Scientific Journal of Impact Factor(SJIF): 3.134
e-ISSN(O): 2348-4470
p-ISSN(P): 2348-6406

International Journal of Advance Engineering and Research Development

Volume 2,Issue 11, November -2015

A Study of Congestion Control Strategy of TCP New Reno Protocol

Bhoomika K.Sharma [1], Prof. Dhaval A. Parikh [2]

[1] Department of computer science and engineering ,L.D. Engineering collage, Ahmedabad,india
[2] Department of computer science and engineering ,L.D. Engineering collage, Ahmedabad,india

***Abstract*—** *Now a day's use of wireless networks for access the Internet is increasing. It shows that use of the main transport layer protocol TCP will be increase in near future. The traditional TCP suffer from packet losses and throughput degradation over wireless link. To improve performance of the TCP New Reno, so many variants of the TCP have been proposed like TCP Reno, New Reno, Tahoe, and Westwood etc. But no one has been deploying successfully except TCP New Reno.TCP New Reno shows improvement in throughput over Reno and SACK in wireless network. But TCP New Reno does not differentiate issue of packet loss due to congestion or bit error. So it always decrease cwnd (congestion window) in all issues of packet loss and degrade the performance. This paper proposes to improve the performance of TCP New Reno by differentiate issue of packet loss through either congestion or bit error. In our study we seen that New Reno gives us better performance in wireless link for throughput, delay and packet delivery ratio. In recent research, New reno can differentiate bit error and congestion control loss.*

***Index Terms*— TCP New Reno, Congestion window, Wireless Network, Congestion Control.**

I. INTRODUCTION

Today in the Internet majority of traffic uses connection oriented services of TCP [4]. The TCP is trustworthy protocol because it uses acknowledgment mechanism. In the Internet many applications like mail, www, ftp, telnet etc, uses TCP protocol. The TCP performs better in wire network [4] but in wireless network; degrade the performance of the TCP. So need to improve mechanism of TCP congestion detection and congestion control as well as distinguishing congestion loss from random loss in wireless link. Many algorithms have been proposed for improve performance of TCP New Reno. In this paper, we proposed a method for differentiate packet loss from either congestion or bit error and with the help of this technique we improve throughput of the TCP New Reno. Many TCP variants have been proposed for improve throughput of the TCP like TCP Reno, TCP New Reno, TCP SACK, TCP Tahoe, TCP Vegas, TCP Westwood [8]. Among these protocols variants only TCP New Reno gives better performance and successfully deploy now a days in Linux operating system. But all these variants of the TCP and original TCP are still unable to sense the cause of packet loss [4]. Hence all loss in the TCP New Reno services is treated as congestion loss, not consider bit error and hence reduce window size and data flow [6]. Finally degrade performance of the TCP New Reno. So in this paper we suggest a new idea for increase performance of the TCP New Reno by using method of differentiate issue of packet losses. At last this paper also shows algorithm description and compared simulation results.

II. TCP CONGESTION CONTROL ALGORITHM

As show in figure-1, TCP congestion control algorithm has four phases: 1) Slow Start 2) Congestion Avoidance 3) Fast Retransmit 4) Fast Recovery [1].

A. SLOW START. TCP uses slow start mechanism to control transmission rate of the sender. This phase has been accomplished by receiving rate of the acknowledgment from receiver. When TCP establishes connection, the slow start algorithm set congestion window to one segment. At this phase cwnd = MSS (maximum segment size). When acknowledgment is return by receiver, the congestion window increase by one segment for each acknowledgment received. This phase is actually not so much slow, because every time when ACK received, congestion window increase at double rate, i.e. when sender gets first ACK, sender increases cwnd by two segments, when sender gets other two ACKs, sender increase cwnd by four segments, so on. At threshold level cwnd reaches at maximum level and packets loss will trigger and sender goes into congestion avoidance mode [1].

B. CONGESTION AVOIDANC. A point during slow start that network is forced to drop one or more packets due to congestion. If this happens, congestion avoidance is used [1]. In congestion avoidance algorithm, the sender knows about loss of packets due to congestion when duplicate ACK receive by sender.

The sender immediately reduces the cwnd by one half of current window size, but to at least two segments. If timeout occurs due to congestion, cwnd reduce or reset to one segment, which automatically puts the sender into slow start mode. However in this phase slow start is only use up to the halfway point where congestion originally occurred. After this halfway point, cwnd is increased by one segment. If the congestion was noticed by DUPACK (duplicate acknowledgment), starts fast retransmission and fast recovery algorithm.

C. FAST RETRANSMIT. When DUPACK received by sender, it does not know the actual reason that the segment was lost or simply that segment was delayed. Typically no more than one or two duplicate ACKs should be received when simple segment has been delayed [7]. But when more than two DUPACKs received by the sender, it is a strong indication that at least one segment has been lost due to congestion. When three DUPACK are received, the sender does not wait for time out and immediately retransmit lost segment. This procedure is called fast retransmission.

D. FAST RECOVERY. With the help of DUPACK, the sender know about other segments receive successfully at receiver. This is a strong indication that serious congestion may not happen and loss of the segment due to delayed. So instead of reducing window abruptly by going all the way into slow start, the sender only enters congestion avoidance phase [7]. The sender does not set cwnd to one segment as in slow start phase, but resumes transmission with larger window and continuous incrementing. This allows better throughput and performance under moderate congestion. To summarize this section figure 1 show what typically data transfer phase using TCP congestion control might look like.

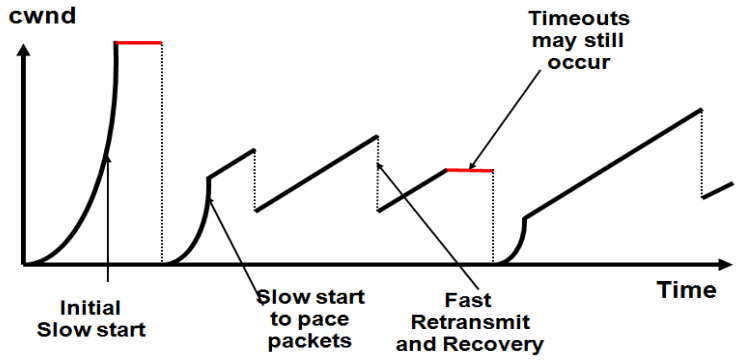


Figure-1. Congestion Control Algorithm in TCP

III. PERFORMANCE OF TCP OVER WIRELESS LINK.

The TCP is reliable and pervasive transport layer protocol. Traditional TCP suffer from congestion issue like packet losses due to congestion or timeout. So traditional TCP is slower protocol compare to UDP. So several variants of the TCP have been proposed like TCP Reno, TCP New Reno, TCP Tahoe, TCP Westwood, TCP Vegas and TCP SACK [2]. In this section we discuss performance of different variants of the TCP and then we discuss our modified algorithm for improve performance of TCP by modifying TCP New Reno [6].

A. **TCP RENO.** In 1990, TCP Reno has been developed and TCP Reno uses previous slow starts [7] and retransmits timer mechanism & improves by adding fast recovery algorithm and prevent from empty transmission path or pipeline. For indication of packet loss, TCP Reno uses 3 DUPACK (duplicate acknowledgment) mechanisms, i.e. whenever we receive 3 DUPACK then it assume that packet loss during transmission and retransmit packet without waiting of timeout [2]. Then it reduces window size and set cwnd (congestion window) to half [2]. But limitation of Reno are, it does not suitable when multiple packets loss in single window, window does not continuously modified and Reno leads to half window size after recovering from first packet loss so subsequent losses can't be identify 3 DUPACK. Another problem associated with Reno is packets arriving out of order at receiving end can yield DUPACK when in fact there is no loss or error. TCP Reno does not work for small window size like less than 4 packets because it does not provide 3 DUPACK [2].

B. **TCP NEW RENO.** Limitations of TCP Reno overcome by another variant call TCP New Reno [5]. At 1996, Hoe gives new variants called New Reno. The New Reno performs better compare to Reno when multiple packets losses occur. New Reno modified fast recovery or fast retransmit algorithm to improve throughput of TCP and over here fast indicates it does not wait for time out when not getting an ACK for a packet [5]. TCP New Reno uses two types of ACKNOWLEDGMENTS 1) a full ACK 2) a partial ACK. Full ACK acknowledges all the outstanding packets and

partial ACK acknowledges some outstanding packets. With the help of partial ACK, TCP New Reno does not exit from fast recovery but indication of packets immediately following the acknowledged packet has been lost and retransmitted immediately. In this manner TCP New Reno modified fast recovery algorithm. But limitation of TCP New Reno is, it can't detect delay and it is limited to resend at one lost packet per RTT [5].

C. **TCP TAHOE.** In 1988, TCP Tahoe was design as modification of traditional TCP for improves performance of TCP. It uses different mechanism of TCP like slow start, congestion avoidance and fast retransmits [8]. Fast retransmit is main advantage of TCP Tahoe because sender does not wait for timeout when any segment loss during transmission. When any loss occur in network and receive DUPACKs, sender immediately retransmit segment without waiting of timeout period. But limitation of TCP Tahoe is that packet loss is detected after whole timeout interval. So Tahoe degrade performance in this case, when loss detected after time out. Tahoe uses modified RTT (round trip time) estimator [8].

D. **TCP WESTWOOD.** Westwood is modified version of TCP Reno [8]. During loss of segments, Reno reduces size of congestion window (cwnd) to half and degrades the performance whereas Westwood estimates the available bandwidth of the connection with the help of rate at which ACKs received. This delivery rate is calculated from ACK information. The estimated bandwidth uses for fast recovery when packet loss occurred [6]. TCPW tries to select appropriate cwnd and ssthresh (slow start threshold). Selected cwnd base on bandwidth estimator improves the performance compare to TCP Reno. TCPW can't differentiate issue of segment loss through either congestion or random loss in wireless link. [6]

E. **TCP VEGAS.** Compare to other TCP variants, TCP Vegas uses better bandwidth estimation scheme [7]. Vegas estimates bandwidth using current data flow rate and expected data flow rate. Vegas stores current values of system clock and segment transmission time. So it is able to know exact RTT for each sent segment [8]. Vegas does not wait for packet loss, it adjust cwnd as soon as it detects congestion in the network. Also retransmission mechanism is better than other variants, it retransmits loss packet as soon as it receives a single DUPACK [7]. Vegas does not wait for 3 DUPACK as in Reno. It does not reduce congestion window unnecessarily. When sender receives single DUPACK, it checks if (current time – packet transmission time) > Round Trip Time. If this condition is true, the sender provides a retransmission without waiting of timeout or 3 DUPACK [8].

F. **TCP SACK.** SACK is a technique that can help reduce unnecessary retransmission on the part of sender [5]. In the TCP SACK, receiver can offer the feedback to the sender about successful receiving segments in the form of selective acknowledgment option. It uses option field of the TCP header. In the SACK option fields tell the sender which contiguous segments it has received [5]. Receiver includes SACK information in the TCP header only when arrival of out of order packet at receiving ends. It enters the fast retransmit phase when loss occurs and it exists when all the sent data has been acknowledged. Limitation of SACK is, it can't differentiate loss due to congestion or bit error on the wireless network [5].

IV. PERFORMANCE ISSUE OF TRADITIONAL TCP NEW RENO.

TCP New Reno is implemented in linux operation system and successfully deployed. But TCP New Reno does not differentiate reason of packet loss either through congestion or through bit error on wireless link [6]. For recovery of packet loss, traditional TCP used time out and set cwnd to 1. TCP Tahoe used fast recovery algorithm to retransmission of segments, but it also set cwnd to 1. TCP Reno used 3 DUPACK mechanisms and enters into fast recovery mode when gets 3 DUPACK but does not work with multiple loss of segments. TCP New Reno deal with multiple losses but not deal with bandwidth delay [5]. SACK used selective ACKs, but does not differentiate reason of loss [5]. Compare to other variants TCP New Reno performsbetter and improve the throughput on wireless link [9]. But it also reduces congestion window either at time out or at 3 DUPACKs. But there is no need to reduce cwnd to each and every case like bit error or random loss. TCP New Reno always reduce cwnd either set to half or set to 1 [9]. Simply TCP New Reno does not distinguish congestion loss from bit error loss on wireless link. We observe that when buffer size of the router is small and less than 10% of BDP (bandwidth delay product), queue management scheme not work properly with TCP New Reno. Main limitation of the TCP New Reno is cuts down its CWND constantly upon loss [6].

V. BACKGROUND:

In "Modified TCP NewReno for Wireless Networks" [1] Ahmead and kabir said that the overloaded routers do not get flooded with thenew segments and get some time to drain out their queues without dropping the segments. This action helps the network to ease the congestion if there is real congestion in the network. Hotheyver, if segments are lost due to bit error then there is no gain in reducing the transmission rate. In this case, the sender should continue transmitting at the original rate and try to deliver as many segments as possible in the midst of random bit errors. Throttling transmission rate will not do any good in this scenario. Hence, some new strategies should be introduced in TCP such that it can detect segment loss due to bit error and act accordingly. They keep a running count of the number of timeouts and the number of 3-dupacks experienced during an interval. Whenever the sender experiences a timeout or 3-dupack event, they compute the ratio of the number of timeouts to the number of 3-dupacks.If the network is congested then the timeout will happen successively. In this case, the time difference bettheyen two consecutive timeout events will be roughly equal to

the timeout interval of the retransmission timer at that instant. On the other hand, when the network is not congested, the TCP sender will only encounter timeout events whenever the cwnd crosses the current network capacity. That will be quickly resolved by the TCP sender by entering into the slow-start phase. Again, if there are random bit errors, then some segments will be damaged and will be rejected by the receiver. Hotheyver, duplicate acknowledgements will be returned from the receiver to the source for subsequent segments which are not lost. This will prevent the source from having timeout events and will also enable the source to solve potential loss of segments using fast retransmit and fast recovery. So, in non-congestion scenario the timeouts will be sparse and the time difference bettheyen two successive timeouts will be much greater than the retransmission timer's estimated timeout interval at that moment. This work can also be used during a timeout or 3-dupack event to detect whether the event is a result of real network congestion or due to random segment losses due to bit error In "TCP NCE: A unified solution for non-congestion events to improve the performance of TCP over wireless networks" [2] they propose a unified solution called Transmission Control Protocol (TCP) for Non-Congestion Events (TCP NCE), to overcome the performance degradation of TCP due to non-congestion events over wireless networks. TCP NCE is capable to reduce the unnecessary reduction of congestion window size and retransmissions caused by non-congestion events such as random loss and packet reordering. TCP NCE consists of three schemes.Detection of non-congestion events (NCE-Detection), Differentiation of non-congestion events (NCE-Differentiation)and Reaction to non-congestion events (NCE-Reaction). For NCE-Detection, theycompute the queue length of the bottleneck link using TCP timestamp and for NCE-Differentiation, they utilize the flightsize information of the network with a dynamic delay threshold value. They introduce a new retransmission algorithm called 'Retransmission Delay' for NCE-Reaction which guides the TCP sender to react to non-congestion events by properly triggering the congestion control mechanism. According to the extensive simulation results using qualnet network simulator, TCP NCE acheives more than 70% throughput gain over TCP CERL and more than 95% throughput improvement as compared to TCP NewReno, TCP PR, RR TCP, TCP Veno, and TCP DOOR when the network coexisted with congestion and non-congestion events. Also, they compared the accuracy and fairness of TCP NCE and the result shows significant improvement over existing algorithms in wireless networks.

In "Performance Improvement of TCP Reno Based on Monitoring the Wireless Packet Loss Rate" [3] they proposes a new modification of TCP Reno based on monitoring the wireless packet loss rate in real time. When the modified Reno cooperates with the router configured with explicit congestion notification (ECN), it is capable of distinguishing the wireless packet losses from the congestion packet losses, and reacting accordingly. At the same time, the sender takes advantage of the monitor result to adjustment the TCP segment size. The simulations in this work show that the modification of TCP is feasible, and the performance of TCP is improved actually.

In "Enhancing TCP Performance in Hybrid Networks with Fixed Senders and Mobile Receivers" aims to address the challenges of high bit error rate in wireless links and long disconnections due to mobility. It keeps the modification in BS and MH minimal without requiring any changes to FH. The BS uses one bit (ECN bit) to enable the MH in distinguishing bettheyen congestion and wireless losses, and thus, suppressing unnecessary duplicate acknowledgments (dupACK) due to wireless losses. In addition, zero window acknowledgment and triplicate dupACK schemes are adopted as part of the handoff procedure in BS. Both simulation and network test-bed results show that TCP-ECN performs significantly better than TCP-Reno and Snoop in the face of high wireless loss, high or low congestion loss, and mobility. Performance improvement is more significant when more wireless receivers are supported.

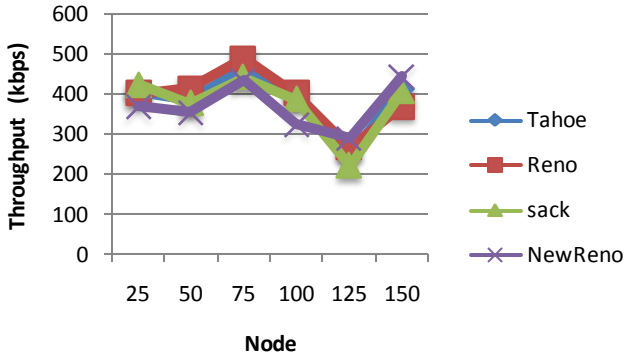
VI. SIMULATION PARAMETERS.

Parameter	Values
Traffic time	TCP
Simulation time (sec)	100
Simulation Area	500 X 500
Simulation Model	TwoRayGround
MAC Type	802.11
Number of nodes	25,50,75, 100,125,150
Connection	10 and 25
Queue Length	50
Routing Protocol	AODV
Link Layer Type	LL
Antenna	Omni Antenna

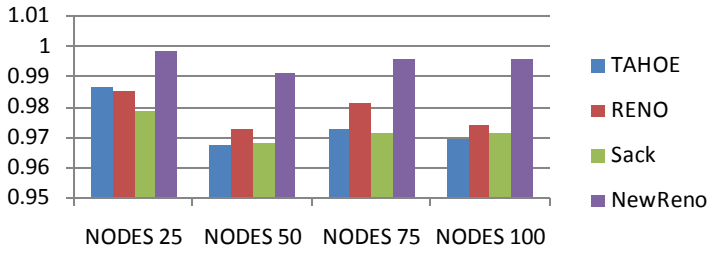
VII. SIMULATION RESULT.

No of nodes:	Tahoe			Reno			New Reno			Vegas			Sack		
	Delay (ms)	Thput (kbps)	PDF	Delay (ms)	Thput (kbps)	PDF	Delay (ms)	Thput (kbps)	PDF	Delay (ms)	Thput (kbps)	PDF	Delay (ms)	Thput (kbps)	PDF
25	490.6	464.9	98.11	459.4	495.98	98.29	459.4	495.98	98.2	78.82	442.06	99.84	502.1	467.46	98.00
50	597.33	318.58	97.98	551.118	310.56	97.71	630.764	308.35	97.63	81.6641	368.33	99.52	566.312	310.42	97.48
75	599.462	354.39	98.03	615.667	341.80	97.85	409.626	372.88	97.73	71.9072	350.24	99.50	475.061	360.31	98.28
100	453.997	411.95	97.83	502.035	377.83	97.62	423.962	459.56	97.79	65.5729	423.75	99.64	457.437	406.36	97.26

A.THROUGHPUT .



B. PACKET DELIVERY RATIO



VIII. CONCLUSION.

Traditional Linux based TCP New Reno does not differentiate issues of loss either through congestion or through bit error [6]. So it reduces congestion window in all cases and degrade the performance. So in this paper we use anovel approach for distinguish issue of segment loss with the help of flag indication of TCP header and continuous monitoring successive receiving segments. At sender end, we set congestion window (cwnd) as per issue of packet loss. Our analysis results shows better throughput compare to traditional TCP New Reno. Using this algorithm we get better performance over wireless link and easily deploy in Linux operation system for better Internet services and real time communication.

REFERENCES

[1] "Modified TCP NewReno for Wireless Networks"Ahmead and kabir Pranab Kumar Dhar, Mohammad Ibrahim Khan, ieee infocom pp 127 – 132 2010 .
[2] TCP NCE: A unified solution for non-congestion events to improve the performance of TCP over wireless networks" ieee-International Workshop on Information and Electronics Engineering (IWIEE)-2012
[3] Kureshi and huh an "Performance Improvement of TCP Reno Based on Monitoring the Wireless Packet Loss Rate" International Journal of Computer Applications Volume 45– No.5, May 2012
[4] Mukesh Kumar Dhariwal, Sanjeev Sharma, "An Improved Mechanism for Congestion Control in TCP for Ad Hoc Network", International Journal of Computer Applications, Volume 20– No.2, April 2011.
[5] Shilpi Gupta, Kamal Kumar Jyotiyana, "Analysis of Different Congestion Avoidance Algorithms", IRACST – International Journal of Computer Networks and Wireless Communications (IJCNWC), Vol.3, No1, February 2013.
[6] Minho Park, Youngsik Youn, "An Improvement of TCP Performance over Wireless Networks", 2013 IEEE.
[7] Yuhua Liu1, Hao Huang1, Kaihua Xu2, Chun Yang1, "Developing Slow Start over Wireless Networks", 2008 International Multi-symposiums on Computer and Computational Sciences.
[8] Kwan L. Yeung, Victor O.K. Li, "On Performance Modeling of TCP New-Reno", 2007 IEEE.
[9] Han Bing, FangYing-lan, Li Ye-bai, "Research and Improvement of Congestion Control Algorithms Based on

TCP Protocol", 2009 IEEE.
[10] S. Servati, H. Taheri and M. Nesary Moghaddam , "Performance Enhancement for TCP Vegas in Slow Start Phase over a Satellite link" , (2008) IEEE.
[11] http://www.isi.edu/nsnam/ns NS–2, Network Simulator, March 2005.
[12] http://C3lab.poliba.it/index.php/Westwood.

