International Journal of Advance Research in Science and Engineering
Vol. No.5, Issue No. 09 , September 2016
www.ijarse.com
IJARSE
ISSN (O) 2319 - 8354
ISSN (P) 2319 - 8346

AN ENHANCE APPROACH OF MOBILE TCP-NEW RENO USING DIFFERENTIATE VARIOUS PACKET LOSSES OVER WIRELESS-LINK

Prof. Bhoomika K. Sharma

Computer Engineering Department, Government Polytechnic, Ahmedabad, (India)

ABSTRACT

TCP is dominant transport protocol which used in internet application like telnet, WWW, FTP and email for data transmission. Tcp is connection oriented protocol. TCP provides a connection orientation, reliable data delivery and end to end mechanism.TCP have developed many variants for improve performance, fast retransmission and recovery of multiple losses packets.TCP variants are Tahoe, Reno, New Reno, Vegas, Sack and many more. In this paper, compare all the variants using simulator NS2.35.TCP New Reno shows better performance than other variants. TCP New Reno has higher throughput, high packet delivery ratio, less end to end delay compare to all the variants.

Keywords*: *TCP/IP, TCP New Reno, Throughput, Packet delivery ratio, End to end delay, 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. 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. Many TCP variants have been proposed for improve throughput of the TCP like TCP Reno, TCP New Reno, TCP SACK, TCP Tahoe, TCP Vegas[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.

IJARSE
ISSN (O) 2319 - 8354
ISSN (P) 2319 - 8346

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].

2.1. 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].

2.2. Congestion Avoidance

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

2.3. 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.

2.4. 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 fig. 1 show what typically data transfer phase using TCP congestion control might look like.

International Journal of Advance Research in Science and Engineering
Vol. No.5, Issue No. 09 , September 2016
www.ijarse.com
IJARSE
ISSN (O) 2319 - 8354
ISSN (P) 2319 - 8346

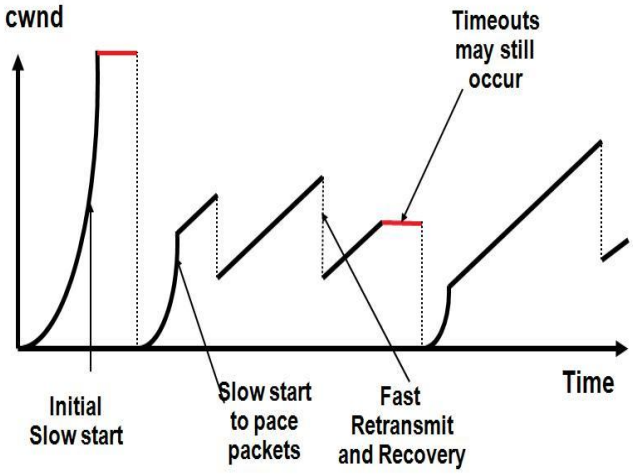

Fig1. Congestion Control Algorithm in TCP

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 Westwood [6].

3.1. TCP Reno

In 1990, TCP Reno has been developed and TCP Reno uses previous slow starts [7] and retransmits timer 2 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].

3.2. 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

IJARSE
ISSN (O) 2319 - 8354
ISSN (P) 2319 - 8346

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].

3.3. 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].

3.4. 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]

3.5. 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].

3.6. 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].

IJARSE
ISSN (O) 2319 - 8354
ISSN (P) 2319 - 8346

IV. PERFORMANCE ISSUE OF TRADITIONAL TCP New Reno

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 performs better 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.

V. PROPOSED ALGORITHM FOR MODIFIED TCP New Reno

In this paper we describe an algorithm for Modified TCP New Reno show in figure-2. Initially client enters into slow start phase of TCP. The traditional TCP New Reno cannot differentiate reason of segment loss through either congestion or bit error on wireless link [9]. Therefore in this paper we proposed new approach for finding reason of loss with the help of flag (F) of TCP Header. We use 1 bit flag of TCP Header for strong indication of congestion or bit error. We use F = 0 for congestion and F = 1 for bit error. This algorithm states if any segment loss due to any reasons during communication, receiver has been received out of order sequence numbers. When receiver receives out of order sequence number, it will start observing timing of next successive segments and continuous retrainsmits DUPACK. If next successive segments suffer from delay, then receiver assume that congestion occurs in the medium and segment loss due to congestion. So set FLAG = 0 in the 3^{rd} DUPACK (duplicate acknowledgment). If next successive segments does not suffer from delay and received continuously without delay, then receiver assume that congestion not occur in medium. So set FLAG = 1 in the 3^{rd} DUPACK. In this algorithm we use 3^{rd} DUPACK because 3^{rd} DUPACK is only indication of packet loss in TCP New Reno. In the TCP New Reno, when the ender receives 3 DUPACKs, it immediately reduces cwnd (congestion window) to half without knowledge of reason of loss through either congestion or bit error [6]. So it degrades the performance in real time communication. So in this paper we modified TCP New Reno protocol to improve the performance of TCP New Reno. In our case when sender receives 3 DUPACKs, it does not immediately reduce congestion window to half. But sender observes the FLAG status of 3^{rd} DUPACK. If F = 0 in 3^{rd} DUPACK, Sender assumes that packet loss due to congestion and set cwnd to half & enter into fast retransmit phase of TCP. If F = 1 in 3^{rd} DUPACK, Sender assumes that packet loss due to bit errorand not reduce cwnd & enter into congestion avoidance phase of TCP. So in this way In the TCP New Reno, when the ender receives 3 DUPACKs, it immediately reduces cwnd (congestion window) to half without knowledge of reason of loss through either congestion or bit error [6]. So it degrades the performance in real time communication. So in this paper we modified TCP New Reno protocol to improve the performance of TCP New Reno. In our case when sender receives 3 DUPACKs, it does

IJARSE
ISSN (O) 2319 - 8354
ISSN (P) 2319 - 8346

not immediately reduce congestion window to half. But sender observes the FLAG status of 3rd DUPACK. If F = 0 in 3rd DUPACK, Sender assumes that packet loss due to congestion and set cwnd to half & enter into fast retransmit phase of TCP. If F = 1 in 3rd DUPACK, Sender assumes that packet loss due to bit errorand not reduce cwnd & enter into congestion avoidance phase of TCP. So in this way we improve the performance and our results show difference performance of TCP New Reno and modified TCP New Reno.

Fig. 2 shows proposed detail algorithm for Modified TCP New Reno. It gives better performance and throughput compare to traditional Linux based TCP New Reno. In this algorithm of MTCP- New Reno, we use notation like cwnd = Congestion Window, F = Flag of TCP header, SEQ = Sequence Number, RX_SEG = Receiving Segment, DUPACK = Duplicate Acknowledgment. This modified algorithm support feature of traditional TCP New Reno like slow start, congestion avoidance, fast retransmit, and fast recovery phase.

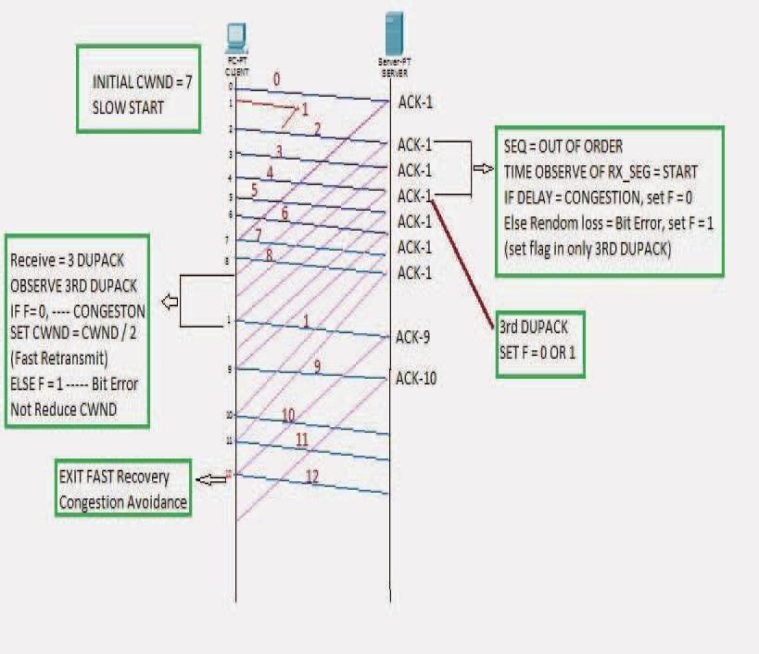

Figure2. Proposed Algorithm for Modified TCP New Reno.

As shown in fig.2, client initially set cwnd = 7, and at particular time t1 client sends frame-0 and server stores frame-0 inside buffer and send back ACK-1. The client sends next successive frames without waiting of acknowledgment because cwnd set to 7. The server receives frames in out of order because frame-1 loss due to any reason (congestion or bit error). The server receives first frame 2 instead of frame 1 so it starts observing time for next successive frames and send back DUPACKs of frame 1. If delay occurs in next successive frames, the server assumes that congestion occurs in the medium and set FLAG = 0 in 3rd DUPACK. When 3 consecutive DUPACK receives by client, it assumes that loss occurs inside medium and without immediately reducing of cwnd, first it checks the status of flag in TCP header of 3rd DUPACK. If FLAG = 0, client assumes that frame loss due to congestion and set cwnd to half, enters into fast retransmit phase of TCP. If FLAG = 1, client assumes that frame loss due to bit error and does not change status of cwnd, enter into congestion avoidance phase of TCP. So using modified TCP New Reno and simulation results of NS-2, we notice that improve throughput compare to traditional TCP New Reno.

International Journal of Advance Research in Science and Engineering
Vol. No.5, Issue No. 09 , September 2016
www.ijarse.com
IJARSE
ISSN (O) 2319 - 8354
ISSN (P) 2319 - 8346

VI. SIMULATION TOPOLOGY

In this paper we use NS-2.35 simulator for analysis our results with TCP New Reno [3]. For our algorithm we consider following topologies. We introduce error 0.001 and 0.01 in following scenarios over wireless link.

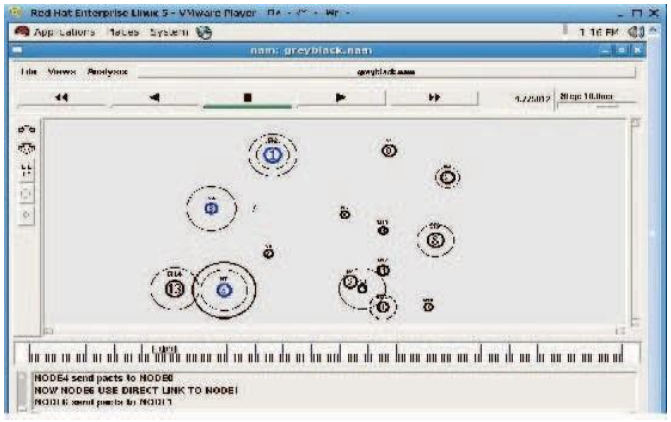

Figure3. Simulation Scenario - 1

As show in fig.3, in this topology we use FTP traffic of TCP and use 15 nodes over wireless link. First scenarios-1 show with congestion and as shown in figure-4, scenarios-2 shows same topology with the error over wireless link. We change error rate in scenario-2 and measure performance of Modified TCP New Reno.

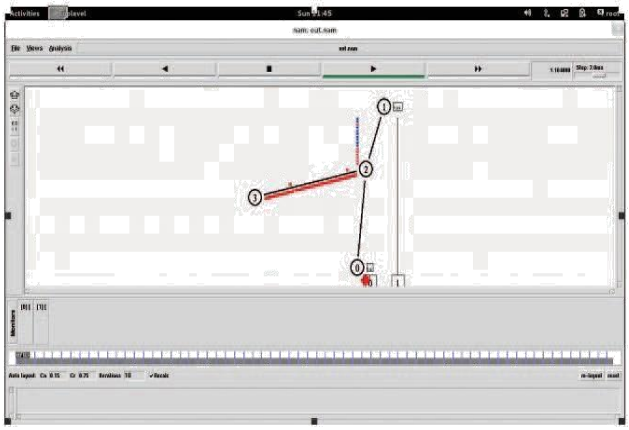

Figure4. Simulation Scenario – 2

As shown in fig.4, we also measured the performance of our algorithm on wired topology of NS-2 with using 4 nodes [3]. After this simulation we get better throughput compare to TCP New Reno. Figure-4 shows wired topology with using modified TCP New Reno with FTP traffic.

So finally our algorithm works better in both topologies like wired and wireless. Modified TCP New Reno can be easily implemented in open source operating system and increase the performance of real time communication.

International Journal of Advance Research in Science and Engineering
Vol. No.5, Issue No. 09 , September 2016
www.ijarse.com
IJARSE
ISSN (O) 2319 - 8354
ISSN (P) 2319 - 8346

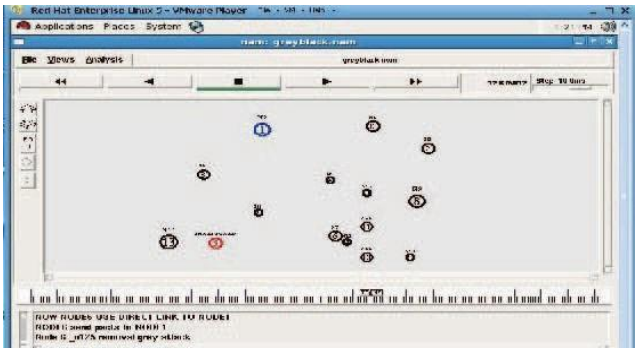

Figure5. Simulation Scenario - 3

VII. SIMULATED RESULTS:

We simulate our result with the help of NS-2.35 using XGRAPH utility [3]. Fig.6 shows the First result, its comparison between TCP New Reno (Green line) and Modified TCP New Reno (RED line) for throughput (Throughput Vs Time). Fig.6 shows the throughput performance of both protocols. Y-axis shows throughput (Mbps) and X-axis shows the time (sec). We run this simulation up to 60 sec. and gets better throughput modified TCP New Reno at error rate 0.001 with high congestion issue. At 0.001 error rate we receive 16048 segments using modified TCP New Reno, whereas using traditional TCP New Reno we receive 15097 segments successfully at receiver end. We use FTP traffic of TCP in both the simulation results. Our simulation results show that we get throughput upto 140 Mbps using modified TCPW whereas using traditional TCP New Reno gives upto 110 Mbps data rate after completed 60 sec simulation in NS-2.35 [11].

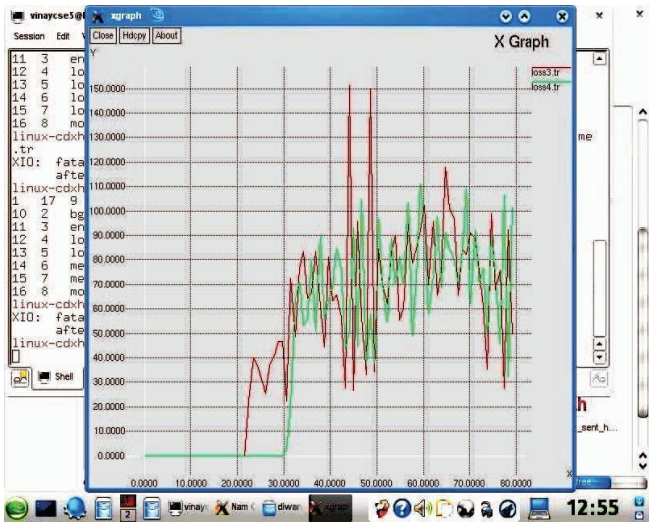

Figure6. Throughput Vs Time

Fig.7 shows results at varied bit error rate, result A gives throughput at 0.00 error rate, result B at 0.001 error rate and result C at 0.01 error rate using modified TCP New Reno. So conclusion of our results is that we get better through compare to TCP New Reno during bit error rate.

International Journal of Advance Research in Science and Engineering
Vol. No.5, Issue No. 09 , September 2016
www.ijarse.com
IJARSE
ISSN (O) 2319 - 8354
ISSN (P) 2319 - 8346

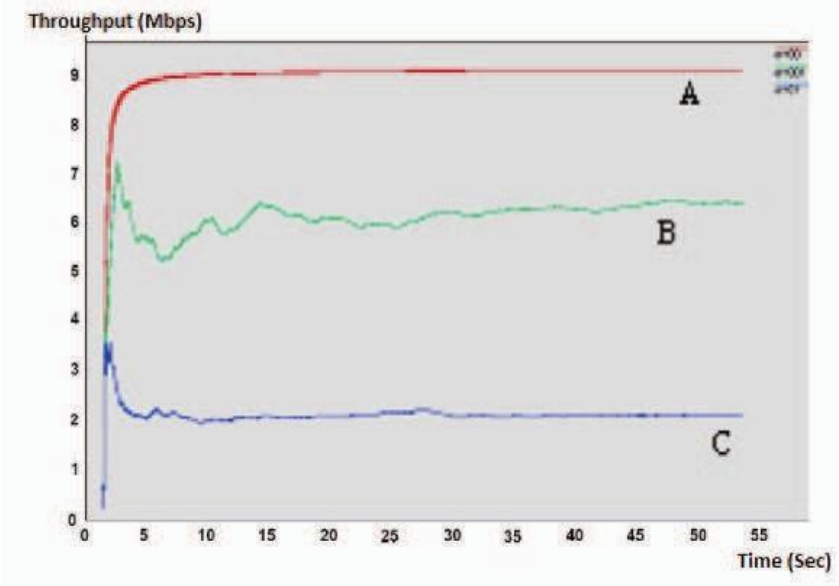

Figure7. Throughput Vs Time (at varied bit error rate)

Fig.8 shows the result of congestion window v/s time. In this case we notice that we get maximum throughput at high congestion window up to 100 compare to traditional TCP New Reno at 90 congestion window during 10 sec simulation on NS-2 [3]. Green line is indication of Modified TCP New Reno and Red line is indication of traditional TCP New Reno.

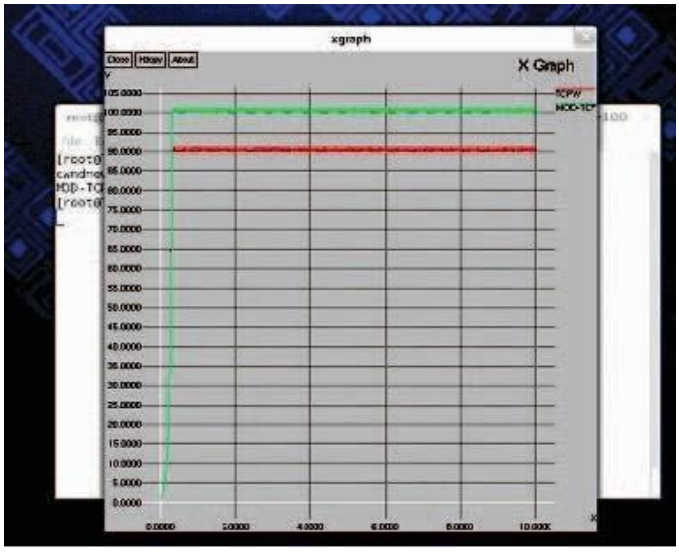

Figure8. CWND Vs Time

IJARSE
ISSN (O) 2319 - 8354
ISSN (P) 2319 - 8346

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 a novel 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] Ahmed Khurshid; Md Humayun kabir; Rajkumar Das,"Modified TCP New Reno for wireless network," in Computer and Information Technology (ICCIT), IEEE on , vol., no., pp.425-427, April 2015

[2] Dhar, P.K.; Khan, M.I.; Deb, K.; Hassan, P.M.M., "A modified New Reno for performance enhancement of TCP in wireless network," in Strategic Technology (IFOST), 2010 International Forum on , vol., no., pp.450-453, 13-15 Oct. 2010

[3] Kowsar, M.M.S.; Islam, M., "TCP performance enhancement over IEEE 802.11," in Computer and Information Technology (ICCIT), 2012 15th International Conference on , vol., no., pp.326-331, 22-24 Dec. 2012

[4] Khurshid, A.; Kabir, M.H.; Prodhan, M.A.T., "An improved TCP congestion control algorithm for wireless networks," in Communications, Computers and Signal Processing (PacRim), 2011 IEEE Pacific Rim Conference on , vol., no., pp.382-387, 23-26 Aug. 2011

[5] Krishnan, S.B.; Moh, M.; Teng-Sheng Moh, "Enhancing TCP Performance in Hybrid Networks with Fixed Senders and Mobile Receivers," in Global Telecommunications Conference, 2007. GLOBECOM '07. IEEE , vol., no., pp.5265-5270, 26-30 Nov. 007

[6] Luo Yongmei; Jin ZhiGang; Zhao Ximan, "A New Protocol to Improve Wireless TCP Performance and Its Implementation," in Wireless Communications, Networking and Mobile Computing, 2009. WiCom '09. 5th International Conference on , vol., no., pp.1-4, 24-26 Sept. 2009

[7] Prasanthi Sreekumari and Sang-Hwa Chung, **"**TCP NCE: A unified solution for non-congestion events to improve the performance of TCP over wireless networks **",** EURASIP Journal on Wireless Communications and Networking, 2011

[8] Prasanthi, S.; Sang-Hwa Chung, "An Efficient Algorithm for the Performance of TCP over Multi-hop Wireless Mesh Networks," in Information Technology: New Generations (ITNG), 2010 Seventh International Conference on , vol., no., pp.816-821, 12-14 April 2010

[9] Dr Neeraj Bhargava, Dr Ritu Bhargava, Manisha mathuriya and Shilpi Gupta, "Analysis of different congestion avoidance algorithms" in IJCNWC journal (2250-3501) volume 3-No.1, pp.4-5, February 2013

[10] RFC: 2001, "TCP Slow Start, Congestion Avoidance, Fast Retransmit, and Fast Recovery Algorithms",January 1997

[11] RFC: 1323, "TCP Extension for High Performance", 1992