2015 The 5th International Workshop on Computer Science and Engineering
doi: 10.18178/wcse.2015.04.095

TCP VenoDR: TCP Enhancement with Adaptive Queue Length Threshold and Dynamic Recovery Mechanism

Nan Ding[1,a], Rui-Qing Wu[1,b] and Hong Jie[1,c]

[1]School of Electronic Engineering, University of Electronic Science and Technology of China, Chengdu, 611731, China

[a]dingnan11@163.com, [b]rqwu@uestc.edu.cn, [c]jieqionghong@126.com

Keywords: wired/wireless networks, loss differentiation, adaptive queue length threshold, dynamic recovery mechanism

Abstract. In the past decades, many end-to-end TCP enhancements have been presented for the wireless networks with high bit-error rates (BER). As a typical example, TCP Veno mainly enhanced the performance suffering in the wireless networks. However, the fixed queue length threshold and traditional fast recovery algorithm make TCP Veno unable to adapt to the network states changes. In this paper, we propose an enhancement based on TCP Veno, called TCP VenoDR, which has two improvements: differentiating the random loss from congestion loss based on an adaptive queue length threshold of bottleneck link, and a dynamic recovery algorithm (DR) in the recovery period by adopting a dynamically adjusted variable to control the congestion window (*cwnd*). The simulation results by NS-2 show that VenoDR obtains a significant performance in throughput, bandwidth utilization and fairness over the traditional TCPs, such as Veno, Westwood, and Reno in wired/wireless networks.

Introduction

With the rapid development of emerging wireless communication technology, the reliable transport control protocol originally designed for wired networks, such as TCP Reno [1], is challenged by the problem of high BER when it is deployed in wired/wireless networks. Since the random BER is negligible in wired networks, traditional TCP regards congestion as the main reason of packet loss. But in wireless networks, the packet loss is often induced by high BER rather than congestion. Therefore, the traditional TCP is unable to differentiate the random packet loss from congestion loss, which results in slowing down the sending rate and performance degradations in wired/wireless heterogeneous networks.

As an end-to-end scheme [2, 3, 4], TCP Veno [5] can provide a significant improvement in the wireless networks and show friendliness towards TCP Reno in simulation and Internet measurements. It estimates the number of packets (N) backlogged in the router buffer and compares N with a fixed threshold β (generally 3) to differentiate the random packet losses from network congestion and adjusts the *cwnd* accordingly. However, the following shortages still exist in TCP Veno when it is applied in wired/wireless networks with high BER. (i) The coarse-grained estimation of N and a fixed threshold make TCP Veno unable to adapt to the network states changes. (ii) TCP Veno inherits the fast recovery (FR) algorithm of traditional TCP, which will reduce the *cwnd* by several times if multiple packets are lost within a window and drive the TCP into timeout.

To solve the performance degradation in FR algorithm mentioned above, many researchers have proposed several algorithms, such as TCP NewReno [6], TCP SACK [7], and TCP FACK [8], etc. They all provide good solutions of this problem, but modifications of both the TCP sender and receiver are needed in SACK and FACK, which make them complex to realize and hard to widespread. Compared to these algorithms, NewReno only has a few changes in TCP senders. And it is more compatible with traditional TCP algorithms.

In this paper, we improve the loss differentiation mechanism in Veno with an adaptive queue length threshold and add an improved dynamic recover algorithm (DR) to Veno based on the FR in TCP NewReno,aiming to improve the TCP performance in wired/wireless networks with high BER.

The organization of the rest paper is as follows. Section 2 introduces the related works of loss differentiation in TCP Veno and FR algorithm in TCP NewReno. In section 3, we propose and discuss the TCP VenoDR. Section 4 descripts the simulation and performance evaluation of TCP VenoDR and other TCP variants. Finally, section 5 concludes with the simulation and observation.

Related works

Overview of TCP Veno. TCP Veno utilizes the idea of congestion control scheme in TCP Vegas [9] and intelligently integrates it into TCP Reno. It refines Vegas' additive increase and multiplicative decrease (AIMD) algorithm, adds the packet loss differentiation mechanism to Reno after receiving three duplicate ACKs (dup-ACKs). In the slow-start phase, Veno uses the same exponential increase algorithm as Reno. In the congestion avoidance phase, Veno calculates backlogged packets number N in the router buffer shown in Eq. 1 and adjusts *cwnd* based on the comparison of N and a fixed threshold β (generally 3) as the following pseudo codes.

$$N = (cwnd/BaseRTT - cwnd/RTT)\ BaseRTT. \quad (1)$$

// The congestion avoidance phase.
if ($N < \beta$) *cwnd*=*cwnd*+1; //available bandwidth not fully utilized
else *cwnd*=*cwnd*+1/*cwnd*; // available bandwidth fully utilized

where *cwnd* is the current TCP window size, *BaseRTT* is the minimum of measured round-trip times, and *RTT* is the smoothed round-trip time measured. When three duplicate ACKs occur, Veno differentiates packet losses as follows.

// The loss differentiation mechanism.

if ($N < \beta$) *ssthresh*=4*cwnd*/5; //random loss due to random BER
else *ssthresh*=*cwnd*/2; //congestion loss

Fast recovery algorithm of TCP NewReno. The realization process of FR algorithm in TCP NewReno is as follows [6]:

Step1: NewReno enters the FR phase, when receiving three dup-ACKs. The slow-start threshold (*ssthresh*) is set to a fixed value equal to max(*cwnd*/2, 2). The *recover_* saves the highest sequence number of transmitted packets.

Step2: Retransmit the lost packets and add three maximum segment size (MSS) to the *cwnd*.

Step3: When receiving a dup-ACK each time, the TCP sender increases the *cwnd* by one MSS and sends a new packet.

Step4: When the sender receives a new ACK acknowledged, all the packets are recorded in *recover_*. Both the first lost packet and the packets transmitted after three dup-ACKs are acknowledged. NewReno will decrease the *cwnd* to min(*ssthresh*, *cwnd*+1) and exit the FR phase. Where the *ssthresh* is set to the value recorded in **Step1**. Otherwise, if the ACK only acknowledges part of the transmitted packets, namely a partial-ACK, the *cwnd* will be decreased to a value equal to the difference of the *cwnd* and the number of partial acknowledged packets. Then the TCP sender resets the timeout timer and goes back to **Step3**.

Though NewReno is to avoid the performance degradation caused by multiple packets losses within a window, it still has deficiencies: (i) NewReno can not effectively get the enough information of the network bandwidth after receiving a partial-ACK, and resulting in low data transmission efficiency. (ii) The value of *cwnd* for exiting FR phase in **Step4** has been determined before NewReno goes into FR phase, namely the *ssthresh* in **Step1**. The *cwnd* is not suitable for the network states when there are packet losses again during FR phase, which will result in a frequent change between the FR and non-FR phase.

TCP VenoDR

TCP VenoDR has two improvements.The one is an improved loss differentiation mechanism with an adaptive queue length threshold based on TCP Veno, and the other is the DR mechanism on the basis of TCP NewReno.

Estimating the queue length. We note that a packet's round-trip time (RTT) consists of three parts: queuing delay, transmission delay and propagation delay. The sum of transmission delay and propagation delay is generally a constant, it equals to the *BaseRTT*. And the queuing delay of bottleneck link is equal to *L*/*BW*, where *L* means the queue length in bottleneck link, *BW* means the bottleneck bandwidth. Therefore, the *RTT* can be formulated as Eq. 2, and then we can deduce the formula of *L* as Eq. 3.

$$RTT = L/BW + BaseRTT. \quad (2)$$

$$L = (RTT - BaseRTT)\, BW. \quad (3)$$

Veno gets the coarse-grained estimation of bottleneck router backlogged packets number *N*, and VenoDR estimates the queue length *L* in bottleneck router based on *RTT*, *BaseRTT* and *BW*.

Loss differentiation based on adaptive threshold. In order to estimate the congestion status of bottleneck link, we set an adaptive queue length threshold *T*:

$$T = \alpha\, Lmax. \quad (4)$$

$$Lmax = (RTTmax - BaseRTT)\, BW. \quad (5)$$

where L_{max} is the largest value of *L* calculated by TCP sender, and α is a constant between 0 and 1. Here α is an experience value which is set to 0.55 to maximize the throughput and fairness of VenoDR.

When three dup-ACKs are received, if *L*<*T*, VenoDR regards the packet loss as a random loss and reduce *ssthresh* by a smaller amount (1/5). Otherwise, VenoDR assumes the packet loss caused by congestion and reduces the ssthresh by half. From Eq. 3 and Eq. 4, we note that *L* and *T* both have the factor *BW*, and divide both sides of the inequality *L*<*T* by *BW* and the inequality will still be true. Therefore, the *BW* is not necessary and it is set to 1.

Dynamic recovery mechanism. In order to solve the problem discussed in 2.2, we propose a DR mechanism based on the FR algorithm in NewReno and the flowchart of DR mechanism is shown in Fig.1.

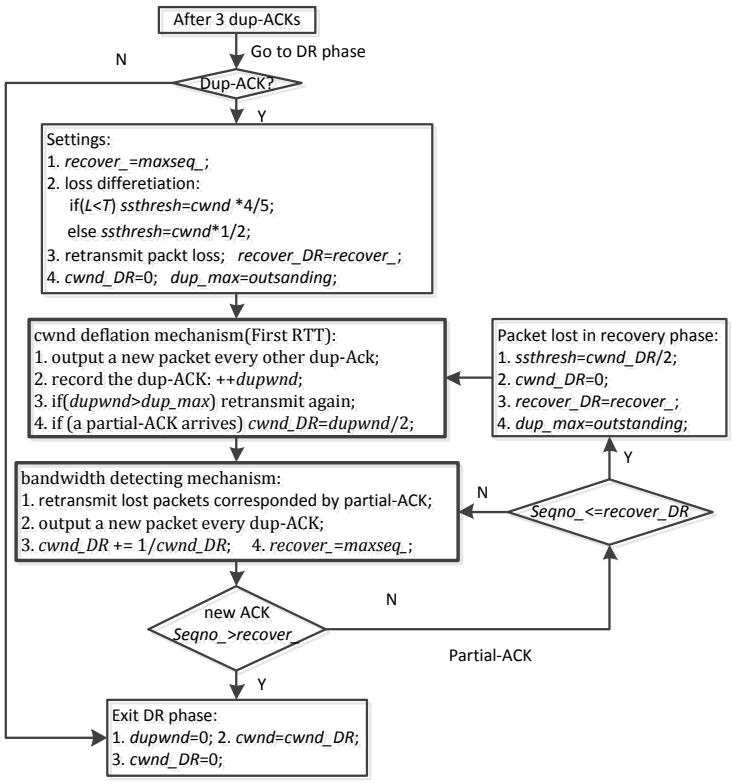

Fig.1 Flowchart of DR mechanism (including the fast retransmit)

The DR mechanism adopts a dynamic adjusted variable *cwnd_DR*, namely the packets transmitted in the current *RTT*, but not the *cwnd* determined before FR phase in NewReno. We note that the *cwnd* used in FR can't reflect the number of packets transmitting currently in networks. It represents all the transmitted packets, which consists of both the packets transmitted in current RTT (*cwnd_DR*) and the packets transmitted in the previous RTT including the lost packets and unacknowledged packets. Only the first part of the *cwnd* should be adopted to adjust the value of congestion window when TCP exits the FR phase. Therefore, DR mechanism uses *cwnd_DR* to control the recovery and passes the value of *cwnd_DR* back to *cwnd* when it exits the DR phase. The initialization and dynamic update of *cwnd_DR* are shown in Fig.1. Specific meanings of the other parameters are given below. The variable *recover_* denotes the highest sequence number before the most recent dup-ACK received. The *recover_DR* means the highest sequence number when the first packet loss of the previous RTT happened. The *dup_max* represents the number of transmitting and unacknowledged packets. The *dupwnd* records the number of dup-ACKs during *cwnd* deflation phase. And *seqno_* means the sequence number indicated by the current ACK.

The DR mechanism contains two parts compared to NewReno: cwnd deflation mechanism and bandwidth detecting mechanism. The first part is to recover the first packet loss, including the situation that the packet lost again during retransmission period (when *dupwnd>dup_max*), and to give the rights of congestion control back to *cwnd* through passing *cwnd_DR* to *cwnd* when it exits the DR, which will realize the deflation of *cwnd*. When the sender receives a partial-ACK, which means multiple packets are lost within a window, it enters the bandwidth detecting phase. The bandwidth detecting phase can not only recover the rest of packet losses corresponded by partial-ACK but also detect the available network bandwidth by adopting an additive increase algorithm, namely increasing the *cwnd_DR* by one MSS every *RTT*. Then if a partial-ACK is received, we judge whether the recovery phase with packet loss or not by comparing the *seqno_* and *recover_DR*. If *seqno_<=recover_DR*, there are packet losses in DR again, we need to update the *cwnd_DR*, *recover_DR* and *dup_max*. Then the recovery will go back to the *cwnd* deflation phase. Otherwise, it will go to the bandwidth detecting phase. The sender will exit the DR phase when all packet losses are recovered and pass the value of *cwnd_DR* back to *cwnd* as shown in Fig.1.

Performance Evaluation

We evaluate the performance of TCP VenoDR using NS-2 network simulator in throughput, bandwidth utilization, fairness and friendliness compared with the other three representative algorithms, namely TCP Reno, TCP Veno and TCP WestwoodNR [10]. We use the topology depicted in Fig. 2 to simulate the throughput performance, and the topology shown in Fig. 3 to evaluate the fairness and friendliness performance.

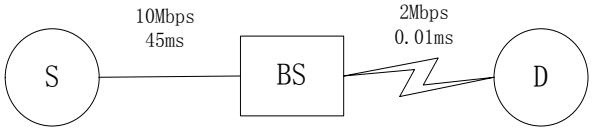

Fig.2 Topology for throughput

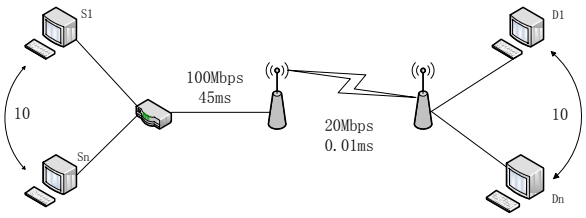

Fig.3 Topology for fairness and friendliness

Throughput performance. The connection between the source node and base station is an error free link. The base station is linked to the wireless destination node with BER varying from 0.01% to 10%. The link bandwidth and one-way delay are shown in Fig.2. A single TCP connection runs a FTP application from 0s to 100s. The maximum segment size is 1400 bytes. We run the simulation for VenoDR, Veno, Reno and WestwoodNR respectively. The router buffers are equal to the pipe size.

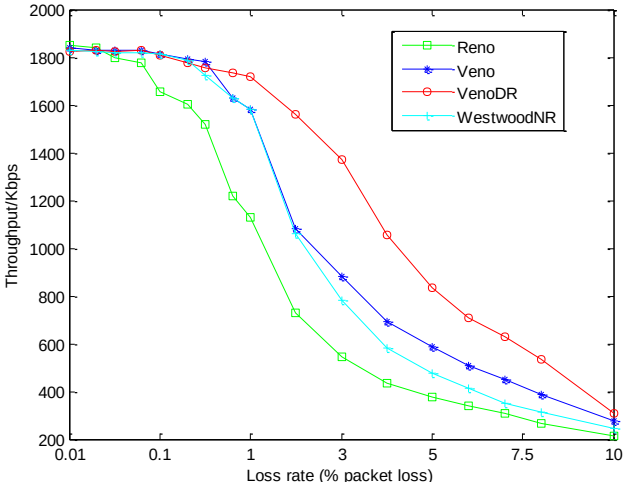

Fig.4 Mean throughput comparison

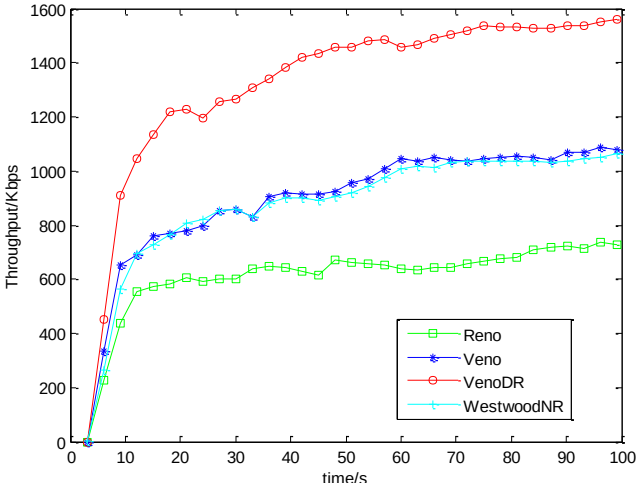

Fig.5 Comparison with 2% BER

Fig.4 shows the mean throughput comparison of 4 algorithms under different wireless BER from 0.01% to 10%. When the loss rate is smaller than 0.5%, the throughput curves are almost the same. From 0.5%, TCP VenoDR is superior to other 3 algorithms along with the growth of BER, especially when the BER is higher than 1%. Fig. 5 shows the throughput comparison in real time with 2% BER. It is obvious that the throughput of VenoDR at 100s outperforms Veno by 44.1%, WestwoodNR by 47.0% and Reno by 113.8%. And the advantages of VenoDR still exist in various bottleneck bandwidths as shown in Fig. 6.

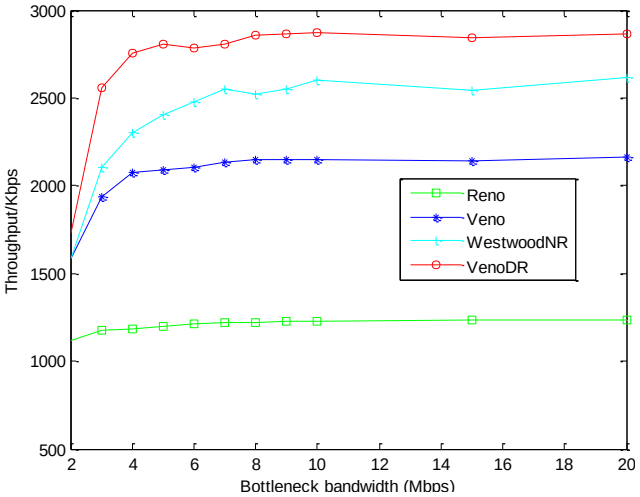

Fig.6 Comparison of various BW

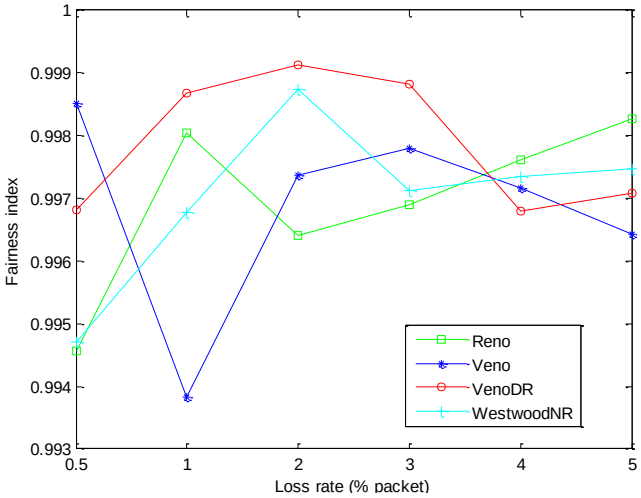

Fig.7 Fairness index comparison

Fairness and friendliness performance. TCP fairness means that multiple senders running the same TCP algorithm share the network resources fairly when the network congestion occurs. As shown in Fig. 3, there are a total of 10 same TCP flows share a 20Mb/s bottleneck link. The BER at the wireless bottleneck link varies from 0.5% to 5%. We simulate the four TCP schemes respectively and calculate the fairness index proposed in [11] to evaluate the fairness of TCP schemes. The results are shown in Fig. 7.

The fairness indexes of VenoDR with different BER are all higher than 0.996, which is also higher than the other three algorithms. That means VenoDR has a fairly satisfactory fairness performance. Moreover, we can see from Fig. 8, which shows the total throughput of 10 TCP VenoDR connections compared to other three TCP algorithms, the throughput of TCP VenoDR is much higher than others. In conclusion, TCP VenoDR not only has a better fairness but also provide the highest bandwidth utilization compared to others.

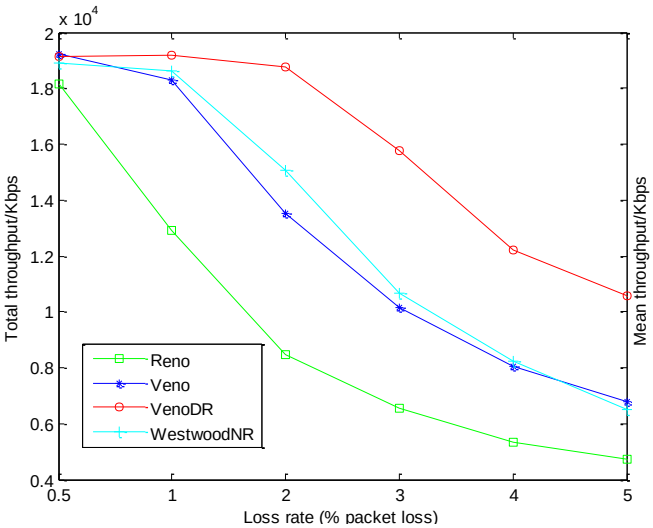

Fig.8 Bandwidth utilization

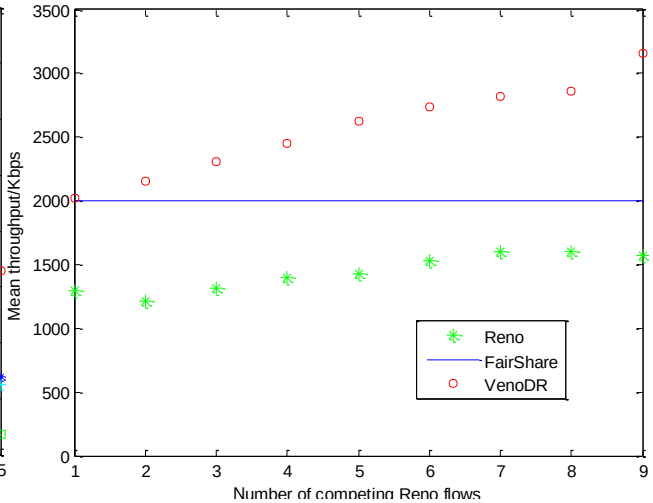

Fig.9 Friendliness of VenoDR towards Reno

A friendly TCP scheme should be able to coexist with other TCP variants. We simulate the friendliness of VenoDR towards Reno using the topology with total 10 TCP connections of two schemes sharing a 20Mbps bottleneck in Fig.3. The simulation results are shown in Fig.9, where the horizontal axis represents the number of competing Reno connections, the remaining connections being VenoDR. The vertical axis represents the average throughput of Reno and VenoDR respectively. From Fig. 9 we can see that VenoDR achieves a little higher throughput than its fair share, and the throughput of Reno is below the fair share but within a tolerable range. Therefore, VenoDR is friendly towards Reno.

Conclusions

In this paper, we have described a novel congestion control mechanism called TCP VenoDR with adaptive queue length threshold and DR mechanism. It utilizes the RTT measurements to estimate the queue length of bottleneck link and differentiates packet loss based on an adaptive threshold. Its DR algorithm can probe the available bandwidth of the network, recover not only the first packet loss but also the packets lost later in the recovery period, and adopt a dynamically adjusted variable to control the congestion window when it exits the recovery phase. Simulation results by NS-2 show that VenoDR provides a significant performance improvement in throughput, bandwidth utilization and fairness over other TCP variants. It is also friendly towards the traditional TCP Reno.

References

[1] M. Allman, V. Paxson, W. Stevens. *TCP congestion control.* RFC2581, Apr. 1999.

[2] Yanxiang Z, Fang S, Mingyan K, A fuzzy packet loss differentiation algorithm based on ack-timeout times ratio in heterogeneous network. Communications and Mobile Computing (CMC), International Conference on. IEEE 1, pp. 453-457, 2010.

[3] Geethu Wilson, Robin Cyriac., An Enhancement to TCPW BBE by Modifying the Bandwidth Estimation Using Modified EWMA. *International Journal of Advanced Research in Computer Science and Software Engineering*, 2(6), pp. 278-281, June 2012.

[4] K. Xu, Y. Tian, and N. Ansari, Improving TCP Performance in Integrated Wireless Communications Networks. *Journal of Computer Networks*, 47, pp. 219–237, February 2005.

[5] Fu C P, Liew S C., TCP Veno: TCP enhancement for transmission over wireless access networks. *IEEE Journal of Selected Areas in Communications*, 21(2), pp. 216-228, 2003

[6] S. Floyd, T. Henderson. *The NewReno modification to TCP's fast recovery algorithm*. RFC2582, Apr. 1999.

[7] M. Mathis, J, Mahdavi, S. Floyd, and A. Romanow. *TCP selective acknowledgement options.* RFC2018, Oct. 1996.

[8] M. Mathis, J. Mahdavi., Forward acknowledgement: refining TCP congestion control. Proceedings of the SIGCOMM, August 1996.

[9] L.S. Brakmo, S.W. O'Malley, L.L. Peterson. TCP Vegas: New techniques for congestion detection and avoidance. In Proc. ACM SIGCOMM, Oct 1994.

[10] Ns-2 modules of TCP Westwood with the New Reno feature [Online]. Available: http://193.204.59.68/mascolo/tcp%20westwood/modules.htm

[11] Jain R, Chiu D M, Hawe W R. A quantitative measure of fairness and discrimination for resource allocation in shared computer system. *Eastern Research Laboratory, Digital Equipment Corporation*, 1984.