ISSN: 2413 – 6999
Journal of Information, Communication, and Intelligence Systems (JICIS)
Volume 2, Issue 3, September 2016

Methodical Improvement of Congestion Control Mechanism in Distinctive Data Transmission

Raghad T. Abdulrazzaq Al-Hassani

***Abstract*— Data transmission using protocols such as TCP (Transmission Control Protocol) between wireless and wired systems, is still include a concrete difficult. The congestion control appliance does not allow the flow of data to obtain a full bandwidth on present system acquaintances. To resolving this problematic, various protocols is hosted with a high performance. Consequently, this article improved congestion control algorithm and congestion window (*cwnd*) is suggested to increase TCP performance by growing the cycles number of the next *cwnd* to improving the number of sent packets. The suggested procedure uses a novel technique supported by the existing bandwidth of links to detecting the capacity of the network channel to develop a steady clocking mechanism in congestion avoidance phase. The effort provided in this article is based on Network Simulation (NS-2) to modeling the putative mechanism.**

***Index Terms*—Congestion Control, Network Simulation, Data Transmission.**

I. Introduction

The NS-2 simulator (Network Simulator 2) [1] is a free network simulator object-oriented and discrete events, which offers a arrangement to form the prototype network. NS-2 identifies the data as input factors, the outcome of investigating data and generating consequences. The two central motives for the broad impact of NS-2 is since it is unrestricted, which fits academics in research laboratory and universities, also is the enormous range of components and system items that you can implement via NS -2 [2]. On other hand, TCP is the greatest and commonly used protocol for the wireless and wired systems, while TCPs were not initially considered in real time application or for wired or wireless network.

Raghad T. Abdulrazzaq Al-Hassani is now with Electrical & Electrical Engineering Department, Faculty of Engineering, Al-Fateh University, Tripoli, Libya (e-mail: eng_raghadtariq@yahoo.com).

At that time, we must to developing a n innovative TCP varieties, or at least select appropriate TCP type for each advanced system to making it more proficient and more dependable with these networks.

Initially, the congestion control tools provided in early 1980 and was considered predominantly to break the failure in link congestion. In current years, the huge transmitted data and the improved capacity of traffic created by real time application force the developers to deals with high amount of transmission packets and that delivers a congestion situation and then the link should control the packets amount [3].

In this study, we will explore the possibility to improve the act of the proposed congestion control mechanism, which established to be capable to transmission high-rate of packet over large bandwidth and low latency links. Besides, when applying typical TCP used over cellular systems, the outcome data in end-to-end production will be very weak because of the radio links and interference. This is since the dynamic features of TCP due to its doesn’t developed to run over and wireless networks [4].

TCP protocols bounds data transmission rate by adjusting the sent packets in *cwnd*, where this window represents the amount of packets that can be transferred in the link pipe. Normally, the period among the transmission and receive of ACK packets is called RTT (Round Trip Time), the TCP protocol sender can send to *cwnd* a packets through only one RTT. When TCP send a packet to the *cwnd*, it must send the next packet only after this packets arrives by sending ACK to that sender. So, the typical amount of packets in TCP *cwnd* is practically the magnitude of window distributed via RTT [5].

The significant idea in TCP protocol is the congestion control mechanism and *cwnd*, where the window is the number of packets that have been transferred, but haven't received yet an acknowledgment. Commonly, the TCP variants are different in congestion control mechanism, and each TCP type has been documented as a standard TCP in RFC2581 [6], and usually mentioned to TCP Reno, while the other TCP's is Newreno [7], and the variant of TCP which is relative with selective ACK is called Sack TCP [8].

To Enhancing TCP protocols performance over next generation networks, many TCP versions are mainly used and each are differ on the functions and the behavior of congestion control in congestion situations, and that produced TCP source variants and the other variants. Presently, the TCP types are six; Reno, Newreno, Vegas, Fack, Sack, and Tahoe, with other comprehensive TCP alternatives, where every type has isolated congestion control procedure or established a previous mechanism, to be effective and dependable with a new end-to-end applications.

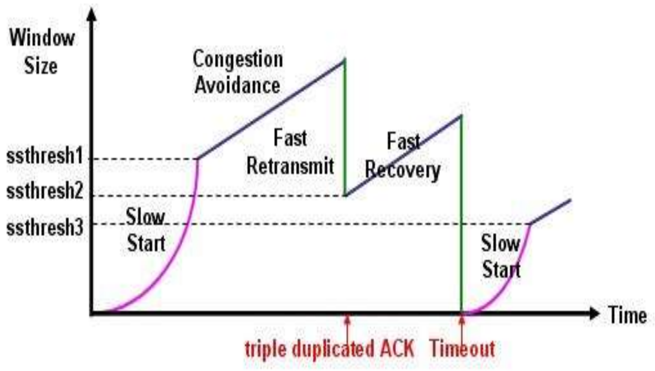
II. Congestion Control Mechanism

To controlling the packet congestion, TCP should adjust *cwnd* of transmitted packets at TCP sender sides in a way that is stopping packets excess in receiver. To accomplishing this, TCP used additional parameters to regulating *cwnd*. Then, we can identify the congestion control represents a amount of packets that can be inserted in to the network connection without producing congestion situation. The challenges is to taking advantages of the existing bandwidth in the network channel. Furthermore, TCP suppose or detect the network congestion when the retransmission timer is expire, and that it relates with network congestion status by regulating *cwnd* two mechanisms, slow-start and the congestion avoidance, as presented in the Fig. 1.

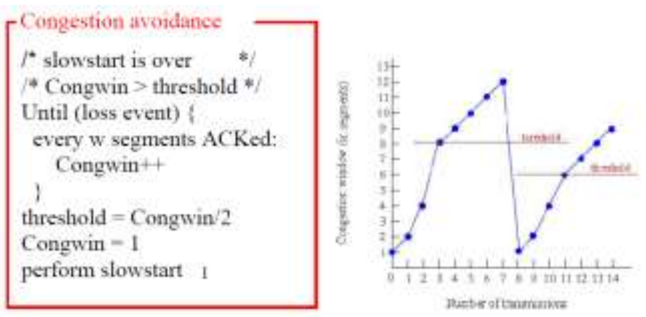
In phase of slow-start, and after connection establishes, at first TCP sets *cwnd* to 1, and then each ACK is received, *cwnd* is increased to be:

$$cwnd = cwnd + 1$$

and that means that *cwnd* will duplicating for each RTT.



(a)



```
Congestion avoidance
/* slowstart is over      */
/* Congwin > threshold */
Until (loss event) {
   every w segments ACKed:
      Congwin++
   }
threshold = Congwin/2
Congwin = 1
perform slowstart   1
```

(b)

Fig.1 Slow-tart phase and congestion avoidance phase in Typical TCP

The speedy development of *cwnd* remains till the packet losses was detected, then reach to the level of slow-start threshold (*ssthresh*) which will updated to:

$$ssthresh = cwnd/2$$

Then once a packet dropping happen, the link starts from slow-start once more by setting *cwnd* to be equal 1, and is growing exponentially till *cwnd* becomes equal to *ssthresh*. At the moment, the TCP packet transmitting goes to the next phase in congestion control mechanism, the congestion avoidance phase, where *cwnd* becomes :

$$cwnd = cwnd + 1/cwnd$$

This will indicates a linear behavior instead of the exponential growing, and continuing on increasing till detect the packet losing.

The proposed congestion control mechanism that introduced in this paper is based on an innovative technique to avoid congestion that estimating the expected throughput with the vision of advanced efficiency, nevertheless of the network congestion situation. This new mechanism is based on using the existing size on network connections to detecting the cumulative or declining the size of *cwnd* to obtaining an efficient congestion avoidance strategy. Surely, the suggested procedure providing an increase in the network connection around 20-30%, and that permits to increasing the bottleneck size too, even the link is suffer of congestion situation.

In the offered mechanism, we are used typical exponential increase in slow-start mode, and the *cwnd* supposed to stay less than the value of *ssthresh* and the *cwnd* size surges by 1 per RTT, as shown in Eq. (1):

if ($cwnd < ssthresh$)

Then:

$$cwnd = cwnd + 1 \qquad (1)$$

Besides, the TCP sender will update the size of *cwnd* in the congestion avoidance mode in corresponding to the Eq.(2) after each ACK receiving from TCP receiver:

$$cwnd = cwnd + (f / cwnd) \qquad (2)$$

Where *f* represents a control factor. In Eq. (2), we suppose that the *cwnd* size growths by *f* segments per RTT, where the focal purpose of the proposed mechanism is to adjust the *f* value adaptively and dynamically, and in standard TCP Reno the value of *f* is set to be 1.

In next section, we describe the effect to changing *f* according to the link and network situation in addition to the resent TCP throughput. The standard mechanism is shown below:

$$\text{cwnd\_} = cwnd\_ + (f / cwnd\_)$$

Where:

*cwnd_* : is the real *cwnd* variable in NS-2.

*ssthresh_*: is the real *ssthresh* variable in NS-2.

f : control parameter, and f=1 in Reno.

In the suggested congestion avoidance mechanism, we require to updating *f* explained in Eq. (2) when TCP sender getting a new ACK.

Actually, the mechanism implementation depends on main four key factors or parameters to control *f* values per RTT. These parameters demonstrated below:

- *ssthresh*: slow-start threshold of network path.
- *cwnd*: the last value of *cwnd*.
- wnd_const: packets per RTT.
- K_parameter: k parameter in binomial controls.

These parameters are used to estimating *f* values, and the increment step of *cwnd* turn out to be very big when the existing TCP link throughput is extreme under the objective throughput.

On other hand, the big *f* values can causes burst situation in packets losses and that result a performance degradation because of re-transmission timeouts. Alternatively, when the network links has enough remaining bandwidth, the steps degree of incrementing on *cwnd* turn out to be less than one. As a result, the mechanism limits the minimum and maximum of *f* values. So, per RTT, the next value of *f* come to be:

$$f = ssthresh \text{ x wnd\_const} \qquad (3)$$

Eq. (3) illustrates that *f* will be equivalent to the existing bandwidth; and that can regulate the transfer period of packet according to the existing bandwidth in network link. Then, if the suggested procedure in Eq. (3) has gotten a higher throughput as described, we must minimizing the value of *f* to avoid packet loss. The regulator of this problem is via dividing the last values of *f* that obtained by Eq. (3) using the previous value of *cwnd* and multiplied it by the exponent *cwnd* value to the *k* factor in a binomial controller as presented in Eq. (4):

$$f = f / (cwnd \text{ x pow}(cwnd, \text{k\_parameter})) \qquad (4)$$

That means, the suggested congestion avoidance mechanism takes the bandwidth from computing the flows in network links to achieving the required bandwidth.

In briefly, the suggested mechanism is update *f* value when TCP sender is receive a next new ACK via the formula shown below:

$$f = \text{wnd\_const\_} \times ssthresh\_/cwnd\_ \times \text{pow}(cwnd\_,\ \text{k\_parameter\_});$$

$$cwnd\_ = cwnd\_ + (f / cwnd\_);$$

III. Results Analysis

Initially, we are checked if the innovative mechanism run effectively over the available bandwidth in the network connection. In that tests, we achieved that the data transmission by the TCP- based new mechanism via changing the network bottleneck bandwidth and save all last parameters.

As presented in Fig. 2, we are used 100 Mbps as a bandwidth and experimented the *cwnd* behavior of the enhanced TCP and comparing it with the standard TCP Reno.

The outcomes of experimenting *cwnd* through this test achieved that the *cwnd* of the enhanced TCP is done well if compared it with *cwnd* of TCP Reno, furthermore, the clocking to the enhanced TCP is also improved than TCP Reno. Additionally, the enhanced TCP provided a flat slow-start behavior like Reno.

Fig.3 shows that the big increase in *cwnd* between enhanced and original TCP, and this alteration return to the increasing in the bandwidth from 100Mbps in first experiment to 200Mbps.

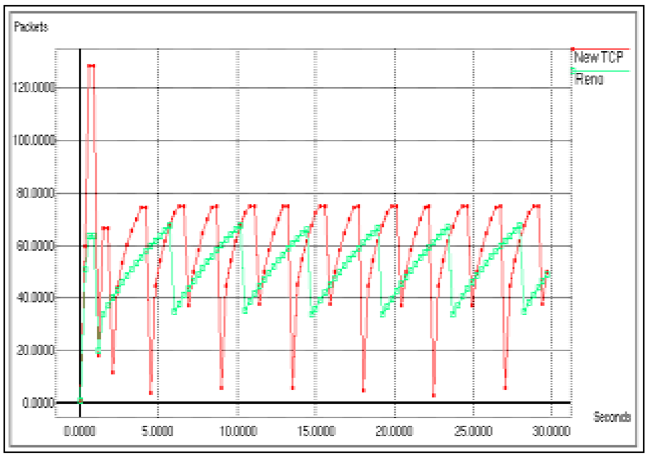


Fig.3 The behavior of enhanced TCP with standard TCP Reno when BW is 200Mbps.

That increase allows to the proposed technique to exposed the *cwnd* to reaching to 75 packets, but typical Reno keep the packets level in both experiments to smaller than 65 packets. The cause of this return back to the differences in the congestion avoidance technique that are used new and original TCPs. The difference in performance of both TCPs seems clearly in the next experiment, when the used bandwidth upgraded to 500Mbps as expressed in Fig. 4.

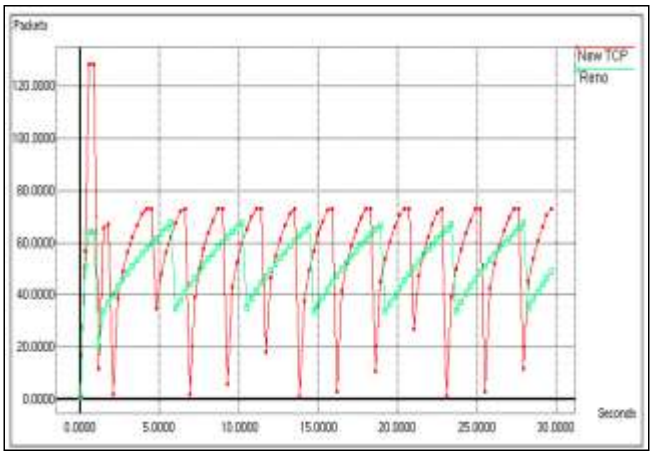


Fig.2 The behavior of enhanced TCP with standard TCP Reno when BW is 100Mbps.

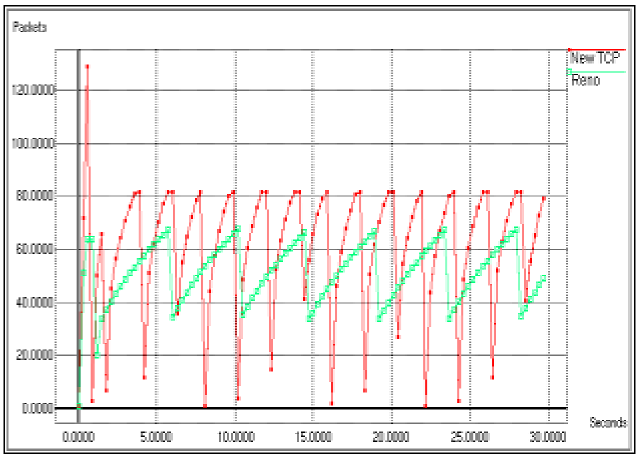


Fig.4 The behavior of enhanced TCP with standard TCP Reno when BW is 500Mbps.

That causes to obtaining a *cwnd* more than 80 packets in enhanced TCP, while it remains in same level and constant *cwnd* typical Reno. Moreover, Fig.4 illustrates that the existing bandwidth can gives a throughput during a cross traffic in network links.

IV. Conclusion

The paper provided the details of using new technique to improving the congestion avoidance phase in TCP congestion control. The proposed technique based on modifying the steps of increment in *cwnd* size per RTT after TCP connection is established by sensing the current bandwidth of the network link. The improved technique offers 125 packets as a *cwnd* level before the congestion status, while standard TCP Reno can reach a level of *cwnd* with 65 packets only. Furthermore, the enhanced TCP provided faster performance more than twice that got by standard TCP Reno. So when it finishes one duration of *cwnd* in standard TCP Reno, we can obtain more than two duration in the enhanced TCP, and that achieve that the proposed technique can send about twice packets amount in new the mechanism.

References

[1] J. Olsén, Stochastic modeling and simulation of the TCP protocol. 2003: Matematiska Institutionen.

[2] G. A. Abed, M. Ismail, and K. Jumari, "Distinguishing Employment of Stream Control Transmission Protocol over LTE-Advanced Networks," *Research Journal of Information Technology,* vol. 3, pp. 207-214, 2011.

[3] M. Allman, V. Paxson, and W. Stevens. TCP congestion control. RFC 2581, April 1999.

[4] S. Floyd and T. Henderson. The NewReno modification to TCP’s fast recovery algorithm. RFC 2582, April 1999.

[5] M. Mathis, J. Mahdavi, S. Floyd, and A. Romanow. TCP selective acknowledgment options. RFC 2018, October 1996.

[6] S. Floyd, J. Mahdavi, M. Mathis, and M Podolsky. An extension to the selective acknowledgement (SACK) option for TCP. RFC 2883, July 2000.

[7] G. A. Abed, M. Ismail, And K. Jumari, "The Evolution To 4g Cellular Systems: Architecture And Key Features Of LTEAdvanced Networks," Spectrum, Vol. 2, 2012.

[8] G. A. Abed, M. Ismail, And K. Jumari, "Architecture And Functional Structure Of Transmission Control Protocol Over Various Networks Applications," Journal Of Theoretical And Applied Information Technology, Vol. 34, 2011.