Tuning the TCP Congestion Control Parameters to Optimize Client-Server Systems

Jia Uddin[1], Jong-Myon Kim[1,*], Myeong-Jae Yi[1] and Tae-Gong Kim[2]

[1]*School of Electrical Engineering, University of Ulsan, South Korea*
jia@mail.ulsan.ac.kr, jmkim07@ulsan.ac.kr, ymj@ulsan.ac.kr

[2]*Department of Computer Engineering, Inje University, Busan, South Korea*
sun@inje.ac.kr

Abstract

The demand of high volume data communication over the internet is mounting day by day. In general, the performance of a communication system depends on the loss of data packets. It happens as there are multiple paths exist in wireless networks. For the reliable and secure data communication over Internet Transmission Control Protocol (TCP) is playing a significant role. On the development of TCP, a number of approaches are already designed and tested in the communication system. Generally, TCP congestion parameters are set in sender site in a communication system. In this paper, we investigate the download system performance tuning the TCP congestion parameters at the Ethernet port of receiver side. Based on the experimental study, it is concluded that tuning TCP parameters at receiving side improves the download system performance by reducing packet loss, increasing download speed and maintain stable I/O and time/sequence graphs.

Keywords: *Transmission Control Protocol (TCP), Westwood Congestion Algorithm, Congestion Window, Internet.*

1. Introduction

Wireless communication technology is playing a significant role in access networks [1, 2, 3]. These access networks are usually associated with Internet that consists of wired/wireless backbone networks. The rapid mounting of Internet traffic has introduced new requirements in terms of resources utilization and Quality of Service (QoS) support. Transmission Control Protocol (TCP) is widely used today and likely to be adopted in future networks in order to address these requirements due to its fast, secure and reliable communication capability maintaining high QoS support. It uses a number of sophisticated mechanisms of flow and congestion control algorithms in order to share resources by avoiding congestion [4]. And the first congestion control function was initiated into TCP in 1988 [5]. In wired network a congestion router is indeed the probable ground of packet loss. On the other hand, fading radio channel causes the packet loss in wireless networks [6]. However, in wireless networks or mixed of wired and wireless networks, packet loss causes by wireless-channel characteristics cannot be ignored.

Normally, congestion occurs in the networks when the offered traffic beats the available transmission capacity. The result of congestion control is that resources are mutual between flows for the duration of periods of congestion. TCP flow control algorithm uses a window and end-to-end acknowledgments to provide reliable data transfer across a network [5]. The congestion window size determines the amount of data that can be stupendous at any instant. This is a means of stopping the link between two places from getting congested with high

* Corresponding author.

traffic. The size of this window is calculated by estimating how much congestion there is within the two places. Normally sender maintains the congestion window. When a connection is set up, the congestion window is set to the maximum segment size permissible on that connection. It means that if all segments are received and the acknowledgments reach the sender on time, some constant is added to the window size. The window increasing linearly until a timeout occurs or the receiver reaches to its limit. If a timeout occurs, the window size is halved [7].

A primary function of TCP is to ensure the properly match the transmission rate of the sender and the receiving rate of receiver in a network. TCP provides reliable and efficient data transfer over different communication paths, such as wired, ground radio, and satellite links increasing bandwidth-delay product [8, 9, 10].

It has been observed by our rigorous study that a number of researchers work on TCP congestion algorithms, congestion windows in both wired and wireless networks [1-3, 5-8, 11]. In some papers authors tune these TCP parameters on the sending server side [5-8] and some of them tuned in routers [12, 13]. On the other hand, tuning TCP parameters in receiving side is lack of generality of our observation, it is expected that that approaches also may improve the download performance. Thus, this paper investigates the impact of tuning two commonly used TCP parameters, TCP Westwood congestion control algorithm and TCP congestion window at the receiver sides on downloading a large file from a distance server though Internet.

The rest of this paper is organized as follows. In section II, the experimental model is presented. Section III provides the simulation results and discussions. Section IV concludes the paper.

2. Experimental Environment

Figure 1 illustrates the experimental client-server network model used in this study, where server holds a data file to download and client downloads it by the Internet. In our experiment, we used the domain *"speedtest.bbned.nl"* representing the IP address *62.0.0.0* as the server. This can be retrieved from the URL *http://speedtest.bbned.nl/* and located at Netherland. A file with size 100 MB was used to investigate the downloading performance obtained in the client side. There are a number of protocol analyzers, such as tcpdump, wireshark, or tcpprobe, etc are available. We utilized the WIRESHARK network protocol analyzer in a LINUX computing platform for general experimental results due to its user-friendly Graphical User Interface (GUI) and cross-platform supporting [14].

Figure 1. Block Diagram of the Experimental Client-server Network Model for Tuning the TCP Parameters at the Receiver Side

For experimental evaluation, we installed the WIRESHARK tool in the client machine. First, we downloaded the experimental file from the server to the client through the Internet without tuning the TCP parameters (Westwood algorithm and congestion window size) at the client side. We then tuned the TCP parameters at the

client side and examined the performance of downloading the same experimental file. The *net.ipv4.tcp_rmem* parameter is set which alters the receive buffer size of the TCP protocol. The size of the receive buffer can be set by modifying the net.ipv4.tcp_rmem variable. It takes three different values: *min*, *default* and *max*. The *min* value defines the minimum receive buffer size even when the operating system is under hard memory pressure. The *default* is the default size of the receive buffer, which is used together with the TCP window scaling factor to calculate the actual advertised window. The *max* defines the maximum size of the receive buffer.

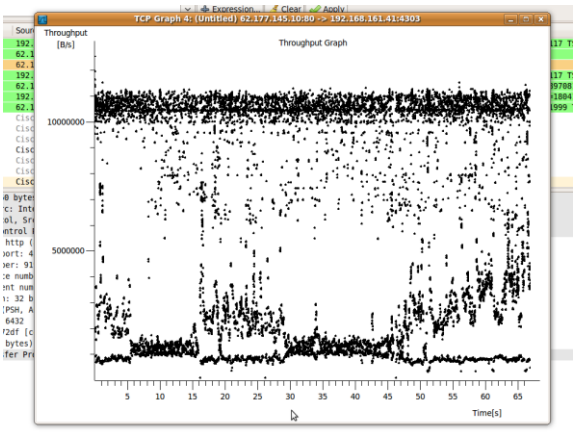

Figure 2(a). Throughput Graph before Tuning the TCP Parameters

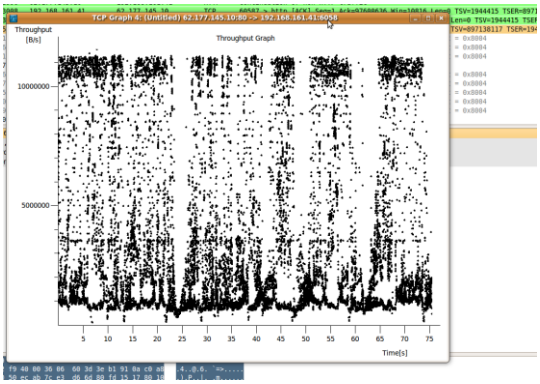

Figure 2(b). Throughput Graph after Tuning the TCP Parameters

3. Experimental Results and Analysis

Experimental results obtained before and after tuning the TCP parameters are illustrated in Figure 2, 3 and 4. Figure 2 illustrates the throughput comparison of before and after the TCP tuning parameters. In Figure 2(a), the throughput rate is irregular due to high packet loss. However, after tuning TCP Westwood algorithm and the congestion window size, we get an overall regular throughput as presented in Figure 2(b) by exhibiting less packet loss. Similar characteristic curves were also found in the time/sequence graph depicted in Figure 3. Figure 4 shows the I/O graph before and after tuning the TCP parameters. Similar to the throughput and time/sequence graphs, I/O graph also exhibited unstable characteristics curve before Tuning TCP parameters.

However, we observed more stable I/O graph as shown in Fig. 4(b) after tuning the TCP parameters.

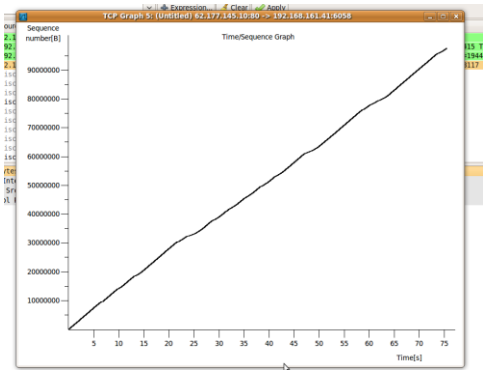

Figure 3(a). Time/sequence Graph before Tuning the TCP Parameters

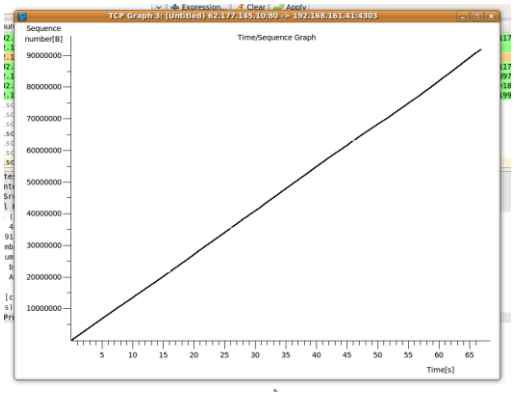

Figure 3(b). Time/sequence Graph: after Tuning the TCP Parameters

We collected a number of statistics before and after tuning the TCP parameters from the WIRESHARK tool. From the results, we observe that the total packet loss is reduced after tuning the TCP parameters, as we receive total 98690 packets before tuning and 106836 packets after tuning the TCP parameters. The throughput was also increased after tuning, as the rate of receiving are1182.592 packets/sec and 1300.367 packets/sec before and after tuning the TCP parameters, respectively.

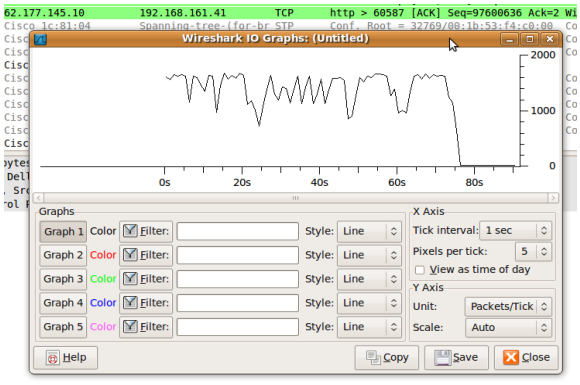

Figure 4(a). I/O Graph: before Tuning the TCP Parameters

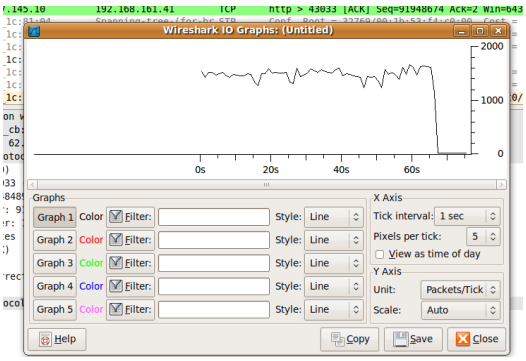

Figure 4 (b). I/O Graph: after Tuning the TCP Parameters

4. Conclusion

In this paper, we tuned two significant TCP parameters, such as, the congestion control algorithm and the congestion window size at receiver side and analyzed the results of a client-server system in order to optimize the performance in a real-world scenario. Experimental results were obtained by executing the tuned TCP implementation in a LINUX computing platform downloading a 100 MB sized file from a server to a client machine through Internet. Experimental results showed that tuning the TCP parameters at the receiver side significantly improve the download performance by reducing packet drop, increasing total captured packets, and average throughput. In addition, tuning the TCP parameters, we get more stable I/O and Time/sequence graphs. In the future, we will investigate the impact of other TCP parameters on system performance.

Acknowledgements

This work was supported by the National Research Foundation of Korea (NRF) Grant Funded by the Korean Government (MEST) (No. 2011-0017941) and by the Network-based Automation Research Center (NARC) at the University of Ulsan.

References

[1] C. C. Cheung, Proceeding of International Conference on Telecommunications, **(2008)** 16-19 June; St. Petersburg, Russia.
[2] M. Gerla, S. S. Lee and G. Pau, Proceeding of IEEE Conference on Global Telecommunications, (**2001**) 25-29 November; Texas, USA.
[3] C. C. Cheung, Proceeding of 6th International Symposium on Wireless and Pervasive Computing, **(2011)** 23-25 February; Hong Kong, China.
[4] L. A. Grieco, S. Mascolo, Proceeding of International Conference on High Speed Networks, **(2002)** 22-24 April; Berlin, Germany.
[5] V. Kononenko, S. Kravchuk and M. Alieksieiev, proceeding of 11th International Conference of the Experience of Designing and Application of CAD Systems in Microelectronics, **(2011)**, 23-25 February; Svalyava, Ukraine.
[6] M. Tekala and R. Szabo, Procedure of International Conference on Computer Systems and Applications, **(2006),** 8-11 March; Dubai/Sharjah, UAE.
[7] Q. Wang and D. Yuan, Proceeding of International Symposium on Performance Evaluation of Computer and Telecommunication Systems, **(2010)**, 11-14 July; Ottawa, Canada.
[8] I. Lengliz, H. Touati, F. Kamoun and M. Y. Sanadidi, Proceeding of 3rd Annual Mediterranean Ad Hoc Networking Workshop, **(2004)**, 27-30 July; Bodrum, Turkey.
[9] C. W. Hsu, T. C. Hou and C.S. Wu, Proceeding of International Conference on Wireless Communications and Networking, 18-21 April; Sydney, Australia.

[10] T. Dunigan, M. Mathis and B. Tierney, Proceeding of the ACM/EEE conference on Supercomputing, **(2002)**, 16 – 22 November, Baltimore, USA.
[11] A. Srivastava, R. J. Friday, M. W. Ritter, W. S. Filippo, Proceeding of International Conference on Vehicular Technology, **(2001)**, 06 -09 May; Rhodes, Greece.
[12] M. Kawarasaki, K. Suzuki, G. Itoh, Proceeding of 4thUKSim European Symposium on Computer Modeling and Simulation, **(2010)**, 17-19 November; Pisa, Italy.
[13] M. Tekala, R. Szabo, Proceeding of International Conference on Computer Systems and Applications, **(2006)**, 8-11 March; Dubai, UAE.
[14] L. Chappell, The Official Wireshark Certified Network Analyst Study Guide. Protocol Analysis Institute, **(2010)**, dba, Chappell University.

Authors

Jia Uddin received the B.Sc. degree in Computer & Communication Engineering from International Islamic University Chittagong (IIUC), Bangladesh, in 2005, the M.Sc. degree in Electrical Engineering emphasis on Telecommunications from Blekinge Institute of Technology (BTH), Sweden, in 2010. Currently, he is pursuing Ph.D. in Computer Engineering in University of Ulsan, South Korea. He is an Assistant Professor in Faculty of Science & Engineering at IIUC, Bangladesh. His research interests include Wireless Networks and MANETs.

Jong-Myon Kim received the BS degree in electrical engineering from the Myongji University, Yongin, Korea, in 1995, the MS degree in electrical and computer engineering from University of Florida, Gainesville, in 2000, and the PhD degree in electrical and computer engineering from the Georgia Institute of Technology, Atlanta, in 2005. He is an assistant professor of Electrical Engineering at University of Ulsan, Korea. His research interests include multimedia specific processor architecture and embedded system.

Myeong-Jae Yi received his BS degree in Computer Science from the Seoul National University, Republic of Korea, in 1987. He also received the MS and PhD degrees in Computer Science from the Seoul National University in 1989 and 1995, respectively. He was a part-time lecturer at the Department of Computer Science of the Seoul National University and the Sookmyung Women's University from 1991 to 1996. He is currently a professor in the School of Electrical Engineering, University of Ulsan, Korea. Prof. Yi is also a deputy-director of NARC (Network based Automation Research Center) at the University of Ulsan and is a member of KIISE and KIPS.

Tae-Gong Kim received his BS degree in Computer Science and Statistics from the Seoul National University, Republic of Korea, in 1983. He also received the MS and PhD degrees in Computer Science and Statistics from the Seoul National University in 1985 and 1994, respectively. He is currently an associate professor in the School of Computer Engineering, University of Inje, Korea. His interests include software engineering, refactoring, code smell detection, AOSD, generative programming.