World Applied Sciences Journal 22 (6): 877-881, 2013
ISSN 1818-4952
© IDOSI Publications, 2013
DOI: 10.5829/idosi.wasj.2013.22.06.524

Demonstration of the Congestion Control Algorithms Implemented by TCP

[1]*Ikram Ud Din*, [2]*Saeed Mahfooz*, [3]*Rahat ullah and* [1]*Muhammad Zeeshan*

[1]Department of Information Technology, University of Haripur, Pakistan
[2]Department of Computer Science, University of Peshawar, Pakistan
[3]Institute of Business and Management Sciences,
Agricultural University, Peshawar, Pakistan

Abstract: In this paper, we have set up a network in which the Transmission Control Protocol (TCP) is its end-to-end transmission protocol. We have analyzed the congestion window size with different methods. The purpose of this study is to examine the congestion control algorithms implemented by TCP. The article presents a number of scenarios to examine these algorithms. We have tested the scenarios for FTP that transfers files of 1MB and 10MB sizes. The simulation results show the performance and effect of the file's size. The performance parameters in this work are: Sent Segment Sequence Number, Received Segment Ack Number and Congestion Window Size. The OPNET Modeler simulation results show a good estimation of the traffic examined in the network.

Key words: TCP · FTP · Congestion window size · Sent segment sequence number · Received segment Ack number

INTRODUCTION

TCP is one of the primary protocols of the transport layer of the TCP/IP suit, also called Internet model. At application level, it makes it possible to handle data coming from or going to the network layer of the model. TCP segments are encapsulated into IP datagrams when they are provided to the IP protocol. TCP reassembles the datagrams when received from the IP protocol. It permits data to be shaped in variable length segments in order to provide them to the network layer [1]. Congestion window increases the probability of contention and packet losses and consequently degrades performance of the TCP [2]. Using the TCP protocol, applications can communicate securely thanks to the TCP protocol's acknowledgements system independently from the lower layers. The TCP protocol makes it possible to ensure reliable data transfer, although it uses the IP protocol, which does not include any monitoring of datagram delivery.

In reality, to ensure mutual receipt of data, the TCP protocol has an acknowledgement system. When a segment is issued, a sequence number is assigned to it. Upon receipt of a segment, the receiver returns a data segment where the Ack flag is set to 1 (in order to indicate that it is an acknowledgement) along with an acknowledgement number equal to the preceding sequence number [3].

TCP normally performs congestion control, flow control and so on, which can control the rate at which applications send traffic. Other protocols degrade application's performance in case of collisions and retransmissions [4].

When an application is sending a large amount of packets over a high-latency network and a high-bandwidth, TCP window sizes must be ample to allow TCP to send various packets in a row without having to wait for acknowledgements. TCP sends data only when the amount of sent but not yet acknowledged packets is less than the minimum of the sender window, receiver window and congestion control window sizes. If any window is less than the "bandwidth-delay product", TCP is obliged to wait for acknowledgments. The bandwidth-delay product is (2* Bandwidth* Propagation Delay). Due to the congestion control, flow control, sender/receiver window size, TCP lessens the rate at which the application can send packets [4].

When TCP examines packet loss, it assumes that the network is congested. This causes TCP to reduce the rate at which applications can send traffic. Retransmissions increase the chance of TCP windowing bottlenecks,

Corresponding Author: Ikram Ud Din, Department of Information Technology, University of Haripur, Pakistan.

because retransmissions cause TCP to reduce the window size. The large Protocol/Congestion delay is caused by a small TCP window and retransmissions. Both of these can reduce the rate at which the FTP server sends messages. The FTP Server sends packets to the client and waits to receive acknowledgement from the client before sending more packets. This is not a good procedure because it delays the transmission process and extends the file transfer. The transmission delay is caused by size of the packet. Lower the transmission speed, larger will be the delay and vice versa [4].

The purpose of this paper is to study the FTP that transfers files of different sizes. The simulations have been done in OPNET Modeler 14.0, because OPNET Modeler is a powerful tool that presents an inclusive development environment for the simulation, measurement and performance testing of communication networks [5-9].

Related Work: Computer networks and Internet are experiencing an unstable growth. Most daily activities and transactions are done through the network in the form of transmitting of data, sharing of files and computing resources. This rising growth of computer network thus has created inflexible congestion problems. In responding to data loss or congestion, TCP uses retransmission and timeout techniques. For that reason, the endpoints generally assume that the timeout occurs because of simple delay in the network. For congestion resolution, the presumed loss datagrams are retransmitted which nevertheless deteriorate the congestion instead of resolving the problem. When data segments are received in out of order fashion, TCP generates an instantaneous duplicate acknowledgment to let the sender know that the sent segments are received out of order.

Congestion avoidance in Reno technique is achieved after fast retransmit sends the missing segment, because the lost segment is a sign of congestion. This algorithm is known as fast recovery [10]. Congestion is a situation of severe delay produced by an excess of datagrams at router [11]. Conversely, congestion control is a distributed technique for sharing network resources among the users [12]. Control is being carried out by TCP [10]. Congestion takes place if the resource demands exceed the capacity and packets are lost as a result of too much queuing in the routers. Congestion increases the path delay and drops the network throughput to zero. A congestion control method helps networks to recover from the congestion state, while a congestion avoidance technique permits a network to function in the area of high throughput and low delay. Such methods keep away a network from entering the congestion state. Simply it is said that congestion control is a recovery procedure while congestion avoidance is prevention technique [13-17]. At the beginning of a connection and after idle times, TCP uses the Slow-Start technique which delays the transport of segments especially if the round-trip time is large, which is unfavorable for voice or video applications [18]. One of the TCP's serious duties is to determine which packets are lost in the network, as a base for control actions (i.e., flow control and packet retransmission). Two TCP's contemporary implementations are timeout and fast retransmit. Detection through timeout is inevitably a time-consuming operation; fast retransmit, while much quicker, is only efficient for a small fraction of packet loss [19-20]. The principal reasons for TCP performance degradation are strife between hidden terminal problems, sharing terminals and packet loss in the MAC layer. In addition, exponential retransmission backoff in the TCP layer, path disconnections arising from mobility and reordering also exacerbate operations. In order to keep the probability of contention loss in the system to a minimum, it is indispensable to bound the TCP congestion window size [2], [21]. A lab is designed in [22] to examine the congestion control algorithms implemented by TCP. The idea of the congestion control procedure is to suffocate how fast TCP sends segments to keep the sender from overloading the network. The concept of TCP congestion control is for each source to verify how much capacity is offered by the network so that it knows how many segments it can safely have in transit.

OPNET SIMULATION

The main focus of this study is to observe the functioning of FTP transferring different size of files. Using OPNET Modeler 14.0, three networks (i.e., Drop_NoFast, Drop_Fast and No_Drop) were simulated in which the TCP parameters i.e., Congestion Window Size, Sent Segment Sequence Number and Received Segment Acknowledgement Number were tested.

Applications Parameters: OPNET Modeler allows choosing different parameters for TCP [23-24] e.g., Congestion Window Size, Received Segment Ack Number and Sent Segment Sequence Number, etc. The Congestion Window Size defines the amount of data in segments that a TCP sender can send before waiting for an acknowledgment to precede. It provides the transport level with a flow control mechanism (end-to-end) and it makes sure that data is accurately delivered. The Sent Segment sequence number is used for the synchronization of the sequence numbers. The acknowledgement number does not relate to the number of the last segment received but it relates to the sequence number of the next segment expected [3].

RESULTS

In this section, three scenarios were tested that compare the performance of FTP transferring files of 1MB and 10MB. As in [12], in the 1st scenario, namely No_Drop, the fast retransmit and fast recovery techniques were disabled. In 2nd scenario, namely Drop_NoFast, the packet discard ratio was assigned a value of 0.05%. And in the 3rd scenario (Drop_Fast), the fast retransmit method was enabled and the fast recovery was assigned Reno attribute which also has 0.05% packet loss.

Fig. 1 and 2 show the Sent Segment Sequence Number of 1MB and 10MB files, respectively. The Received Segment Acknowledgement Number with 1MB and 10MB files are shown in Fig. 3 and 4, respectively. The Congestion Window size with 1MB file is given in Fig. 5; Fig. 6 shows the effect of the Congestion Window size with 10MB file.

Performance Evaluation: We compare our results with the work done in [22]. In the Drop_NoFast scenario, the packet discard ratio was assigned a value of 0.05%. In transferring of 1MB file (Fig. 1), it has the slowest growth in sequence number compared to other two scenarios (i.e., Drop_Fast and No_Drop), while in 10MB file transmission, it has the highest growth in sequence number (Fig. 2) and same is the case with segment Ack number (Fig. 3 and 4).

Similarly, the graph for congestion window size in No_Drop scenario, in which fast retransmit and fast recovery techniques were disabled, is straight for both 1MB and 10MB files. And in the Drop_NoFast scenario, the congestion window size for 1MB file's transmission is pretty low compared to that of 10MB file's transmission (Fig. 5 and 6).

In each of the following figures, the X-axis shows the amount of simulated time, while theY-axis shows the value of Sent Segment Sequence Number in Fig. 1 and 2, the Received Segment Acknowledgement Number in Fig. 3 and 4 and Congestion window size (in bytes) in Fig. 5 and 6, respectively. All the networks were simulated for 7 minutes.

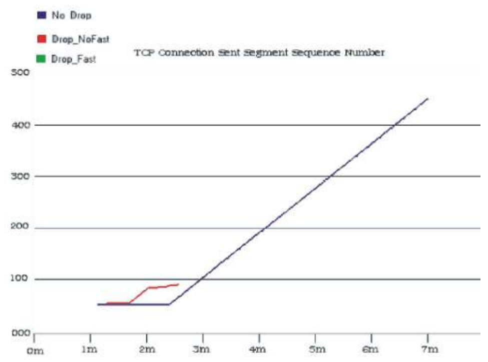

Fig. 1: Sent Segment Sequence Number with 1MB file

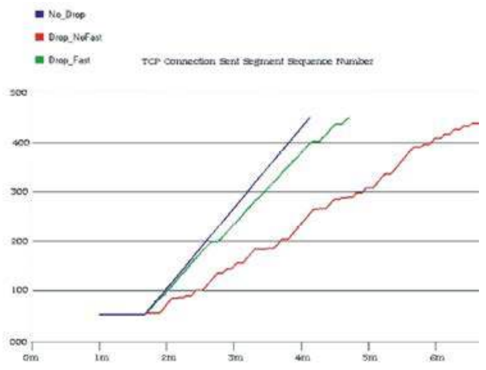

Fig. 2: Sent Segment Sequence Number with 10MB file

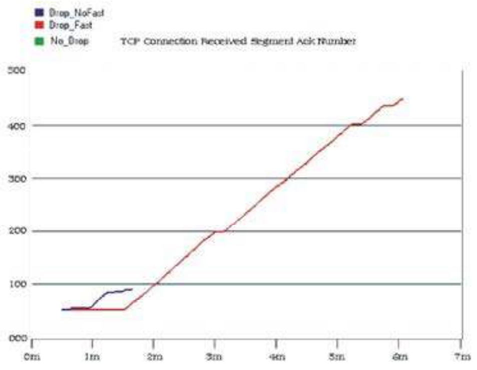

Fig. 3: Received Segment Ack Number with 1MB file

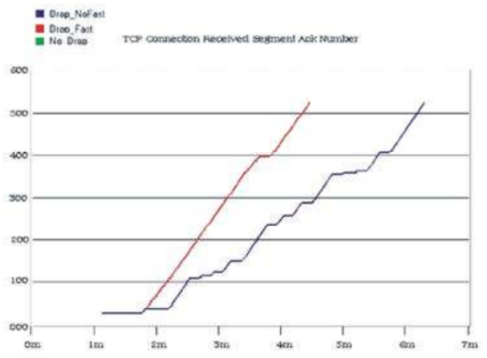

Fig. 4: Received Segment Ack Number with 10MB file

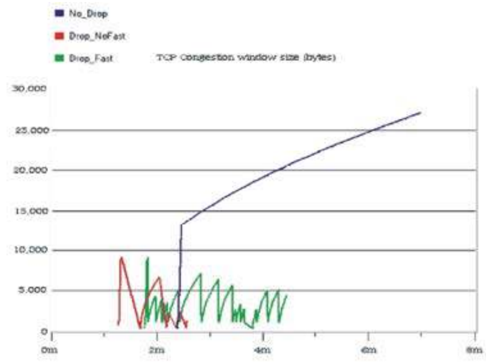

Fig. 5: Congestion Window Size with 1MB file

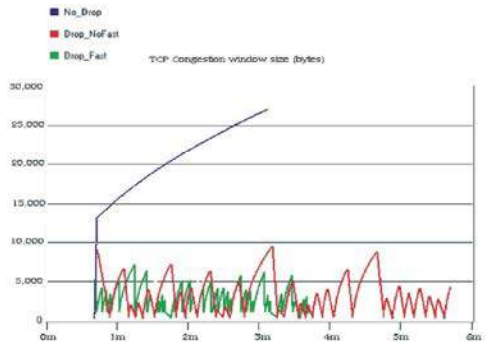

Fig. 6: Congestion Window Size with 10MB file

Concluding Remarks: This article indoctrinates the congestion control algorithms implemented by TCP. The study shows a number of scenarios to examine the effect of these algorithms. The performance was tested using FTP transferring files of different sizes. The performance is examined with 1MB and 10MB file sizes. A good estimation of the traffic investigated in the network has been shown using OPNET simulations.

REFERENCES

1. http://en.kioskea.net/contents/internet/tcp.php3.
2. Huh, I., J.Y. Lee and B.C. Kim, 2006. Decision of Maximum Congestion Window size for TCP performance Improvement by Bandwidth and RTT measurement in Wireless Multi-hop Networks. International Journal of Information Processing Systems.
3. Thai, B., R. Wan, A. Seneviratne and T. Rakotoarivelo, 2003. Integrated personal mobility architecture: a complete personal mobility solution. Mobile Networks and Applications, 8(1): 27-36.
4. Cisco Systems, Inc, Cisco Application Analysis Solution ACE Tutorials and Examples, Vol. Release 1.0, Corporate Headquarters, 170 West Tasman Drive: San Jose, CA 95134-1706 USA.
5. Nazri, M.I. and A.M. Zin, 2008. Emulation Network Analyzer Development for Campus Environmetn and Comparison between OPNET Application and Hardware Network Analyzer. European Journal of Scientific Research, EJSR, 24(2): 270-291.
6. Nazri, M.I. and A.M. Zin, 2009. Evaluation of Software Network Analyzer Prototyping Using Qualitative Approach, EJSR, 26(3): 170-182.
7. Sood, A., 2007. Network Design by Using OPNET™ IT GURU Academic Edition Software", Rivier College Academic Journal, 3(1): 1-8.
8. Nazri, M.I. and A.M. Zin, 0000. A Simulation Model Design and Evaluation for Aggregate Traffic Over Local Area Networks", Int. J. of Adv. Comp Eng (IJACE).
9. Nazri, M.I. and A.M. Zin, 2008. Measurement and Characterization of Network Traffic Utilization between Real Network and Simulation Modeling in Heterogeneous Environment, Int. J. of Comp Sc and Net Sec [IJCSN], 8(3): 326-337.
10. Onwutalobi, A.C., 2008. TCP Congestion Control, Helsinki Finland, Codewit Global Network, pp: 1-6.
11. Commer, D.E., 2006. Internetworking with TCP/IP, Principles, protocols and Architecture, Vol. 1, 4th Editiion.
12. Jacobson, V., 1998. Congestion avoidance and control, ACM SIGCOMM Computer Communication Review, 18(4): 314-329.
13. Thai, B., R. Wan, A. Seneviratne and T. Rakotoarivelo, 2000. Integrated personal mobility architecture: a complete personal mobility solution. Mobile Networks and Applications, 8(1): 27-36.
14. Jain, R. and K.K. Ramakrishnan, 1998. Congestion Avoidance in Computer Networks with a Connectionless network Layer Part 1: Concepts, Goals and Methodology, Proceedings of the Computer Networking Symposium; IEEE, Washington, DC, pp: 134-143.
15. Pang, Q., S.C. Liew, C.P. Fu, W. Wang and V.O. Li, 2003. Performance study of TCP Veno over WLAN and RED router. In IEEE Global Telecommunications Conference, 2003. GLOBECOM'03, 6: 3237-3241.
16. Han, H., S. Shakkottai, C.V. Hollot, R. Srikant and D. Towsley, 2006. Multi-path TCP: a joint congestion control and routing scheme to exploit path diversity in the internet. IEEE/ACM Transactions on Networking (TON), 14(6): 1260-1271.

17. Analoui, M. and S. Jamali, 2006. A conceptual framework for bio-inspired congestion control in communication networks. In Proceedings of the 1st international conference on Bio inspired models of network, information and computing systems, ACM, pp: 38.
18. Scharf, M., 2009. Performance evaluation of fast startup congestion control schemes. In networking, Springer Berlin Heidelberg, pp: 716-727.
19. Nahur, F. and M. Crovella, 2005. Bayesian Packet Loss Detection for TCP, INFOCOM, pp: 1-12.
20. Jiang, X., F. Yang and H. Zou, 2003. A novel architecture to customer service management for the NGN, ICCT2003. Int. Con, pp: 123-126.
21. Ebrahimi-Taghizadeh, S., A. Helmey and S. Gupta, 2005. A Systematic Simulation-based Study of Adverse Impact of Short-lived TCP Flows on Long-lived TCP Flows, IEEE/INFOCOM 2005.
22. Guo, J., W. Xiang and S. Wang, 2007. Reinforce networking theory with opnet simulation. Journal of Information Technology Education, 6: 215-226.
23. Bawazir, S.A. and S.H. Al-Sharaeh, 2006. Performance of Infrastructure Mode Wireless LAN Access Network Based on OPNET^{TM} Simulator.
24. Nazri, M.I. and A.M. Zin, 2008. Development of Simulation Model in Heterogeneous network Environment: Comparing the Accuracy of Simulation Model for Data Transfers Measurement over Wide Area Network", Info. Tech. J. (Assian Net for Sc. Info), 6(7): 897-903.