A Robust Wireless Transmission Control Protocol to Cope with Channel Errors in a Long Round-Trip Delay Environment[1]

Shu-wen Teng
VIA Technologies, Inc. - 533 Chung-Cheng Road 8F., Hsin-Tien, - Taipei 231, Taiwan
e-mail: KellyTeng@via.com.tw

Jin-Fu Chang[2]
National Chi Nan University - Department of Electrical Engineering - Puli, Nantou, Taiwan 545
e-mail: jfchang@ncnu.edu.tw

ABSTRACT

Extending the TCP (transmission control protocol) to the wireless domain is a hot research topic at the present time. It is also implemented in a long round-trip delay, e.g., satellite, channel. Performance degradation due to TCP's misinterpretation of corruption loss as congestion loss has been reported in the literature. This paper proposes a threshold-based TCP to prevent it from reacting to cell losses due to corruption, that is, channel error. It is demonstrated through numerical experiments that the proposed scheme can effectively mitigate the negative impact of wireless channel errors on the performance of TCP.

1. INTRODUCTION

A. Motivation

As we have marched into the second year of the twenty-first century, it is more clear then ever that computer networks has become an inalienable part of human civilization. Computer systems are weaved through communication facilities to form a networked cyberspace. Nowadays, a computer system does far more than its traditional role of data crunching and word processing. Networked computers provide us with a cyber-world of info-diversity. A good wealth of multimedia and on-demand contents has been cultivated in the Internet cyber-land. A far apart website is merely one click away. But how does the task of networking eventually become a reality? A crucial ingredient clearly is the sophisticated communication protocols that make different computer systems talk to each other.

Communication protocols are structured into the division-of-labor layers so that each layer is commissioned a unique responsibility. Although different layers are independent and non-interfering, they are stacked together to form a function-able suite so that a lower layer protocol serves the one sitting on its top.

In the family of protocols, TCP/IP (transmission control protocol/ internet protocol) undoubtedly is the builder of the Internet. TCP is responsible for congestion control whereas IP takes care of routing.

Although TCP/IP was originally engineered for wired links which are nowadays mostly optical fiber lines. It has been implemented in both terrestrial and satellite wireless channels. To extend TCP/IP from the scope of wired links to radio channels, modification is a mandate and has been enthusiastically pursued by researchers recently. In this paper, we concentrate on the discussion of wireless TCP in a long round-trip time environment such as a satellite channel.

A contrast between an optical fiber link and a radio link lies in their degree of cleanness and serenity. An optical fiber link nowadays is almost immune to channel errors; while a radio channel is contaminated by not only all kinds of noise but also fading induced largely by multi-path interference. That is to say packet or cell losses due to irrecoverable channel errors are almost nonexistent in a wired link but do exist in a radio channel. Cell losses are perceived by TCP as indication of congestion and should be acted upon. If TCP is utilized in the wireless world without modification, corruption losses are misinterpreted as congestion losses and overreacted then the throughput performance of TCP is seriously harmed. This phenomenon of performance is now widely recognized, e.g. [1]-[3], and needs to be cured. A number of remedial measures have also been proposed, e.g. [1]-[3]. The purpose of this paper is to propose a novel threshold-based wireless TCP that is simple but efficient. This threshold-based scheme is designed to fit a long round-trip time such as satellite environment.

In a link routed through a geo-synchronous satellite, the round trip propagation delay maybe as long as 558 ms. Even in the environment of LEO (low earth orbit) or MEO (medium earth orbit) satellites, the delay may range from several mini-seconds to 80 ms.

B. A Quick Look at TCP

It is not the purpose of this paper to furnish a thorough review of TCP. Rather we shall take only a quick look at its spirits. Please consult [4] for detailed description of TCP.

TCP is hung at the fourth or transport layer of OSI (open system interconnect)'s 7-layer structure. It sits on top of IP.

[1]This work is performed while the authors were with the Graduate Institute of Communication Engineering, National Taiwan University, Taipei, Taiwan 10764. Presentation of this paper is made possible by a grant from the National Science Council, Taiwan (NSC 91-2219-E-260-001).
[2] Corresponding author

TCP/IP is the soul of Internet. TCP performs both error and congestion control via the routing services provided by IP to create a reliable connection oriented data services to support a wide variety of Internet applications.

TCP's error control is accomplished through a sliding window mechanism in conjunction with positive acknowledgements and retransmission timeouts. Although congestion control is also handled by the network layer, it is mostly handled by TCP. TCP controls congestion through two windows: the advertised window at the receiving end and the congestion window at the sending end. TCP uses two phases, slow start and congestion avoidance, to adjust the width/size of the congestion control window denoted by *cwnd*.

Switch from slow start to congestion avoidance is controlled by the parameter ssthreshold in which ss stands for slow start. Another important parameter affecting the performance of TCP is the retransmission time out (RTO). TCP uses sampled round trip times (RTTs) to adjust the RTO.

We wish to point out that irrecoverable lost packets are treated by TCP as indications of congestion and acted upon by shrinking *cwnd* to 1 whenever congestion is detected.

To expedite recovery of lost packets a modified TCP with fast retransmit and fast recovery is proposed [5].

Since the original TCP was invented [4], a good variety of TCP modifications have been proposed. They include TCP Tahoe [5], TCP Reno [5], TCP New Reno [6], TCP SACK [7], TCP Vegas [8], TCP Westwood [9], and etc. A common goal of these different versions of TCP is to seek better throughput performance.

C. Prior Works

We have pointed out earlier that in extending the operation of TCP from a clean wired medium to a noisy radio channel, a potential harm is that corruption losses, i.e., packet losses due to channel errors, are misinterpreted as congestion losses and wrongly acted upon by the congestion control of TCP. This problem has been widely recognized and a number of cures have been proposed.

The cure can be arranged in at least either of the following two directions or a combination of both: link layer enhancements and TCP layer enhancements.

Enhancements at the link layer maybe done through either forward error correction (FEC) or automatic repeat request (ARQ) or a hybrid of them. Note that link layer resides at the second of the 7 layers of OSI and enhancement at this layer aims at curing channel impairment done on individual packets. Error control at the link layer has been very thoroughly treated by the research community and is not the focus of this paper.

At least two kinds of enhancement at TCP layer have been proposed: split-connection protocol and end-to-end protocol [1]. The idea of split-connection is to split an end-to-end TCP connection into a wired plus a wireless sub-connection by placing a TCP agent at the router that separates these two sub-connections. It is the TCP agent not the sender that handles retransmission of lost packets in the wireless channel. When a packet sent from sender arrives at the agent an ACK is returned to the sender as if the agent were the end of the connection. In other words, a TCP connection is essentially split into two independently run TCP connections. This approach is called indirect TCP [1]. A similar approach is called snoop TCP [1] . The end-to-end feature is maintained in snoop TCP meaning that the TCP connection is not terminated at the agent. The duplicate of a packet received from sender by the agent is stored at the agent and no ACK is returned. Non-duplicate ACK returning from receiver is forwarded to sender; but duplicate ACKs are intercepted by the agent and retransmission of corruption losses is taken care by the agent. The agent is further equipped with a timer to estimate if a packet has timed out at sender so that retransmission can start earlier at the snoop agent. Clearly, this snoop TCP also has a split-correction nature.

In the end-to-end protocol, the idea is to try to distinguish congestion losses from corruption losses. Whenever a corruption loss is detected, the TCP layer is prevented from taking any congestion avoidance action.

The rest of this paper is organized as follows. In sec. II, we propose a threshold-based scheme to differentiate corruption from congestion losses. In Sec III we conduct a series of numerical experiments to demonstrate that the throughput performance of a TCP connection is indeed unaffected by the presence of a wireless channel. Finally, conclusions are given in section IV.

2. A THRESHOLD-BASED SCHEME TO DIFFERENTIATE CORRUPTION FROM CONGESTION

If we are able to differentiate corruption from congestion losses, the congestion control mechanism of TCP does not have to be invoked in the wake that corruption losses are detected. In other words cells lost due to channel errors are retransmitted, mostly at the data link layer, without adjusting the congestion window.

In the operation of TCP, the sampled *rtt* is used to trigger the adjustment of congestion window. When *rtt* gets larger, the congestion window has to be made smaller; and vice versa. Let us demonstrate a relationship between *rtt* and *cwnd* by performing an experiment for a simple TCP connection using the Network Simulator [10] developed at the Lawrence Berkeley Laboratory. The result is plotted in Fig. 1 where one sees a strong correlation between the behavior of *rtt* and *cwnd*. In Fig. 1 (the sampled) rtt is expressed in terms of the number of TCP ticks and the value of a TCP tick is set at 0.01sec.

The correlation between *rtt* and *cwnd* seems to suggest that as long as *rtt* is small the corresponding TCP connection is not very likely to fall into congestion and cell losses henceforth are due mostly to channel errors. This then gives an idea to use a threshold to differentiate congestion from corruption. If *rtt* appears to be smaller than this threshold then losses are interpreted as corruption and otherwise. Each TCP connection has a minimal *rtt* value known as the base *rtt*. The threshold must be chosen to be larger than this base *rtt*. How large a threshold should be will be answered when we present numerical experiments.

This threshold-based approach is very easy to implement and involves very little modification to the original TCP design. It requires only two extra memory spaces at the sending end to store the base *rtt* and the threshold; and a simple comparison with the threshold.

For that TCP Reno is the most widely used version of TCP, we shall in the sequel use it for the purpose of demonstrating our idea. But our approach is definitely applicable to any version of TCP.

3. NUMERICAL EXPERIMENTS

We shall in the following examine how our proposed threshold-based scheme behaves to react to congestion losses through numerical experiments conducted on the Network Simulator. In these experiments the following three channel error rates are tried: 10^{-5}, 10^{-6}, and 10^{-7}. We consider a continual transmission of a large file, that is, a typical FTP application. Each simulation run continues for a period of 100 sec. The round trip signal propagation time is in the order of 500ms to emulate a satellite link.

A. Single Connection Case

We use Fig. 2 to demonstrate the behavior of *cwnd* before and after the application of the threshold scheme. In this figure the threshold is set at 550 ms or 55 TCP ticks. The difference is clearly noticed.

In Fig. 3, we plot the throughput performance versus threshold value for BER (bit error rate)= 10^{-5}, 10^{-6}, and 10^{-7}.In this figure, base *rtt* is equal to 51 TCP ticks. We observe first that in each BER throughput performance improves as the threshold value is pushed up. Second, throughput performance already gets significant improvement when the threshold is put at 53 and starts to level off when the threshold is pushed further. Third, the improvement is most profound in BER=10^{-5}, then 10^{-6} and 10^{-7}. This trend is of course reasonable.

B. Plural Connection Case

We have also conducted simulations for the case of two and three connections. Similar trends are seen in these experiments. Figs. 4-7 are results we have obtained.

We show in Fig. 8 the performance of our proposed threshold scheme in a short delay environment where the round trip time is 50ms and the base *rtt* is set at 6 TCP ticks. We again observe the improvement in throughput due to the implementation of a threshold scheme.

C. No Disruption to the Performance of a wired TCP connection

We use Fig. 9 to demonstrate that our threshold scheme causes no disruption to the performance of a wired connection.

4. Conclusions

We have in this paper engineered a threshold-based TCP to prevent it from reacting to losses due to channel errors for a long round-trip delay environment. This design involves only very minor modification to the current TCP design. Through extensive numerical experiments we have conducted one indeed witness the mitigation of the negative impact of channel errors on the behavior of TCP. We further demonstrate that our proposed scheme brings no disruption to the performance of a wired TCP connection.

Whether this threshold-based idea works for a very short delay environment such as terrestrial mobile remains to be investigated.

In the real world, an environment containing long and short delay TCP connections in which the problem of fairness arises needs to be dealt with. This is an issue remains to be investigated.

REFERENCES

[1] H. Balakrishnan, V.N. Padmanabhan, S. Seshan, and R.H. Katz “A Comparison of Mechanisms for Improving TCP Performance over Wireless Links,” IEEE/ACM Transactions on Networking, Volume 5, no. 6 , pp. 756 – 769, Dec. 1997.

[2] M. Allman, D. Glover, and L. Sanchez, “Enhance TCP Over Satellite Channels using Standard Mechanisms,” RFC 2488, January 1999.

[3] M. Allman, S. Dawkins, D. Glover, J. Griner, D. Tran, T. Henderson, J. Heidemann, J. Touch, H. Kruse, S. Ostermann, K. Scott, and J. Semke, “Ongoing TCP Research Related to Satellite,” RFC 2760, Feb. 2000.

[4] J. Postel “Transmission Control Protocol,” STD 7, RFC 793, September 1981.

[5] W. R. Stevens, “TCP Slow Start, Congestion Avoidance, Fast Retransmit, and Fast Recovery Algorithms, ” RFC 2001, January 1997.

[6] S. Floyd and T. Henderson, “The NewReno Modification to TCP’s Fast Recovery Algorithms, ” RFC 2582, April 1999.

[7] M. Mathis, J. Mahdavi , S. Floyd, and A. Romanow, “TCP Selective Acknowledgement Options,” RFC2018, October 1996.

[8] L. S. Brakmo and L. L. Peterson, “TCP Vegas: End to End Congestion Avoidance on a Global Internet,” IEEE Journal on Selected Areas in Communications, Vol. 13, No. 8, pp. 1465 -1480, October 1995.

[9] A. Zanella; G. Procissi, M. Gerla, M.Y. Sanadidi, “TCP westwood: Analytic Model and Performance Evaluation,” IEEE Global Telecommunications Conference, 2001, pp. 1703 –1707.

[10] Network Simulator, http://www-nrg.ee.lbl.gov/ns/.

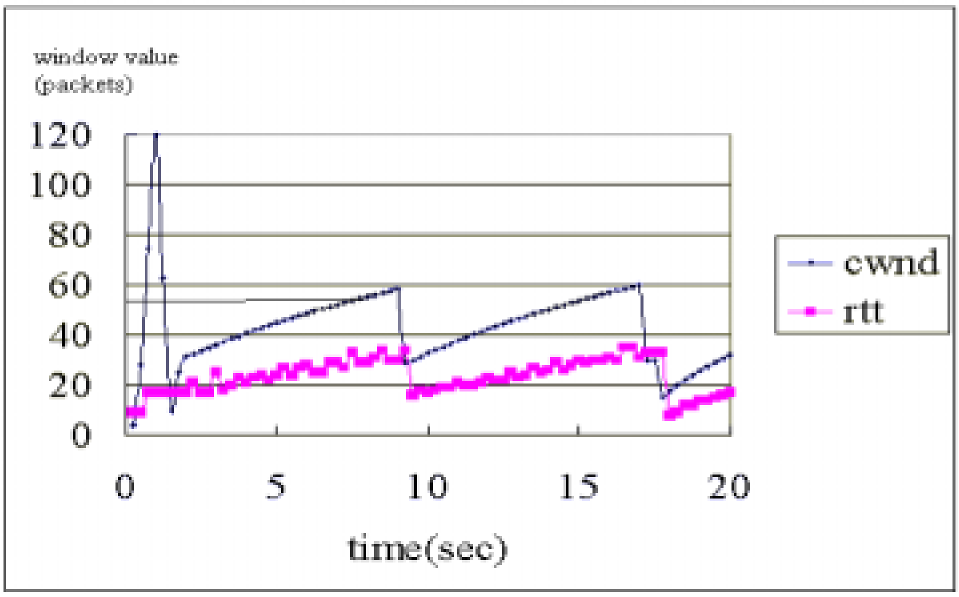

Fig. 1. Relationship between *rtt* and *cwnd*.

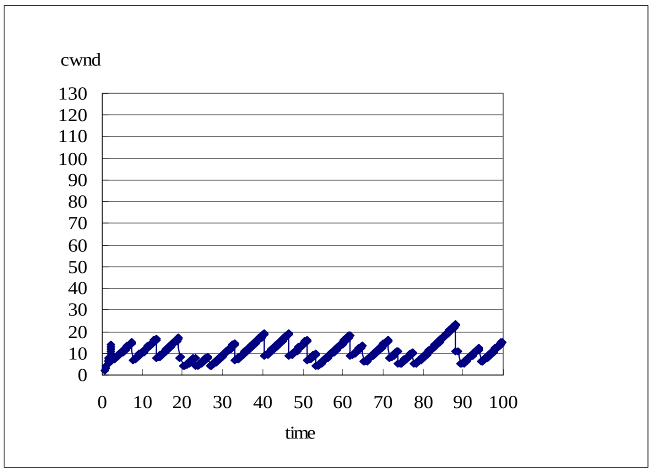

Fig. 2 (a)

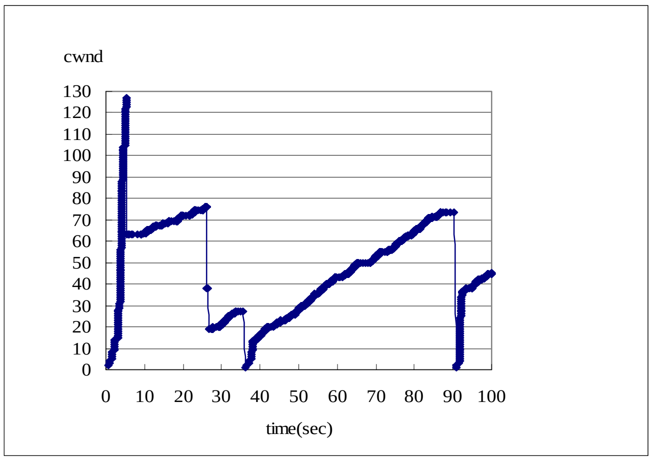

Fig. 2 (b)

Fig. 2. The behavior of *cwnd* before and after applying the threshold scheme for the case of single TCP connection: (a) before and (b) after.

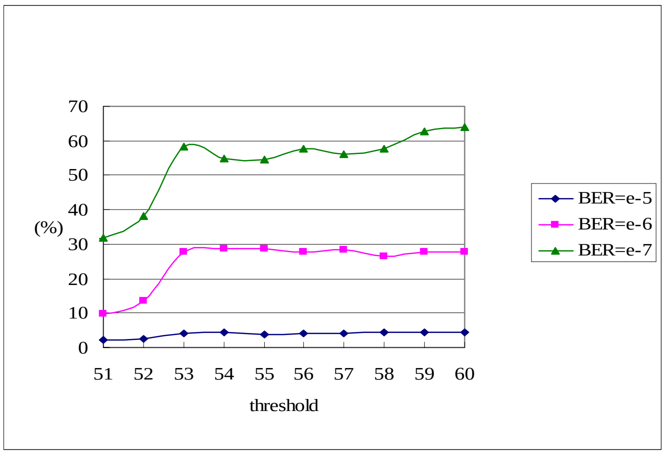

Fig. 3. Throughput performance versus threshold value for single TCP connection under BER=10^{-5},10^{-6}, and 10^{-7}.

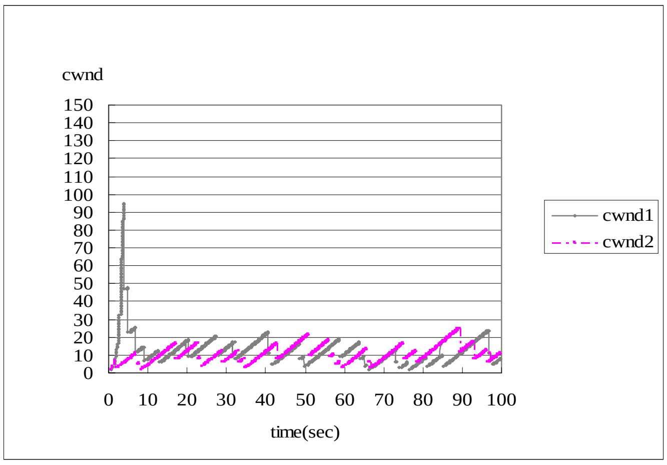

Fig. 4 (a)

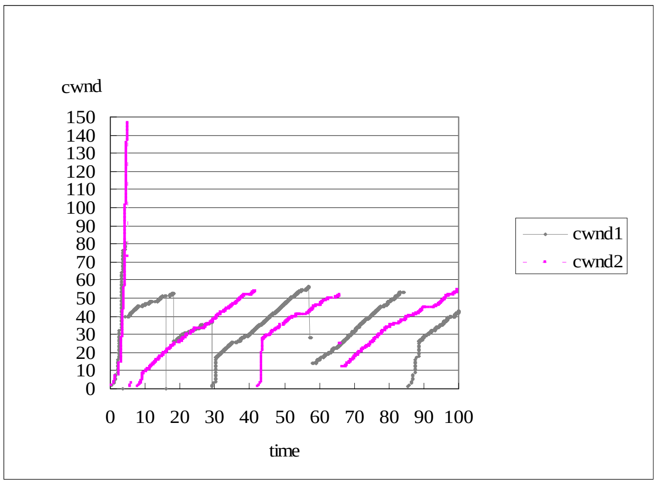

Fig. 4 (b)

Fig. 4. The behavior of *cwnd* before and after the application of the threshold scheme for the case of two connections: (a) before and (b) after.

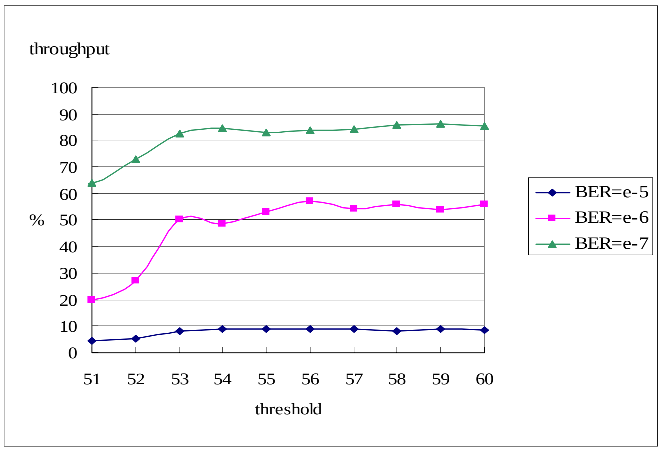

Fig. 5. Throughput performance versus threshold value for two TCP connections under BER=$10^{-5}, 10^{-6}$, and 10^{-7}.

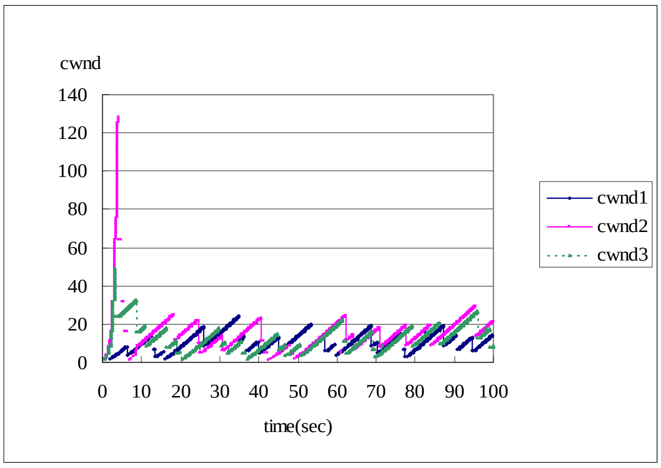

Fig. 6 (a)

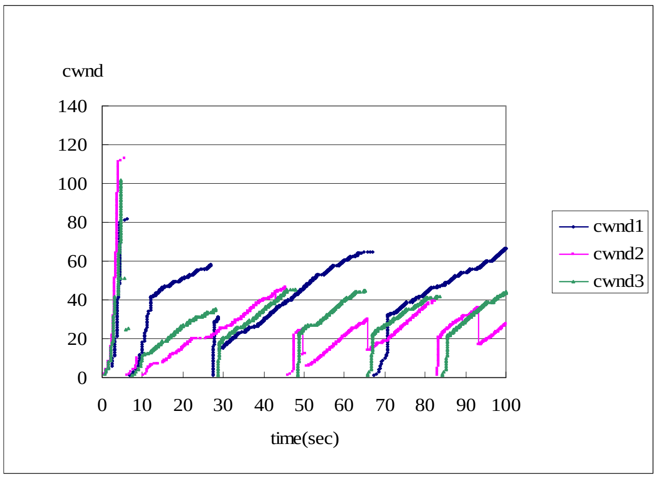

Fig. 6 (b)

Fig. 6. The behavior of *cwnd* before and after applying the threshold scheme for the case of three connections: (a) before and (b) after.

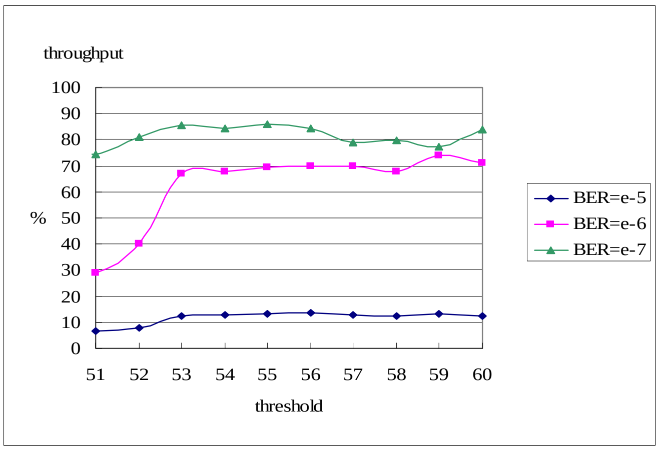

Fig. 7. Throughput performance versus threshold value for three TCP connections under BER=$10^{-5}, 10^{-6}$, and 10^{-7}.

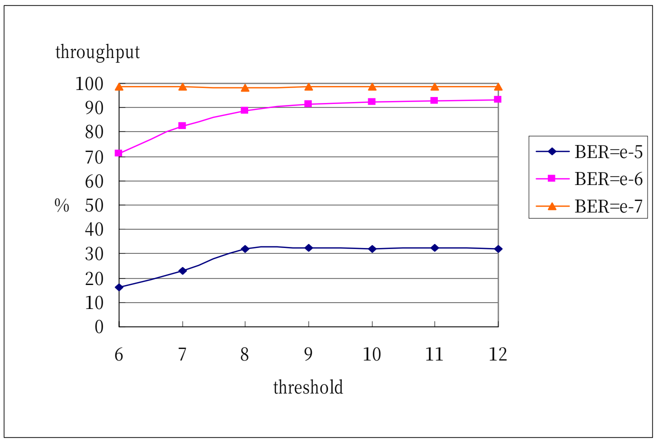

Fig. 8. Throughput performance versus threshold value for a short delay environment.

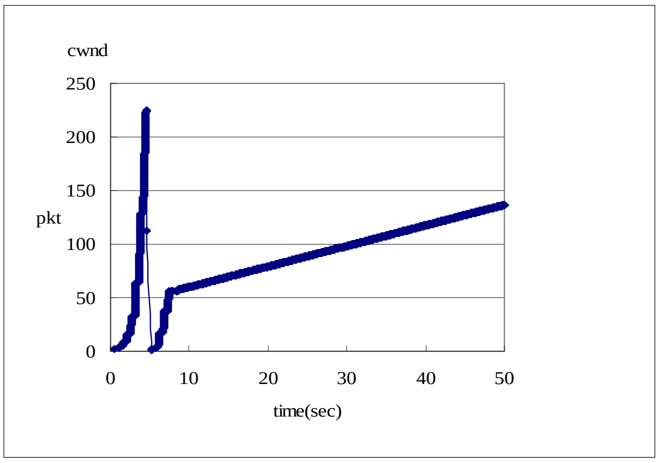

Fig. 9 (a)

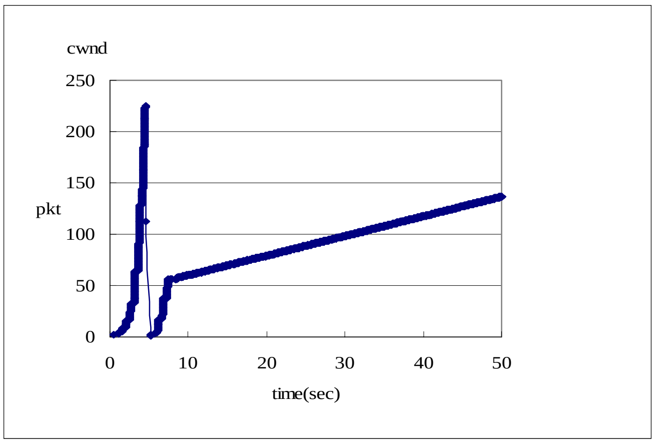

Fig. 9 (b)

Fig. 9. The behavior of *cwnd* before and after the applying the threshold scheme for a wired connection: (a) before and (b) after.