Analysis Of TCP WestwoodNR Protocol in Congested and Lossy Network

Amit M Sheth[#1], Kaushika D Patel[*2], Jitendra P Chaudhari[#3], Jagdish M Rathod[*4]

[#]*Communication System Engineering, Charusat University*
At & Po:Changa-388421, Dist-Anand,India

[*]*Birla Vishwakarma Mahavidhyalaya Engineering College*
Vallabh Vidhyanagar-388120, Dist-Anand, India

Abstract: **We study the performance of TCP WestwoodNR (TCP-WNR), a TCP protocol controls the window using end-to-end bandwidth estimation. This Bandwidth Estimation continuously estimates, at the TCP sender, the packet rate of the connection by monitoring the ACK reception rate. The estimated connection rate is then used to compute congestion window and slow start threshold settings after a packet loss occurred. Resetting the window to match available bandwidth makes TCP-WNR more robust to Random loss as well as in Congestion. Thus Rather than to react unnecessary window reduction after every packet lost, TCP-WNR uses this bandwidth estimation to compute congestion window and slow start threshold. These often cause conventional TCP to overreact, leading to unnecessary window reduction. Experimental studies of TCP-WNR show significant improvements in throughput performance over Reno and SACK, particularly in wired networks. Performance Results are shown that TCP-WNR is the best TCP protocol for link errors as well as congested networks. Performance results also shown that with High Error Rate Environment, TCP-WNR gives the highest throughput among all other TCPs. Analytic results are validated against simulation results.**

Keywords: **ssthresh- slow start threshold,cwnd-congestion window, Congestion Avoidance, TCP WestwoodNR, Bandwidth Estimated, Random Loss (Link Error)**

I. INTRODUCTION

TCP is a connection-oriented, end-to-end reliable protocol designed to fit into a layered hierarchy of protocols which support multi-network applications. The TCP provides for reliable inter-process communication between pairs of processes in host computers attached to distinct but interconnected computer communication networks. Very few assumptions are made as to the reliability of the communication protocols below the TCP layer. TCP assumes it can obtain a simple, potentially unreliable datagram service from the lower level protocols. In principle, the TCP should be able to operate above a wide spectrum of communication systems ranging from hard-wired connections to packet-switched or circuit-switched networks [21].

The primary purpose of the TCP is to provide reliable, securable logical circuit or connection service between pairs of processes. To provide this service on top of a less reliable internet communication system requires basic TCP facilities in the following areas:

- Basic Data Transfer
- Reliability
- Flow Control
- Multiplexing
- Connection
- Precedence and Security

II. TCP VARIANTS

TCP congestion control involves slow start and congestion avoidance phases. In order to improve the performance, several mitigation techniques have been suggested over standard TCP versions like NewReno and SACK TCP. The proactive schemes like, TCP Westwood and TCP WestwoodNR intend to improve flow control and avoid packet losses from estimation of certain network parameters. By improving the basic TCP Tahoe, Other versions Of TCPs are invented. Tahoe TCP consist of slow start, congestion avoidance and fast retransmission algorithms. But the problem with TCP Tahoe is that every time a packet is lost it waits for a timeout. TCP Reno adds “fast recovery” to the Tahoe TCP as additional feature. When a first packet lost is happened, it cuts its cwnd by half. But the problem with TCP Reno is in a single window whenever there is a multiple packet loss, it behaves same like TCP Tahoe. TCP NewReno is a modification made in TCP Reno, where TCP sender retransmit the packet either on reception of three dupacks or expiration of retransmission timer. In New-Reno, partial acks do not take TCP out of Fast Recovery. Instead, partial acks received during Fast Recovery are treated as an indication that the packet immediately following the acknowledged packet in the sequence space has been lost, and should be retransmitted. Thus, when multiple packets are lost from a single window of data, New-Reno can recover without a retransmission timeout, New-Reno remains in Fast Recovery until all of the data outstanding when Fast Recovery was initiated will get acknowledged. But the problem with TCP Newreno is that when large amount of packets dropped from the window of data, TCP data send retransmit time will ultimate expire. TCP Sack option follows the format in the SACK option field contains a number of SACK blocks, where each SACK block reports a non-contiguous set of data that has been received and queued. But the problem with TCP Sack is that currently selective acknowledgments are not provided by the receiver.

TCP Westwood & TCP WestwoodNR introduces ”faster” recovery to avoid over-shrinking cwnd after three duplicate ACKs by taking into account the end-to- end estimation of the bandwidth available to TCP.TCP Westwood uses the badwidth estimate to set the cwnd & ssthresh after a congestion episode. Also it uses the same features with TCP Reno. But the problem with TCP Westwood is that it omits the router's buffer size. Therefore, modifications done to implement TCP WestwoodNR are comparable to the ones implemented in the transition from TCP Reno to TCP Newreno. TCP-WNR was aimed to improve performance under random or sporadic losses. This version was tested through simulation and showed considerable gain in terms of throughput in almost all scenarios.

III. OVERVIEW OF TCP WestwoodNR

In TCP WestwoodNR the sender continuously computes the connection Bandwidth Estimate (BWE) which is defined as the share of bottleneck bandwidth used by the connection. Thus, BWE is equal to the rate at which data is delivered to the TCP receiver. The estimate is based on the rate at which ACKs are received and on their payload. After a packet loss indication, (i.e, reception of 3 duplicate ACKs, or timeout expiration). The sender resets the congestion window and the slow start threshold based on BWE. More precisely, cwin=BWE x RTT.

To understand the rationale of TCP-WNR, note that BWE varies from flow to flow sharing the same bottleneck; it corresponds to the rate actually achieved by each INDIVIDUAL flow. Thus, it is a FEASIBLE (i.e. achievable) rate by definition. Consequently, the collection of all the BWE rates, as estimated by the connections sharing the same bottleneck, is a FEASIBLE set. When the bottleneck becomes saturated and packets are dropped, TCP-WNR selects a set of congestion windows that correspond exactly to the measured BWE rates and thus reproduce the current individual throughputs. The solution is feasible, but it is not guaranteed per se to be “fair share.” An additional property of this scheme, described in Section III, drives the rates to the same equilibrium point and makes it “fair share” under uniform propagation delays and single bottleneck assumptions.

Another important element of this procedure is the RTT estimation. RTT is required to compute the window that supports the estimated rate BWE. Ideally, the RTT should be measured when the bottleneck is empty. In practice, it is set equal to the overall minimum round trip delay (RTTmin) measured so far on that connection (based on continuous monitoring of ACK RTTs)[15].

A. Setting cwin and ssthresh in TCP-WNR

Further details regarding bandwidth estimation are provided in following sections. For now, let us assume that a sender has determined the connection bandwidth estimate (BWE), and let us describe in this section how BWE is used to properly set cwin and ssthresh after a packet loss indication.

First, we note that in TCP-WNR, congestion window increments during slow start and congestion avoidance remain the same as in Reno, , that is they are exponential and linear, respectively. A packet loss is indicated by (a) the reception of 3 DUPACKs, or (b) a coarse timeout expiration. In case the loss indication is 3 DUPACKs, TCP-WNR sets cwin and ssthresh as follows:

```
if (3 DUPACKs are received)
ssthresh = (BWE * RTTmin) / seg_size;
if (cwin > ssthresh) /* congestion avoid. */
cwin = ssthresh;
endif
endif
```

In the pseudo-code, seg_size identifies the length of a TCP segment in bits. Note that the reception of n DUPACKs is followed by the retransmission of the missing segment, as in the standard Fast Retransmit implemented by TCP Reno. Also, the window growth after the cwin is reset to ssthresh follows the rules established in the Fast Retransmit algorithm (i.e. cwin grows by one for each further ACK, and is reset to ssthresh after the first ACK acknowledging new data). During the congestion avoidance phase we are probing for extra available bandwidth. Therefore, when n DUPACKs are received, it means that we have hit the network capacity (or that, in the case of wireless links, one or more segments were dropped due to sporadic losses). Thus, the slow start threshold is set equal to the window capable of producing the measured rate BWE when the bottleneck buffer is empty (namely, BWE*RTTmin). The congestion window is set equal to the ssthresh and the congestion avoidance phase is entered again to gently probe for new available bandwidth. Note that after ssthresh has been set, the congestion window is set equal to the slow start threshold only if cwin > ssthresh. It is possible that the current cwin may be below threshold. This occurs after time-out for example, when the window is dropped to 1 as discussed in the following section. During slow start, cwin still features an exponential increase as in the current implementation of TCP Reno[15].

In case a packet loss is indicated by timeout expiration, cwin and ssthresh are set as follows:

```
if (coarse timeout expires)
cwin = 1;
ssthresh = (BWE * RTTmin) / seg_size;
if (ssthresh < 2)
ssthresh = 2;
endif;
endif
```

B. Bandwidth Estimation

The TCP-WNR sender uses ACKs to estimate BWE. More precisely, the sender uses the following information: (1) the ACK arrival times and, (2) the increment of data delivered to the destination. Let assume that an ACK is received at the source at time tk, notifying that dk bytes have been received at the TCP receiver. We can measure the sample bandwidth used by that connection as $b_k=d_k/(t_k–t_{k-1})$, where t_{k-1} is the time the previous ACK was received. Letting $\Delta t_k=t_k–t_{k-1}$, then

$b_k=d_k/\Delta t_k$. Since congestion occurs whenever the low-frequency input traffic rate exceeds the link capacity, we employ a low pass filter to average sampled measurements and to obtain the low-frequency components of the available bandwidth. More precisely, we use the following discrete approximation of the low pass filter due to Tustin.

Let b_k be the bandwidth sample, and $\hat{b}_k$ the filtered continuous first order low-pass filter using the Tustin estimate of the bandwidth at time t_k. Let α_k be the time-varying exponential filter coefficient at t_k. The TCP-WNR filter is then given by

$$\hat{b}_k = \alpha_k * \hat{b}_{k-1} + (1-\alpha_k) * (\frac{b_k + b_{k-1}}{2})$$

Where

$$\alpha_k = \frac{2\tau - \Delta t_k}{2\tau + \Delta t_k}$$

And $1/\tau$ is the filter cut-off frequency.

Notice the coefficients α_k depend on Δt_k to properly reflect the variable inter-arrival times.[15]

A number of considerations must be taken into account while interpreting the information that a returning ACK carries regarding delivery of segments to the destination. Two of these considerations are:

1. An ACK i received by the source implies that a transmitted packet was delivered to the destination. A DUPACK also implies that a transmitted packet was delivered, triggering the transmission by the receiver of the DUPACK. Thus, DUPACKs are considered in estimating bandwidth.

2. TCP ACKS can be "delayed," i.e., receivers wait for a second packet before sending an ACK, until 200 ms elapse in which case an ACK is sent without waiting. Delayed ACKs are also accounted for by our scheme.

These items are included in our implementation of TCP-WNR under Linux[15].

IV. PERFORMANCE ANALYSIS OF TCP WestwoodNR

In this section a set of results of performance comparison between TCP WestwoodNR and TCPs Reno, Sack and Westwood. In order to be aware of the perturbations and interactions of TCP WestwoodNR, we analyse the impact on Throughput by Two factors (error rate, bottleneck bandwidth).Throughput is a common metric of TCP Performance. All simulations presented in this paper were run using the Network Simulator version 2.35.

A. Impact of Error Rate on Throughput

We create a Link to Link with one source, one router and one destination. We give Duplex Link between source to router and router to destination with 5mb of Bandwidth and 0 second of Delay. Define TCP Agent like Reno, SACK, WestwoodNR with source and give the flow between source to destination. Start All simulation at 0.2 second and stop simulation at 50 second. Total Simulation Time is also 50 second. Following Existing Topology is as under drawn

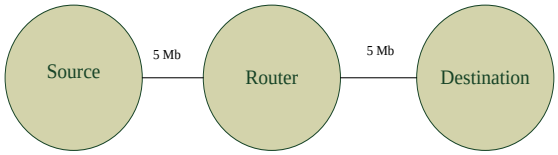

Fig.1 End to End Connection Network

We Analyzed the Throughput of Different TCP Agent like Reno, SACK, WestwoodNR with different Error Rates like 0.001, 0.01, and 0.1 and we see that with Increasing Error Rate, The Throughput Of TCP WestwoodNR increases compare to other TCP Agent.

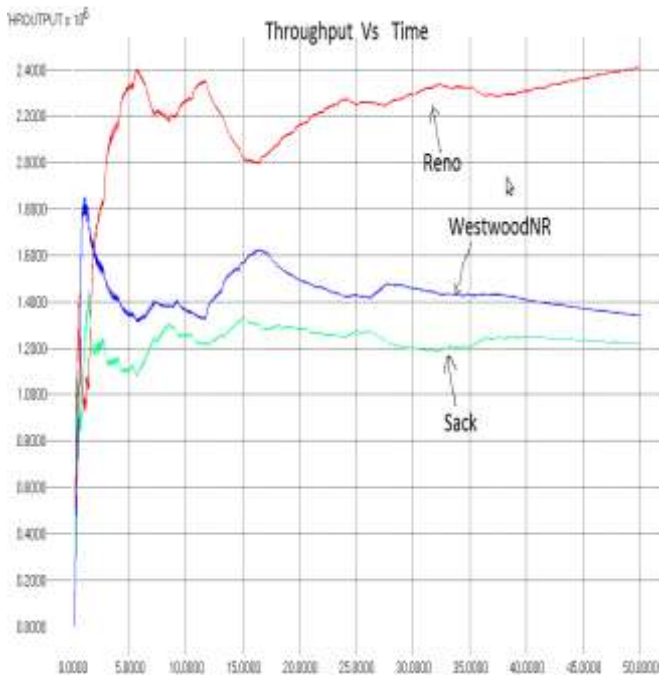

Fig.1.1(a) Throughput Of TCPs with 0.001 Error Rate

As From Fig. 1.1(a),(b),(c) and From Table-I we can see from the simulation that with small Error Rate of 0.001, The Throughput Of Reno much Higher than Other TCPs. Like wise As we increase the Error Rate with 0.01 and then much increase as 0.1 We can clearly see that Our Approach towards throughput of TCP WestwoodNR is also increase with increasing the Error Rate.

Thus it is clearly seen that with Random Drops in Topology of the Network, TCP WestwoodNR is more useful protocol to use in it. Thus Throughput Of TCP WestwoodNR compare with other TCPs is increasing with increasing Error Rate.

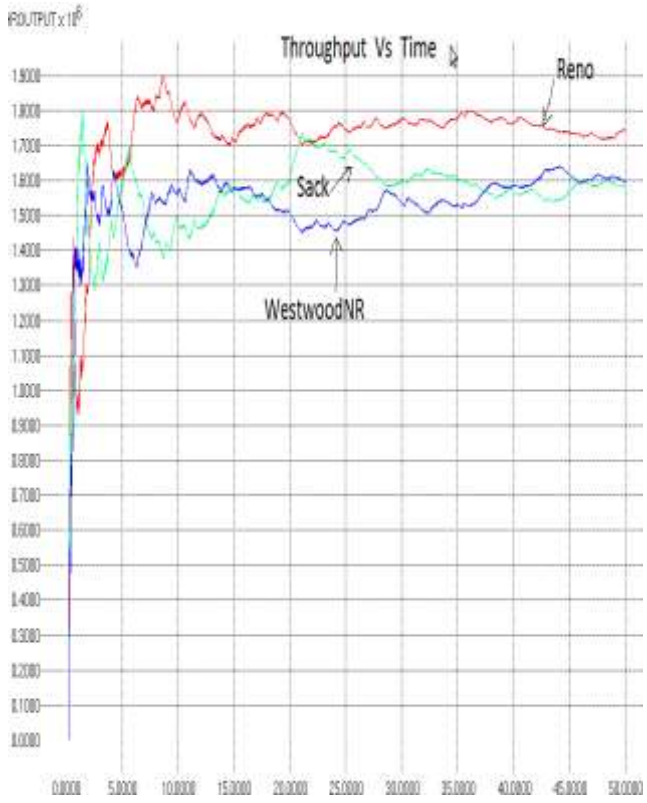

Fig.1.1(b) Throughput Of TCPs with 0.01 Error Rate

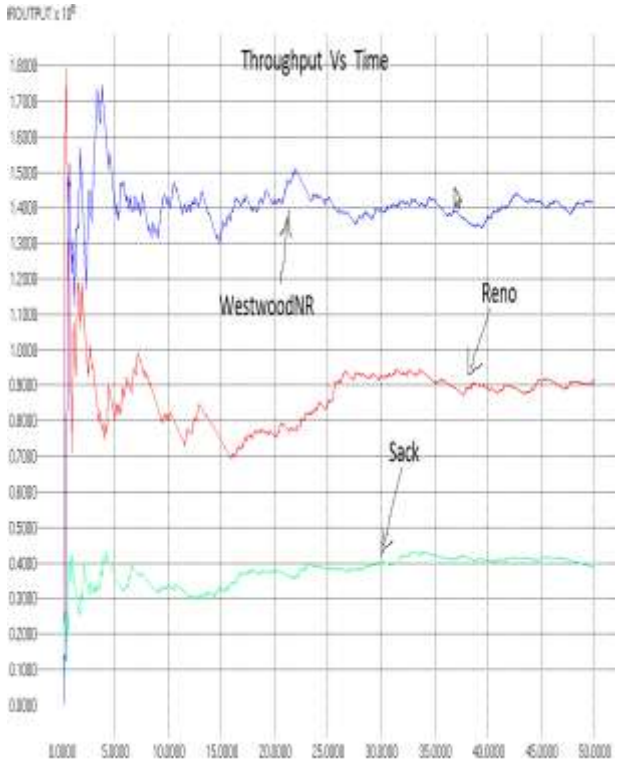

Fig.1.1(c) Throughput Of TCPs with 0.1 Error Rate

Let us we conclude this result in Table:

Table-I

Error Rate	Average Throughput Of Reno in Mbps	Average Throughput Of Sack in Mbps	Average Throughput Of WestwoodNR in Mbps
0.001	2.4	1.22	1.37
0.01	1.75	1.59	1.60
0.1	0.9	0.4	1.41

B. Impact of Bottleneck Bandwidth on Throughput

We first create three sources like source-1, source-2 and source-3. Then we connect these sources with one router and finally it connects to the destination. We give Duplex Link between source-1 to router, source-2 to router, and source-3 to router with 5mb of Bandwidth and 0 second of Delay. Then we give Duplex Link between router to destination with 2mb of Bandwidth and 0 second of Delay. Thus clearly we have given 2mb of Bottleneck in this Topology. Define TCP Agent/Sack1 with source-1 and give the flow between source-1 to destination. Define TCP Agent/Westwood with source-2 and give the flow between source-2 to destination. Define TCP Agent/WestwoodNR with source-3 and give the flow between source-3 to destination. Start All simulation at 0.2 second and stop simulation at 50 second. Total Simulation Time is also 50 second. Following Basic Topology is as under drawn.

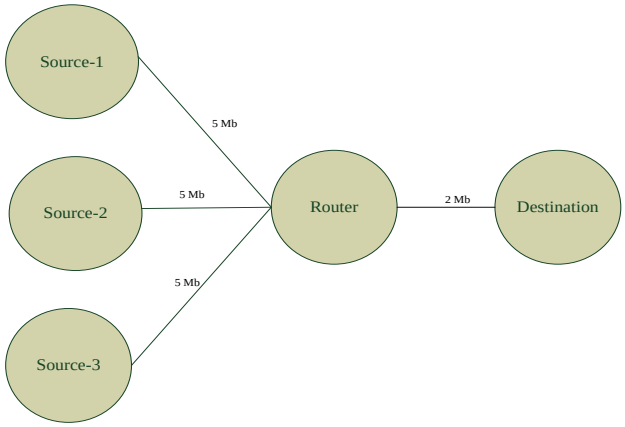

Fig.2 Bottleneck Network

We Analyzed the Throughput of Different TCP Agent like SACK, Westwood, and WestwoodNR with Bottleneck Bandwidth of 2 Mbps.

Let us Analyzed From Firg.2.1(a) and (b) We have clearly seen that with a Bottleneck Bandwidth Of 2 Mpbs and without Error Rate given TCP WestwoodNR has the Highest Average Throughput of 0.72 Mbps, TCP Westwood has Average Throughput of 0.68 Mpbs and TCP Sack has lowest Average Throughput of 0.58 Mbps. Also from Fig 2.1(b) We have clearly seen that As We increase the Error Rate in Bottleneck TCP WestwoodNR has much better Average Throughput compare with some other TCPs.

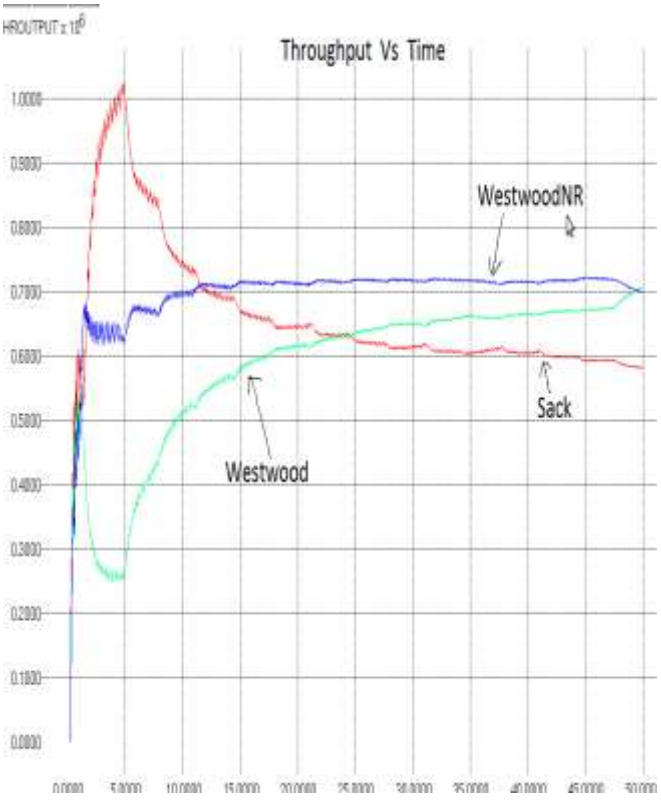

Fig.2.1 (a) Throughput Of TCPs with Bottleneck Bandwidth without Error Rate

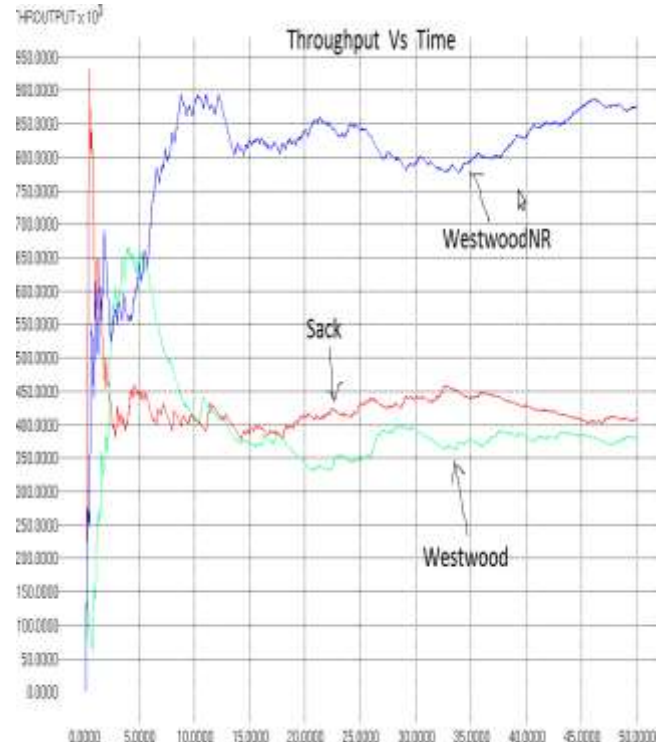

Fig.2.1(b) Throughput Of TCPs with Bottleneck Bandwidth with 0.1 Error Rate

C. Impact Of Bottleneck Bandwidth on Throughput with UDP Traffic

We first create a two sources like source-1 and source-2. Then we connect these sources with one router and finally it connects to the Destination. We give Duplex Link between source-1 to router with 1mb of Bandwidth and 0 second of Delay and source-2 to router with 10mb of Bandwidth and 0 second of Delay. Then we give Duplex Link between router to destination with 5mb of Bandwidth and 0 second of Delay. Thus clearly we have given 5mb of Bottleneck in this Topology. Define UDP Agent with source-1 and give the flow between source-1 to destination. Define TCP Agent/Reno first and then Define TCP Agent/WestwoodNR with source-2 and give the flow between source-1 to destination. Start all simulation at 0.3 second and stop simulation at 50 second. Total Simulation Time is also 50 second. Following Basic Topology is as under drawn:

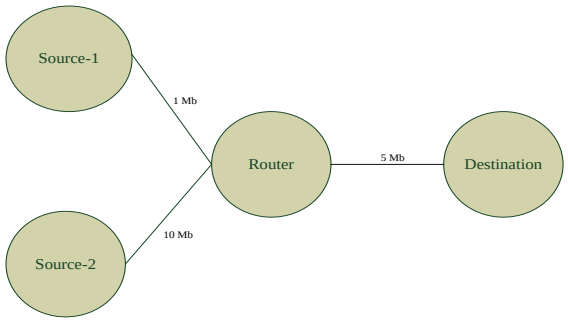

Fig.3 Bottleneck with UDP Traffic Network

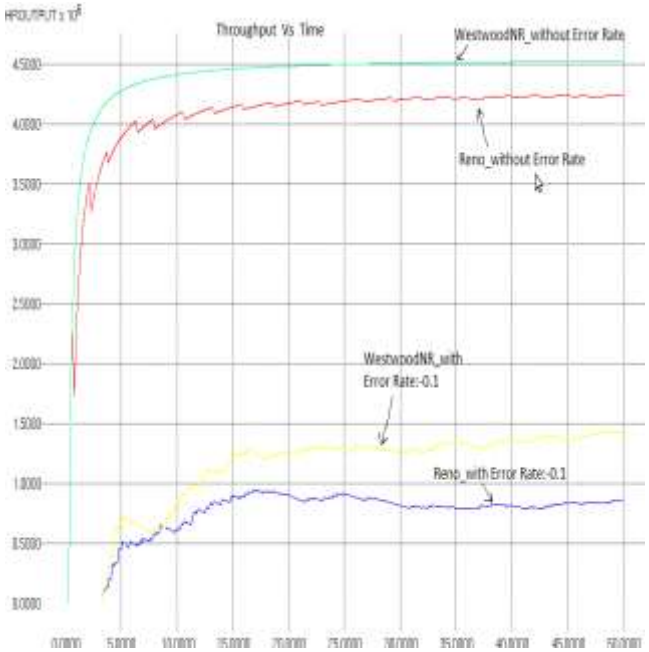

Fig.3.1 Throughput Of TCPs with UDP Traffic

- *Bottleneck Link With UDP Traffic Without Error Rate*

First We analyzed of TCP Reno with UDP Traffic without given any Error Rate. From Fig 4.1 we have seen that the Average Throughput of Reno in this simulation is around 4.2 mbps.

Then we analyzed of TCP WestwoodNR with UDP Traffic without given any Error Rate. From Fig 4.1 we have seen that the Average Throughput of WestwoodNR in this simulation is around 4.5 mbps.

- *Bottleneck Link With UDP Traffic With 0.1 Error Rate*

Here we analyzed of TCP Reno with UDP Traffic with given Error Rate of 0.1. From Fig 4.1 we have seen that the Average Throughput of Reno in this simulation is around 0.8 mbps.

Then we analyzed of TCP WestwoodNR with UDP Traffic with given Error Rate of 0.1. From Fig 4.1 we have seen that the Average Throughput of WestwoodNR in this simulation is around 1.42 mbps.

Thus we can say that TCP WestwoodNR has much better Throughput with Random Drops Error. So it is very useful Protocol in Random Drops Error in Network.

V. CONCLUSION

TCP WestwoodNR estimates bandwidth and adjusts the cwnd and ssthreh after loss detection. It sets bandwidth to the measured rate currently experienced by the connection, rather than using the conventional MD scheme.TCP WestwoodNR (TCP-WNR) differs from Reno in that it adjusts the congestion window after a loss detection by setting it to the measured rate currently experienced by the connection, rather than using the conventional multiplicative decrease scheme (i.e., divide the current window by half). Most important, it can handle losses caused by link errors than TCP Reno. Moreover, if TCP-WNR and Reno coexist on a bottleneck with error induced losses with UDP Traffic, TCP-WNR outperforms Reno mainly because it can make better use of the channel, while "stealing" only a modest fraction of throughput from Reno. TCP-WNR has been implemented in LINUX and has been tested extensively in a NS-2. The performance measured in the NS-2 and it confirms the simulation results. In particular, It confirms that whenever there is given high error rate, TCP WestwoodNR has highest Average Throughput among all other TCPs.

ACKNOWLEDGMENT

The author is thankful to Dr. Niraj Shah, and Prof. Brijesh Shah, for their support and encouragement during the research endeavour. We would like to thank V. T. Patel Department of Electronics and Communication, CHARUSAT University, India, for cooperation in the research work.

REFERENCES

[1] Geethu Wilson, Robin Cyriac "*An Enhancement to TCPW BBE by Modifying the Bandwidth Estimation Using Modifieed EWMA*" International Journal Of Advanced Research in Computer Science and Software Engineering, June-2012

[2] Shimaa Hagag, Ayman EI-Sayed(IEEE Senior Member)"*Enhanced TCP Westwood Congestion Avoidance Mechanism(TCP WestwoodNew)"* International Journal Of Computer Application, May-2012

[3] Kau Lan, Niu Sha "*A CMT Congestion Window Updates Mechanism Based on TCP Westwood*" International Conference on Mechatronic Science, Electric Engineering and Computer, August-2011

[4] Neng-Chung Wang, Jong-Shin Chen, Yung-Fa Huang, Chi-LunChiou "*Performance Enhancement Of TCP in Dynamic Bandwidth Wired and Wireless Network*" Wireless Pers Commun, Springer Science+Business Media, sMarch-2008

[5] Kenshin Yamada, Ren Wang, M. Y. Sanadidi, Mario Gerla "*TCP With Sender-Side Intelligence to Handle Dynamic, Large, Leaky Pipes*" IEEE Communication Society, Feb-2005

[6] McCanne, S. and Floyd, S. *NS Network Simulator Version 2.35*,2005

[7] Kazumi Kaneko, Jiro Katto "*Reno Friendly TCP Westwood Based On Router Buffer Estimation*"International Conference on Autonomic and Autonomous System and International Conference on Networking and Services, IEEE Computer Society, 2005

[8] Kenshin Yamada, Ren Wang, M. Y. Sanadidi, Mario Gerla "*TCP Westwood with Agile Probing: Dealing with Dynamic, Large, Leaky Pipes*" IEEE Communication Society, 2004

[9] S. Floyd, T.Henderson, A.Gurtov "*The New Reno Modification to TCP's Fast Recovery Algorithm,RFC-3582*" Networking Working Group,April-2004

[10] S. Floyd, T. Henderson, A. Gurtov, Y. Nishida "*The New Reno Modification to TCP's Fast Recovery Algorithm,RFC-6582*" Internet Engineering Task Force, 2004

[11] M. Gerla, B.K.F.Ng, M.Y.Sanadidi, M.Valla,R.Wang "*TCP Westwood with adaptive bandwidth estimation to improve efficiency/friendliness tradeoffs*" UCLA Computer Science Department, Los Angeles, CA 90095, USA,Feb-2003

[12] Claudio Casetti, Mario Gerla, Saverio Mascolo, M.Y.Sanadidi,Ren Wang "*TCP Westwood: End-to-End Congestion Control For Wired/Wireless Networks*" Kluwer Academic Publisher, 2002.

[13] Ren Wang, Massimo Valla, M.Y.Sanadidi, Mario Gerla "*Adaptive Bandwidth Share Estimation in TCP Westwood*" UCLA Computer Science Department, Los Angeles, CA 90005, USA.

[14] S.Mascolo, C.Casetti, M.Gerla, S.S. Lee, M.Sanadidi "*TCP Westwood: Congestion control with faster recovery*"

[15] Mario Gerla, M.Y.Sanadidi, Ren Wang, Andrea Zanella "*TCP Westwood: Congestion Window Control Using Bandwidth Estimation*"

[16] Kevin Fall and Sally Floyd "*Simulation-based Comparisons of Tahoe, Reno, and SACK TCP*"

[17] M.Allman, V.Paxson "*TCP Congestion Control, RFC-2581*" Network Working Group, April-1999

[18] S.Floyd, T.Henderson "*The New Reno Modification to TCP's Fast Recovery Algorithm,RFC-2582*" Networking Working Group, April-1999

[19] M.Mathis, J.Mahdavi "*TCP Selective Acknowledgment Options, RFC-2018*" Network Working Group, October-1996

[20] V.Jacobson, R.Braden "*TCP Extensions For Long-Delay Paths, RFC-1072*" Network Working Group, 1988

[21] Marina Del Ray "*Transmission Control Protocol, RFC-793*" Defence Advanced Research Projects Agency, Arlington, Virginia, Sept-1981.