veno does benefit from its *ssthresh* adjustment strategy, having its throughput significantly improved in heterogeneous environment, especially when the wireless loss rate is about 1%.

2.3.3 Performance Issues

In TCP Veno, the decision of whether a packet loss is due to network congestion or wireless transmission error is based on the estimation of network state. Thus, the performance of the protocol is dependent on the accuracy in determining the cause of the packet loss. It is therefore natural to expect that congestion state measurement could be accurate or the packet loss could be correctly associated with the congestion state. The question, however, subsequently arises: Can such expectation always be

achieved? Unfortunately, the answer is no. Some performance problems have been observed in previous studies [13] [35] [50]:

- The congestion packet loss and increase in round-trip time is weakly correlated, which was stated in [50] that the increase of round-trip time is not a reliable signal for predicting congestion packet loss, especially in network where a TCP flow is only a fraction of the total traffic.
- The packet loss, either congestion loss or random loss, may not necessarily be associated with the congestion state that is correctly identified [13].
- The congestion state, however, can be underestimated or overestimated under some conditions [35].

In this thesis, we address the above issues in TCP VenO. The details are depicted in Chapter 4 and Chapter 5.

2.4 Other TCP Variants

We now look at some well-known TCP variants that have been proposed in previous works. We introduce the approaches adopted to cope with the performance problems that may arise over heterogeneous environment.

2.4.1 TCP NewReno

In the case of multiple packet losses from within the same window, *partial* new ack in TCP Reno (the new ack that does not acknowledge all the outstanding segments) forces the Fast Recovery to be terminated and eventually leads to timeout.

TCP NewReno [22] [31] solves the problem by using the information of *partial* ack subtly. *Partial* ack can indicate that the outstanding segment right after the acknowledged packet has been lost. TCP NewReno immediately retransmits the lost segment without having to leave Fast Recovery phase, and thus avoids unnecessary timeout.

2.4.2 TCP Vegas

TCP Vegas [3] is a sender-side modification of TCP that tries to deduce the building up of network congestion. The network congestion is estimated by the algorithm that is used by TCP Veno. The key innovation in TCP Vegas is its refined congestion avoidance driven by the receipt acks coupled with congestion state. In TCP Vegas, the operation of window adjustment is divided into three phases: window increasing phases, window remaining phase and window decreasing phase. The three phases are linked with non-congestive state, moderate congestive state and severe congestive state, respectively. In this way, TCP Vegas attempts to smooth the window evolution, maintain the network in equilibrium and avoid packet loss due to congestion.

2.4.3 TCP Santa Cruz

Similar to TCP Vegas, TCP Santa Cruz [57] tries to monitor the increase of network congestion and use this information in the operation of its congestion control mechanism. The operation of network state estimation is done over an interval, which is the time taken by a source to send a window of packets and receive all the acks. TCP Santa Cruz adjusts (increase, decrease or maintain as the current size) its window according to the current congestion state in order to avoid the swing in the

throughput that TCP Reno exhibits.

TCP Santa Cruz also proposes the modification in Slow Start and retransmission and recovery algorithms, which can effectively deal with the problems rising from asymmetric links, the loss of acks and multiple losses in a window of data.

2.4.4 TCP Westwood

TCP Westwood [53] is another TCP extension that aims at improving TCP's performance in wireless/wired network. The major difference between TCP Westwood and TCP Reno is that TCP Westwood uses the estimated bandwidth to set *ssthresh* as well as congestion window upon a packet loss, which is detected by either three duplicate acks or timeout. This way TCP Westwood avoids abrupt reduction of congestion window and Slow Start threshold.

The bandwidth measurement is carried out at the TCP sender-side. the TCP sender estimates the dispersion between two no-duplicate acks and uses this information to calculate the bandwidth available to the connection. This is based on the assumption that the rate of returning acks fully complies with the rate at which the acknowledged data segments are served in the forwarding path.

2.5 Bandwidth Measurement Technique

2.5.1 Bandwidth Related Metrics

Two bandwidth-related metrics frequently used are path capacity and available bandwidth. The path capacity of a route is the capacity of the lowest bandwidth link

(bottleneck link) of that route. As long as the route between the two hosts does not change, the path capacity of this route will remain the same and will not be affected by the network traffic. Specifically, if n is the number of links in the path, C_i is the capacity of link i , the path capacity is

$$C = \min_{i=0\dots n} C_i \quad (2.5)$$

The available bandwidth is the maximum capacity the route can offer at a specific point in time, given the current traffic load in the network. If U_i is the current utilization of link i , the unused capacity of link i is $C_i(1 - U_i)$, and the available bandwidth of the path is

$$C_a = \min_{i=1\dots n} C_i(1 - U_i) \quad (2.6)$$

2.5.2 Technique Classification

Bandwidth measurement techniques can be classified in several ways. A frequently used classification of measurement techniques is as follows:

- Active measurement techniques inject probe packets into the network and observe their behavior.
- Passive measurement techniques observe the existing traffic without perturbing the network.

Currently, there aren't any passive techniques that can efficiently estimate bandwidth.

The second classification is to divide the techniques into end-to-end and hop-by-hop. End-to-end techniques measure the path capacity and available bandwidth

of a path [77] [80] [86]. Hop-by-hop techniques measure path capacity C and if possible, available bandwidth of each link along the path [78] [79] [87] [88]. End-to-end metrics are more directly relevant to applications and transport protocol. Hop-by-hop metrics are considered to be more useful for network operations, debugging, and traffic engineering, especially in those cases where traffic engineers have limited view or access to the entire path.

A third classification method of bandwidth measurement techniques is whether the data are network manager-gathered or actively probed. Operators have used tools such as MRTG [90] to measure utilization of individual links with information obtained from router management software via SNMP. These techniques use counters and configuration parameters maintained by routers, and normally are accurate. However, SNMP-based techniques require access to the router, which is usually limited for widespread deployment. Active techniques, in general, require only the cooperation of path end-point. These techniques estimate bandwidth metrics by actively injecting the probe traffic into the path.

2.5.3 Packet Dispersion and End-to-End Bandwidth Measurement

Packet Dispersion is the time interval between two packets [80]. This dispersion reflects the rate at which the packets are served in the network, given the size of packets. Suppose that two packets are sent into the network from an endpoint, traverse through several links and arrive at another endpoint, as long as we can calculate the packet dispersion from the arrival time of the packets, we may estimate

the **end-to-end** service rate that is currently available to these packets, although we might not know the capacity of each link. The service rate estimated could be caused by either the available capacity of the path or the path capacity (bottleneck link capacity).

Fig. 2.6 illustrates the principle of packet dispersion, if the source sends a pair of packets back-to-back, these packets will be serviced by the bottleneck link and exhibit an inter-arrival time equivalent to the service time on the second packets. If the packet size is L , the path capacity is C , then

$$\Delta = L/C \quad (2.7)$$

where Δ is the dispersion between two consecutive probing packets at the destination.

Figure 2.6: Packet Pair Dispersion

The main problem that most packet dispersion techniques encounter is the impact of cross traffic interference. Cross traffic injected into the path may interfere with the probing packets, distorting the dispersion and eventually make it greatly distorted the measured results. “Packet Triplet”, our proposed end-to-end path capacity measurement technique proposed in our thesis, is developed based on packet dispersion

principle, dealing with the challenge of cross traffic. Packet Triplet technique will be presented in Chapter 3. Before that, we will introduce some related works that address the challenges in the area of bandwidth measurement.

2.5.4 Related Work

Over the past decades, a lot of research have focused in the area of bandwidth measurement. We introduce some typical techniques which use packet dispersion.

The concept of packet dispersion measurement technique, as a burst of packets traverses the narrow link of a path, was originally described in [36].

In [39], Keshav also studied the same idea in the context of congestion control, he recognized that the inter space of packet pairs is not related to the available bandwidth when the router are using First-Come-First-Served (FCFS) queueing discipline. He found that if all the routers use a fair queueing discipline, then the cross traffic is isolated and the packet pair technique can correctly estimate the available bandwidth, not necessarily the bottleneck in the path.

Bolot [75] used the packet pair mechanism to characterize the inter-arrivals of cross traffic. The early works on packet pair dispersion focus on robust statistical filtering techniques. Cater and Crovella [77] developed bprobe, in which several packet pair measurements, originating from packets of different sizes, are processed using union and intersection filtering to produce the final capacity estimate.

K.Lai and M.Baker [86] used kernel density estimator as their statistical filtering mechanism. However, the filtering algorithm in [86] and [77] is based on the assumption that the valid samples should be closely clustered around the correct value,

while incorrect samples should not be clustered around any one value, that implies the distribution of capacity estimates are commonly in unimodal. In fact, the assumption may not hold due to cross traffic interference. The multimodal distribution of bandwidth measurements was firstly observed by V.Paxon in his research [91].

Dovrolis and Ramanathan [80] had done meticulous researches on multiple modes of bandwidth measurement, the queueing effects that caused the multiple modes. Based on their research results, they developed an estimate tool Pathrate to detect the bandwidth metrics from the multimodal distribution.

CapProbe [85] is another Packet Dispersion technique that estimates the path capacity. CapProbe uses packet pair to probe the path of interest. Of each packet pair, CapProbe estimates the end-to-end delay of each probing packets and calculate the sum of these delays. The basic idea is that the pair of probing packets that have the minimum end-to-end delay sum catch the path capacity with the dispersion between them, because they are not delayed by any cross traffic in the path. The packet pairs that do not satisfy the *minimum delay sum* condition are filtered out as useless samples, because they are interfered by cross traffic.

Chapter 3

Path Capacity Measurement

3.1 Overview

Bandwidth estimation is of great interests to Internet service providers and network managers. Accurate bandwidth estimation can be used to improve end-to-end or peer-to-peer transport performance. Bandwidth estimation are also critical for traffic engineering and resources allocation. Bandwidth estimation techniques measure the these two bandwidth related metrics: path capacity and available bandwidth. Path capacity is the maximum bandwidth that a link or a path can have [77]. Available bandwidth is the unused bandwidth of a link or a path. In our works, we focus on the measurement of path capacity [87].

We propose a novel approach called Packet Triplet to pursue an accurate measurement of path capacity. Packet Triplet is an end-to-end based measurement technique. That is, the measurement is performed at the endpoints of the path without requiring the participation of intermediate nodes along the path. It is especially useful for

those who want to know the path capacity, but have no access to related information directly from the routers.

Packet Triplet is based on the principle of Packet Pair dispersion [77]. Unlike Packet Pair technique [77], Packet Triplet emits three back-to-back packets to probe the path from one end and obtain the dispersions of two pairs of two in-order packets at the other end. The filtering technique in Packet Triplet compares the difference between the two dispersions, using such information to estimate path capacity.

In Section 3.2, we present the principle of Packet Triplet. Section 3.3 proposes the analysis on the delay due to cross traffic. In Section 3.4, we show the experimental results.

3.2 Packet Triplet

Packet Triplet sender emits three back-to-back packets P_k ($k = 1, 2, 3$) of identical size L to probe a path, which may traverse N links with each link of capacity C_i ($i = 1 \dots N$). Herein at the receiver side, from the arrival time of each packet, we can calculate the dispersion (Δ_1) between P_1 and P_2 , and the dispersion (Δ_2) between P_2 and P_3 , as illustrated in Fig. 3.1.

Figure 3.1: Packet Triplet probe under ideal condition

Obviously, these two dispersions form a dispersion pair (Δ_1, Δ_2) . The main idea behind Packet Triplet is that undistorted Packet Triplet would have two identical dispersions. That is, if the probing packets go through the path without any disturbance, both dispersions $(\Delta_1$ or $\Delta_2)$ are L/C , where $C = \min_{i=1\dots N}(C_i)$. In fact, C is the actual capacity of bottleneck link along the whole path. Therefore, the path capacity C is estimated by $B_1 = L/\Delta_1$ or $B_2 = L/\Delta_2$. In ideal situation, $\Delta_1 = \Delta_2$, thus $B_1 = B_2$.

Such equivalence is very difficult to reach in reality because cross traffic along the path may vary from time to time, resulting in large variation of queueing delay and processing time. Nonetheless, in Packet Triplet, dispersion pair can be used to filter those distorted probing packets, and significantly improve the accuracy of path measurement. In the following part, we can see how the relation between two dispersions is mined to conduct such path capacity measurement. On the contrary, Packet Pair only has single dispersion measured during each probe, and lacks flexibility.

We assume that m Packet Triplet probes are successfully sent out. Thereafter, we can get m estimate pairs $(B_{j,1}, B_{j,2})$, $j = 1 \dots m$ at the receiver side. In real networks, some latency variation¹ is inevitable when probing packets traverse routers or some other intermediate nodes. Thus, such latency variation will result in a random variation between the dispersion $\Delta_{j,1}$ and $\Delta_{j,2}$ or a difference between $B_{j,1}$ and $B_{j,2}$. That is $|B_{j,1} - B_{j,2}| = L \times \frac{|\tau_b - \tau_b + \sigma|}{\Delta_{j,1}\Delta_{j,2}} \rightarrow 0$, where τ_b is transmission delay of a probing packet at bottleneck link and σ denotes the random variation, which is due to the

---

¹The latency is due to the packet propagation at the speed of light (propagation delay) plus the processing time that a router takes to look up routes in a routing table, and plus other fixed per-packet delays that a router incurs before it can forward a packet [87] [76]. There will be a little variation between the latency of different probing packets due to variability in tasks performed in a router, although they follow the same path.

latency variation. Overall, according to [76] [87], these variations can be neglected due to their small variations. For example, the measurements in [76] show that the typical latency for a Cisco core router is about $224\mu s$, while the variance of end-to-end latency tests is of the magnitude of just 10^{-11} to 10^{-10} .

However, when conducting measurement along an Internet path, the influence of cross traffic has to be taken into account. Fig. 3.2 gives an illustration of such interference between probing packets.

Figure 3.2: Distorted Packet Triplet Probe

For example, cross traffic packets may be injected between the first two probing packets in some cases, but in other cases, they may be injected between the last two probing packets, leading to dispersion extension; On the other hand, the dispersion may experience compression if the consecutive probing packets are queued to be served at a router after the bottleneck link.

3.3 Delay Analysis

Consider a path of M links with capacity $C = \{C_1, C_1, C_2, \dots, C_M\}$. Each link uses FIFO queueing discipline. Three back-to-back probing packets of size L are sent from the source to the destination. For each packet, the transmission delay of a link

i is $\tau_i = L/C_i$ and the end-to-end transmission delay is $\tau = \sum_{i=1}^N \tau_i$. Moreover, each packet experiences network latency, which is due to packet propagation at the speed of light (propagation delay), the time for a intermediate router to look up routes in a routing table, and other fixed per-packet delays that a router incurs before it can forward a packet. The latency for the probing packets is denoted by lat_1 , lat_2 and lat_3 , respectively.

In addition to transmission delay and latency, the probing packets may suffer queueing delay. We assume that the interval between two Packet Triplet probes is sufficiently large, therefore the probing packets from different Packet Triplet will never queue in the same buffer. Let d_1 , d_2 and d_3 denote the end-to-end queueing delay of the first, the second and the third probing packet, respectively. According to our assumption, d_1 is thus the end-to-end cross traffic induced delay that the first probing packet suffers along the path. d_2 and d_3 is due to cross traffic and/or the probing packets.

Let D_1 , D_2 and D_3 denote the end-to-end one way delay for the probing packets, respectively. The end-to-end one way delay is defined as the time between the departure of a probing packet at the sender and the arrival of that packet at the receiver.

We have:

$$\begin{cases} D_1 = d_1 + \tau + lat_1 \\ D_2 = d_2 + \tau + lat_2 \\ D_3 = d_3 + \tau + lat_3 \end{cases} \quad (3.1)$$

We assume that the transmission delay of the sender is very small and can be

Figure 3.3: Packet Triplet Dispersions

neglected. As Fig. 3.3 shows, at the receiver, the packet dispersions are:

$$\begin{cases} \Delta_1 = D_2 - D_1 = d_2 - d_1 + lat_2 - lat_1 \\ \Delta_2 = D_3 - D_2 = d_3 - d_2 + lat_3 - lat_2 \end{cases} \quad (3.2)$$

We define the difference between two dispersions as:

$$|\Delta_1 - \Delta_2| = |((d_3 - d_2) - (d_2 - d_1))| + |(lat_3 - 2lat_2 + lat_1)| \quad (3.3)$$

If there is no cross traffic interfering with probing packets, the queueing of probing packets will take place only at the bottleneck buffer and thus $d_2 - d_1 = \tau_b$ and $d_3 - d_2 = \tau_b$, where τ_b is the transmission delay for a probing packet at bottleneck link.

So we have:

$$\begin{cases} \Delta_1 = \tau_b + (lat_2 - lat_1) \end{cases} \quad (3.4)$$

$$\begin{cases} \Delta_2 = \tau_b + (lat_3 - lat_2) \end{cases} \quad (3.5)$$

The difference between Δ_1 and Δ_2 is:

$$|\Delta_1 - \Delta_2| = |(lat_3 - lat_2) - (lat_2 - lat_1)| \quad (3.6)$$

Equation 3.6 shows that in ideal condition, where a Packet Triplet probe is not distorted, the difference between Δ_1 and Δ_2 is only due to the difference between the

latency of probing packets. According to [87], such variation is however very small if the path is stable. So in such a case, we can claim that $|\Delta_1 - \Delta_2| \rightarrow 0$ or Δ_1 equals to Δ_2 . A crucial question arise in practice, when can we identify two dispersions the “equal” dispersions by taking latency factor into account in the Packet Triplet measurement?

We use a metric δ to reflect the difference between the two variables x_1 and x_2 :

$$\delta = \frac{\max(x_1, x_2) - \frac{x_1 + x_2}{2}}{\frac{x_1 + x_2}{2}}$$

In practice, we calculate the difference between bandwidth estimations B_1 and B_2 instead of dispersions. Such translation of unit is commonly used in the delay analysis of packet dispersion based measurement technique. By substituting x_1 and x_2 with B_1 and B_2 , the above equation is written as $\delta = \frac{\max(B_1, B_2) - \frac{B_1 + B_2}{2}}{\frac{B_1 + B_2}{2}}$. In Packet Triplet, given the value α (which is initially set to 0.01), if the difference δ is smaller than α , we claim that B_1 equals to B_2 (or Δ_1 equals to Δ_2). We refer to this relation as *equivalent condition*.

In the presence of cross traffic, the difference between the dispersion pair can be expressed as:

$$|\Delta_1 - \Delta_2| = |((d_3 - d_2) - (d_2 - d_1))| \quad (3.7)$$

where the second term of equation 3.3 plays no role and are eliminated. The left term in equation 3.7 shows the contribution of delay due to cross traffic and/or probing packets. The behavior of delay for each probing packet can be radically different depending on the interaction between the cross traffic and probing packets, therefore leading to difference between two dispersions. We now categorize and discuss the different effects of cross traffic on the dispersion pair:

1. The cross traffic create $\delta > \alpha$. This is created by different amount of cross traffic between probing packets. Such cases can be easily detected and discarded.
2. $\delta \leq \alpha$. **a)**. Equal amount of cross traffic between consecutive packets. In this case, two dispersions are in fact distorted, being either compressed or expanded. Compression leads to underestimation and expansion leads to overestimation. As they satisfy the equivalent condition, these samples has to be included in path capacity estimation. This is what we did not expect to occur. Fortunately, our experimental results show that the probability of such occurrence is lower than that of undistorted Packet Triplet. **b)**. Due to queueing at link further downstream from the bottleneck link. So in this case, the capacity of the downstream link might be taken as the path capacity. We will later show that the occurrence of this case can be reduced by selecting the probing packet size carefully.

3.4 Associated Filtering Technique

Based on the above investigations, we develop a receiver-side filtering technique to retrieve the valid data from the measured results.

The filtering algorithm works in the following manner:

1. Given the value α , if the sample pair $(B_{j,1}, B_{j,2})$ satisfies the equivalent condition ($\delta_j \leq \alpha$): this available sample pair is classified into different sets (i.e. [0-1Mbps], [1-2Mbps], [2-3Mbps]...). The width of each set is 1Mbps. The number of all available sample pairs ($\#avbl\_samples$) is calculated. if the sample pair

fail to meet the equivalent condition, Step 1 is repeated to a limit of 30 times.

We deem that the cross traffic is highly dynamic and has distorted the Packet Triplet dispersions quite a lot, and go to step 2.

2. α is increased by 20% to accommodate the more serious distortion due to cross traffic and helps to make the filtering process converge faster.
3. If the *#avl_samples* obtained is larger than 75², the measurement process is stopped; otherwise, the probing process repeat itself from step 1. When the estimation process finish, the sample set that has the largest frequency (i.e. contains the largest number of sample pairs involved in this set), is selected as the path capacity set. The mean value of the samples of this selected set is considered to be the path capacity.

3.5 Performance Evaluation

3.5.1 Experiment Topology

We use ns-2 to conduct our experiment to evaluate the performance of Packet Triplet. Fig. 3.4 shows the experimental topology. The capacity of each link along the path is $P = \{50, 20, 30, 10, 25, 40\}$ Mbps, thus the path capacity is 10Mbps. This link capacity setting applies for all the experiment scenario below. All the routers use FIFO queueing discipline. The buffer size of each link is 20. The round-trip time is 120ms. The cross traffic here is in Pareto distribution; the shape parameter for Pareto distribution is 1.7; both are similar to citeDovrolis.

---

²the value 75 is carefully chosen based on the results of multiple experiments to give attention to the requirement of not only accuracy but also speed of Packet Triplet measurement.

Figure 3.4: Experiment Topology

3.5.2 The size of probing packets

In this section, we investigate the effect of probing packet size on the performance of Packet Triplet measurement.

Many researches have been conducted to measure the effect of probing packet size on the accuracy of Packet Pair measurement [77] [80] [85]. It has been shown that different packet sizes may lead to different distorted measurement results, i.e the capacity could be under-estimated or over-estimated.

Such bias in measurement may also occur in Packet Triplet. To achieve the required accuracy, Packet Triplet must receive sufficient number of probes that do not suffer from cross traffic interference.

According to equation 2.7, reducing the probing packet size reduce the packet dispersion and thus reduce the chance for cross traffic being injected between probing packets. As the cross traffic between probing packets may cause the original dispersion to expand, reducing the probing packet size thus reduce the possibility of path capacity under-estimation.

However, reducing the packet size may also increase the probability of over-

estimation. Suppose the time taken for the queueing of first packet at a link is longer than the dispersion between the first and the second packet by the time the first packet reaches the link and just before the first packet is being served and so on for the queueing of the second packet, the Packet Triplet dispersions are thus compressed, leading to path capacity over-estimation.

On the other hand, Packet Triplet probes with increasing packet size suffer the opposite problem. The probability of over-estimation reduces but the probability of under-estimation increases.

The experiment brings us vivid understanding on the above discussion. The probing packet sizes are 100bytes and 1500bytes, respectively. Packet Triplet probes were emitted every 200ms, ensuring that the sending rate will not significantly increase the path rate. The cross traffic packet is uniformly distributed between 40 and 1500bytes. All links are 50% loaded by cross traffic. Fig. 3.5(a) shows the distribution of Packet Triplet estimation when the probing packet size is 100bytes. Fig. 3.5(b) illustrates the result when the probing packet size is 1500bytes.

Figure 3.5: Packet Triplet Estimation (Mbps) for different probing packet size, (a) 100bytes, (b) 1500bytes

As Fig. 3.5(a) depicts, the path capacity is over-estimated as the set standing around slot 22Mbps has highest frequency, while the set of correct path capacity (9 - 10Mbps) sample is yet significant. In Fig. 3.5(b), the sets of underestimated samples explicitly cluster around slots 3Mbps to 5Mbps, and no accurate samples can be obtained. For 100bytes packet size, the experiment needs about 640 probes to finish, while for 1500bytes, it requires about 1450 probes.

We can thus conclude that the packets of smaller size have higher chance of yielding the correct path capacity estimates and is more efficient, but on the other hand, the negative effect (over-estimation) due to small size has to be taken into account.

As a consequence, for Packet Triplet to be accurate and efficient, the packet size must be carefully set to achieve a balance between avoiding under-estimation and eliminating over-estimation. Through experiments, we found that 400bytes is the optimal size that can meet the requirement. Fig. 3.6 illustrates the percentage of correct estimation against *#avl_samples* (the number of all available samples) for different packet size under different cross traffic load. As the figure shows, when the cross traffic load is light (i.e. 20% load), the measurement using smaller size packets is more efficient than larger size packets. As the cross traffic load increase (i.e. 60% and 80% load), the aforementioned distortion effects (overestimation or underestimation) related to probing packet size become significant and the measurement using packet of 400bytes has the best performance.

Figure 3.6: Effect of packet size on Packet Triplet path capacity estimate accuracy

3.5.3 Experimental Results

Fig. 3.7 illustrates the experimental results of path capacity estimation distribution under different cross traffic load. The packet size of cross traffic injected on each hop is in uniform distribution between $[40, 1500]$ bytes. Fig. 3.7(a) shows the Packet Triplet path capacity estimation when each link is loaded with 20% of its capacity with cross traffic. The sample set of path capacity in the figure has the highest frequency and is determined as the accurate estimation, while the other sets of samples with much lower frequency can be eventually excluded. In addition, a so-called *post narrow capacity* [80] set stands along at slot 25Mbps. If three probing packets, which have gone through the narrow link, queue back-to-back again at any downstream congested link before finally arriving at the receiver without further being impacted by cross traffic, the dispersions introduced by that downstream link result in higher estimation, but not path capacity. Such link was so-called *post narrow link* [80].

In Fig. 3.7(b), all the links are 40% utilized. The path capacity set still dominates among all the estimates such that the filtering algorithm works. The result under 60% cross traffic load shown in Fig. 3.7(c) is similar to that in Fig. 3.7(b).

Figure 3.7: Packet Triplet path capacity estimation (Mbps) for cross traffic size in uniform distribution, (a) 20% cross traffic load, (b) 40% cross traffic load, (c) 60% cross traffic load, (d) 80% cross traffic load

Fig. 3.7(d) illustrates the estimation distribution when all links are 80% utilized. As the figure shows, Packet Triple's filtering mechanism is still effective in retrieving the path capacity from all the samples.

Table. 3.1 summarized the estimated path capacity and number of Packet Triplet

Table 3.1: Packet Triplet path capacity estimation

cross traffic load	path capacity (Mbps)	num of probes	time
20%	10.05	121	0'18
40%	10.16	387	1'00
60%	10.218	430	1'06
80%	10.075	722	1'54

probes that the sender needs to send and time taken (in minutes and seconds) for effective measurement, e.g. 121 probes under 20% cross traffic load. The results are accurate and number of probes required is dependent on the path load condition. More probes are required with increased cross traffic load along the path. In this experiment, we adjusted the time interval between consecutive probes based on round-trip time. The interval is set by $1.2 * \max(rtt)$, which ensures that consecutive probes won't affect each other. Packet Triplet probe period ranges from a few seconds to minutes. The current interval setting and *#avbl_samples* (refer to filtering step (4)) threshold in our experiment is conservative. We found that the Packet Triplet can converge much faster but yet effective if more aggressive strategies are used, for example, setting the *interval* = $\max(rtt)$ and *#avbl_sample* to 60. With such setting, the time taken for 80% cross traffic load is about only 1 minute 13 seconds and for 20% cross traffic load is about 12 seconds.

Fig. 3.8(a) and Fig. 3.8(b) show the distribution of about 500 Packet Pair samples when the path is 60% and 80% utilized, respectively. The experimental settings are the same as those for Packet Triplet in the previous section. As Fig. 3.8(a) and Fig. 3.8(b) illustrate, Packet Pair experiences serious distortion due to cross traffic and thus is unable to probe the path capacity accurately by just choosing the sample set with largest frequency. If that is done, in case of 60% cross traffic load, the

Figure 3.8: Packet Pair path capacity estimation (Mbps), (a) 60% cross traffic load, (b) 80% cross traffic load

estimated path capacity is 5.37Mbps and under 80% cross traffic load, the result is 4.41Mbps, both of which are quite different from the correct path capacity of 10Mbps.

Table 3.2: CapProbe Estimation

cross traffic load	path capacity (Mbps)	num of probes	time
20%	10	72	0'07
40%	10	207	0'30
60%	10	663	1'32
80%	10	885	2'36

We also evaluate Packet Triplet where there is a combination of three or four cross traffic packet sizes (40bytes, 550bytes, and 1500bytes). The probing packet size is 400bytes. The percentage of 40bytes, 550bytes and 1500bytes packets in aggregated cross traffic is 50%, 25% and 25%, respectively, which are similar to [80]. Fig. 3.9 illustrates the good performance of Packet Triplet's filtering process in mining the correct path capacity estimation, while the distribution of estimation is rather in the "discrete" form because of cross traffic of certain sizes, especially for heavy load (60% and 80%), as compared with the "continuous" dispersion in Fig. 3.7.

Figure 3.9: Packet Triplet Estimation (Mbps) for cross traffic size in triform distribution, (a) 20% cross traffic load, (b) 40% cross traffic load, (c) 60% cross traffic load, (d) 80% cross traffic load

We compared Packet Triplet with CapProbe [85], which uses Packet Pair to probe the path. The experimental setting for evaluating CapProbe is the same as those for Packet Triplet. Table 3.2 summarizes the result. For light and medium load (20% and 40%), CapProbe converges faster than Packet Triplet. However, for heavy load (60% and 80%), Packet Triplet converges faster than CapProbe. In CapProbe, an undistorted sample (which means its packet pair dispersion Δ satisfies $\Delta = L/C$, where C is path capacity) could even be regarded as *incorrect* sample because the

sample may not satisfy the *minimum delay sum* condition. For example, a pair of probing packets could be queued *back-to-back* due to cross traffic in bottleneck link buffer, therefore, experiencing a correct dispersion after leaving the bottleneck link. So the *minimum delay sum* condition is not satisfied because of queueing delay in bottleneck link, even if two probing packets can arrive at the endpoint with the correct dispersion. As the path load increase, this problem becomes more serious. Since Packet Triplet is solely based on the packet dispersion, this problem does not occur.

In the experiment, CapProbe is superior to Packet Triplet in terms of accuracy, only because CapProbe measurement will not be affected by the clock drift, which does not exist between the two ends of the path in ns-2 simulation environment.

3.6 Summary

Packet Pair is less effective in path capacity estimation when the cross traffic load varies. In this section, we introduce and study a new active path capacity measurement technique, called Packet Triplet. Three same size packets are sent to probe the path of interests. Packet Triplet uses the difference between two dispersions found between consecutive probing packets to filter out inaccurate samples that are distorted by cross traffic.

Our experimental results showed that Packet Triplet is able to estimate the path capacity accurately and effectively under different cross traffic load. The comparison between Packet Triplet and CapProbe shows that when the cross traffic load is heavy, Packet Triplet converge faster than CapProbe. This feature is especially useful in the

environment where the path capacity changes frequently, for example for wireless links.

Packet Triplet is easy to implement, as it is an end-to-end technique, which has no requirement of the participation of intermediate nodes in the path. It is also a friendly technique that will not aggravate the load of the probed network path.

Chapter 4

Packet Loss and Congestion State in TCP Veno

4.1 Overview

TCP Veno relies on round-trip delay to estimate the network congestion state. The estimated congestion state is used to infer the reason packet loss in heterogeneous environment: congestion loss or wireless transmission error. Based on the result of estimation and inference, TCP Veno carries out its refined congestion window reduce scheme and additive increase mechanism, aiming at improving TCP throughput in heterogeneous environment.

A previous work [25] has shown that it is TCP Veno's packet loss distinguishing and congestion window reducing scheme that contribute to improved throughput performance for TCP Veno compared to its additive increase algorithm. The improvement due to refined additive increase is slight in TCP Veno [25].

In this section, we revisit the congestion state estimation and packet loss distinguishing in TCP Veno in different network conditions. We study TCP Veno's performance when coexisting with TCP Reno connections under different cross traffic load. We observed that in some cases the accuracy in determining the cause of congestion loss using congestion state estimated is low when the network is heavily loaded. We revealed that the problem is due to the reason of congestion state underestimation [74].

We also investigate the effect of aggregation of multiple TCP Veno flows in different network conditions. By releasing the restriction of congestion loss rate, we observed that TCP Veno yields very different features of interest that had never been covered by the study of [10].

We define performance metrics in Section 4.2. In section 4.3, we introduce the experiment setting. In section 4.4, we evaluate the congestion state measurement and packet loss distinguishing accuracy. In section 4.5, we make a summary of our investigation.

4.2 Performance Metrics

In TCP Veno, the judgment of whether a packet loss is due to network congestion or wireless transmission error is deduced by the estimation of network state. Some exceptions that we cannot perceive may occur, resulting in erroneous diagnosis: the actual network state could be associated with an irrelevant packet loss or a packet loss is linked with the network state that is mistakenly inferred.

We define two performance metrics to evaluate the accuracy of different packet

loss recognitions:

- $P_{N \geq \beta|c}$: This metric is the conditional probability of identifying a packet loss as congestion loss given that it is due to congestion. It indicates the accuracy of congestion loss distinguishing.
- $P_{N < \beta|w}$: This metric is the conditional probability of identifying a packet loss as random loss given that it is random loss. It indicates the accuracy of random loss distinguishing

We substitute $P_{N \geq \beta|c}$ with $P_{c|c}$ and substitute $P_{N < \beta|w}$ with $P_{w|w}$ in the rest of this thesis.

In addition, we introduce the following metrics that will be employed for purpose of analysis in the rest of this thesis.

- $P_{N \geq \beta}$: This metric indicates the probability of $N \geq \beta$.
- $P_{N < \beta}$: This metric indicates the probability of $N < \beta$.

The value of N is sampled at the end of each *tagged RTT* during TCP Veno's congestion avoidance phase. For accuracy, the sampling is done throughout the lifetime of each TCP Veno connection and then we calculate the above metrics from sampled data. Since the congestion state estimation in TCP Veno is done during the congestion avoidance phase, in which the increment speed of congestion window depends on the congestion state the network is supposed to be in, the metric $P_{N \geq \beta}$ and $P_{N < \beta}$ ¹ thus approximately reflect the statistical feature and the "shape" of TCP Veno's congestion avoidance phase.

---

¹Note that $P_{N \geq \beta} = 1 - P_{N < \beta}$.

We will later see that these metrics play crucial role a in the investigation and analysis of TCP Veno's packet loss distinguishing.

4.3 Experiment Settings

In our experiments, we constructed two different network models. The first network model is used to investigate bandwidth sharing between TCP Veno and TCP Reno. This model has two different scenarios. In scenario 1, a TCP Veno connection and a TCP Reno connection share the bottleneck link with cross traffic between R1 and R2. In scenario 2, two Reno connections share the bottleneck link with cross traffic between R1 and R2.

The second network model consists of multiple TCP Veno connections sharing a bottleneck link, which we use for investigating the effect of aggregation of multiple TCP Veno flows in different network conditions.

Note that the parameters defined in the next section are applied in the rest of this thesis.

4.3.1 The network model of Bandwidth sharing between TCP Veno and TCP Reno

To compare TCP Veno's performance when it co-exists with TCP Reno connections under different cross traffic load, the experimental model is constructed as shown in Fig. 4.1.

As Fig. 4.1 shows, the link between R1 and R2 is a wired bottleneck link. The bottleneck link uses FIFO queueing discipline. The links between R2 and all TCP

Figure 4.1: Network Model

sinks are simulated as wireless links. The aggregating Pareto [68] cross traffic sources with shape parameters of 1.7 (< 2) produce Long Range Dependent (LRD) traffic.

Let τ denote the one way latency of bottleneck link, which does not include queueing delay; B represents the bandwidth of the bottleneck link. μ is the buffer size of the bottleneck link. The links between R2 and all TCP sinks are simulated as wireless links, with 2Mbps bandwidth [13], 0.1ms propagation delay and buffer size of 50 assigned. Let γ be the random loss rate (number of error packets/s) over wireless link. The links between R1 and TCP sources represent the wired links with 10Mbps bandwidth, 0.1ms round-trip time, and buffer size of 50. All the parameters are applied in the forward and backward direction.

Let CL be the cross traffic load (i.e., the percentage of bottleneck capacity approximately occupied by cross traffic). In this network model, we use different CL and γ to simulate different network conditions. Each experiment runs for 500 seconds.

μ , B and τ take 28, 4Mbps and 60ms, respectively. The buffer size of all the other links are set to 50 by default. These setting are applied for scenario1 and scenario 2.

In scenario 1, the packet size of TCP Veno and TCP Reno connection is 1460bytes. The links between TCP sources and R1 are assigned with 10Mbps and 0.1ms one-way propagation delay in both directions. The wireless links between R2 and TCP sink have 2Mbps capacity and 0.01ms one-way propagation delay in both directions. In scenario 2, the packet size of two TCP Reno connections are 1460bytes, the other settings are same as scenario 1.

4.3.2 Multiple TCP Veno connections

The model we use for studying the aggregation of multiple TCP Veno flows is shown in Fig. 4.2. Multiple TCP Veno connections start synchronously and run in a heterogeneous wired/wireless environment and share the same wired link between R1 and R2.

Figure 4.2: Experiment Topology

In this experiment, we calculate performance metrics according to the logged information. The simulation time is 1010 seconds. Each connection has a lifetime of 1000 seconds and always has data to send. The packet size of TCP Veno connections is 1460bytes. We vary μ , τ and B , respectively, to simulate different network conditions.

For clarity and convenience of analysis, when any one of these parameters is varied, the others are kept constant.

Unlike the experiment in [10], in our experiment *congestion loss rate* is not kept constant.

The parameter settings in the experiments are summarized as follows:

- γ is set to 1%. The experiment result with other values are similar.
- τ range from 5ms to 125ms, which means the round trip propagation delay is in the range 10ms to 250ms. When τ is fixed, we only show the result for $\tau = 50ms$, because the results with other values are similar.
- μ is set in the range 5 to 40. When μ is kept constant, we set it to 15.
- B is in the range 200kbps to 4Mbps. When B is kept constant, we set it to 4Mbps.
- N_c is the number of TCP Veno connection. N_c is set to 3 by default.

4.4 Bandwidth sharing between TCP Veno and TCP Reno

Table. 4.1, Table. 4.2 and Table. 4.3² (in page: 74, 75, 76) summarized the results for CL of 30%, 50% and 70% , respectively in scenario 1. We can observe the following characters: a) $P_{c|c}$ decreases and $P_{w|w}$ increases when CL is increased.

²In these tables, NUM_W is number of all random loss; $NUM_{W,N \geq \beta}$ is number of random loss in case of $N \geq \beta$; $NUM_{W,N < \beta}$ is number of random loss in case of $N < \beta$; NUM_C is number of all congestion loss; $NUM_{C,N \geq \beta}$ is number of congestion loss in case of $N \geq \beta$; $NUM_{C,N < \beta}$ is number of congestion loss in case of $N < \beta$.

Table 4.1: Experiment results for loss distinguishing in TCP Veno, $CL = 30\%$

γ	Throughput(Mbps)		Actual Type of Packet Loss						$P_{w w}$	$P_{c c}$
	TCP Veno	TCP Reno	Random Loss			Congestion Loss				
			NUM_W	$NUM_{W,N \geq \beta}$	$NUM_{W,N < \beta}$	NUM_C	$NUM_{C,N \geq \beta}$	$NUM_{C,N < \beta}$		
0	1.288	1.242	0	0	0	221	208	13	N/A	0.941
0.002	1.373	1.169	100	33	67	151	136	15	0.67	0.901
0.004	1.369	1.111	188	61	127	105	99	6	0.676	0.943
0.006	1.380	1.06	311	65	246	70	64	6	0.79	0.914
0.008	1.31	1.003	350	57	307	46	38	8	0.86	0.826
0.01	1.328	0.888	525	49	476	17	14	3	0.907	0.824

Table 4.2: Experiment results for loss distinguishing in TCP Veno, $CL = 50\%$

γ	Throughput(Mbps)		Actual Type of Packet Loss						$P_{w w}$	$P_{c c}$
	TCP	TCP	Num of Random Loss			Num of Congestion Loss				
	Veno	Reno	NUM_W	$NUM_{W,N \geq \beta}$	$NUM_{W,N < \beta}$	NUM_C	$NUM_{C,N \geq \beta}$	$NUM_{C,N < \beta}$		
0	1.041	0.909	0	0	0	278	173	105	N/A	0.622
0.002	0.967	0.802	62	17	45	248	157	91	0.726	0.633
0.004	1.007	0.822	115	22	93	241	122	119	0.809	0.506
0.006	1.027	0.781	217	25	192	188	73	115	0.885	0.388
0.008	1.032	0.777	326	28	298	158	81	77	0.914	0.513
0.01	1.041	0.69	365	31	334	141	56	85	0.915	0.392

Table 4.3: Experiment results for loss distinguishing in TCP Veno, $CL = 70\%$

γ	Throughput(Mbps)		Actual Type of Packet Loss							$P_{w w}$	$P_{c c}$
	TCP	TCP	Num of Random Loss			Num of Congestion Loss					
	Veno	Reno	NUM_W	$NUM_{W,N \geq \beta}$	$NUM_{W,N < \beta}$	NUM_C	$NUM_{C,N \geq \beta}$	$NUM_{C,N < \beta}$			
0	0.714	0.570	0	0	0	387	58	329	N/A	0.15	
0.002	0.746	0.556	44	3	41	386	45	341	0.931	0.117	
0.004	0.708	0.560	83	4	79	386	42	344	0.952	0.109	
0.006	0.748	0.527	131	6	125	343	22	321	0.954	0.064	
0.008	0.692	0.531	179	8	171	334	23	311	0.955	0.069	
0.01	0.681	0.488	203	8	195	287	32	255	0.961	0.111	

b) $P_{w|w}$ increases and $P_{c|c}$ exhibits fluctuation with the increase of γ , while CL is constant. c) When the network is heavily congested ($CL = 70\%$), TCP Veno is unable to differentiate between congestion loss and wireless loss ($P_{c|c}$ is very small).

(a). Let us look at the first observation. Referring to definition in [25], the backlog can be written as term $cwnd * (1 - BaseRTT/VRTT)$. When a TCP Veno connection is established in a heavily loaded network, $BaseRTT$ may be overestimated and thus the term $(1 - BaseRTT/VRTT)$ is smaller than the actual value. Consequently, the backlog is underestimated, resulting in misclassification of the cause of packet loss. Moreover, since TCP Veno connection occupies only a small fraction of total network capacity - about 15% of bottleneck capacity, the impact of this connection on congestion level is insignificant. This implies that $BaseRTT$ may still be overestimated even though TCP Veno drops congestion window to reduce the rate, because the network is not out of congestive state. The reason for increasing trends of $P_{w|w}$ can be explained as follows. If a random loss occurs when the network is in congestion state (at least b packets are queueing up in bottleneck buffer and buffer is not overflowed), the random loss will be regarded as congestion loss. Recall that on one hand, as CL increases, underestimation of network congestion state worsens the accuracy of congestion loss distinguishing.

(b). In the second observation, $P_{w|w}$ increases and $P_{c|c}$ exhibits fluctuation with increase in γ with CL constant. Frequent occurrence of random loss reduces the chance for TCP Veno to reach congestion state, thus reduces the probability of identifying random loss as congestion loss. We can see that under relatively small CL (30%, 50%), this trend is more significant than under extremely heavy CL (70%), because

a TCP flow which uses a very small fraction of network capacity makes minimal influence on network state, as what we have discussed above. Heavy cross traffic load and random loss results in small $P_{c|c}$ and large $P_{w|w}$.

(c). Third observation is that $P_{c|c}$ reduces with increase in CL. In extreme case that CL is 70% and γ is 0, $P_{c|c}$ is 0.15. It implies that only one in about each 7 congestion losses is accurately classified while the other 6 are misdiagnosed as random losses and correspondingly, congestion window is dropped only by 1/5. In such a case, one may think that a TCP Veno connection is very aggressive due to its low accuracy of congestion loss distinguishing and will grab more bandwidth resource from the co-existing TCP Reno connection. However, we found that in such an extreme case, TCP Veno can still keep its characteristic of compatibility - TCP Veno continues to work friendly with TCP Reno without "stealing" the bandwidth from TCP Reno. The congestion window of TCP Veno has been heavily restricted by frequent congestion losses in case of extreme heavy load. Thus, the difference between congestion window dropping by 1/2 and 1/5 is insignificant. For example, in Fig. 4.4, when CL is 70%, the average congestion window of the TCP Veno connection is only about 5.1 while the average congestion window of the TCP Reno connection is 4.9. Fig. 4.3 illustrates the picture of congestion window evolution for a TCP Reno and a TCP Veno connection and queue-length fluctuation for CL of 70% without artificially introducing random loss.

On the other hand, as depicted in Fig. 4.3, TCP Reno has the opportunity to increase its window size (e.g., around time slot 174s) beyond TCP Veno (Fig. 4.3(a)), although sometime (e.g. around time slot 178s) TCP Veno can maintain higher win-

(a) TCP Veno/TCP Reno congestion window

(b) Bottleneck queue length

Figure 4.3: TCP Veno/TCP Reno congestion window vs bottleneck queue length (no random loss) CL = 70%

dow size due to misclassification of packet loss when the network is heavily congested (Fig. 4.3(a) and Fig. 4.4). For instance, we found that around time slot 174s when a congestion loss took place but was diagnosed as random loss, TCP Veno tried to achieve more efficient utilization by the smaller window-dropping factor of $1/5$. This in turn expedited the advent of next congestion loss, which will be misdiagnosed again. Furthermore, in Fig. 4.4, it can be seen that Time-Out occurs with higher frequency, making either TCP Veno or TCP Reno connection drop its congestion

window to 1 and meanwhile giving other connections a chance to resynchronize their throughput. All these make TCP Veno's average congestion window comparable with the average congestion window of TCP Reno.

Figure 4.4: TCP Veno and TCP Reno Congestion window evolution (no random loss)

Figure 4.5: TCP Veno and TCP Reno throughput vs. avg throughput of Two TCP Reno connections

To verify the above statement, we do the experiment (scenario 2) with two TCP Reno connections rather than a TCP Veno and a TCP Reno connection, keeping the other setting in the aforementioned scenario 1 unchanged. The average throughput of

two TCP Reno connections under different CL (0, 10%, 30%, 50% 70%) are compared with the throughput of the TCP Reno and TCP Veno connection in scenario 1, γ is 0. As Fig. 4.5 depicts, the performance of TCP Reno in scenario 1 is scarcely impacted by TCP Veno, as compared with the average throughput of two TCP Reno connections.

4.5 Evaluation of Multiple TCP Veno model

4.5.1 Effect of Buffer Size

Figure 4.6: $P_{c|c}$ and $P_{w|w}$ versus bottleneck buffer size

We first evaluate the effect of buffer size on TCP Veno's packet loss identification. As Fig. 4.6 illustrates, $P_{c|c}$ increases while $P_{w|w}$ decreases with increase in μ , when the other parameters are kept constant. We note that in this experiment, as the buffer size increases, the congestion loss rate of TCP Veno connections decrease. This is

Figure 4.7: $P_{N \geq \beta}$ versus bottleneck buffer size

different from [10], in which the authors artificially kept the congestion loss rate constant by adjusting the load of background traffic load. Thus, the corresponding analysis model proposed in [10] cannot be applied here.

In order to explain the observation, we refer to the metrics $P_{N \geq \beta}$ and $P_{N < \beta}$. We observed that increasing of μ lead to increase of $P_{N \geq \beta}$ and i.e. the decrease of $P_{N < \beta}$, as illustrates in Fig. 4.7. The reason can be explained as follows. Let N_{max} be the maximum N that a TCP Veno connection can meet in one of its congestion avoidance phase and so the $cwnd_{max}$ accordingly. So we have

$$N_{max} = cwnd_{max} * (1 - \frac{BaseRTT}{VRTT})$$

As the buffer size increases while other parameters are kept constant, the buffering ability of the path increases, so $cwnd_{max}$ increases and thus increases N_{max} . Moreover, as the congestion window increases linearly, variation trends for $P_{N \geq \beta}$ thus follows the variation of the term $\frac{N_{max}-3}{N_{max}}$, which increases when the buffer size increases.

Figure 4.8: TCP Veno's congestion window Evaluation versus different bottleneck buffer size, (a), $\mu = 10$; (b), $\mu = 30$

Due to the definition of $P_{N \geq \beta}$, the larger $P_{N \geq \beta}$ implies that a TCP Veno connection can operate longer in the region of congestion and thus the probability that loss due to congestion in this region increases, which means $P_{c|c}$ increases. On the other hand, the probability that the loss due to wireless transmission error in the region of congestion also increases, and so $P_{w|w}$ decreases.

In order to show the above effect clearly, we conducted a scenario of a single TCP Veno connection in the absence of random loss, as Fig. 4.8 illustrates. Fig. 4.8(a) shows the TCP Veno congestion window when the buffer size is 10, and Fig. 4.8(b) shows the TCP Veno congestion window evolution when the buffer is 30. From the figures, we can see that when the buffer size is increased, the proportion of congestion duration that a TCP Veno connection experiences in a congestion avoidance phase increases.

Figure 4.9: $P_{c|c}$ and $P_{w|w}$ versus bottleneck bandwidth (Mbps)

4.5.2 Effect of Bottleneck bandwidth

We also explore the performance of packet loss identification with different bottleneck bandwidth (B). As Fig. 4.9 shows, $P_{c|c}$ increases and $P_{w|w}$ decreases when B increases. Similar to the effect of buffer size, we observed the increase of $P_{N \geq \beta}$ when B increase from 100kbps to 4Mbps, which can be used to explain the result of evaluation in this par, as Fig. 4.10 illustrates. The result is different from that of similar experiment in [10], in which they claimed that $P_{c|c}$ decreases and $P_{w|w}$ increases when B increases. Their conclusions however is drawn from the restriction that they kept the congestion loss rate constant.

In fact, this observation can be indirectly explained by our observation and conclusion in [74]: increasing the bottleneck bandwidth increases the fair share of each TCP Veno connection and thus decrease the probability of congestion state underestimation, and vice versa.

Figure 4.10: $P_{N \geq \beta}$ versus bottleneck bandwidth (Mbps)

4.5.3 Effect of Number of TCP Veno connections

We consider the effect of concurrent TCP Veno connections on the packet loss identification. We increase the TCP Veno connections from 3 to 11. The experiment with more connections has similar result. Fig. 4.11 shows the result: $P_{c|c}$ decreases and $P_{w|w}$ increases when the number of TCP Veno connections increases.

Increasing the number of connections decreases the bandwidth share of each connection, the result can also be explained using the aforementioned analysis about $P_{N \geq \beta}$ or congestion state underestimation.

4.5.4 Effect of RTT on packet loss identification

In this section, we investigate the effect of different RTT on TCP Veno's packet loss identification. Fig. 4.13 illustrates that $P_{c|c}$ decreases and $P_{w|w}$ increases when RTT increases.

Figure 4.11: $P_{c|c}$ and $P_{w|w}$ versus number of TCP Veno connections

The observations can be explained by the following mathematical analysis. The TCP Veno backlog $N = cwnd * (1 - \frac{BaseRTT}{VRTT})$ can be reconstructed as:

$$N = cwnd * (1 - \frac{BaseRTT}{BaseRTT + \Delta RTT}) \quad (4.1)$$

where ΔRTT is the delay due to congestion.

Referring to $N = cwnd * (1 - \frac{BaseRTT}{VRTT})$, we denote $cwnd_\beta$ as the congestion window when the backlog N equals to threshold β , and so the ΔRTT_β . So we have

$$\beta = cwnd_\beta * (1 - \frac{BaseRTT}{BaseRTT + \Delta RTT_\beta}) \quad (4.2)$$

Similarly, let $cwnd_{max}$ be the maximum congestion window when a packet loss occurs, and so the N_{max} and ΔRTT_{max} . We have

$$N_{max} = cwnd_{max} * (1 - \frac{BaseRTT}{BaseRTT + \Delta RTT_{max}}) \quad (4.3)$$

As Fig. 4.12 shows, the variation of $\frac{cwnd_\beta}{cwnd_{max}}$ reflects the variation of $P_{N \geq \beta}$ and

Figure 4.12: TCP Veno's Congestion Avoidance

Figure 4.13: $P_{c|c}$ and $P_{w|w}$ versus RTT

$P_{N<\beta}$. That is, the larger value of $\frac{cwnd_{\beta}}{cwnd_{max}}$ implies a larger $P_{N<\beta}$ and thus smaller $P_{N\geq\beta}$ and vice versa.

The term $\frac{cwnd_{\beta}}{cwnd_{max}}$ is:

$$\frac{cwnd_{\beta}}{cwnd_{max}} = \frac{\beta}{N_{max}} \times \frac{\Delta RTT_{max}}{\Delta RTT_{\beta}} \times \frac{2\tau + \Delta RTT_{\beta}}{2\tau + \Delta RTT_{max}} \quad (4.4)$$

where 2τ is approximately the $BaseRTT$.

As μ and B are kept constant and the congestion window increase linearly, N_{max} ,

ΔRTT_β and ΔRTT_{max} in equation 4.4 is thus fixed. As τ increases, the third term $\frac{2\tau + \Delta RTT_\beta}{2\tau + \Delta RTT_{max}}$ on the right side of equation 4.4 increases, so $\frac{cwnd_\beta}{cwnd_{max}}$ increases.

Figure 4.14: $P_{N \geq \beta}$ versus RTT

We can conclude not only from the above analysis but also from the experiments that $P_{N \geq \beta}$ decreases and $P_{N < \beta}$ when RTT increases, which can explain the experiment result illustrated by Fig. 4.13 – the decrease of $P_{N \geq \beta}$ predicts the decrease of $P_{c|c}$; while the increases of $P_{N < \beta}$ implies the increase of $P_{w|w}$. The $P_{N \geq \beta}$ sampled from the experiment is shown in Fig. 4.14.

Chapter 5

TCP VenO enhancement

5.1 Overview

In this section, we discuss the impact of bursty congestion on TCP VenO. Authors in [13] claimed that the transient congestion is a reason of misclassification for packet loss since a transient congestion loss is similar to a random wireless loss. We found that frequent transient congestion leads to poor quality of congestion loss identification and results in aggressive behavior of TCP VenO. In addition, the “step” congestion is another reason that was not mentioned in [13] and probably makes TCP VenO aggressive as well. The reason is that TCP VenO’s congestion state estimation scheme is incapable of tracing a sudden increase of congestion level if such increase (either transient or “step”) occurs within most recent round-trip time. Based on this observation, our definition of bursty congestion therefore consists of two parts. One is transient congestion, another is “step” congestion.

The congestion state measurement in TCP VenO often lags slightly behind the

current network state. This appears to be an inherent drawback in the presence of steep increase of network load (bursty congestion), as TCP Veno cannot associate the current fluctuation of congestion level with the packet loss due to buffer overflow, leading to incorrect inference as to the cause of packet loss. There are at least two scenarios when this condition occurs: 1). the transient congestion that last for less than one round trip delay; 2). “step-like jump” increase in network congestion level. The congestion lasts for more than one round-trip time. Unlike transient congestion, such “step” congestion can be recognized after a tagged RTT. Transient congestion and “step” congestion are defined as *bursty congestion*.

We propose an enhanced TCP Veno, called TCP Veno enhancement, which can track the dynamic of network congestion level. We develop a statistical model of TCP Veno throughput, which is the function of accuracy of packet loss identification. Experimental results demonstrate that the quality of identification of congestion loss in TCP Veno enhancement is improved in network with bursty congestion. We show that there is a tradeoff for our modification between accuracy of congestion loss identification and accuracy of wireless loss identification. Moreover, we observed that TCP Veno using our solution, TCP Veno enhancement, yields better friendliness while it competes with other TCP connections.

5.1.1 Impact of Bursty Congestion

Revisit Congestion Estimation In TCP Veno

TCP Veno uses *diff* to estimate the number of packets accumulated in bottleneck buffer. Let N be the backlog at the bottleneck queue. We have

$$N = diff * BaseRTT = cwnd * (1 - BaseRTT/VRTT) \quad (5.1)$$

TCP Veno assign a backlog threshold (β) ¹ to be the crossover point for differentiating between non-congestive and congestive state. If $N < \beta$ when a packet loss is detected, the network is supposed to be in non-congestive state and the packet loss is identified as random loss. If $N \geq \beta$, the network is believed to be congested and the packet loss is identified as congestion loss.

The Problem

In equation. 5.2, the smoothed average round-trip time $VRTT$ is calculated as:

$$VRTT = v\_sumRTT/v\_cntRTT \quad (5.2)$$

We define the round-trip time taken by TCP Veno to send a window of data segments and have all the ack back as tagged RTT. The data segment with largest sequence number within this tagged RTT is defined as tagged segment. Within the tagged RTT, for each data segment and its received ack, TCP Veno calculates the fine RTT, cumulatively adds it to v_sumRTT and adds v_cntRTT by one until the sequence number of received ack is larger than the sequence number of tagged data

---

¹In TCP Veno and TCP Veno enhancement, β is set to 3

segment, indicating that a tagged RTT is finished and next tagged round is starting. By using equation. 5.2 to average v_sumRTT of finished tagged RTT, we get $VRTT$.

In other words, TCP Veno is using the estimation obtained in the last finished tagged RTT to predict the current network congestion state. When there is no congestion or the variation of congestion level is smooth and slow, such lag effect is insignificant. However, if there are abrupt change of network traffic load, the estimation of backlog in the previous RTT may not be able to reflect the queueing size in bottleneck buffer that has been drastically changed (increasing or decreasing) in the current RTT, leading to misclassification of packet loss. For example, if the network state was estimated to be non-congested (i.e. $N < \beta$) during the finished tagged RTT, while currently the queueing size in the bottleneck buffer is suddenly stepped up by the incoming traffic, the packet loss due to the buffer overflow in the current RTT will be wrongly identified as wireless loss.

5.2 TCP Veno enhancement Mechanism

To detect the bursty congestion, we modified congestion state measurement and loss identification scheme in TCP Veno. The enhanced TCP Veno is called TCP Veno enhancement. The basic idea is that if we can instantly trace the abrupt change of queue size in the bottleneck buffer within the current tagged round, it is possible for us to distinguish between loss due to bursty congestion and wireless loss. We define a new metric N_{inst} , which is calculated by

$$N_{inst} = cwnd * (1 - baseRTT/VRTT_{inst}) \quad (5.3)$$

where $VRTT_{inst}$ is smoothed RTT for each data segment and its received ack measured in current tagged RTT. We use a low-pass filter to calculate $VRTT_{inst}$. The low-pass filter is an exponential weighted moving average (EWMA) [63]. The calculation is as follows:

Algorithm 3 The calculation of $VRTT_{inst}$

---

if ($v\_cntRTT = 1$) **then**

$VRTT_{inst} = MRTT$ {This is the first received ack after the finished tagged RTT}

else if ($v\_cntRTT > 1$) **then**

$VRTT_{inst} = \alpha MRTT + (1 - \alpha) VRTT_{inst}$

end if

---

where $MRTT$ is the round trip time measured for each data segment and its received ack. Here we set α to 0.2, with which 80% of each new $VRTT_{inst}$ comes from the previous measurement and 20% is from the current estimation. Of TCP Reno's smoothed RTT estimator, α is recommended to be 0.9 [63], which is less sensitive than our filter to the variation of RTT. Because we need to smooth our measurement such that the estimate of N_{inst} will not be too sensitive to tiny and fast change of queue length, but meanwhile, hope that $VRTT_{inst}$ won't response too slowly to the change of congestion state. At least, the response should be more sensitive than TCP Reno.

Our refined TCP Veno's packet loss identification scheme is:

The possible situations of which N and N_{inst} take different values are summarized as follows: 1) If $N_{inst} \geq \beta$ and $N \geq \beta$, it is obvious that network is in congestive state. 2) If $N_{inst} \geq \beta$ and $N < \beta$, a bursty congestion episode takes place currently

Algorithm 4 Refined packet loss identification scheme

---

if ($N_{inst} \geq \beta$ or $N \geq \beta$) **then**

The packet loss is congestion loss;

else if ($N_{inst} < \beta$ and $N < \beta$) **then**

The packet loss is wireless loss;

end if

---

right after the finished tagged RTT. 3) If $N_{inst} < \beta$ and $N \geq \beta$, we claimed that network is in congestive state, which is a decision more conservative than considering that the network is non-congested. 4) If $N_{inst} < \beta$ and $N < \beta$, the network is deemed to be non-congested.

5.3 Effectiveness of Packet Loss labeling

We derive a mathematical model to reflect the effect of packet loss identification $P_{c|c}$ and $P_{w|w}$ on TCP Veno throughput performance. For simplicity, we ignore the effect of timeout.

As aforementioned, TCP Veno's additive window increase may go through two phases, one is the same as TCP Reno, increases the congestion window by $1/w$ upon every ACK reception, another increases the congestion window by $1/w$ upon every other ACK reception. The window is increased as follows:

$$w_s(n+1) = \begin{cases} w_s(n) + \frac{1}{W} & \text{if } N < \beta \\ w_s(n) + \frac{1}{2W}(1-x) & \text{if } N \geq \beta \end{cases} \quad (5.4)$$

where N is the backlog and β is the threshold for packet loss identification; $x = 0$ or 1 , indicating that in case of $N \geq \beta$, window increase by $\frac{1}{W}$ every other ACK reception.

So the expected increment of the congestion window W per update step is

$$\begin{aligned} E[\Delta W_+] &= \frac{1}{W}(P_{N<\beta} + \frac{1}{2}P_{N\geq\beta})(1 - P_c)(1 - P_w) \\ &= \frac{A}{W} \end{aligned} \quad (5.5)$$

where P_c is congestion loss probability, P_w is wireless loss probability, $P_{N\geq\beta}$ is the probability of $N \geq \beta$, $P_{N<\beta}$ is the probability of $N < \beta$ and $A = (P_{N<\beta} + \frac{1}{2}P_{N\geq\beta})(1 - P_c)(1 - P_w)$

When TCP Veno detects packet loss, the window W is decreased according to:

$$\Delta W = \begin{cases} \frac{1}{5}W & \text{if } N < \beta \\ \frac{1}{2}W & \text{if } N \geq \beta \end{cases} \quad (5.6)$$

So the expected decrement of the congestion window W per update step is

$$\begin{aligned} E[\Delta W_-] &= W(\frac{1}{5}(P_w P_{w|w} + P_c P_{w|c}) + \frac{1}{2}(P_c P_{c|c} + P_w P_{c|w})) \\ &= WD \end{aligned} \quad (5.7)$$

where $D = \frac{1}{5}(P_w P_{w|w} + P_c P_{w|c}) + \frac{1}{2}(P_c P_{c|c} + P_w P_{c|w})$.

The expected change in the congestion window per update step is:

$$E[\Delta W] = \frac{A}{W} - WD \quad (5.8)$$

For TCP Veno, the interval ν between update steps is approximately:

$$\nu = \frac{RTT}{W} \quad (5.9)$$

The expected change in the sending rate r per unit time is approximately²:

$$\frac{dr(t)}{dt} = \frac{A}{RTT^2} - r^2(t)D \quad (5.10)$$

---

² $r(t) = \frac{W(t)}{RTT}$

By rearranging and integrating we have³:

$$\int_0^{r(t)} \frac{dr(t)}{\frac{A}{RTT^2} - r^2(t)D} = \int_0^t dt \quad (5.11)$$

$$r(t) = \frac{1}{RTT} \sqrt{\frac{A}{D}} \left(\frac{1 + \rho e^{-2at}}{1 - \rho e^{-2at}} \right) \quad (5.12)$$

where ρ depends on initial conditions and a is given by

$$a = \frac{1}{RTT} \sqrt{AD} \quad (5.13)$$

The steady state throughput of TCP Veno is thus given by:

$$\begin{aligned} r_V &= \lim_{t \rightarrow \infty} r(t) = \frac{1}{RTT} \sqrt{\frac{A}{D}} \\ &= \frac{1}{RTT} \sqrt{\frac{(P_{N<\beta} + \frac{1}{2}P_{N\geq\beta})(1 - P_c)(1 - P_w)}{\frac{1}{5}(P_w P_{w|w} + P_c P_{w|c}) + \frac{1}{2}(P_c P_{c|c} + P_w P_{c|w})}} \end{aligned} \quad (5.14)$$

Since $P_{w|w} + P_{c|w} = 1$ and $P_{w|c} + P_{c|c} = 1$, the above equation can be deduced as:

$$r_V = \frac{1}{RTT} \sqrt{\frac{(P_{N<\beta} + \frac{1}{2}P_{N\geq\beta})(1 - P_c)(1 - P_w)}{P_c(\frac{1}{5} + \frac{3}{10}P_{c|c}) + P_w(\frac{1}{2} - \frac{3}{10}P_{w|w})}} \quad (5.15)$$

Equation. (5.15) reveals the effect of accurate packet loss identification on TCP Veno's throughput. We assume that the packet loss rate, $P_{N<\beta}$ and $P_{N\geq\beta}$ are fixed, thus the higher value of $P_{w|w}$ (the accuracy of random loss identification) leads to higher throughput, while higher value of $P_{c|c}$ leads to lower throughput. This has been demonstrated by our experiment, where we can see that superior accuracy of loss identification of TCP Veno enhancement reduce the throughput slightly as compared with TCP Veno. But on the other hand, improvement in accuracy of loss

³The integrating is according to $\int \frac{dx}{a^2 - x^2} = \frac{1}{2a} \ln \left| \frac{a+x}{a-x} \right| + \rho$

identification make TCP Veno enhancement more friendliness towards TCP Reno connections compared to TCP Veno.

5.4 Experimental Setting

We compare the accuracy of packet loss identification in TCP Veno enhancement with TCP Veno. Comparisons are done with different round trip delay, different bottleneck buffer size and different background traffic rate. We also investigated how the modification affects TCP Veno's performance of friendliness and fairness, with co-existing TCP Reno connections.

Fig. 5.1 illustrates the hybrid wired/wireless network topology, which is same as the topology shown in Fig 4.1. The link between R1 and R2 represents the wired bottleneck link. μ indicates the buffer size of bottleneck link; τ denotes the one way latency (does not include queueing delay) of bottleneck link; B is the bandwidth of bottleneck link. All the parameters are applied for both forward and backward direction. The links between R2 and TCP sinks represent the wireless links, with 2Mbps bandwidth, 0.1ms round-trip time and buffer size of 50 assigned. Let γ be the wireless error rate in unit of packet. The links between R1 and TCP sources represent the wired links with 10Mbps bandwidth, 0.1 ms round-trip time, and buffer size of 50. We use VBR (variable bit rate) traffic in Poisson distribution to produce the background traffic in bottleneck buffer. As Fig. 5.1 shows, the aggregated background traffic enter into network at R1, traverse through the bottleneck link, and leave from R2.

To investigate the fairness in multiple TCP connection scenario, we use the fairness

Figure 5.1: The Simulation Topology

index [70]:

$$FairnessIndex = 1 - \frac{\sum_{i=1}^n |T_i - Avg|}{2(n-1)Avg}$$

where T_i is the throughput of i th TCP flow, Avg is average throughput of all TCP flows, and n is number of TCP flows. This fairness index is shown to be more sensitive in response of throughput change than the fairness index proposed in [13].

5.5 Performance Evaluation

We conduct the following experiment to evaluate the efficiency of our refined packet loss identification scheme in response to different network conditions. Three TCP Veno connections share the bottleneck link with bursty background traffic. B is 4Mbps. Each connection transfers 40Mbytes of data from sources to sinks to ensure that corresponding metrics can be accurately exploited with enough data packets. TCP packet size is 1460bytes. We vary μ , τ and background traffic rate respectively. In the experiment, when any one of these parameters varies, the others are kept constant.

Figure 5.2: The accuracy for different buffer size, RTT=60ms, background traffic load=10%

5.5.1 Impact of Bottleneck Buffer Size

First, we validate the performance of TCP Veno enhancement with different bottleneck buffer size. Fig. 5.2 illustrates average value of P_{clc} and $P_{w|w}$ for TCP Veno and TCP Veno enhancement connections. As the figure shows, P_{clc} is improved under different buffer size, while there is a little degradation in $P_{w|w}$.

We propose a simple mathematic analysis to explain the improved performance in congestion loss identification in TCP Veno enhancement. We introduce another variable $\tilde{N}$ for analysis. According to the refined packet loss identification scheme, we define the following substitution using $\tilde{N}$:

$$\begin{cases} N \geq \beta \text{ or } N_{inst} \geq \beta \longrightarrow \tilde{N} = \max(N, N_{inst}) \\ N < \beta \text{ and } N_{inst} < \beta \longrightarrow \tilde{N} = \max(N, N_{inst}) \end{cases} \quad (5.16)$$

From equation. 5.16, we can conclude that $\tilde{N} \geq N$. In another word, the new loss identification scheme is built on a metric ($\tilde{N}$) that is more sensitive to the increase of congestion than the one (N) used by the unmodified loss identification scheme.

Figure 5.3: The accuracy for different round trip delay, $\mu=15$, background traffic load=10%

As Fig. 5.2(a) shows, the accuracy of congestion loss identification is significantly improved. On the other hand, some wireless losses may occur during the episode of bursty congestion and are probably misclassified as congestion loss. Thus, using a loss identification scheme being more sensitive to congestion will also increase the opportunity of such wireless loss misclassification and thus, decrease the accuracy of wireless loss identification, as Fig. 5.2(b) illustrates.

5.5.2 Impact of RTT

Fig. 5.3 shows P_{clc} and $P_{w|w}$ in different round trip delay. $BaseRTT$ can be written as $BaseRTT = 2\tau + \sigma$, where σ is a delay variation, for example, the transmission time, the process time for packet in router or endpoint. Then, we have $\tilde{N} = cwnd * (1 - \frac{2\tau + \sigma}{RTT + \Delta RTT})$ and $N = cwnd * (1 - \frac{2\tau + \sigma}{RTT})$, where ΔRTT is a random variable that represents the smoothed variation of RTT, which can be zero or positive. Thus, $\frac{\partial \tilde{N}}{\partial \tau} = -cwnd * \frac{2}{RTT + \Delta RTT}$ and $\frac{\partial N}{\partial \tau} = -cwnd * \frac{2}{RTT}$. Since $\frac{\partial N}{\partial \tau} < \frac{\partial \tilde{N}}{\partial \tau} < 0$, as τ increases,

the decreasing speed of $\tilde{N}$ is slower than N , given the other network parameters unchanged. As $P_{c|c}$ of TCP Veno and TCP Veno enhancement follow the trend of $\tilde{N}$ and N respectively, we can expect that with increase of τ , $P_{c|c}$ of TCP Veno enhancement drops slower than TCP Veno, just as shown in Fig. 5.3(a). Furthermore, the decrease in $\tilde{N}$ also indicates that the probability of $\tilde{N} < \beta$ increases, which implies that the probability of wireless loss at congestive state ($\tilde{N} \geq \beta$) reduces. That's why as RTT increases, the degradation in $P_{w|w}$ of TCP Veno enhancement become less significant, as compared with $P_{w|w}$ of TCP Veno, as illustrated in Fig. 5.3(b).

Figure 5.4: The accuracy for different background traffic load, $\mu=15$, RTT=60ms

5.5.3 Impact of Traffic Load

We also compare the quality of packet loss identification between TCP Veno and TCP Veno enhancement under different background traffic load, as illustrated in Fig. 5.4. As Fig. 5.4(a) shows, the congestion loss identification of TCP Veno enhancement outperforms TCP Veno, but $P_{c|c}$ of TCP Veno enhancement decreases

Figure 5.5: The comparison of throughput and fairness, RTT=30ms

faster than that of TCP Veno as the background traffic load is increased. For $P_{w|w}$, the difference between TCP Veno enhancement and TCP Veno reduces with the increasing of background traffic load, as shown in Fig. 5.4(b). The reason is that as the load of background traffic increases, the system is more likely to be maintained in persistent congestion rather than in "bursty" congestion. Thus, the probability of underestimation of congestion state, which leads to frequent misclassification of congestion loss and high accuracy of wireless loss identification, becomes significant in bursty congestion.

5.5.4 Friendliness to TCP Reno

So far, we have investigated the affect of our idea under different network conditions in the presence of bursty traffic, showing that $P_{c|c}$ can be efficiently improved in TCP Veno with our idea incorporated and $P_{w|w}$ is slightly reduced. It is reasonable to expect that the improvement of $P_{c|c}$ should enhance the friendliness of TCP

Veno enhancement in the presence of bursty congestion. We use two scenarios to do the investigation. For one scenario, two TCP Veno and two TCP Reno connections share the bottleneck link. For another scenario, two TCP Veno enhancement and two TCP Reno connections share the bottleneck link. In both scenarios, μ is 15, RTT=30ms, and TCP packet size is 1460bytes. We experiment with different rates of background traffic, calculating the throughput and fairness index of TCP connections in two scenarios. Fig. 5.5 illustrates the results of throughput and fairness gathered from two scenarios. It is shown from the comparison in Fig. 5.5(a) and Fig. 5.5(b) that, friendliness of TCP Veno using refined control mechanism is improved.

5.6 Summary

TCP Veno makes use of measured congestion state to discriminate between different packet loss. We have shown that the measurement of congestion state in TCP Veno is not able to follow the fluctuation of bursty congestion, the packet loss due to such bursty congestion is likely to be misclassified. We propose a solution, called TCP Veno enhancement, to minimize this problem.

The experiment results show that our solution can significantly improve the accuracy of congestion loss identification in TCP Veno in the presence of bursty congestion, at the cost of a slight reduction in the accuracy of wireless loss identification. Moreover, we observed that friendliness of TCP Veno enhancement is improved, as compared with TCP Veno.

Chapter 6

Conclusions and Future work

6.1 Summary

In this thesis we firstly focus on how to determine the bottleneck bandwidth. Active measurement techniques like Packet Pair (which derive from the principle of self-clocking in TCPs congestion control) have been increasingly used to determine the bottleneck bandwidth. One of the major issues is to maximize the statistical correctness of the measurement. We proposed a new end-to-end based measurement technique, called Packet Triplet, to achieve a higher probability of finding the bottleneck link. A new filtering technique, is introduced into Packet Triplet to further refine such measurement approach. It has shown better performance in terms of better “accuracy” even in heavily loaded network. The different effects that cross traffic may bring to the measurement are studied and experimental results strongly back up this idea.

Next, we investigate the performance of transport protocol, TCP. TCP is widely

accepted as the transport layer protocol in today's Internet. It makes up a large proportion of the traffic on the Internet. Our research work studied the cause of packet loss and the estimated queued packets (to estimate current state of network congestion) along the path in TCP's congestion control. We choose TCP Veno, which is a refinement designed for better TCP performance in heterogeneous environment, to be the object of our study. TCP Veno uses packet loss as well as congestion state monitored based on queued packet in the path as its congestion control signals.

Based on our investigation, we have further refined the state estimation algorithm in TCP Veno to give a more accurate classification of the loss in the presence of bursty congestion. The enhanced TCP Veno is called TCP Veno enhancement.

In Section 6.2, we draw a summary of the contributions of this thesis. We present the directions of future works in Section 6.3.

6.2 Research Contributions

6.2.1 Packet Triplet

Packet Triplet aims at estimating the bottleneck bandwidth of a network path. It is designed based on the principle of packet pair dispersion. Packet Triplet sends three back-to-back probing packets at one end of path of interest. At the other endpoint, a pair of dispersions can be found from the arrival of two consecutive probing packets. A packet triplet probe that comes with two different dispersions is distorted by the cross traffic. We study the effect of delay due to cross traffic on the dispersions of Packet Triplet. Based on our study, a filtering algorithm is proposed to filter out the

distorted samples and find the accurate estimations of bottleneck bandwidth.

We also investigate the effect of size of probing packets on the accuracy and convergence of Packet Triplet measurement. The Packet Triplet of different packet size can be distorted in different manner and thus lead to different distributions of path capacity estimation. Decreasing the packet size reduces the probability of bandwidth underestimation, but on the other hand, increases the probability of bandwidth overestimation; while increasing the packet size results in the opposite effect. To give attention to diminishing the impact of different distortions, the probing packet size is carefully selected.

The experimental results show that Packet Triplet is able to estimate the path capacity accurately and effectively even if the path is heavily loaded by cross traffic. Packet Triplet converges fast. This is a very important feature for network environment in which the routing and capacity change fast, such as wireless network.

Packet Triplet is easily deployed in a network. It is an end-to-end approach of which the measurement process does not require any change at intermediate nodes in the path. The measurement is done at the two ends of the path.

Packet Triplet is also friendly. It uses a sophisticated approach to keep the probing flow from increasing the load of the path.

6.2.2 Packet Loss and Congestion State in TCP Veno

TCP Veno is a TCP variation proposed for improving TCP throughput performance in heterogeneous environment. It makes use of a mechanism borrowed from TCP Vegas to estimate the network congestion state. TCP Veno associates the esti-

mated congestion state with its refined congestion avoidance algorithm as well as its packet loss identification and window reduction strategy.

In this thesis, we investigate the performance of congestion state estimation and packet loss identification in TCP Veno. Before our work, [10] had investigated the ability of TCP Vegas's congestion state estimation algorithm as a potential packet loss predictor. Unfortunately, they obtained negative results that TCP Vegas's congestion state estimation algorithm does not always perform well. Nevertheless, their observations cannot be applied to TCP Veno, because: all the necessary parameters and performance metrics in [10] are estimated based on a single Reno connection. Consider that TCP Veno's behavior is quite different from Reno, the proposed results thus may not accurately reflect the situations in the presence of data flow controlled by TCP Veno. Further, some experiments in [10] were conducted by artificially fixing the congestion loss rate of a Reno connection when the other network parameters change. This restriction makes it easy to explain the observations supported by a simple mathematical model given in [10], but it is not practical. In reality, the congestion loss rate of a TCP connection can change due to the variation of many factors in the network, i.e. link bandwidth, bottleneck buffer size or round trip time.

To address the above problems, there is no restriction of congestion loss rate in our experiments. We define a set of metrics to help indicate and analyze the performance of interest under different network condition. We investigate the effect of variation of network factors in different aspects, i.e, bottleneck bandwidth, bottleneck buffer size, round trip time, traffic load, number of TCP Veno connections and wireless loss rate. We found the problem of congestion state underestimation when the network

is heavily loaded. This problem may seriously reduce the accuracy of congestion loss identification. The reason of underestimation is that $BaseRTT$ could be overestimated. We also observed that, because of removing the restriction on congestion loss rate, the results from our investigation are opposite to the conclusions in [10], for example, we found that when the bottleneck bandwidth decreases, the accuracy of congestion loss identification reduces; while in [10], they claimed that this metric increases, given that the congestion loss rate is artificially kept constant.

By studying TCP Veno's congestion avoidance algorithm that uses the congestion state information to adjust the window's increasing speed, we revealed that the refined congestion avoidance algorithm also plays a crucial role in affecting and explaining the variation trends of TCP Veno's packet loss identification.

As a summary, the experiments and analysis we presented in this thesis suggest that TCP Veno cannot always performs well in congestion state estimation and packet loss identifications.

6.2.3 TCP Veno enhancement

In addition the reasons revealed in the above works, another fact that contributes to the inaccuracy of packet loss identification is bursty congestion, which can adversely affect TCP Veno's performance of congestion loss identification especially even when the network is lightly loaded. We found that the reason is that the state estimation responds too slowly to the occurrence of bursty congestion. Based on our investigation, we have further refined the state estimation algorithm in TCP Veno to give a more accurate classification of the loss in the presence of bursty congestion, which is

called TCP Veno enhancement.

To have more accuracy of congestion loss identification, we introduce two parameters in TCP Veno enhancement: instantaneous round trip time ($VRTT_{inst}$) and number of queue packets (N_{inst}). The refined estimation algorithm using these parameters makes TCP Veno enhancement catch up with the bursty congestion that can suddenly increase the congestion level in the period of a RTT. The accuracy of congestion loss identification is significantly improved in the presence of bursty congestion. As a consequence, TCP Veno enhancement uses our solutions exhibits better friendliness while it competes with other TCP connections.

6.3 Challenges and Future Works

We now discuss several directions and challenges for future works. Both are the extension of the works presented in this thesis.

6.3.1 Packet Triplet and Bandwidth Measurement

So far, we have evaluated Packet Triplet as an active bandwidth measurement technique and demonstrated the effectiveness of Packet Triplet for our experimental settings. However, several challenges remain in our work.

First, we assume that all the routers are using FCFS (First-Come First-Served) queueing discipline. It is thus necessary to make further investigation on how Packet Triplet performs in the system using other queueing algorithms, e.g. fair queueing algorithm or classful queueing for differentiated service control.

Second, the challenges of high speed bandwidth (e.g. 1Gbps or even higher) in

near future. One challenge is that the higher the network bandwidth, the dispersion between two packets will be shorter, resulting in granular resolution. A dispersion cannot be accurately measured when it is smaller than or much close to the resolution of system timer. For example, the current Unix system time resolution is $1\mu s$, so if two packets of 1500bytes are transmitted across a 10Gbps network, the dispersion is about $1.2\mu s$, which cannot be accurately measured by the current time resolution. Another challenge is that the network bandwidth exceeds the physical I/O speed of measurement host. The low speed host using Packet Pair dispersion based techniques thus can only measure the capacity of measurement host but not network bandwidth. [82] [83] advised that the Packet Train based algorithm, which is less sensitive to timer resolution, is a feasible way of high-speed bandwidth measurement in the future. So we are to use packet train with N packets (N is even and is larger than 3) to yield detectable dispersions (i.e. the dispersion between the 1th and the $(1+n)/2$ th packet, and between the $(1+n)/2$ th and n th packet, respectively). However, using packet train increases the probability of cross traffic interference. So the length of packet train must be taken into account in the future design.

Finally, we will draw our attention in future on how to use bandwidth measurement in network performance management, such as network routing, traffic engineering, resource reservation and distribution as well as end-to-end admission control.

6.3.2 Develop a new end-to-end congestion control algorithm

The experiments and analysis proposed in this thesis have revealed the feature of TCP Veno in respects of congestion state estimation, efficiency of packet loss

identification. Our work shows that there are several drawbacks in TCP Veno's congestion control mechanisms. The congestion state could be underestimated due to the overestimation of $BaseRTT$; the threshold for judging the state of connection (congestion or non-congestion) is fixed independent of network conditions.

Moreover, [50] found that the correlation between packet loss and the increase of RTT is very weak, especially in high-speed network where a TCP flow consumes a small fraction of total bandwidth. This observation therefore implies that TCP Veno's packet loss distinguishing based on RTT increment has poor accuracy in such circumstance.

Therefore, developing a new end-to-end congestion control algorithm that can overcome the above drawbacks of TCP Veno is a potential effort in future direction. An alternative approach that can evade the problem is to use bandwidth estimation in TCP's congestion control. TCP Westwood [53] uses estimated available bandwidth to adjust its congestion window and slow-start threshold and therefore achieves performance improvement in heterogeneous environment. The bandwidth in data forwarding direction (from the TCP sender to the TCP receiver) is estimated from the dispersion between two acknowledgement packets at the TCP sender. This idea is actually a variant of packet pair algorithm. However, if the dispersion between two acknowledgement packets were distorted in the reverse path, the network bandwidth will be erroneously estimated. The way to avoid this problem is to estimate the network bandwidth using data packets at the TCP receiver, like the suggestion in [82] and [83]. To do this, some related issues are to be considered in the future, such as the filtering algorithm in bandwidth estimation or the aforementioned challenges that

comes up with the increase of network bandwidth.

In addition to using bandwidth measurement in TCP design, the performance of TCP (include TCP Veno) in high-speed network are to be addressed. In high-speed network, TCP's linear increase is too slow; Multiplicative decreases in window occur when packet loss is too drastic. The end-to-end loss rate needed by TCP to attain a very large congestion window is impossible to be achieved in practice. There are several works to address the difficulties of TCP in high-speed network [32] [38] [44] [53]. These algorithms has more aggressive linear increase and more gentle multiply decrease, as compared to conventional TCP. HighSpeed TCP [32] and Scalable TCP [44] are equation-based solutions which rely on the accurate estimation of end-to-end loss rate. However, the loss rate is difficult to be accurately estimated, especially in low congestion environment where the loss rate is very small [38]. Fast TCP [38], as opposed to equation-based solutions, borrows the idea of TCP Vegas [3], using the increase of RTT to infer the congestion and addressing the problem of TCP in high-speed network. Fast TCP is however observed to experience the problem that TCP Vegas also has: TCP ACK packets traversing in the reverse path could be delayed by other traffic [14] [21]. TCP Westwood [53], which makes use of bandwidth estimation to adjust congestion window, also has the problem caused by cross traffic on reverse path. We are aiming to address the limitations of these TCP variants. For example, one possible solution is to do the bandwidth estimation at TCP receiver. We can also take the advantages of ideas in the above algorithms into our design in the future.

Bibliography

- [1] Allman, M.; Paxson, V.; Stevens, W.; “TCP Congestion Control”, RFC 2581, Apr. 1999.
- [2] Arauz, J.; Banerjee, S.; Krishnamurthy, K.; “MAITE: a scheme for improving the performance of TCP over wireless channels”, in Proceedings of the IEEE Vehicular Technology Conference (VTC), Volume 1, Page(s):252-256, Oct. 2001.
- [3] Brakmo, L.S.; Malley, S.W.O.; Peterson, L.; “TCP Vegas: New Techniques for Congestion Detection and Avoidance”, in Proceedings Of ACM SIGCOMM94, Page(s): 24-35, London, Oct. 1994.
- [4] Balakrishnan, H.; Seshan, S.; Katz, R.H.; “Improving reliable transport and hand-off performance in cellular wireless networks”, ACM/Baltzer Wireless Networks Journal, Volume 1, No. 4, Page(s): 469-481, Dec. 1995.
- [5] Bakre, A.; Badrinath, B.R.; “I-TCP: Indirect TCP for Mobile Hosts”, in Proceedings of ICDCS, Page(s): 136-143, May 1995.
- [6] Balakrishnan, H.; Seshan, S.; Katz, R.H.; “A Comparison of Mechanisms for Improving TCP Performance over Wireless Links”, Networking, IEEE/ACM Transactions on Networking, Volume 5, Issue 6, Page(s): 756-769, Dec. 1997.

-
- [7] Bakshi, B.; Vaidya, N.H.; “Improving performance of TCP/IP over wireless networks”, in Proceedings of the 17th International Conference on Distributed Computing Systems, Page(s): 365, 1997.
  - [8] Brown, K.; Singh, S.; “M-TCP: TCP for Mobile Cellular Networks”, ACM SIGCOMM Computer Communication Review, Volume 27, Issue 5, Page(s): 19 - 43, Oct. 1997.
  - [9] Biaz, S, Mehta, M, West, S, Vaidya, N.; “TCP over Wireless Networks Using Multiple Acknowledgements”, Technical Report: TR97-001, 1997.
  - [10] Biaz, S.; Vaidya. N.H.; “Distinguishing Congestion Losses from Wireless Transmission Losses: A Negative Result”, Seventh International Conference on Computer Communications and Networks (IC3N), New Orleans, Page(s):722 - 731, Oct. 1998.
  - [11] Bansal, D.; Balakrishnan, H.; “TCP-friendly congestion control real-time streaming application”, MIT technical report, May 2000.
  - [12] Balan, R.K.; Lee, B. P.; Kumar, K.R.R.; “TCP HACK: TCP Header Checksum Option to Improve Performance over Lossy Links”, INFOCOM 2001, Volume 1, 22-26, Page(s):309-318, Apr. 2001.
  - [13] Barman, D.; Matta, I.; “Effectiveness of Loss Labeling in Improving TCP Performance in Wired/Wireless Networks”, IEEE International Conference on Network Protocols, Pairs, France, 12-15, Page(s): 2-11, 2002.
  - [14] Bullo, H.; Cottrell, R.L.; Hughes-Jones, R.; “Evaluation of Advanced TCP

- Stacks on Fast Long-Distance Production Networks”, in J. of Grid Computing, Volume 1, Page(s): 345C359, Dec. 2003.
- [15] Chandran, K.; Raghunathan, S.; Venkatesan, S.; Prakash, R.; “A Feedback based scheme for improving TCP performance in ad-hoc networks”, in Proceedings of the The 18th International Conference on Distributed Computing Systems, Page(s): 472-478, May 1998.
- [16] Cceres, R.; Iftode, L.; “Imroving the Performance of Reliable Transport Protocols in Mobile Computing Environments”, IEEE Journal of Selected Areas in Communication, Volume. 13, No. 5, Page(s): 850-857, June 1995.
- [17] Cardwell, N.; “Modeling the performance of short TCP connections”, Tech. Rep.; Department of Computer Science and Engineering, Univ. of Washington, 1998.
- [18] Chockalingam, A.; Zorzi, M.; Tralli, V.; “Wireless TCP performance with link layer FEC/ARQ”, ICC99, Volume 2, Page(s):1212-1216, June 1999.
- [19] Cardwell, N.; Savage, S.; Anderson, T.; “Modeling TCP Latency”, INFOCOM 2000. Nineteenth Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings. IEEE, Volume 3, 26-30, Page(s):1742-1751, 2000.
- [20] Deshpande, N.; “TCP Extensions for Wireless Networks”, *http* :
 //www.cse.wustl.edu/ jain/cis788 – 99/ftp/tcp_wireless/.
- [21] Dalton, L. A.; Isen, C; “A study on high speed TCP protocols”, Global Telecommunications Conference, GLOBECOM’04. IEEE, Volume 2, Page(s): 851-855, Nov. 2004.

-
- [22] Fall, K.; Floyd, S.; "Simulation-based Comparison of Tahoe, Reno, and SACK TCP", Computer Communication Review, Volume 26, Issue 3, July 1996.
- [23] Fu, C.P.; Chung, L.C.; Liew, S.C.; "Performance Degradation of TCP Vegas in Asymmetric Network and Its Remedies", IEEE ICC 2001, Volume 10, Page(s): 11-14, June 2001.
- [24] Fu, C.P.; Lu, W.; Lee, B.S.; "TCP Veno Revisited", IEEE Globecom 2003, Volume 6, 1-5, Page(s): 3253-3258 vol.6, Dec. 2003.
- [25] Fu, C.P.; Liew, S.C.; "TCP Veno: TCP Enhancement for Transmission over Wireless Access Networks", IEEE (JSAC) Journal of Selected Areas in Communications, Volume 21, Issue 2, Page(s): 216 - 228, Feb. 2003.
- [26] Fu, C.P.; Li, X.; Nie, N.; Li, L. M.; "Veno II: Universal Transport in Next Generation Communications", Mainland - Hong Kong Academic Exchange, June, 2005.
- [27] Floyd, S.; Jacobson, V.; "On traffic phase effects in packet switched gateways", Internetworking:Res. Experience, Volume 3, Page(s): 115-156, 1992.
- [28] Floyd, S.; "TCP and Explicit Congestion Notification", ACM Computer Communication Review, V. 24 N. 5, Page(s): 10-23, Oct. 1994.
- [29] Floyd, S.; Handley, M.; Padhye, J.; "A comparison of equation-based and AIMD congestion control", www.icir.org/tfrc/aimd.pdf, May 12, 2000.
- [30] Floyd, S.; Handley, M.; Padhye, J.; "Equation-based congestion control for unicast application", in Proceedings of ACM SIGCOMM, Page(s):45-58, Aug. 2000.

-
- [31] Floyd, S.; Fall, K.; “Promoting the Use of End-to-End Congestion Control in the Internet”, IEEE/ACM Transactions on Networking, Volume 7, Issue 4, Page(s):458-472, 2000.
- [32] Floyd, S.; Ratnasamy, S.; Shenker, S.; “Modifying TCP’s Congestion Control for High Speeds”, Rough draft, May 2002.
- [33] Goff, T.; Moronski, J.; Phatak, D.S.; “Freeze-TCP: A true end-to-end TCP enhancement mechanism for mobile environments”, in Proceedings of INFOCOM 2000. Nineteenth Annual Joint Conference of the IEEE Computer and Communications Societies. Volume 3, 26-30 Page(s):1537-1545, March 2000.
- [34] Gurtov, A.; Floyd, S.; “Modeling Wireless Links for Transport Protocols”, CM CCR, 34(2): 85-96, April 2004.
- [35] Hengartner, U.; Bolliger, J.; Gross, T.; “TCP Vegas revisited”, in Proceedings of the IEEE INFOCOM, Tel Aviv, Israel, Volume 3, Page(s):1546-1555, March 2000.
- [36] Jacobson, V.; “Congestion Avoidance and Control”, in Proceedings ACM SIGCOMM, Page(s): 314-329, 1988.
- [37] Jacobson, V.; “Modified TCP Congestion Avoidance Algorithm”, end2end-interest mailing list, April 30,1990.
- [38] Jin, C.; D. Wei, Low, S.H.; Buhrmaster, G.; Bunn, J.; Choe, D.H.; Cottrell, R.L.A.; Doyle, J.C.; Feng, W.; Martin, O.; Newman, H.; Paganini, F.; Ravot, S.;

- Singh, S.; "FAST TCP: from theory to experiments", *Network, IEEE*, Volume 19, Issue 1, Page(s): 4-11, Jan.-Feb. 2005.
- [39] Keshav, S.; "A Control-Theoretic Approach to Flow Control", in *Proceedings of ACM SIGCOMM'91*, Page(s): 3-15, Sept. 1991.
- [40] Karn, P.; Partridge, C.; "Improving Round-Trip Time Estimates in Reliable Transport Protocols", *ACM Transactions on Computer Systems (TOCS)*, Volume 9, Issue 4, Page(s): 364-373, 1991.
- [41] Kelly, F.; "Mathematical modeling of the Internet", *Proceedings of the Fourth International Congress on Industrial and Applied Mathematics*, Page(s): 105-116, 1999.
- [42] Kim, D.K.; Toh, C.K.; Choi, Y.H.; "TCP-BuS: Improving TCP Performance in Wireless Ad Hoc Networks", *Journal of Communications and Networks*, Volume. 3, No. 2, Page(s): 1-12, June. 2001.
- [43] Konrad, A.; Zhao, B.Y.; Joseph, A. D.; Ludwig, R.; "A Markov-based channel model algorithm for wireless networks", *ACM/Baltzer Wireless Networks*, Volume 9, No.3, Page(s): 189 - 199, May 2003.
- [44] Kelly, T.; "Scalable TCP: improving performance in high speed wide area networks", in *Processings of 1st Intern. Workshop on Protocols for Fast Long-Distance Networks*, Geneva, Switzerland, Feb. 2003.
- [45] Lakshman, T.V.; Madhow, U.; "The Performance of TCP/IP for Networks with

- High Bandwidth-Delay Products and Random Loss”, IEEE/ACM Transactions on Networking, Volume 5, No. 3, June 1997.
- [46] Lakshman, T.V.; Madhow Bernhard Sute, U.; “Window-based error recovery and flow control with a slow acknowledgement channel: a study of TCP/IP performance”, INFOCOM, Volume 3, 7-11, Page(s):1199-1209, 1997.
- [47] Ludwig, R.; “A Case for Flow-Adaptive Wireless Links”, Technical Report UCB// CSD-99-1053, University of California at Berkeley, May 1999.
- [48] Ludwig, R.; Katz, R.H.; “The Eifel Algorithm: Making TCP Robust Against SpuriousRetransmissions”, ACM Computer Communication Review, Volume 30, No. 1, Jan. 2000.
- [49] Mo, J.; La, R.J.; Anantharam, V.; Walrand, J.; “Analysis and Comparison of TCP. Reno and Vegas”, In Proceedings of INFOCOM '99, Page(s): 1556-1563, March 1999.
- [50] Martin, J.; Nillson, A.; Rhee, I.; “Delay-based congestion avoidance for TCP”, IEEE/ACM Transactions on Networking, Volume 11, Issue 3, Page(s): 356-369, June 2003.
- [51] Mathis, M.; Mahdavi, J.; Floyd, S.; Romanow, A.; “TCP Selective Acknowledgement Options”, RFC 2018, Oct. 1996.
- [52] Mathis, M.; Semke, J.; Mahdavi, J.; Ott, T.; “The macroscopic behaviour of the TCP congestion avoidance algorithm”, Computer Communication Review, Volume 27, Page(s): 67-82, 1997.

-
- [53] Mario, G.; Sanadidi, M. Y.; Wang, R.; Zanella, A.; “TCP Westwood: congestion window control using bandwidth estimation”, Global Telecommunications Conference, GLOBECOM '01. Volume 3, 25-29, Page(s): 1698-1702, 2001.
- [54] Mascolo, S.; Casetti, C.; Gerla, M.; Sanadidi, M.Y.; Wang, R.; “TCP Westwood: Bandwidth Estimation for Enhanced Transport over Wireless Links”, International Conference on Mobile Computing and Networking, Page(s): 287-297, 2001.
- [55] Postel, J.; “Internet Protocol”, RFC 791, Sept. 1981.
- [56] Postel, J.; “Transmission Control Protocol”, RFC 793, September 1981.
- [57] Parsa, C.; Garcia-Luna-Aceves, J.J.; “Improving TCP Congestion Control Over internets with Heterogeneous Transmission Media”, IEEE ICNP 99, Page(s): 213-221, 1999.
- [58] Padhye, J.; Firoiu, V.; Towsley, D.; “A stochastic model of TCP Reno congestion avoidance and control”, Tech. Rep. UMASS-CS-TR-1999-02.
- [59] Parsa, C.; Garcia-Luna-Aceves, J.J.; “Improving TCP Performance over Wireless Networks at The Link Layer”, Mobile Networks and Applications, Volume 5, Issue 1, Page(s): 57-71, 2000.
- [60] Padhye, J.; Firoiu, V.; Towsley, D.F.; Kurose, J. F.; “Modeling TCP Reno Performance: A Simple Model and Its Empirical Validation”, IEEE/ACM TRANSACTIONS ON NETWORKING, Volume 8, No. 2, Page(s): 133-145, Apr. 2000.

-
- [61] Richard J.L.; Walrand, J.; Anantharam, V.; “Issues in TCP Vegas”, <http://www.eecs.berkeley.edu/~ananth/1999-2001/Richard/IssuesInTCPVegas.pdf>
  - [62] Ramani, R.; Karandikar, A.; “Explicit congestion notification in TCP over wireless networks”, Proc. IEEE Int. Conference on Personal Wireless Communications (ICPWC’00), Page(s): 495-499, 2000.
  - [63] Stevens, R.W.; “TCP/IP illustrated”, Addison-Wesley, 1994.
  - [64] Suter, B.; Lakshman, T.; Stiliadis, D.; Choudhury, A.; “Design considerations for supporting TCP with per-flow queuing”, in Proceedings of IEEE INFOCOM98, Volume 1, Page(s): 299-306, 1998.
  - [65] Sisalem, D.; Schulzrinne, H.; “The loss-delay based adjustment algorithm: a TCP-friendly adaptation scheme”, In Proceedings of NOSSDAV’98, Cambridge, England, July 1998.
  - [66] Sinha, P.; Venkitaraman, N.; Nandagopal, T.; Sivakumar, R.; Bharghavan, V.; “A Wireless Transmission Control Protocol for CDPD”, IEEE Wireless Communications and Networking Conference, New Orleans, LA. 21-24, Volume 2, Page(s): 953-957, 1999.
  - [67] Sinha, P.; Venkitaraman, N.; Sivakumar, R.; Bharghavan, V.; “WTCP: A Reliable Transport Protocol for Wireless Wide-Area Networks”, ACM Mobicom 99, Seattle. WA. Page(s): 231-241, August 1999.
  - [68] Taqqu, M.S.; Willinger, W.; Sherman, R.; “Proof of a Fundamental Result in

- Self-Similar Traffic Modeling”, ACM Computer Communications Review, Page(s): 5-23, Apr. 1997.
- [69] Tsaoussidis, V.; Badr, H.; Verma, R.; “WWP - An energy saving transport protocol for mobile IP-Devices”, in Proceedings of the Seventh Annual International Conference on Network Protocols, Page(s): 301, 1999.
- [70] Vardalis, D.; Tsaoussidis, V.; “Efficiency/Fairness Tradeoffs in Networks with Wireless Components and Transient Congestion”, The Journal of Supercomputing, Volume 23, Page(s): 281-296, 2002.
- [71] Wright, G.R.; Stevens, W.R.; “TCP/IP Illustrated”, Volume 2 (The Implementation), Addison Wesley, Jan. 1995.
- [72] Zhang, C.; Tsaoussidis, V.; “TCP-Real”, Computer Networks: The International Journal of Computer and Telecommunications Networking, Volume 40, Issue 4, Page(s): 477-497, Nov.2002.
- [73] Zhang, Y.; Delucia, D.; Ryu, B.; Dao, S.; “Satellite communications in the global Internet: issues, pitfalls, and potential”, in Proceedings of INET 1997.
- [74] Zou, Z.X.; Lee, B.S., Fu, C.P.; “Packet Loss and Congestion State In TCP Veno”, IEEE International Conference on Networks, Volume 2, 16-19, Page(s):731-735, 2004.
- [75] Bolot, J.C.; “End-to-End Packet Delay and Loss Behavior in the Internet”, in Proceedings of ACM SIGCOMM, Page(s): 289-298, September 1993.

- [76] Bovy, C.J.; Mertodimedjo, H.T.; Hooghiemstra, G.; Uijterwall, H.; Mieghem, P. V.; “Analysis of End-to-End Delay Measurements in Internet”, in Proceedings of Passive and Active Measurements Workshop (PAM2002), Ft.Collins, March 2002.
- [77] Carter, R.L.; Crovella, M. E.; “Measuring Bottleneck Link Speed in Packet-Switched Networks”, Technial Report BU-CS-96-006, Boston University, Page(s): 8-11, 1996.
- [78] Downey, A.B.; “Using pathchar to Estimate Internet Link Characteristics”, in Proceedings of ACM SIGCOMM’99, Page(s): 241-250, 1999.
- [79] Downey, A.B. “clink: a Tool for Estimating Internet Path Characteristics”, <http://rocky.wellesley.edu/downey/clink/>, June 1999.
- [80] Dovrolis, C.; Ramanathan, P.; Moore, D.; “What do packet dispersion techniques measure?”, INFOCOM 2001. Twentieth Annual Joint Conference of the IEEE Computer and Communications Societies, Volume 2, Page(s): 2-7, 2001.
- [81] IP Performance Metrics, <http://www.ietf.org/html.charters/ippm-charter.html>
- [82] Jin, G. J.; “Algorithms and Requirements for Measuring Network Bandwidth”, Lawrence Berkeley National Laboratory, technical report LBNL-48330, 2003.
- [83] Jin, G. J.; Tierney, B. L.; “System Capability Effects on Algorithms for Network Bandwidth Measurement”, in Proceedings of the 3rd ACM SIGCOMM conference on Internet measurement, Page(s): 27-38, 2003.
- [84] Jacobson, V.; “pathchar: a Tool to infer characteristics of Internet paths”, Apr. 1997, <ftp://ftp.ee.lbl.gov/pathchar/msri-talk>.

-
- [85] Kapoor, R.; Chen, L. J.; Sanadidi, M. Y.; and Gerla, M.; “CapProbe: A Simple and Accurate Technique to Measure Path Capacity”, Technical Report TR040001, UCLA CSD, 2004.
- [86] Lai, K.; Baker, M.; “Measuring Bandwidth”, in Proceedings of IEEE INFOCOM. Apr. 1999.
- [87] Lai, K.; Baker, M.; “Measuring Link Bandwidth Using a Deterministic Model of Packet Delay”, ACM SIGCOMM Computer Communication Review , Volume 30, Issue 4, Aug. 2000.
- [88] Lai, K. Baker, M.; “Nettimer: A Tool for Measuring Bottleneck Link Bandwidth”, In Proceedings of USITS’ 01, March 2001.
- [89] Mansour, Y.; Patt-Shamir, B.; “Jitter control in QoS networks”, Foundations of Computer Science, 1998. in Proceedings of 39th Annual Symposium, Page(s): 50-59, Nov. 1998.
- [90] Oetiker, T.; “MRTG: Multi Router Traffic Grapher”, <http://people.ee.ethz.ch/~oetiker/webtools/mrtg/paper/>.
- [91] Paxson, V.; “Measurements and Analysis of End-to-End Internet Dynamics”, PhD thesis. University of California, Berkely, Apr. 1997.
- [92] Paxson, V.; Alems, G.; Mahdavi, J.; Mathis, M.; “Framework for IP Performance Metrics”, RFC 2330, May 1998.

List of Publications

Journal

Zou Z.X., Lee B.S.; Fu C.P.; Song J.; “Packet Triplet: A Novel Approach to Estimate Path Capacity”, IEEE Communications Letters, Volume 9, Issue 12, Page(s): 1076-1078, Dec. 2005.

Conference

[1] Zou Z.X.; Lee B.S.; Fu C.P.; Song J.; “Packet Triplet: An Enhanced Packet Pair Probing for Path Capacity Estimation”, Asia Pacific Advanced Network, Page(s): 311-314, 2003.

[2] Zou Z.X.; Fu C.P.; Lee B.S.; “Congestion State and Packet Loss in TCP Veno”, IEEE International Conference on Network, Volume 2, 16-19, Pages: 731-735, Nov. 2004

[3] Zou Z.X.; Fu C.P.; Lee B.S.; “A refinement to Improving TCP Veno Performance under Bursty Congestion”, IEEE Global Telecommunications Conference, GLOBE-COM, Volume 2, Page(s): 5 pp. Dec. 2005.