A New Variant of TCP for Heterogeneous Networks

Moumita Deb, Soumya Sarkar, and Abhijit Bagchi

***Abstract*—TCP being the most widely used routing protocol for wired network considers packet loss as an indicator of congestion and calculates its congestion window according to that, but this approach is not suitable for wireless network where packet loss occurs due to various reasons other than congestion. Taking into consideration heterogeneous network, in this paper we explore a new variant of TCP, which has only sender side modification, end-to-end reliability and dynamic window calculation technique, which gives better result compared to existing known variants of TCP. We call it TCP Rcc. Here, we have used fixed window concept because it proves to produce better result.**

***Index Terms*—Adaptive window calculation, bandwidth estimation, random *waiting* time, throughput comparison.**

I. Introduction

TCP is a transport layer protocol used by applications that require guaranteed delivery. It is a sliding window protocol which provides both timeouts and retransmissions. TCP establishes a full duplex, virtual connection between two endpoints. Each endpoint is defined by an IP address and a TCP port number. TCP provides a communication service at an intermediate level -between an application program and the Internet Protocol (IP). That is, when an application program desires to send a large chunk of data across the Internet using IP, instead of breaking the data into IP-sized pieces and issuing a series of IP requests, the software can issue a single request to TCP and let TCP handle the IP packaging. The byte stream is transferred in segments. The window size determines the number of bytes of data that can be sent before an acknowledgement from the receiver is received. TCP works particularly well in links with low error rates. However situation is different in wireless links which are characterized by high bit error rate, packet corruption or link failure [1]. So, in wired links most packet losses occur due to congestion, but in wireless links packet loss may not always occur due congestion only. Older variants like TCP Reno or TCP NewReno[2]cannot distinguish between congestion loss and link loss. So blindly reduce congestion window size when there is a packet loss. Newer variants of TCP try to solve this problem by finding out techniques to differentiate between packet loss owing to congestion and due to link errors [3]. In conventional congestion control schemes after a packet loss is detected the window is halved or reinitialized to 1MSS (Maximum Segment Size) and enter slow start again. This lead to unutilized bandwidth and reduced throughput.

Manuscript received July 20, 2012; revised September 21, 2012.

The authors are with the Department of Information Technology,RCC Institute of Information Technology Kolkata (e-mail: moudeb@gmail.com, portkey1996@gmail.com, abhijit070590@gmail.com).

A fixed window approach [4] produces better throughput and goodput compared to "blind" increase/decrease in window sizes which is followed in TCP Reno.

In this paper we have suggested a variant of TCP keeping focus on optimum bandwidth estimation and window calculation. The goal is to explore possibility of a new approach by mixing best approaches of different available techniques.Constant window approach [5] is followed here as long as there is no packet loss. Once a packet loss is detected via timeout or any duplicate acknowledgement, bandwidth is recalculated once again for the network and window is reinitialized to continue with the data transfer. As most TCP transmissions are short lived this scheme provides considerable benefit in throughput compared to other techniques. It is not intended to radically change the present TCP. The technique proposed, adds some modules in the sender side and the receiver side keeping the existing techniques of congestion detection like timeout or duplicate packets unchanged. Extensive analysis indicates that compared to other techniques like link layer methods, split connection based techniques which either leads to poor end to end throughput due to shielding of the wireless from the wired section of the network or leads to expensive changes in the intermediate nodes; end-to-end solutions require changes only to the sender and the receiver, and is the best method for providing congestion control in TCP which has been utilized in our variant.

This paper is organized as follows. In Section 2,provides a brief description of the Related works done by other people.TCP Rcc(the new variant) is introduced in Section 3 while the simulation and implementation details is given in section 4.In section 5 an overall conclusion is provided.

II. Related Works

A number of TCP variants have been proposed by various authors to control congestion problems. The first approach to solve congestion control was proposed by Van Jacobson named as TCP Tahoe [6]. TCP is based on a principle of 'conservation of packets', i.e. if the connection is running at the available bandwidth capacity then a packet is not injected into the network unless a packet is taken out as well. In TCP an acknowledgement means that a packet was taken off the wire by the receiver. It also maintains a congestion window (CWD) to reflect the network capacity. However there are certain issues, which need to be resolved to ensure this equilibrium like determining present bandwidth, maintaining astute equilibrium and reacting to congestion.

To meet all these criteria Tahoe suggested that whenever a TCP connection starts or re-starts after a packet loss it should go through a procedure called 'slow-start' [7]. It sets the congestion window to 1 and then for each ACK received it

increases the CWD by 1, so in the first round trip time (RTT), one packet is sent, which is doubled in the second transmission time and increased so forth. Thus window is increased exponentially until a packet is lost which is a sign of congestion. When congestion is encountered sending rate is decreased and congestion window is reduced to one and restarted again. For congestion avoidance Tahoe uses 'Additive Increase/ Multiplicative Decrease'. A packet loss is taken as a sign of congestion and Tahoe saves the half of the current window as a threshold value. It then sets CWD to one and starts slow start until it reaches the threshold value. After that, it increments CWND linearly, until it encounters a packet loss. Thus it increases its window slowly as it approaches the bandwidth capacity. The problem with Tahoe is that it takes a complete timeout interval to detect a packet loss. In fact, in most implementations it takes even longer because of the coarse grain timeout .This leads to a major cost in high band-width delay product links.

In the solution to Tahoe's problems a new variant called Tcp Reno was proposed. Reno depends on duplicate ACKs and the timer associated with each packet Tcp sends. Reno suggests a new algorithm called *'Fast Re-Transmit'* in which whenever 3 duplicate ACK's are received it is taken as a sign that the segment was lost, so the segment is re-transmitted without waiting for timeout. After Fast Re-Transmit Reno enters into a stage called fast recovery where instead of reinitialising the window size to 1MSS it is initialised to half the current window .This is contrary to the emptying of the pipe completely as done in Tahoe. Reno however, faced problems in case of multiple packets drops which was resolved in New Reno.Like Reno, New-Reno also enters into fast-retransmit when it receives multiple duplicate packets, however it differs from Reno by the fact that it doesn't exit Fast-recovery until all the data which was out standing at the time it entered fast recovery is acknowledged.

In a newer approach was found for congestion control which was is known as proactive techniques. It was found more suited to solve congestion problems compared to its reactive counterpart. It does not depend solely on packet loss as a sign of congestion. It detects congestion before the packets losses occur via probing of the network .Such approaches were implemented by TCP Vegas [8].Vegas keeps track of each packet sent and monitor round trip times and maintains an estimate of packet transmission times. Whenever a duplicate acknowledgement packet is received it compares packet transmission time with its own estimate to decide on retransmissions.

TCP Westwood (TCPW) [9] is a sender-side modification of the TCP congestion window algorithm that improves upon the performance of TCP Reno in wired as well as wireless networks. The improvement is most significant in wireless networks with lossy links, since TCP Westwood relies on end to-end bandwidth estimation to discriminate the cause of packet loss (congestion or wireless channel effect) which is a major problem in TCP Reno. An important distinguishing feature of TCP Westwood with respect to previous wireless TCP "extensions" is that it does not require inspection and/or interception of TCP packets at intermediate (proxy) nodes.

Rather, it fully complies with the end-to-end TCP design principle. The key innovative idea is to continuously measure at the TCP source the rate of the connection by monitoring the rate of returning ACKs. The estimate is then used to compute congestion window and slow start threshold after a congestion episode, that is, after three duplicate acknowledgments or after a timeout [10]. The rationale of this strategy is simple: in contrast with TCP Reno, which "blindly" halves the congestion window after three duplicate ACKs, TCP Westwood attempts to select a slow start threshold and a congestion window which are consistent with the effective bandwidth used at the time congestion is experienced.

In Table I. We have summarized the basic properties of these variants and their disadvantages.

TABLE I: COMPARISON OF SEVERAL VARIANTS OF TCP.

TCP Variants	Characteristics	Limitations
TCP Tahoe	Implement Additive increase and Multiplicative decrease technique.	i.Takes a Complete Timeout interval to detect congestion and sometimes even longer due to Coarse grain timeouts. ii.Pipeline Emptied every time packet lost.
TCP Reno	Implement Fast-Re-Transmit	Cannot detect multiple packet loss in the same window.
TCP New Reno	Modifies Fast-Recovery phase.	Takes single RTT to detect each packet loss.
TCP West wood	i.Sender-side modification. ii.End-to-end band width estimation. iii.Continuously monitor returning acks.	Since slow start is used, a part of the bandwidth remains unutilized.
TCP Vegas	i.Implement modified Re-transmission mechanism. ii. It determines congestion by a decrease in sending rate as compared to the expected rate. iii.Use Modified Slow-Start.	i.Cannot compete with more aggressive TCP Reno connections. ii.Vegas may not stabilize if buffers are small, leading to behavior that is similar to that of TCP Reno

III. PROPOSED ALGORITHM

TCP Rcc is sender side only modification of TCP Reno where the congestion window is set according to the bandwidth available in the network. The window is kept fixed until any packet loss or duplicate acknowledgement. After congestion is detected, a random amount of time is waited before re-estimation of bandwidth and recalculation of window.

The algorithm used is as follows:-

Step1. At first, the congestion window size is set to 1MSS like TCP Reno here; MSS is meant by the maximum segment size of a frame i.e. the byte of data that is allowed to be sent in one frame.

Step2. After the acknowledgement for the 1st transmission is received by the sender from the receiver, the sender would be able to judge the bandwidth of the network. Every packet sent by the sender contains a timestamp. The time required to send the data (1MSS) from source to destination (γ) is

calculated from the timestamps. Finally, bandwidth estimation (BWE) is done.

$$BWE = (\text{data sent during the first transmission}) / \gamma$$

Here BWE is nothing but throughput - the amount of data that the network can transfer per unit of time.

Step3. Let, $k = BWE \times RTT$

(Amount of traffic the network can transmit to the destination throughout the entire round trip time)

$$Cwnd = k*\beta \ (0.5<\beta<1)$$

Here, cwnd is the congestion window set by the tcp. β is a constant used here whose value may range from 0.5 to 1(In the testing the value of β is ideally taken to be 0.9874) .K's value is measured in MSS ,thus in bytes.

Step4. while (there is any message to sent)
begin
if(3 timeouts or 3 duplicate ACKs occur at point of time)
wait for a random amount of time,
then, cwnd=1MSS and recalculate the BWE,
set, new cwnd again
else
for each successful transmission of cwnd increase cwnd by 1MSS
until complete transmission of data i.e.
cwnd=cwnd+1MSS
end

The algorithm is put in a loop as seen above until all is sent. Unlike approaches of other TCP variants no threshold of the data packets to be sent to the receiver is maintained by the sender. Adaptive window calculation is performed at each stage of congestion detection to improve performance of the sender .

IV. Implementation

There are two ways to establish the credibility of the algorithm developed –either to simulate the algorithm in some network simulator or to mathematically prove it. Since, testing provides a more concrete evidence of the successful running of the algorithm therefore, testing the algorithm in network simulator ns2 is chosen over mathematical estimation.

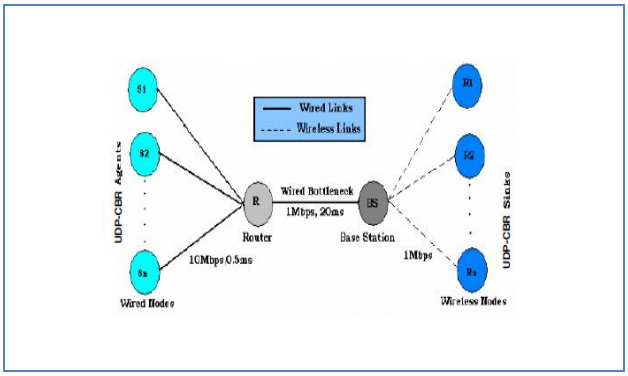

Fig. 1. Simulation scenario.

In the network simulator ns2, two domains one for wired and one for wireless network is built like in Fig.1. The wired domain and the wireless domain communicate with each other via a base station which acts as the intermediate node as well as the bottleneck link. The nodes act as traffic generators following FTP application layer protocol and TCP as transport layer protocol. The mobile nodes would relay packets between each other following DSDV as routing protocol. Up to four wired node and four wireless nodes have been considered. The reason for this type of heterogeneous topology is to simulate the condition faced by the TCP Rcc in real life.

Performance comparisons with various existing tcp variants have been performed at different error rates lying between 5%, and 15% to suit dynamic needs of modern network scenario.

V. Results

The simulation on successful completion gave satisfactory result of the superiority of the proposed algorithm over other TCP variants like Tahoe, Reno, Vegas, New Reno and Westwood.

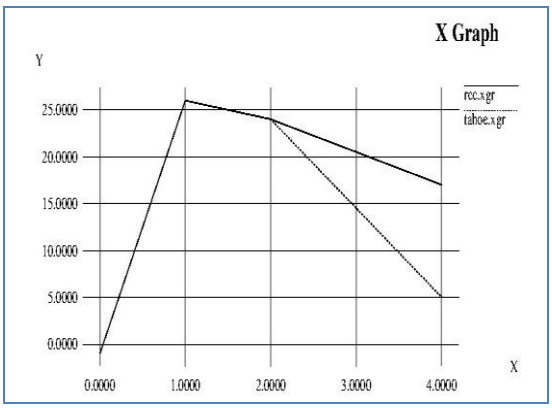

Fig. 2. Throughput comparison of the *TCP rcc vs TCP tahoe* in the simulation with 10% error rate.

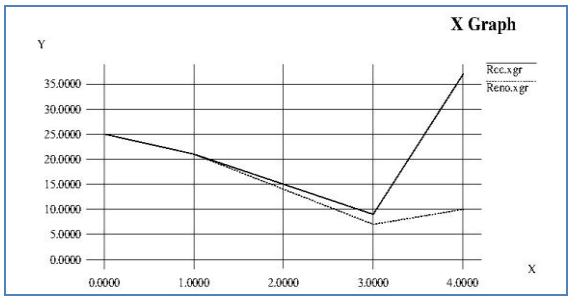

Fig. 3. Throughput comparison of the *TCP rcc vs TCP Reno* in the simulation with 15% eror rate.

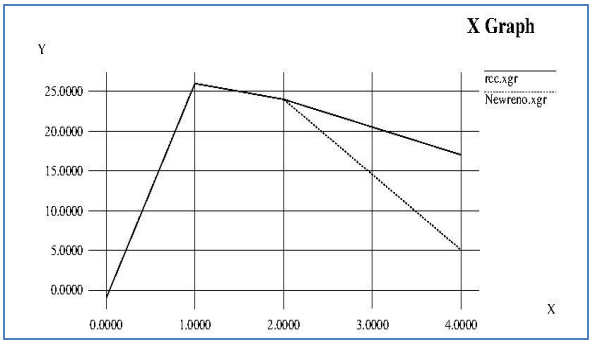

Fig. 4. Throughput comparison of the *TCP rcc vs TCP new Reno* in the simulation with 10% error rate.

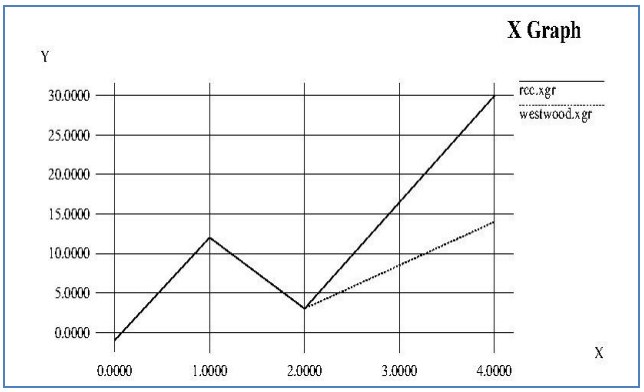

Fig. 5. Throughput comparison of the *TCP rcc vs. TCP westwood* in the simulation with 15% error rate.

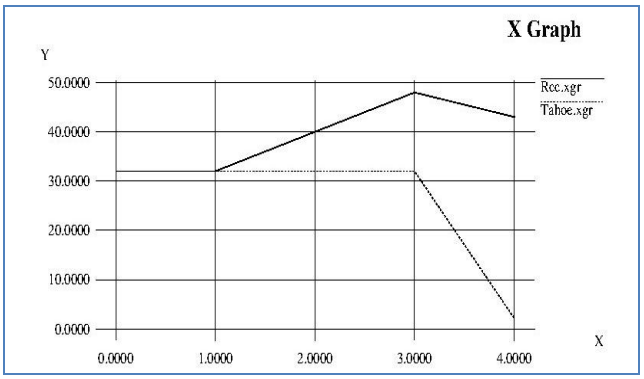

Fig. 6. Throughput comparison of the *TCP rcc vs TCP tahoe* in the simulation with 5% error in the 8 node topology.

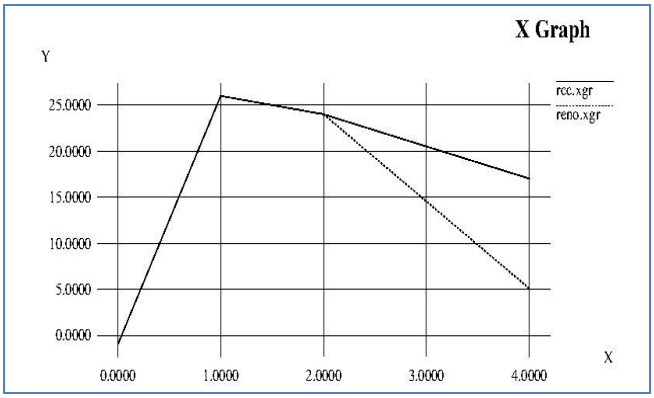

Fig. 7. Throughput comparison of the *TCP rcc vs TCP reno* in the simulation with 10% error rate.

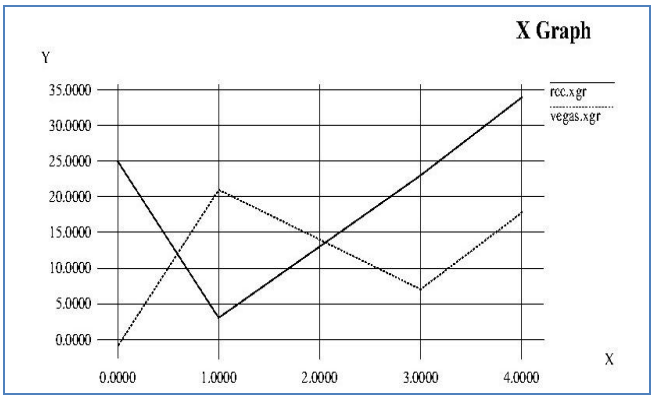

Fig. 8. Throughput comparison of *the TCP rcc vs TCP vegas* in the simulation with 2% error rate.

From the above results it can be concluded that tcp-rcc is functioning equal or superiorly over some established algorithms.

VI. Conclusion and Future Work

We have used drop tail queue structure while designing nodes with constant queue size. However in real world various queue structures are followed and their sizes are not fixed. We are presently working on this queuing problem. Besides we are working on proving the merit of the algorithm statistically particularly in terms of throughput and fairness. We have not tested the situation if TCP Rcc is compatible with other variants of TCP in same network.

Reference

[1] R. Paul and L. Trajković, *Selective-TCP for Wired/Wireless Networks.*
[2] S. Floyd and T. Henderson, "The new Reno modification to TCPs fast recovery algorithm," *IETF RFC 2582*, Apr. 1999.
[3] V. N. Padmanabhan, S. Seshan and R. H. Katz, *.A Comparison of Mechanisms for Improving TCP Performance over Wireless Links Hari Balakrishnan*.
[4] A. K. Ghosh, A. Mukherjee and D. Saha, "Some simulation studies to characterize TCP window control behavior in wired/wireless internetworks," P*roceedings of IEEE International Conference on Personal Wireless Communications (ICPWC) 2005*, India. Jan. 23-25, 2005.
[5] A. K. Ghosh, S. Das, R. Roy, and A. Mukherjee, *Constant Congestion Window Approach for TCP Effect on Fairness*.
[6] V. Jacobson, "Congestion avoidance and control," *the ACM SIGCOMM*, vol. 88.
[7] W. Stevens, "TCP slow start, congestion avoidance, fast retransmit, and fast recovery algorithms," *RFC 2001*, Jan. 1997.
[8] J. La, J. Walrand, and V. Anantharam, "Issues in TCP vegas richard," *Department of Electrical Engineering and Computer Sciences University of California at Berkeley*.
[9] S. Mascolo, P. D. Bari, C. Politecnico, D. Torino, M. Gerla, M. Y. Sanadidi, and R.Wang, "TCP westwood: Bandwidth estimation for enhanced transport over wireless links," *UCLA Computer Science*.
[10] M. Valla, M. Y. Sanadidi, and M. Gerla, *Adaptive Bandwidth Share Estimation in TCP Westwood Ren Wang*.