http://ansinet.com/itj

ITJ

ISSN 1812-5638

INFORMATION TECHNOLOGY JOURNAL

ANSInet
Asian Network for Scientific Information
308 Lasani Town, Sargodha Road, Faisalabad - Pakistan

Information Technology Journal 10 (3): 663-668, 2011
ISSN 1812-5638 / DOI: 10.3923/itj.2011.663.668

Behavior of *cwnd* for TCP Source Variants over Parameters of LTE Networks

Ghassan A. Abed, Mahamod Ismail and Kasmiran Jumari
Faculty of Engineering and Built Environment, National University of Malaysia,
UKM, 43600 Bangi, Selangor, Malaysia

Abstract: This study presents results from an experimental study of TCP source variants, Reno, Tahoe, Newreno, Sack, Fack and Vegas by monitoring the behavior of *cwnd* (Congestion Window) of each variant with using Long Term Evolution (LTE) network parameters such as, bandwidth, propagation delay and packet size. All simulations performed using NS-2 network simulator. In this study, simulation results showing the *cwnd* behavior TCP variants under different network conditions with a standard topology construct of six terminal nodes connected to main node through different bandwidths and different delays and this main node connected to other with constant bandwidth and delay. After getting the trace for each scenario we can analyze the behavior of six TCP variants with six different parameters. We chose LTE networks because it's provide a new specifications for a new radio-access technology geared to higher data rates, low latency and greater spectral efficiency and technology supporting flexible bandwidth deployments.

Key words: LTE, TCP variants, *cwnd*, NS-2

INTRODUCTION

The LTE network architecture is designed with the goal of supporting packet-switched traffic with seamless mobility, quality of service (QoS) and minimal latency. Fundamentally, it is a flattened architecture that enables simplified network design while still supporting seamless mobility and advanced QoS mechanisms. This is a major change relative to traditional wireless networks with many more network nodes using hierarchical network architecture (Khan, 2009).

The reason behind the variations of TCP is that each type possesses some special criteria. Such as the base TCP has become known as TCP Tahoe. TCP Reno adds one new mechanism called Fast Recovery to TCP Tahoe. TCP Newreno uses the newest retransmission mechanism of TCP Reno. The use of Sacks permits the receiver to specify several additional data packets that have been received out-of-order within one dupack, instead of only the last in order packet received. TCP Vegas proposes its own unique retransmission and congestion control strategies. TCP Fack is Reno TCP with forward acknowledgment (Islam *et al*., 2009).

In this study, we proposed a network simulation topology used a six terminal nodes connected directly to other node and each link have a different bandwidth and propagation delay and this represents a main part of simple bottleneck topology with an end node with a constant link parameters of 100 Mbps as a bandwidth and 5 msec as a delay, but all simulation is used a one TCP variants in all links and use a same packet size.

Originally, flow control of TCP was governed simply by the maximum allowed window size, advertised by the receiver and the policy that allowed the sender to send new packets only after receiving the acknowledgment for the previous packet. At the very beginning Tahoe was introduced with three congestion control algorithms: slow start, congestion avoidance and fast retransmit. Later on another additional algorithm called fast recovery was provided by Reno. Besides the receiver's advertisement window (*awnd*), congestion control of TCP introduced two new variables for the connection: the congestion window (cwnd) and the slow start threshold (*ssthresh)*. The window size of the sender, w, was defined as w = min (*cwnd*, *awnd*), instead of being equal to *awnd,* the congestion window can be thought of as a counterpart to advertised window (Rahman *et al.,* 2008). The Slow-Start, actually increases exponentially the size of the congestion window. It is used by a TCP entity at the beginning of a transmission or after detecting a packet loss. The purpose of Slow-Start is to fill as soon as possible a transmission channel.

As explained in Fig. 1, when the congestion window has reached the threshold value, the Congestion Avoidance algorithm is employed.

The timeout signals the loss of the packet. This leads to the retransmission of that packet and halving of the Slow-Start threshold. The congestion window is also set

Corresponding Author: Ghassan A. Abed, Faculty of Engineering and Built Environment, National University of Malaysia, UKM, 43600 Bangi, Selangor, Malaysia

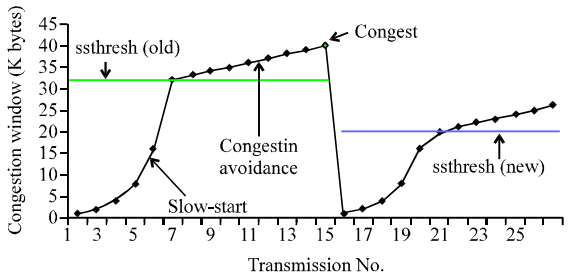

Fig. 1: Slow-start and congestion avoidance

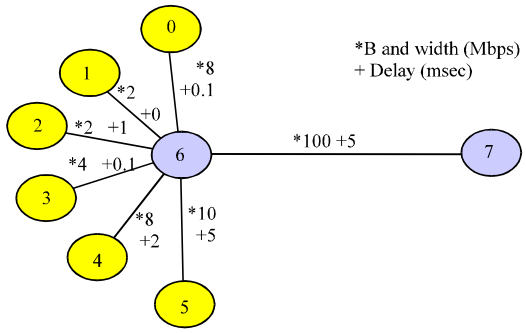

Fig. 2: The proposed network simulation topology

to the value of the loss window (Moraru *et al.*, 2003). Various TCP algorithms and techniques have been proposed to improve congestion and reduce the non-congestion related packet loss. TCP Tahoe, TCP Fack, TCP Reno, TCP Reno with Selective Acknowledgement (SACK), TCP Newreno and TCP Vegas, are examples of proposed end-to-end solutions (Subedi *et al.*, 2008).

Experimentation of TCP will help to get good imagination of real systems that use TCP as a transport protocol and characterize the performance and behavior of the different parts of any networks and is the best way is to use on of the famous network simulator, that is NS-2 (NS-2, 2000). This simulation is very enough and very efficient with TCP simulating over different type of networks, then, in our simulation we used one of latest versions of it, NS-2.32, under windows XP, using Cygwin, where Cygwin is a Linux-like environment for Windows. Typically, TCP is used in wired communication systems with very small bandwidth in wireless links and small data transfer but when use this regular TCP's variants in huge data transfer and with wireless links, then, we can see a large number of Bit Error Rate and some handover scenarios within that. This leads to analyze the behavior of TCP parameters over different type of data and see the limitation of each TCP variants, so one of those important parameters is the *cwnd*. Many models have been built to study the performance of LTE. However, no open source LTE model is found. It is difficult to reuse and evaluate those models. NS-2 is a discrete event simulator targeted at networking research. NS-2 provides substantial support for simulation of TCP, routing and multicast protocols over wired and wireless networks. In the NS-2 simulation, all the data in the network is available, thus the performance of the network can be easily analyzed. What's more, NS-2 is free and open source code and suitable to build system level simulation, so it is deployed to simulate LTE (Qiu *et al.*, 2009).

Figure 2 shows the proposed network simulation topology using NS-2 and shows the parameters of each link. After mapping the topology shown in Fig. 2, the real model is presented with a variable parameters and condition to be flexible with many and different types of TCP, bandwidths, delays and packet size, to cover the all sides of the experimentation in one general topology, to terminate any suspicions in results.

Present aim in this study is to examine how well the behavior of *cwnd* for different TCP variants compares across NS-2 and a simple LTE network testbed, using the same TCP code in each case.

Simulation and network modeling: TCP over LTE is expected to improve substantially end-user throughput, cell capacity and transmission latency. Given the popularity of the Transmission Control Protocol TCP and Internet Protocol (IP) for carrying all types of traffic, LTE supports TCP and IP-based traffic with end-to-end quality of service (Pacifico *et al.*, 2009).

The requirements for the long-term evolution (LTE) can be summarized as follows (Kliazovich *et al.*, 2007):

- Peak data rates of 100 Mb/s in the downlink and 50 Mb/s in the uplink leading to spectrum efficiency of up to 5 bit/s/Hz
- Reduced user- and control-plane latency to less than 10 msec and less than 100 msec, respectively

Figure 3 shows the real model that used in our simulation, where the nodes 0, 1, 2, 3, 4 and 5, are represent a six different node with different delay and different bandwidth. The simulated system scenario is shown in Fig. 3. The topology consist of six nodes (0, 1, 2, 3, 4 and 5), connected to the access routers (6 and 7) via different link parameters which illustrate in Table 1.

The core of network is the link between nodes 6 and node 7, which is the bottleneck link, are interconnected with Ethernet links of 100 Mbps and propagation delay of 5 msec. The terminal nodes are having data connection

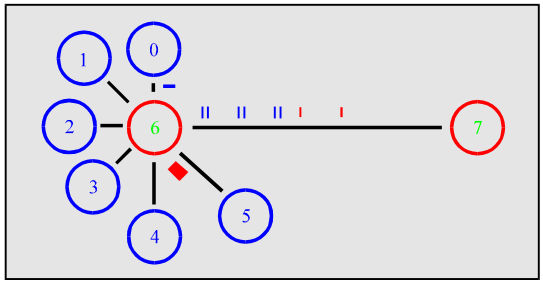

Fig. 3: The real NS-2 network model

Table 1: Parameters of links

Link	Link	Bandwidth (Mbps)	Delay (msec)
Node 0	Node 6	8	0.1
Node 1	Node 6	2	0
Node 2	Node 6	2	1
Node 3	Node 6	4	0.1
Node 4	Node 6	8	2
Node 5	Node 6	10	5
Node 6	Node 7	100	5

of file transfer of FTP, in addition, the maximum packet size is set to 1500 bytes and window size was 128 Kbytes with one drop packet. The maximum scenario duration period was set to be 10 sec only, to observe the behavior of *cwnd* clearly and permitting to see all points of change in each trace.

After we presented all details and parameters, a six TCL scripts programs used to get the *cwnd* traces for each TCP variant in an independent manner, but the all scripts kept to the same parameters and limitations. In each script, we created all links going in both directions between nodes and then, established the layout of model (nam, which shown in Fig. 3), then we created the queues between these nodes with creation of FTP source and associate them with the TCP senders. At last, we finished the simulation after 10.0 sec.

Analysis of *cwnd* behavior: The *cwnd* trace results of six TCP variants, is shown in figures between 4 to 9, each graph contains six curves are represent the scenario of data transfer from node 1, 2,.., 5 to node 7, respectively, so if we said: Node4-*cwnd* when TCP Sack used, that's mean the *cwnd* of TCP Sack for the link between node 4 to node 7 through node 6.

Behavior of *cwnd* for TCP Tahoe: Figure 4 shows the *cwnd* of TCP Tahoe obtained from the proposed model, where the results of the Tahoe simulations are similar in all scenarios except with link of node 2, when the bandwidth and delay set to 2 Mbps and 1 msec respectively, so we can get a maximum performance with node 0, when the parameters was set to 8 Mpbs and 0.1 msec.

The Tahoe sender recovers with a Fast Retransmit followed by Slow-Start regardless of the number of packets dropped from the window of data. For connections with a larger congestion window, Tahoe's delay in slow-starting back up to half the previous congestion window can have a significant impact on overall performance.

Behavior of *cwnd* for TCP Reno: Figure 5 shows the *cwnd* of TCP Reno, where its easy to observe that all links have the same behavior except link one (node 0-node 7), that have long slow-start periods than others. With TCP Reno, Reno's Fast Recovery algorithm gives optimal performance in this scenario, where the sender's congestion window is reduced by half, incoming duplicate Acks are used to clock outgoing packets and Slow-Start is avoided. Reno requires that we receive immediate Ack when a segment is received.

The logic behind this is that when we receive a duplicate acknowledgment, then his duplicate Ack could have been received if the next segment in sequence expected, has been delayed in the network and the segments reached there out of order or else that the packet is lost.

Behavior of *cwnd* for TCP Newreno: Figure 6, shows how *cwnd* seems under TCP Newreno, where no difference with *cwnd* under TCP Reno and its clear to know that is Reno and Newreno perform very well over TCP when the packet losses are small, But with multiple packet losses in one window then Reno and Newreno doesn't perform too well and it's performance is almost the same as Tahoe under conditions of high packet loss.

In fact, Newreno TCP is a variant of Reno with a little modification in Fast Recovery algorithm. This was done in order to solve the timeout problem when multiple packets are lost form the same window, although it can retransmit only one packet per Round Trip Time (RTT). For this reason we can't find large difference in behavior of *cwnd* for two Reno's.

But, Newreno suffers from the fact that it's taking one RTT to detect each packet loss. When the ACK for the first retransmitted segment is received only then can we deduce which other segment was lost.

Behavior of *cwnd* for TCP Sack: TCP with selective acknowledgement called Sack and is an extension of TCP Reno and it works around the problems face by TCP RENO and TCP Newreno, namely detection of multiple lost packets and re-transmission of more than one lost packet per RTT.

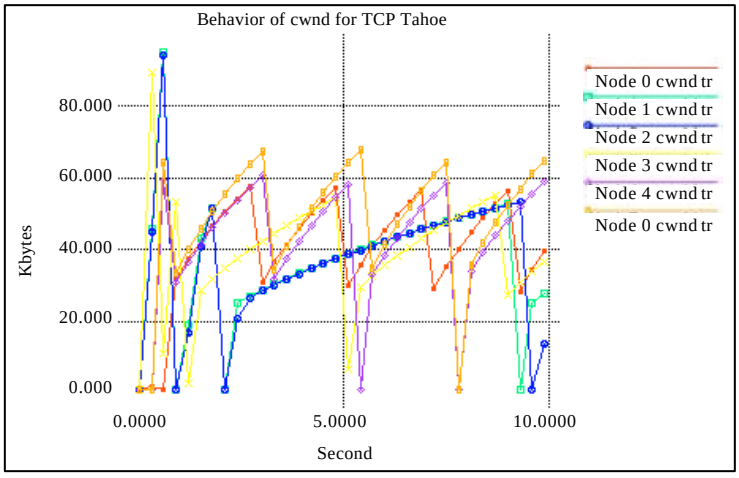

Fig. 4: The behavior of cwnd for TCP Tahoe

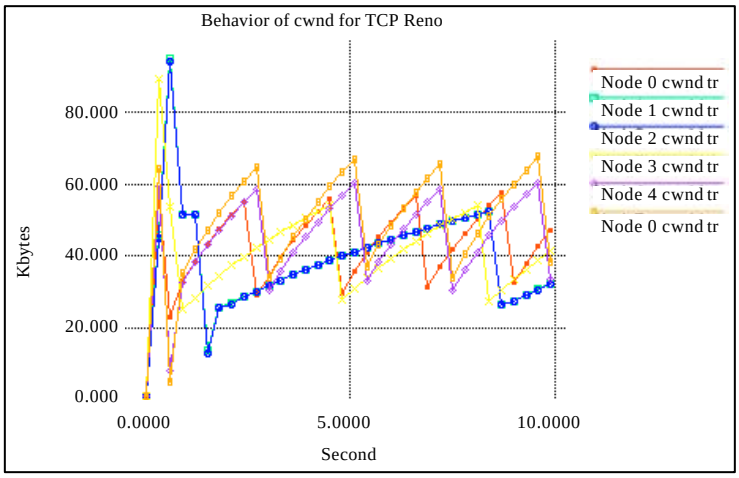

Fig. 5: The behavior of cwnd for TCP Reno

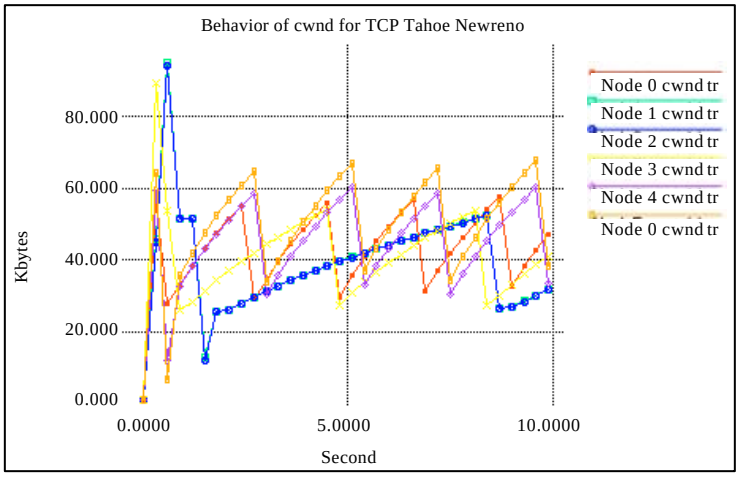

Fig. 6: The behavior of cwnd for TCP Newreno

In Fig. 7, we can see the behavor of *cwnd* in six scenarios, is a combination between *cwnd* in Reno and Tahoe, because of Sack TCP implementation preserves the properties of Tahoe and Reno TCP of being robust in the presence of out-of-order packets and uses retransmit timeouts as the recovery method of last resort.

The large difference between the Sack and Reno implementation is in the behavior when multiple packets are dropped from one window of data. In Sack in during fast recovery, the sender will send data, either new or retransmitted when the value of the pipe less than *cwnd*-number of segments less than *cwnd* value.

Behavior of *cwnd* for TCP Fack: Forward acknowledgment Fack TCP algorithm, is depend on the basic principles of congestion control *cwnd* was designed to be used with Sack TCP option, so, the main aim of Fack algorithm is to provide an exact *cwnd* within recovery by keeping an accurate estimation of the data in the network. In our simulation we can observe, the slow start period for all nodes still need more time to get a more stability and to be regulated because the TCP Fack, regulates the amount of out data in network to be within only one segment of *cwnd* (Fig. 8).

Behavior of *cwnd* for TCP Vegas: In Fig. 9, *cwnd* behavior Vegas TCP, is very different if compared with other variants, because Vegas does not wait for loss to trigger *cwnd* reductions and keeps track of the time each segment is sent. Vegas used this concept to decide if it should increase or decrease the window. We can see, *cwnd* size depends to the bandwidth of link and when the

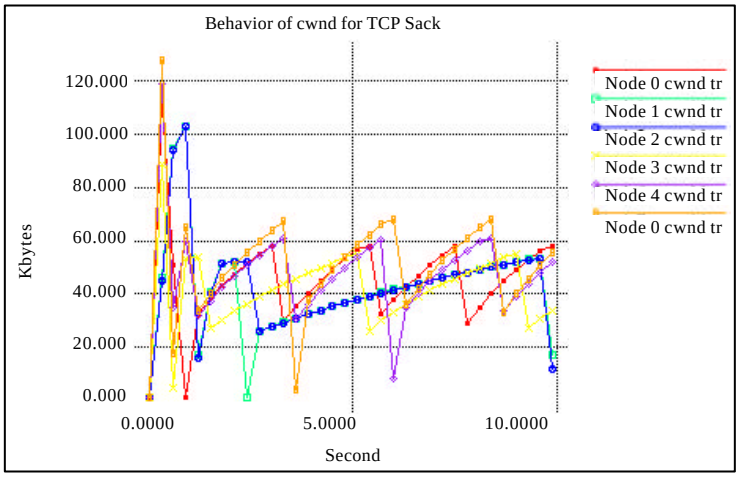

Fig. 7: The behavior of *cwnd* for TCP Sack

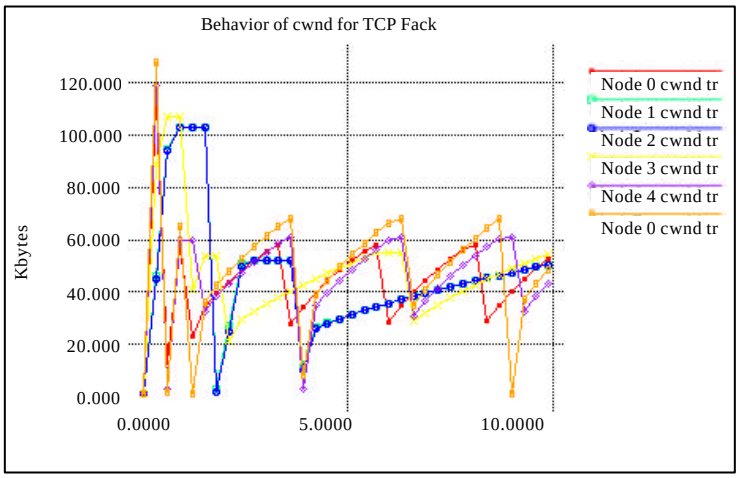

Fig. 8: The behavior of *cwnd* for TCP Fack

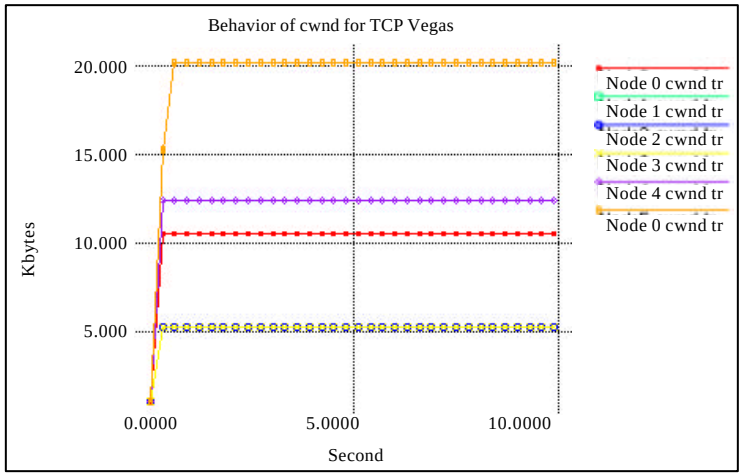

Fig. 9: The behavior of *cwnd* for TCP Vegas

bandwidth increased, it would increase so. In other side, for the links started with nodes 1, 2 and 3, were appear with the same *cwnd* behavior because of the small values of bandwidths.

CONCLUSION

In this study, we presented a *cwnd* behavior over six TCP source variants with parameters of LTE network using NS-2 networks simulation and we discussed the suggested model and the parameters of the network model. Each TCP variants tested over different links with six nodes connected through bottleneck link and the *cwnd* trace graph for each scenario is illustrated.

We will study the behavior of other TCP variants with many drop packets, so we can add more complex scenarios such as handover cases and then measure the throughput and the *cwnd* behavior.

ACKNOWLEDGMENT

This study is sponsored by Universiti Kebangsaan Malaysia (UKM) through the university research grant UKM-OUP-ICT-36-182/2010.

REFERENCES

Islam, M.S., M.A. Kashem, W.H. Sadid, M.A. Rahman, M.N. Islam and S. Anam, 2009. TCP variants and network parameters: A comprehensive performance analysis. Proceedings of the International MultiConference of Engineers and Computer Scientists, IMECS, Mar. 18-20, Hong Kong, pp: 1-3.

Khan, F., 2009. LTE for 4G Mobile Broadband: Air Interface Technologies and Performance. Cambridge University Press, New York, pp: 4-6.

Kliazovich, D., F. Granelli, S. Redana and N. Riato, 2007. Cross-layer error control optimization in 3G LTE. Proceedings of the IEEE Global Communications Conference (GLOBOCOM), Nov. 26-30, Washington D.C., pp: 2525-2529.

Moraru, B., F. Copaciu, G. Lazar and V. Dobrota, 2003. Practical analysis of TCP implementations: Tahoe, reno, newreno. Proceedings of the 2nd RoEduNet International Conference on Networking in Education and Research, (NER'03), University of Cluj-Napoca, Bogdan, pp: 125-130.

NS-2, 2000. Network simulator. http://www.isi.edu/nsnam/ns/.

Pacifico, D., M. Pacifico, C. Fischione, H. Hjalmasson and K.H. Johansson, 2009. Improving TCP performance during the intra LTE handover. Proceedings of the IEEE Global Telecommunications Conference, GLOBOCOM, Nov. 30-Dec. 4, Honolulu, H.I., pp: 1-8.

Qiu, Q., J. Chen, L. Ping, Q. Zhang and X. Pan, 2009. LTE/SAE model and its implementation in NS 2. Proceedings of the 5th International Conference on Mobile Ad-hoc and Sensor Networks, (MSN'09), China, pp: 299-303.

Rahman, M.A., A.H. Kabir, K.A.M. Lutfullah, Z. Hassan and M.R. Amin, 2008. Fair comparisons of different TCP variants for future deployment of networks. Proceedings of the International Conference on Electronics, Computer and Communication, (ICECC'08), University of Rajshahi, Bangladesh, pp: 260-263.

Subedi, L., M. Najiminaini and L. Trajkovic, 2008. Performance evaluation of TCP tahoe, reno, reno with SACK and newreno using OPNET modeler. Proceedings of the 12th Annual Conference on OPNET Technology, (OPNET'08), Simon Fraser University, Washington, D.C. pp: 1-7.