A Comparison and Analysis of Congestion Window for HS-TCP, Full-TCP, and TCP-Linux in Long Term Evolution System Model

Ghassan A. Abed Mahamod Ismail Kasmiran Jumari
Department of Electrical, Electronic & Systems Engineering
Faculty of Engineering & Built Environment Universiti Kebangsaan Malaysia, UKM
43600 UKM Bangi Selangor Darul Ehsan, Malaysia
Email: {ghassan | mahamod | kbj}@eng.ukm.my

Abstract— **TCP performance over high bandwidth network still represents the major challenge, since the large numbers of data flows through bottleneck, TCP forces a high packet loss rate and that causing delays that users are likely to notice. Congestion window (cwnd) is maintained, which indicates the number of Transmission Control Protocol (TCP) segments that the existing connection is capable of holding safely. In high bandwidth networks, such as 4th Generation (4G) Long Term Evolution (LTE) systems, we expect a high rate of traffics, and then each TCP variants perform a different behavior of cwnd. In this study, three TCP's variants: TCP-Linux, FullTCP, and High Speed TCP (HSTCP), observed and the cwnd of each variant analyzed and compared with other two variants, over a model of high bandwidth network topology using network simulator (NS-2). All these TCP's tested under high rate of data transferred through a bottleneck. We found that each variant can provide a good cwnd size performance and stability but the differences was in cwnd phases such as slow start threshold (ssthresh), congestion avoidance, slow-start, and maximum congestion points.**

Keywords– FullTCP; HSTCP; TCP-Linux; cwnd;LTE; NS-2

I. INTRODUCTION

The concepts of the mechanism of TCP congestion control depending dynamically on organize the size of window according to the network path state. TCP has critical design and the congestion control permits to TCP to adjust the network end-to-end connection by control the data rate according to the available bandwidth. However, TCP give poor performance in high bandwidth channels due to the slow response with large congestion window [1]. If TCP sender not receives Acknowledgment (ACK) for TCP receiver after transmission segment, the connection goes to timeout where the first phase of start transmission called slow-start. In slow start phase the packets sending and the window increasing exponentially in parallel with increasing the round trip time (RTT).

The window increment not continues without limits, because that will cause packet losses when the packets injected in network connection larger than the available space in network bottleneck and receiver buffers. To overcome this defect of TCP slow start, the researchers have proposed many improved methods. Fig. 1 shows the phases of standard cwnd. The basic congestion control mechanism supported by TCP include following phases, Slow-Start, Congestion Avoidance, Fast Retransmit, and Fast Recovery.

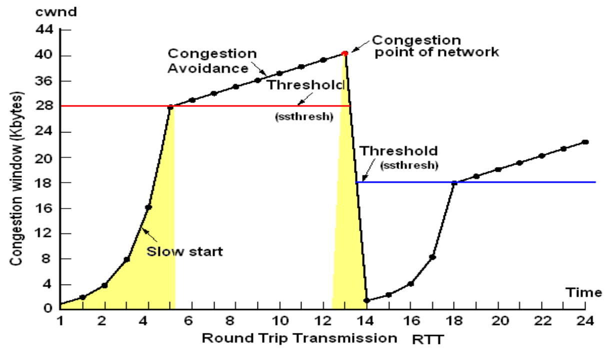

Fig. 1 Slow-Start and Congestion Avoidance

TCP congestion control includes two main phases, slow start phase and congestion avoidance phase, and these two phases used by TCPO sender to adjust the amount of data flow through network. TCP sender starts by sending packets in form of segment and waiting to receive the ACK to proving the sent packets if it's received or lost. If segment acknowledged, the sender sends two segments instead of one. After other ACK, the increment in congestion window duplicates to four. Actually, there is no exponential increment in congestion window due to sometimes TCP sender receives delayed ACK and the sender transmits two segments every ACK. The possibility of congestion to occur in network connections, when the network resources supply a large amount of data exceed the available bandwidth and these problem represents a big challenge in modern networks due to its cause up normal transferring in packets, like delayed packets and lost packets, where that will force the sender to resend these packets and the performance of network will suffering from degradation [2].

Each transmission period starts with slow start phase and the segment transfer from TCP sender to its peer until the congestion window become equal to the present threshold (ssthresh) of the network pipeline. At this point, the network enters to congestion avoidance period and the window starts increasing linearly. In other words, the window enlarges by one segment per RTT. [3]. There are two other phases which support some other TCP variant, these are fast retransmit and fast retransmit. Fast retransmit mechanism allows to TCP sender to retransmit a segment after specific timeout period and no need to wait ACK from receiver.

This mechanism considers the packet lost if ACK delay [4]. In other hand, when fast retransmit detects three duplicate ACKs, the fast recovery process starts from congestion avoidance region and use ACKs in the pipe to transmitting packets. This article provides comparison and investigation to the behavior of congestion window for three TCP versions; HSTCP, FullTCP, and TCP-Linux with analyzing the congestion window clocking and timing when these three TCP's experimented over a traffic model of LTE network. This paper organized as follows. Section II presents an overview of HSTCP, FullTCP, and TCP-Linux mechanisms and the specifications of LTE systems. Section III describes the modeling and simulation of the proposed topology. Section IV illustrates and analyzes the results of cwnd for TCP's. Section V is dedicated to the conclusion.

II. OVERVIEW OF TCP'S AND LTE SYSTEMS

HSTCP introduce a new mechanism based on enhancement the time of loss recovery of classic TCP by variation the standard algorithm of TCP's Additive Increase Multiplicative Decrease (AIMD). The developed mechanism of HSTCP works effectively with large congestion windows. That mean, when the congestion window less than the threshold, HSTCP uses AIMD algorithm to control congestion in network connections, but if congestion window high its use the algorithm of high speed AIMD [5]. Actually, designing of HSTCP is depending on the response variation in environments of the low congestion occurring in bottleneck and to the response activation of standard TCP in environments of packet loss rates [6].

FullTCP can only apply with the congestion control of TCP Reno. However, FullTCP must be provided with fully algorithms of congestion control such as Tahoe, Vegas, and Sack [7]. In TCP-Linux, it's introduced three main characteristics to developing the performance of TCP-Linux. These three elements are [8]:

- Standard interface for congestion control algorithms.
- Redesigned loss detection module.
- New event queue scheduler that increases the test speed.

As we mentioned before, this study and the experimental test depended on analysis the behavior of three TCP's over LTE network topology. LTE is the 3GPP specification for the fourth generation of mobile networks and referred to as Evolved UMTS Terrestrial Radio Access (E-UTRA). LTE systems aim to provide a step forward in wireless and mobile systems by providing high speed data transmission to users by providing low latency links an improved spectral efficiency. Comparing with Wideband Code Division Multiple Access (WCDMA) and HSPA, LTE provides higher rate of data with downlink reaches more than 100Mbps uplink of 50 Mbps. In fact, LTE systems released wide achievements in telecommunication networks by providing lower user costs than other systems, better spectrum efficiency, and a very small latency [9, 10].

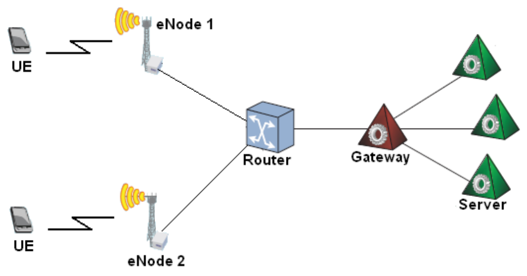

Fig. 2 Basic architecture of LTE

In March 2009, the standardization of LTE firstly published as a part of specification of 3GPP Release 8. In spite of LTE represent the evaluated generation of 3G, but LTE does not offer new approaches to improving the spectral efficiency (the ratio of bps per Hz)if comparing it with 3G. However, LTE enhances the performance of the system used over it by providing larger bandwidth when the spectrum is available [11]. Fig. 2 illustrates the basic architecture of LTE systems, where eNode acts as a base station, and serving gateway, with server as a FTP source.

LTE systems is under developments now, and the TCP protocol is the most widely used protocol for wired and wireless systems, although TCP was not originally designed for real time applications and not for wireless networks then, we need to develop a new TCP mechanisms, or at least choose a suitable TCP variant for each new network, to be more efficient and more reliability with this network.

III. PROPOSED TOPOLOGY AND SIMULATION

As we explained, we used NS-2 network simulator to modulate the proposed topology of LTE and also the experiments to monitor the behavior of TCP versions used in these article performed according to NS-2, where NS-2 is a better simulation deals with TCP's protocols. NS-2 not just a simulator, but it's a discrete event aimed to support the research and studies that deals with communications and networks analysis. Also, NS-2 provides environments to simulate and modeling multicast protocols, networks traffics, handovers, and other networks resources and conditions for wireless and wired channels [12]. In our research, we used NS-2 version 2.32 , and this version installed over Windows XP or using Cygwin, where Cygwin provides a Linux-like environment under Windows, because NS-2 already supported a Linux operating system only, then we need to get a virtual environment.

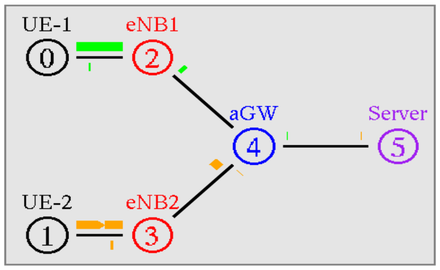

Fig. 3 NS-2 Network Animator

TABLE I. SIMULATION PARAMETERS

Parameter	Value
TCP protocols	HSTCP, FullTCP,TCP-Linux
Propagation Delay of all links	2 msec
Bandwidth of aGW link	10 Mbps
Bandwidth of Server link	100 Mbps
Bandwidth of UE link	1Mbps
Packet Size	1500 Bytes
Window size	100 Kbytes
Simulation Time	50 sec

As shown in Fig. 3, we proposed six nodes elements. Two nodes as mobile nodes (UE's), two base stations (eNB's), one gateway router, and one FTP server. The proposed LTE model is assumed a bottleneck connection with bandwidth of 100Mbps and delay of 2 msec.

The parameters of modeling and simulation presented in Table 1, where all links are set to unified propagation delay of 2 msec. The maximum packet size of TCP was set to 1500 Bytes, with minimum window size of 100 Kbytes. NS-2 representation of links depend on bandwidth of the link, type of link (duplex or simplex) and the propagation delay of packets over this link. In LTE networks simulation, we have many bandwidths and two types of links but only one delay for all links.

As we mentioned before, NS-2 provides the agents of most of TCP's including the three TCP's studied here. TCP-Linux not provided directly in NS-2.32, but it need to install it by using a compatible patch and process some modification in NS-2 environments. In TCP-Linux, TCP agent name change from (Agent/TCP/Sack1) to (Agent/TCP/Linux), and the TCP destination set to Sack1.

However, to implementing TCP-Linux in TCL script (TCL is the language used in programming inside NS-2), we must choose one of two available agents form, either using the agent in the form of: (Agent/TCPSink/Sack1) or using the agent in form: (Agent/TCPSink/DelAck). As shown in script below:

```
/* TCP-Linux agent */

set tcp [new Agent/TCP/Linux]
set tcpsink [new Agent/TCPSink]
```

In HSTCP, the structure of agents differs from that used in TCP-Linux. The implementation of HSTCP is controlled by the manner to adjusting the congestion control. This TCP variant depends on modifying the response function and only determines the effect when the congestion window is high.

In NS-2, implementing of HSTCP to test the congestion window behavior or for any other application also needs to use TCP source and destination with different agents.

When HSTCP is applied in NS-2, must use the form shown below:

```
/* HSTCP agent */

set tcp [new Agent/TCP/Sack1]
set tcpsink [new Agent/TCPSink/Sack1]
```

The implementation of FullTCP in NS-2 is still under development because of its adding newly to the TCP suite of the agents which supported by NS-2 configuration. FullTCP agent is differ from the other agents used under NS-2 due to Full TCP agent is classified as bidirectional protocol. That means, FullTCP is a two way TCP where its provide bidirectional data transmission. In addition, FullTCP support a new data structure, where it uses the bytes instead of packets in representing the sequence numbers. FullTCP can implementing in TCL script in the form shown below:

```
/* FULLTCP agent */

set tcp [new Agent/TCP/FullTCP]
set tcpsink [new Agent/TCP/FullTCP]
```

IV. RESULTS ANALYSIS AND DISCUSSIONS

The results obtained from experiments divided into four parts, three parts to analyze the slow-start and congestion avoidance phases of each TCP, and the last compares the full period of cwnd of three TCP's in one experiment.

As shown in Fig. 4, the slow-start phase of TCP-Linux shows that it reaches to congestion within 1.25 sec, and the maximum window reach to 100 Kbyte. It have a congestion avoidance after 1.5 sec and then it start a new slow-start phase with new ssthresh at 1.5 sec to a new congestion point at 50 Kbytes. That's mean ssthresh=cwnd/2, since TCP-Linux implementation evaluates the initial ssthresh from the cwnd size of the previous TCP connection.

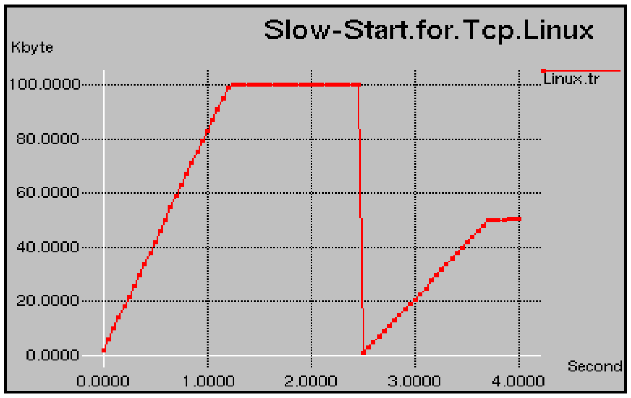

Fig. 4 Slow-Start phase of TCP-Linux

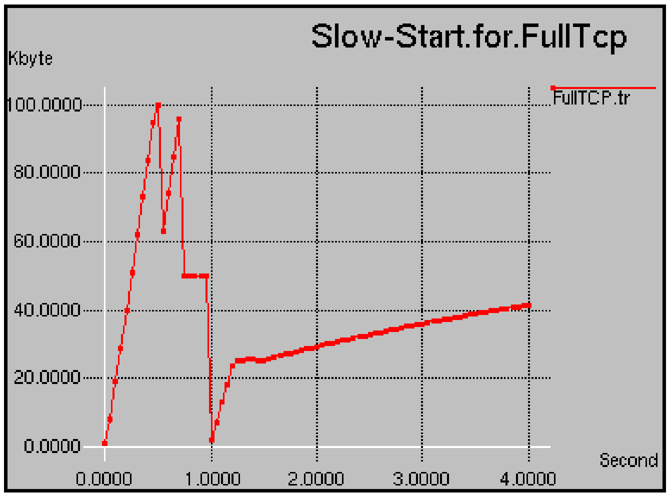

Fig. 5 Slow-Start phase of FullTCP

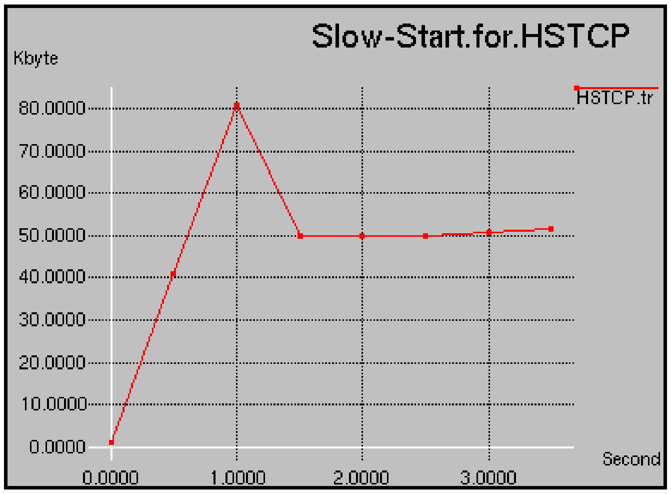

Fig. 6 Slow-Start phase of HSTCP

In Fig. 5, slow-start phase of FullTCP needs only to 0.5 sec to reach window of 100 Kbytes, and no clearance congestion avoidance here, but we can note some fast recovery with some packets lost. The new slow-start phase starts after 1 sec to reach a new congestion point at 60 Kbytes. Slow-start of FullTCP, starts unstable, but after first 1 sec did well and provided a stable behavior of cwnd.

Fig. 6 illustrates the slow-start of HSTCP, where it reached within 1 sec, to 80 Kbytes as ssthresh, without congestion avoidance. So it needs long slow-start time for new session, reached to 10 seconds. Whatever, it provided a regular cwnd but it had a small window compared with other two variants.

The three TCP's can provide a good and regular cwnd, in spite of the differences is slow-start phase behavior. As shown in Fig. 7, the maximum slow-start founded in TCP-Linux where it exceeded 1 sec, with ssthresh of 50 Kbytes with small congestion avoidance. But cwnd of FullTCP performed a better ssthresh of 60Kbytes. A zooming graph of one complete period of cwnd of three TCP's, shown in Fig. 8, where the maximum congestion point of network got with FullTCP, in other side, the minimum point appeared with HSTCP. The maximum slow-start phase occurred with TCP-Linux with maximum congestion avoidance of 1 Kbyte, where, lager congestion avoidance increment will improve performance.

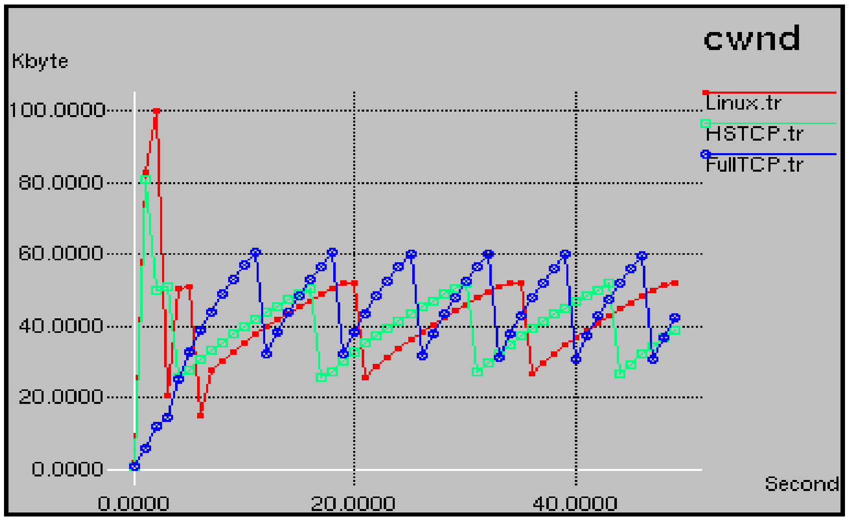

Fig. 7 cwnd of HSTCP, TCP-Linux, and FullTCP

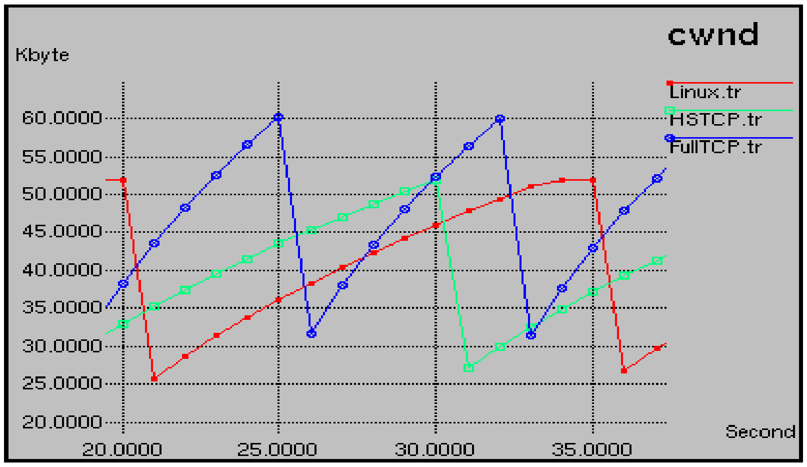

Fig. 8 Enlarged plot of Slow-Start and Congestion Avoidance

When cwnd can reach maximum value, the losses in packets may occur and performance decrease. In the plot here at Fig. 8, the cwnd can be larger than the bandwidth delay product, we only get 60 Kbytes in FullTCP session as a throughput. The losses occur when the window size becomes larger than the point of congestion window of the network, or we can say the packets lost because the network has not enough capacity for storing all the sent packets, so, if the window size is too large, some packets are dropped and performance decreases, since when a loss occur, the cwnd is divide by two.

V. CONCLUSION

This paper provides an investigation to the behavior of congestion window for different TCP versions, HSTCP, FullTCP, and TCP-Linux, over a model of LTE network using NS-2 network simulator. The aim of this work was to establish a bound on the performance of cwnd of these TCP's in high bandwidth networks. The investigation has been carried out by modeling all links and traffics of LTE networks with standard parameters, such as bandwidth and delay, to monitor the cwnd, slow-start, and congestion avoidance of these variants. FullTCP provided a well performance and a big widow size compared with other two variants, so HSTCP had a longest slow-start period with a congestion level near to that produced by TCP-Linux. Generally, the implementation of three TCP's shows the differences in behavior of congestion window phases in spite of it is a little but when used over high bandwidth or any other high speed networks, we can see a large rate of packets losses, because of incompatibility between the congestion window parameters and the link capacity.

ACKNOWLEDGMENT

This study is sponsored by Universiti Kebangsaan Malaysia (UKM) and MIMOS BHD. through the research grant PKT 3/2008.

REFERENCES

[1] Y. Chan, C. Lin, and C. Ho, "Quick Vegas: Improving Performance of TCP Vegas for High Bandwidth-Delay Product Networks," in proceeding of IEICE Transactions 91-B(4), 2008.

[2] I. A. Hassan, "Predicting TCP congestion through active and passive measurements," Master thesis, University of Oslo, May 2005.

[3] X. Chen, H. Zahi, J. Wang, and Y. Fang, "A Survey on Improving TCP Performance over Wireless Networks," Network Theory and Applications, Vol.16, July 2006.

[4] J. Antila, "TCP Performance Simulations Using Ns2", Special Study, Networking Laboratory, Helsinki University of Technology, 2002.

[5] A. Nawaz, X. Hong, J. Wu, and J. Lin, "Impact of QoS Differentiation on Congestion Window Dynamics and throughput of High Speed TCP Protocols on LOBS Testbed ," in proceeding of 6th International Conference on High_Capacity Opticak Networks (HONET), 1999.

[6] E. Souza, and D.A. Agarwal, "HighSpeed TCP study: Characteristics and deployment issues," LBNL, Technical Report,LBNL-.53215, 2003.

[7] K. Fall, S. Floyd, and T. Henderson, "Ns simulator tests for reno fulltcp," U.S. Department of Energy,1997.

[8] D. X. Wei and P. Cao, "NS-2 TCP-Linux: An NS-2 TCP Implementation with Congestion Control Algorithms from Linux," Workshop on NS-2, October 2006.

[9] T. Mahmoodi," Transport Layer Performance Enhancements over Wireless Networks," University of London, August 2009.

[10] Y. Tan,"Active Queue Management for LTE Uplink in eNodeB," Master Thesis, Helsinki University of Technonlogy, February 2009.

[11] A. Racz, A. Temesvary, and N. Reider, "Handover Performance in 3GPP Long Term Evolution (LTE) Systems," Mobile and Wireless Communications Summit, 2007. 16th IST, pp. 1–5, July 2007.

[12] The Network Simulator - NS-2. In: Information Sciences Institute, University of Southern California, USA, 2006.