International Journal of Computer Applications (0975 – 8887)
Volume 45– No.5, May 2012

Enhanced TCP Westwood Congestion Avoidance Mechanism (TCP WestwoodNew)

Shimaa Hagag
South Delta Electricity Company, Cairo, Egypt.

Ayman El-Sayed (IEEE Senior Member)
Computer Science & Eng. Dept., Faculty Of Electronic Eng., Menoufiya University, 32952 Menouf, Egypt.

ABSTRACT

Transport Control Protocol (TCP), the mostly used transport protocol, performs well over wired networks. As much as wireless network is deployed, TCP should be modified to work for both wired and wireless networks. Since TCP is designed for congestion control in wired networks, it cannot clearly detect non-congestion related packet loss from wireless networks. TCP Congestion control plays the key role to ensure stability of the Internet along with fair and efficient allocation of the bandwidth. So, congestion control is currently a large area of research and concern in the network community. Many congestion control mechanisms are developed and refined by researcher aiming to overcome congestion. During the last decade, several congestion control mechanisms have been proposed to improve TCP congestion control. Comparing these mechanisms, showing their differences and their improvements, and we identify, classify, and discuss some of these mechanisms of TCP congestion control such as Tahoe, Sack, Reno, NewReno, Vegas, and Westwood. TCP Westwood works for both wired and wireless network, and we propose a new algorithm called TCP WestwoodNew to increase the performance of TCP-Westwood. By enhanced the congestion avoidance of TCP Westwood by a new estimation to cwnd algorithm based on the network status. Also TCP WestwoodNew introduces a new estimation for Retransmission TimeOuts (RTO). RTO has been reported to be a problem on network paths involving links that are prone to sudden delays due to various reasons. Especially many wireless network technologies contain such links. Spurious RTO often cause unnecessary retransmission of several segments, which is harmful for TCP performance, and unnecessary retransmissions can be avoided. We simulate the proposed algorithm TCP WestwoodNew using the well known network simulator ns-2, by comparing it to the original TCP-Westwood. Simulation results show that the proposed scheme achieves better throughput than TCP Westwood and decreases the delay.

Keywords

TCP, Congestion Control Mechanisms

1. INTRODUCTION

Computer network have experienced an explosive growth over the past few years, that growth cause congestion collapse. When this congestion occurs performance will degrade. The transport layer provides congestion control mechanisms [1], i.e., Transmission Control Protocol (TCP). Transmission Control Protocol (TCP) is the most popular transport layer protocol for the Internet. Due to various reasons, such as multipath routing, route fluttering, and retransmissions, packets belonging to the same flow may arrive out of order at a destination. Such packet reordering violates the design principles of some traffic control mechanisms in TCP and, thus, poses performance problems. In [2], the authors provide a comprehensive and in-depth survey on recent research on packet reordering in TCP. The causes and problems for packet reordering are discussed. TCP is a connection oriented reliable protocol. TCP is end-to-end congestion control where all the work is done by transport layer. It is extensively used in the internet, TCP uses a number of mechanisms to achieve high performance and avoid congestion collapse [3]. Currently, Internet routing protocols select only a single path between a source and a destination. However, due to many policy routing decisions, single-path routing may limit the achievable throughput. In [4], they envision a scenario where multi-path routing is enabled in the Internet to take advantage of path diversity. Using minimal congestion feedback signals from the routers, they present a class of algorithms that can be implemented at the sources to stably and optimally split the flow between each source-destination pair. they then show that the connection-level throughput region of such multi-path routing/congestion control algorithms can be larger than that of a single-path congestion control scheme. In [5], the author studies the stability issue of the average queue length of a Transmission Control Protocol (TCP) model when interacting with Random Early Detection (RED). The model used for the study has shown period doubling bifurcation (PDB) and border collision bifurcation (BCB) in the average queue size at certain values of parameters when original RED is deployed. They adopt a gentle version of RED and a newly derived RED algorithm into the model to study the improvement in stability of average queue size of the system. In [6], the authors analyze the dynamic behavior of a single RED controlled queue interacting with a large population of idealized TCP sources, i.e., sources obeying the rules of linear increase and multiplicative decrease. The aggregate traffic from this population is modeled in terms of the time dependent expected value of the packet arrival rate which reacts to the packet loss taking place in the queue. The queue is described in terms of the time dependent expected values of the instantaneous queue length and of the exponentially averaged queue length. TCP congestion control has been designed to ensure Internet stability along with fair and efficient allocation of the network bandwidth. Congestion control defines the methods for implicitly interpreting signals from the network in order for a sender to adjust its rate of transmission to prevent a sender from overrunning the capacity of the network [7]. Congestion control is built as distributed mechanisms that prevent congestion before happen or even remove the congestion if it happened [8]. The main objective of congestion control mechanisms is to keep the network running pretty close to its rated capacity, even when faced with extreme overload. These objectives could be translated into two main goals, the first is to avoid the occurrence of network congestion before happen and dissolve the congestion if the congestion occurrence cannot be avoided. The second is to provide a fair service to the different connections, along with support various Internet application domains with diverse Quality of Service (QoS) requirements [9]. Generally, there are two ways to implement congestion control: (1) Network-assisted congestion controls; they are approaches taken by routers [10]. These approaches use the router queue size to monitor the congestion state of the network. (2) End-to-End congestion controls; they are approaches taken by the transmission control protocol (TCP) and are mostly achieved in transport layer [11]. Active Queue Management (AQM) [12] routers have been recently proposed to support the end-to-end congestion control in the Internet. In [13], a fuzzy modeling technique is employed to set up a time-delay affine Takagi-Sugeno (T-S) fuzzy model for a Transmission Control Protocol (TCP) network with AQM routers. Based on the proposed time-delay affine T-S fuzzy model, a fuzzy controller design approach is developed for the control of AQM routers. The congestion control protocol considered, evaluated, enhanced in this paper are based on the end-to-end congestion control. This approach is required not only to help the end-user to gain utility from the network, but also to prevent congestion collapse in the network. During the last decade, many congestion control algorithms have been proposed to improve the classic Tahoe/Reno TCP congestion control. In [14], the authors evaluate and compare three control algorithms, which are Westwood+, New Reno and Vegas TCP. This paper aims to comparing the mechanisms of end-to-end user approach, showing their differences and their improvements. We identify, classify, and discuss some of these mechanisms of TCP congestion control such as Tahoe, Sack, Reno, NewReno, Vegas, and Westwood. Also, a Modified TCP Westwood mechanism is proposed. This paper is organized as follows. Section 2 provides a brief description of the Transmission Control Protocol (TCP). The end-to-end congestion control is described in section 3, as the different algorithm and the current congestion control mechanisms. TCP Westwood mechanism is described in detail in section 4, and the new TCP Westwood mechanism depicts on section 5. The results of performance are shown in section 6. Finally, the paper is concluded in section 7.

2. TRANSMISSION CONTROL PROTOCOL

Transmission Control Protocol (TCP) [15, 16, 17] is a reliable, connection-oriented, end-to-end, error free in-order protocol. A TCP connection is a virtual circuit between two computers, conceptually very much like a telephone connection but with reliable data transmission between them. A sending host divides the data stream into segments. Each segment is labeled with an explicit sequence number to guarantee ordering and reliability. When a host receives in sequence the segments, it sends a cumulative acknowledgment (ACK) in return, notifying the sender that all of the data preceding that segment's sequence number has been received. If an out-of -sequence segment is received, the receiver sends an acknowledgement indicating the sequence number of the segment that was expected. If outstanding data is not acknowledged for a period of time, the sender will timeout and retransmit the unacknowledged segments.

3. END-TO-END CONGESTION CONTROL

3.1 Congestion control Algorithms

3.1.1 Slow Start algorithm

When TCP finished the three-way handshake it bursts out as many packets allowed by the agreed window size, wnd. This was not a large problem in the small networking, but as the networks grew, and amount of connected hosts increased, these large bursts turned out to be a cause of problems. Congestion started to occur in network bottlenecks, data adding up faster than it could be forwarded or received. Therefore an algorithm to prevent immediate bursts was introduced. With the incorporation of slow start (SS) [10, 18] two new variables were introduced: the slow start threshold (ssthresh) and the congestion window (cwnd). When starting a transmission cwnd is set to 1 MSS and ssthresh is set to an arbitrary size depending on the OS used. The amount of data the sender is allowed to send is determined by min[cwnd,wnd] and since cwnd = 1 at startup only one packet is allowed. cwnd will then increase by 1 MSS for every ACK received (every RTT). This exponential growth will continue until loss detection or cwnd =ssthresh, when this happens, the congestion avoidance algorithm will take over. Slow Start algorithm is shown as follows:

Slowstart algorithm

```
Initialize: cwnd = 1
For (each segment ACKed)
    cwnd ++;
Until (congestion event or cwnd >ssthresh)
```

3.1.2 Congestion Avoidance algorithm

To avoid congestion on the network the exponential increase of cwnd must be halted. This is usually not a problem in small localized LANs where the usual limitation is the window size. However, in large WANs there are many more hosts that are supposed to share the network capacity and if all hosts would run at full capacity then congestion is hard to avoid. Congestion Avoidance (CA) handles this by lowering the cwnd increase to only 1 packet per RTT, giving cwnd a lower and linear growth. If the Retransmit Time Out (RTO) occurs, CA will consider this as a loss of packet. CA will then set ssthresh to half the current cwnd and after this resets cwnd to one and initiate a SS. Congestion avoidance algorithm is shown as follow:

Congestion avoidance algorithm

```
/* slowstart is over   */
/* cwnd > ssthresh */
  every new ACK:
      cwnd += 1/cwnd
Until (timeout)  /* loss event */
```

3.1.3 Fast retransmit algorithm

Fast Retransmit (FRet) is a short simple algorithm, treating three received DUPACKs as a sign of loss. It is unlikely that the missing packet has gone so far astray from the others that three later packets would arrive before the lost one finds its way to the receiver. FRet was created to remove the need to wait for an RTO by quickly retransmitting the lost packet after three DUPACKs, preventing unnecessary long downtime in the transmission. After the packet has been retransmitted FRet sets ssthresh=1/2cwnd and enters SS. Fast Retransmit algorithm is shown as follows:

Fast retransmit algorithm

```
If receiving 3DUPACK or RTO
Retransmit the packet
ssthresh = cwnd /2
cwnd = 1
perform slowstart
```

3.1.4 Fast Recovery algorithm

Fast recovery algorithm [19] immediately is after Fast Retransmit, after fast retransmit sends what appears to be the missing segment, congestion avoidance, but not slow start is performed. This is the fast recovery algorithm. It is an improvement that allows high throughput under moderate congestion, especially for large windows. The reason for not performing slow start in this case is that the receipt of the duplicate ACKs tells TCP more than just a packet has been lost. Since the receiver can only generate the duplicate ACK when another segment is received, that segment has left the network and is in the receiver's buffer. That is, there is still data flowing between the two ends, and TCP does not want to reduce the flow abruptly by going into slow start. The fast retransmit and fast recovery algorithms are usually implemented together as follows:

Fast retransmit algorithm (Reno)

```
If receiving 3DUPACK or RTO
Retransmit the packet
```

```
After retransmission do not enter slowstart
Enter fast recovery
```

Fast recovery algorithm (Reno)

```
Set ssthresh = cwnd/2 ;
cwnd = ssthresh + 3 ;
/* the three extra packets to compensate for the three packets
leaving the network causing the DUPACKs */
Each duplicate ACK received
 cwnd+ + ;
/* to compensate for the one leaving the network */
    transmit new packet if allowed
If new ACK cwnd=threshold ;
return to congestion avoidance
```

Fast Recovery works according to the following steps:

1. When the third duplicate ACK in a row is received, set ssthresh to one-half the current congestion window, cwnd, but no less than two segments. Retransmit the missing segment. Set cwnd to ssthresh plus 3 times the segment size. This inflates the congestion window by the number of segments that have left the network and which the other end has cached.
2. Each time another duplicate ACK arrives, increment cwnd by the segment size. This inflates the congestion window for the additional segment that has left the network. Transmit a packet, if allowed by the new value of cwnd.
3. When the next ACK arrives that acknowledges new data, set cwnd to ssthresh (the value set in step 1). This ACK should be the acknowledgment of the retransmission from step 1, one round-trip time after the retransmission. Additionally, this ACK should acknowledge all the intermediate segments sent between the lost packet and the receipt of the first duplicate ACK. This step is congestion avoidance, since TCP is down to one-half the rate it was at when the packet was lost.

3.2 Congestion control Mechanisms

3.2.1 Mechanisms

1-TCP Tahoe [16, 17]: was the first algorithm to employ three Congestion control Algorithms: slow start, congestion avoidance, and fast retransmit.

2-TCP Reno: is the most widely adopted Internet TCP protocol. It employs four Congestion control Algorithms: slow start, congestion avoidance, fast retransmit, and fast recovery [20]. When packet loss occurs in a congested link due to buffer overflow in the intermediate routers, either the sender receives three duplicate acknowledgments or the sender's retransmission timeout (RTO timer expires). In case of three duplicate ACKs, the sender activates TCP fast retransmit and recovery algorithms and reduces its congestion window size to half. It then linearly increases congestion window, similar to the case of congestion avoidance. This increase in transmission rate is slower than in the case of slow start and helps relieve congestion. TCP Reno fast recovery algorithm improves TCP performance in case of a single packet loss within a window of data. However ,performance of TCP Reno suffers in case of multiple packet losses within a window of data.

3-TCP NewReno [21]: is a modification of TCP Reno. It improves retransmission process during the fast recovery phase of TCP Reno. TCP NewReno can detect multiple packet losses. It does not exit the fast recovery phase until all unacknowledged segments at the time of fast recovery are acknowledged. Thus, as in TCP Reno, it overcomes reducing the congestion window size multiple times in case of multiple packet losses. The remaining three phases (slow start, congestion avoidance, and fast retransmit) are similar to TCP Reno. TCP NewReno exits fast recovery after receiving acknowledgement of all unacknowledged segments. It then sets congestion window size to slow start threshold and continues the congestion avoidance phase. It retransmits the next segment when it receives a partial acknowledgment. (Partial acknowledgments are the acknowledgments that do not acknowledge all outstanding packets at the onset of the fast recovery). A problem occurs with New Reno when there are no packet losses but instead, packets are reordered by more than 3 packet sequence numbers. When this happens, New Reno mistakenly enters fast recovery, but when the reordered packet is delivered, ACK sequence-number progress occurs and from there until the end of fast recovery, every bit of sequence-number progress produces a duplicate and needless retransmission that is immediately ACKed. New Reno performs as well as SACK at low packet error rates, and substantially outperforms Reno at high error rates.

4-TCP SACK: SACK algorithm [22, 23] allows a TCP receiver to acknowledge out-of order segments selectively rather than cumulatively by acknowledging the last correctly in order received segment. The receiver acknowledges packets received out of order and the sender then retransmits only the missing data segments instead of sending all unacknowledged segments. TCP Reno with SACK behaves similarly to TCP Tahoe and TCP Reno, which are robust in case of out of order packet arrivals. However, TCP with SACK helps improve performance in case of multiple packet losses. During the fast recovery phase, SACK maintains a variable called pipe that represents the estimated number of outstanding packets. The sender only sends new or retransmitted data when the estimated number of packet in a router is smaller than the congestion window. The pipe variable is incremented by one when the sender either sends a new segment or retransmits an old one. It is decremented by one when the sender receives the duplicate ACK with a SACK option [24].

5-TCP Vegas: Until the mid 1990s, all TCPs set timeouts and measured round-trip delays were based upon only the last transmitted packet in the transmit buffer. Researchers introduced TCP Vegas [25], in which timeouts were set and round-trip delays were measured for every packet in the transmit buffer. TCP Vegas detects congestion at an incipient stage based on increasing Round-Trip Time (RTT) values of the packets in the connection unlike other flavors like Reno, NewReno etc. which detect congestion only after it has actually happened via packet drops [26, 27, 28]. The algorithm depends heavily on accurate calculation of the Base RTT value. If it is too small, then throughput of the connection will be less than the bandwidth available, while if the value is too large then it will over run the connection. A lot of research is going on regarding the fairness provided by the linear increase/decrease mechanism for congestion control in Vegas. One interesting caveat is when Vegas is inter-operated with other versions like Reno. In this case, performance of Vegas degrades because Vegas reduces its sending rate before Reno as it detects congestion early and hence gives greater bandwidth to co-existing TCP Reno flows.

6-TCP Westwood [29]: is yet another improvement in the TCP Reno family line. The Fast Recovery algorithm from TCP New Reno has been modified. To help gain faster recovery bandwidth estimation (BWE) algorithm also has been added [30]. This BWE function is what makes TCP Westwood standout. Influenced by TCP Vegas, BWE uses the RTT and the amount of data that has been sent during this interval to calculate an estimate of the currently successful transfer rate. The bandwidth estimate is then used when a loss is detected, setting cwnd and sssthresh at values near the estimation. The main purpose behind this is to improve the throughput in wireless links, where loss is more often caused by link failure than by congestion. There is also the general benefit that starting CA at higher values will lower the recovery time on most networks, thus lowering the transfer times.

3.2.2 Discussion

The Transmission Control Protocol was standardized in 1981 with the publication of RFC 793. After only a short period it was evident that it had some flaws in its behavior and a new version named Tahoe was released. In [31-35], the authors discuss the different mechanisms for TCP congestion control. Figure 1 indicates the TCP Inherence start by TCP Tahoe which added a Slow Start (SS) function, which started the transmission of data slowly but exponentially. An algorithm named Congestion Avoidance (CA) was also added, designed to slow the growth of the senders output lowering the possibility of causing congestion. The final algorithm added in the TCP Tahoe version is called Fast Retransmit (FRet). Fast Retransmit resend the first unacknowledged packet in the send buffer after receiving three DUPACKs after each other instead of waiting for a RTO. TCP SACK is a feature of Selective Acknowledgement, telling the sender what packets have been successfully received at the receiver and not just that a packet has been lost. TCP SACK works exceptionally well, compared to ordinary TCP clones, on a network with problems with multiple packet losses. This help keeping the retransmission queue small and saves time waiting not needing to wait for the next ACK to see if something else is missing. TCP SACK can be used with many later versions of TCP. TCP Reno is an upgrade of TCP Tahoe. Adding an algorithm Fast Recovery (FRec), designed to help TCP recover faster to maximum output after suffering a packet loss. Fast Recovery keeps the flow going instead of performing a SS. TCP New Reno was an improvement of the cooperation between FRet and FRec. To improve the behavior, when faced with rapid multiple packet losses on connections that cannot use the TCP SACK feature. Still keeping the flow going when receiving DUPACKs, TCP New Reno is less careful on how to update the cwnd and ssthresh variables, usually ending up giving them higher values, than its predecessor. TCP Vegas is more of a spinoff TCP clone than part of the evolution of TCP Tahoe. Using a time based estimate of the capacity and limiting the output to avoid congestion, TCP Vegas is a smooth and intelligent TCP clone. However, it does not work well with the TCP Reno family, due to the more aggressive nature of those TCP versions. On its own or together with other TCP Vegas instances it is impressively fair in its sharing and smooth in its throughput. TCP Westwood uses an advanced bandwidth estimation (BWE) to try and figure out the capacity of the network and uses this knowledge to lower the loss in throughput caused by packet loss. This BWE takes the sender output as a measure of the bandwidth of the network and sets the ssthresh accordingly when suffering a loss.

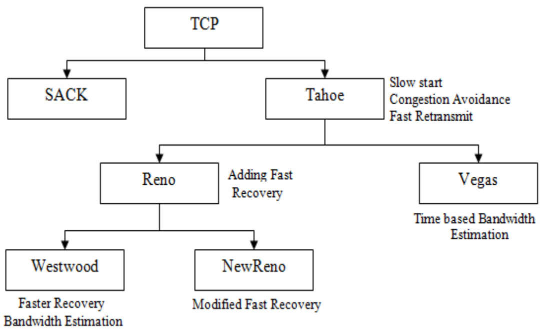

Fig 1: TCP Inherence.

4. TCP WESTWOOD MECHANISM

TCP Westwood congestion control algorithm [36] use a bandwidth estimation, it executed at sender side of a TCP connection. The congestion window dynamics during slow start and congestion avoidance are unchanged. The general idea is to use the bandwidth estimate BWE to set the congestion window (cwin) and the slow start threshold (ssthresh) after a congestion episode. In TCP Westwood the sender continuously computes the connection BWE which is defined as the share bottleneck 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, the sender resets the congestion window and the slow start. Threshold based on BWE. The packet loss is suspected with a reception of three duplicates ACKs or timeout expiration. Another important element of this procedure is the RTT estimation. That is because the congestion window is set precisely to BWE * RTT after indication of packet loss.

4.1 End-to-End Bandwidth measurement

A fundamental design philosophy [37] of the Internet TCP congestion control algorithm is that it must be performed end-to-end. The network is considered as a "black box". A TCP source cannot receive any explicit congestion feedback from the network. Therefore the source, to determine the rate at which it can transmit, must try the path by progressively increasing the input load until feedback signals, that the network capacity has been reached. The key idea of the TCP Westwood, presented before, is to continuously estimate, at the TCP sender, the packet of the connection. This is done by monitoring the ACK reception rate. The estimated connection rate is then used to improve the efficiency of slow start and congestion control algorithms. If an ACK is received at source at time t2, this implies that a corresponding amount of data d2 has been received by the TCP receiver. Therefore, we can measure the following sample of bandwidth used by that connection as: $b2 = d2/(t2-t1)$, where t1 is the time of the previous ACK that was received. An average of the samples is calculated and used to calculate the estimation of the available bandwidth. This bandwidth estimation works in the following way:

Bandwidth estimation (BWE) algorithm
$BWE = b_k = \alpha_k b_{k-1} + (1 - \alpha_k) [(b_k + b_{k-1}) /2]$

```
Where bk = sample bandwidth = dk / t k − t k−1
      where dk = amount of bytes acknowledged by ACK k,
            t k = arrival time of ACK k,
α k = [2τ − Δ(t k − t k−1)] / [2τ + Δ(t k − t k−1)]
   where τ : is the cut- off frequency of this Tustin filter
```

4.2 Round-Trip Time Estimation

When a host transmits a TCP packet to its peer, it must wait a period of time for an acknowledgment. If the reply does not come within the expected period, the packet is assumed to have been lost and the data is retransmitted. The problem is the protocol does not define the length of the period to wait. All modern TCP implementations seek to find a proper waiting time by monitoring the normal exchange of data packets and developing an estimate of how long is "too long". This process is called Round-Trip Time (RTT) estimation [38]. RTT estimates are one of the most important performance parameters in a TCP exchange, especially when you consider that on an indefinitely large transfer, all TCP variants eventually drop packets and retransmit them, no matter how good the quality of the link. RTT is a key component of TCP Westwood algorithm.

4.3 Setting cwnd and ssthresh in TCPW

Let us first assume that a sender has estimated BW, and let us describe in this subsection how is used to properly set cwnd and ssthresh after a packet loss indication. First, we note that in TCPW, congestion window increments during slow start and congestion avoidance remain the same as in Reno, i.e. exponential and linear, respectively. A packet loss is indicated by the follows: (a) the receipt of 3 DUPACKs, or (b) a coarse timeout expiration. In case the loss indication is 3 DUPACKs, TCPW sets cwnd and ssthresh as follows.

4.3.1 Algorithm after three duplicate ACK.

The pseudo code of the TCP Westwood algorithm after three duplicate acknowledgements is:

```
After 3 DUPACKS
If receiving 3 DUPACKS
Set ssthresh =(BWE*RTTmin) /seg_size;
and if cwnd > ssthresh then set cwnd = ssthresh ;
enter congestion avoidance
```

In the pseudo-code, seg_size indicates the length of TCP segments in bits. During the congestion avoidance phase the sender is trying for extra available bandwidth. If three duplicate ACKs are received, the network capacity might have been reached or that in case of wireless links, one or more segments have were dropped due to sporadic losses.

4.3.2 Algorithm after timeout

The pseudo code of TCP Westwood algorithm after timeout is:

```
After Timeout
If RTO then set
    ssthresh = (BWE*RTTmin) /seg_size;
    if (ssthresh < 2) ssthresh =2; end if ;
    cwin = 1;
end if
enter slow start;
```

The rationale of the algorithm above is that after a timeout, cwin is set to equal one and ssthresh is set BWE. A speedy recovery is ensured by setting ssthresh to the bandwidth estimation at the time of timeout expiration.

5. TCP WESTWOOD NEW MECHANISM

5.1 Enhanced TCP Westwood congestion avoidance algorithm

TCP Westwood is a rate based scheme extending the TCP Reno.In Transmission Control Protocol (TCP), the congestion window [39] is one of the factors that determine the number of bytes that can be outstanding at any time. Maintained on the sender, this is a means of stopping the link between two places from getting overloaded with too much traffic. The size of this window is calculated by estimating how much congestion there is between the two places. The sender maintains the congestion window. When a connection is set up, the congestion window is set to the maximum segment size (MSS) then the size doubled every ACK until cwnd >ssthresh then go to congestion avoidance state where size cwnd=cwnd+1/Cwnd until congestion occur we propose enhanced congestion avoidance algorithm as follow: TCP WestwoodNew takes the data-receiving rate as a metric to predict the network conditions. TCP WestwoodNew estimated BW as BWcurrent (the current BW after receive new ACK) then divide it on BWprevious (BW before receive the same new ACK) the result is the BW ratio if the BW ratio < 1 this indicate that there is an increase in the network load therefore the Cwnd should be constant .else if BW ratio > 1 indicates that there is an decrease in the network load therefore the Cwnd should be increased the Cwnd is adjusted based on the network conditions estimate. These modifications constitute the foundation for an efficient congestion avoidance strategy over heterogeneous environments with wire-line or wireless networks.

The TCP WestwoodNew congestion avoidance algorithm:

```
Congestion avoidance
slow start is over */
/*cwnd > ssthresh */
Every Ack:
Estimate BWE
Set BWE = BWcurrent
BWratio = BWcurrent/BWprevious
If (1.5>BWratio >= 1)
  cwnd = cwnd + 1/cwnd
If (BWratio >= 1.5)
   cwnd = cwnd + 2/cwnd
Else if (BWratio < 1)
      cwnd = cwnd + 0
Until (timeout or 3 DUPACKs)
```

Where BWcurrent : the current BW after receive new ACK and BWprevious : BW before receive the same new ACK

5.2 Modified RTO Calculation Algorithm

In the recent years the variety of Internet links with different properties has increased dramatically. The high speed networks have reached Gigabit rates, whereas the increasing number of mobile wireless access networks has introduced a prolific number of mobile hosts attached to the Internet through a slow, wireless links. Moreover, the challenging characteristics of wireless links, in particular high packet loss rate or delays due to various reasons such as link-layer retransmissions or hand-off between the points of attachment to the Internet, have introduced a large set of problems for the Internet transport protocols. The TCP protocol is the dominant Internet transport protocol and its congestion control algorithms are essential for the stability of the Internet. Because these algorithms have a strong effect on TCP performance, finding solutions to improve TCP performance over slow wireless links. The traditional problem regarding the use of TCP over wireless links or other challenging channels has concerned TCP congestion control. If a packet is lost, TCP interprets it as an indication of congestion and a TCP sender needs to reduce its transmission rate. Hence, TCP performance deteriorates with increasing packet loss rate. If the packet loss occurred due to corruption, reducing the TCP transmission rate is, however, the wrong action to take. TCP uses the fast retransmit mechanism to trigger retransmissions after receiving three successive duplicate acknowledgements (ACKs). If for a certain time period TCP sender does not receive ACKs that acknowledge new data, the TCP retransmission timer expires as a back-off mechanism. When the retransmission timer expires, the TCP sender retransmits the first unacknowledged segment assuming it was lost in the network. Because a retransmission timeout (RTO) [40] can be an indication of severe congestion in the network, the TCP sender resets its congestion window to one segment and starts increasing it according to the slow start algorithm. However, if the RTO occurs spuriously and there still are segments outstanding in the network, a false slow start is harmful for the potentially congested network as it injects extra segments to the network at increasing rate. TCP should be able to adjust its RTO value when needed. This is realized according to the identified loss model within the network. First of all, let us note that when congestion is detected within the network, the RTO estimation is not changed and remains similar to the one used by TCP New Reno. Alternatively, in the case of wireless channel errors, no RTO calculation or adjustment is necessary as the network conditions are supposed to be unvaried. In the case of link failure, the RTO value has to be modified based on the characteristics (length, load, and link qualities) of the new route discovered by the routing protocol. So, after link loss recovery by the ad hoc routing protocol, we may observe that both the propagation and queuing delays change suddenly. As RTT is one of the most direct TCP connection characterization parameter that reflects network links conditions, our estimation algorithm will be depending on it. It is obvious that the number of hops as well as the load of the route between the TCP sender and receiver affects the RTT value over that connection. Thus, the characteristics of the new discovered route could be represented by RTT values over that route. Thus, the RTO value would be updated as follows: RTO new = (RTT new /RTT old) RTO old (1), where RTTnew is the new round trip time estimation after congestion recovery and RTTold the round trip time estimation before congestion.

6. PERFORMANCE EVALUATION

6.1 Evaluation criteria

The evaluation criteria are: (1) ***Throughput***, that refers to the number of packets sent by the source and correctly received by destination. (2) ***Delay***, is the required time of two-way communication, it may range from a very few microseconds, it can be measured as per packet transfer times. (3) ***Packet losses*** (packet drop rates), can be measured as the number of dropping packets per unit of time, it may also defined as the packets that are retransmitted again from the source because the packet is either corrupted or lost. (4) ***Congestion window*** is flow control imposed by the sender, while the advertised window is flow control imposed by the receiver. The former is based on the sender's assessment of perceived network congestion, and the latter is related to the amount of available buffer space at the receiver for this connection.

6.2 Simulation setup

The simulations are often used for understanding and prediction of the behavior of protocols and data streams in networks. All simulation results in this paper are obtained using NS2 [41] simulator. Figure 2 shows the network topology that is used for the simulation. The topology has five nodes connected to each other via four TCP connections; each link is labeled with its bandwidth capacity and its delay.

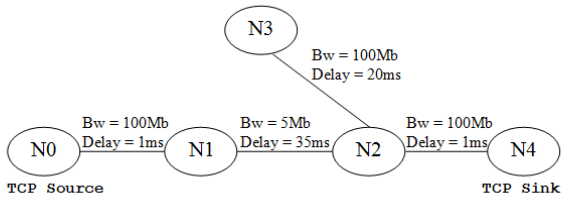

Fig 2: Network topology

6.3 Simulation Results

In Figure 3, the network throughput is presented for TCP Reno, Newreno, Sack, Vegas, Tahoe, Westwood, and Westwoodnew. It is noted that TCP westwoodnew has highest throughput on the steady state time. This is because TCP Westwood new interested by the network status in now and the past, therefore it determines the network status by a large percentage of accuracy TCP Westwood. It determines the network status by estimating the current BW without looking to the previous BW which determines the previous network status, therefore the rate of the sending packets in TCP Westwood and base on that TCP Westwood sending the same rate of packets whether the network status is heavy or not heavy by TCP WestwoodNew determine the rate of packets sending based on the network status if it not heavy TCP Westwood new increase the rate of the sending packets which is increase the throughput performance in the network if network status is heavy TCP WestwoodNew remain the rate of sending packets constant.

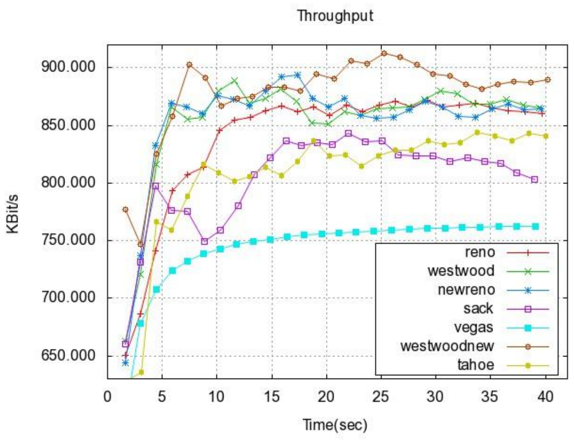

Fig 3: Throughput versus time for Reno, Newreno, Sack, Vegas, Tahoe, Westwood, and Westwoodnew.

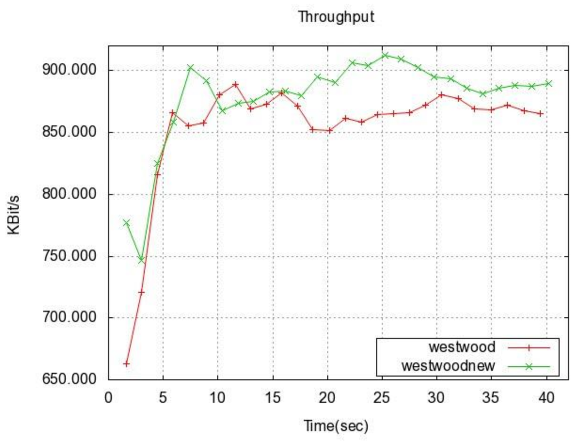

Fig 4: Throughput versus time for Westwood and Westwoodnew.

When comparing the Westwood with westwoodnew, it noted from the zoom of figure 3, in Figure 4 that depicts the throughput Westwoodnew is improved because of taking the network status in now and the past to adapting the congestion window.

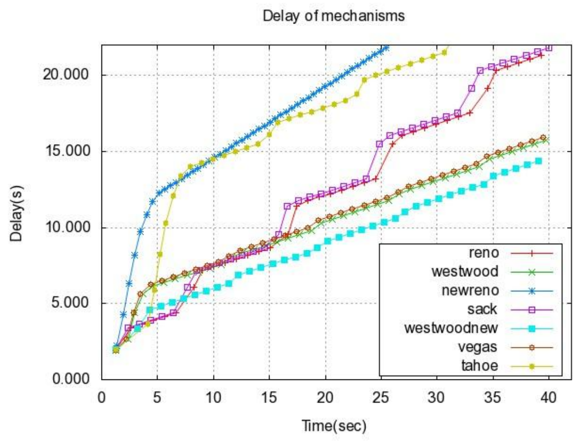

Fig 5: Delay versus the time for Reno, Newreno, Sack, Vegas, Tahoe, Westwood, and Westwoodnew.

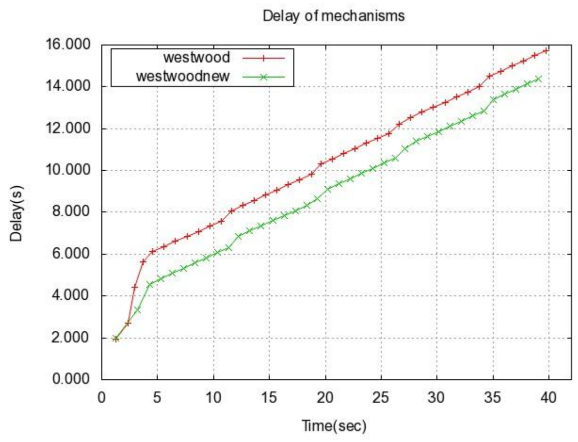

Fig 6: Delay versus the time for Westwood and Westwoodnew.

Figure 5 shows delay changes versus the time for TCP Reno, Newreno, Sack, Vegas, Tahoe, Westwood, and Westwoodnew, The delay measured here as a flow based in terms of per packet transfer time. As shown from the figure, WestwoodNew reduces the delay, unlike Westwood which have higher delay than it. That is because cwnd of WestwoodNew determined by more accurate than cwnd of Westwood this determination based on the degree the network get congested so WestwoodNew could send more new packets depending on the network status that reduces the delay time. Also WestwoodNew determine RTO based on new RTT and old RTT this make it avoid to send the packets once again without necessary to that and avoid a false slow start.. When comparing the westwood with westwoodnew, it noted from the zoom of figure 5, in Figure 6 that depicts the delay of westwood new is improved as the minimum delay.

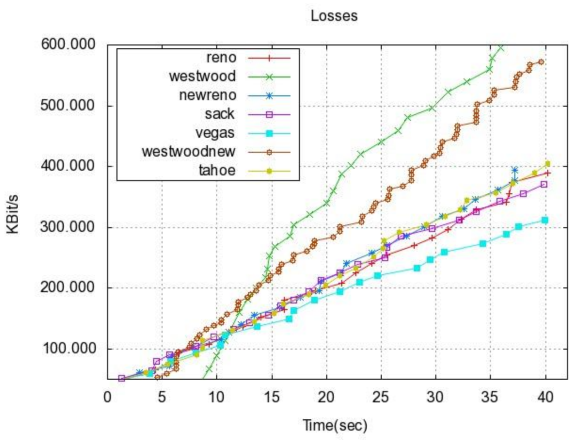

Fig 7: Packet Losses versus the time for Reno, Newreno, Sack, Vegas, Tahoe, Westwood, and Westwoodnew.

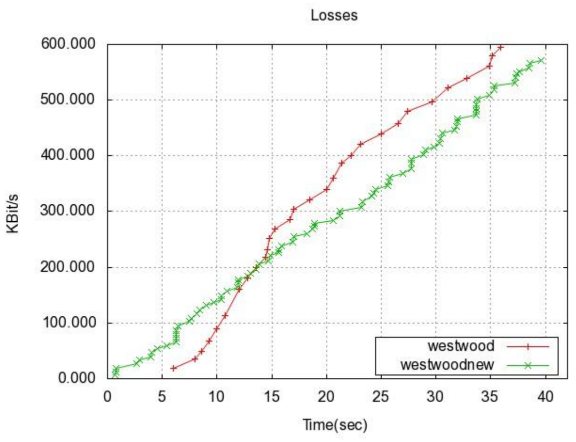

Fig 8: Packet Losses versus the time for Westwood and Westwoodnew.

Figure 7 describes the packet losses versus the time for TCP Reno, Newreno, Sack, Vegas, Tahoe, Westwood, and WestwoodNew. The figure shows that the dropped packets by TCP Westwood have the highest dropping rate over the simulation time cause in fast recovery Westwood send more no. of packets than other mechanisms which reduce the no. of sending packets in fast recovery, WestwoodNew improved Westwood by more adapting to the no. of sent packets in fast recovery depending on the past and current network status. When comparing the westwood with westwoodnew, it noted from the zoom of figure 7, in Figure 8 that depicts the packet losses of westwoodnew is improved more than westwood as less packet losses with highest throughput and minimum delay.

Figure 9 shows the change of the congestion window in packets with the time. With TCP Westwood and TCP WestwoodNew, we find that when congestion detected TCP Westwood and TCP

Westwood New determines the cwnd based on the bandwidth estimation in fast recovery then go to congestion avoidance state with new cwnd in TCP Westwood congestion avoidance we find that when new ACK is received cwnd is increased by 1/cwnd in TCP WestwoodNew congestion avoidance if new ACK is received, the rate of cwnd increasing determine based on network status if heavy cwnd remain constant , else if network status not heavy cwnd increasing by 1/cwnd or 2/cwnd, so we find that the rate of TCP WestwoodNew *Cwnd* larger than the rate of TCP Westwood *cwnd*.

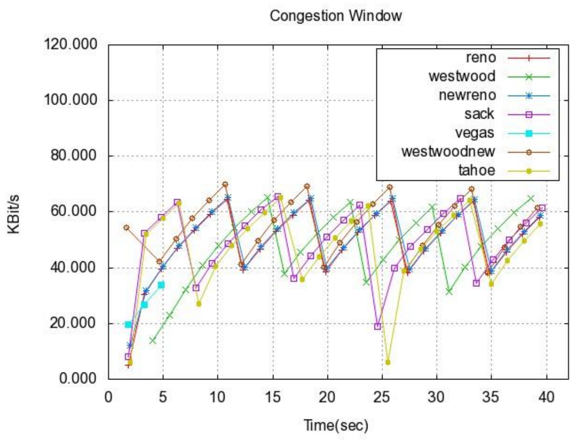

Fig 9: Behavior of the Congestion Window for Reno, Newreno, Sack, Vegas, Tahoe, Westwood, and Westwoodnew.

Figure 10 depicts the Behavior of the Congestion Window for Westwood and WestwoodNew. We note that the *cwnd* of WestwoodNew is growing than *cwnd* of Westwood, because WestwoodNew exploits the available bandwidth by more precious to enlarge its congestion window and sending rate.

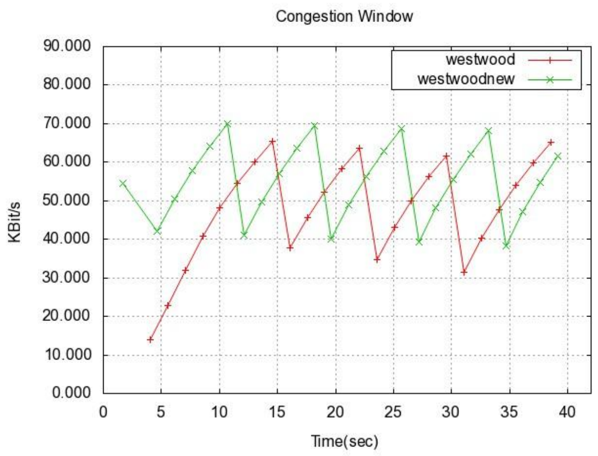

Fig 10: Behavior of the Congestion Window for Westwood and Westwoodnew.

7. CONCLUSION

In this paper we have proposed a new version of the TCP Westwood protocol called TCP WestwoodNew, aimed to improving performance of Westwood under random or sporadic losses. The new version has been tested through NS2 simulation, TCP WestwoodNew introduced a new Congestion Avoidance congestion control algorithm, also introduced a new estimation to RTO based on RTT, TCP WestwoodNew is an enhanced to TCP Westwood congestion control protocol designed to effectively transmit with a rate that utilizes a fair link capacity sharing, and improves the throughput performance in the network. The Congestion Avoidance algorithm was developed by applying dynamic congestion window adaptation mechanism. The proposed mechanism improves the TCP throughput over that of TCP Westwood. TCP WestwoodNew provides good performance in terms of reducing the delay.

8. REFERENCES

[1] M. Kalpana1 and T. Purusothaman, “Performance Evaluation of Exponential TCP/IP Congestion Control Algorithm”, International Journal of Computer Science and Network Security (IJCSNS), VOL.9 No.3, March 2009.

[2] Ka-Cheong Leung, Victor O.K. Li, Daiqin Yang, "An Overview of Packet Reordering in Transmission Control Protocol (TCP): Problems, Solutions, and Challenges," IEEE Transactions on Parallel and Distributed Systems, pp. 522-535, April, 2007

[3] Seifeddine Kadry, Issa Kamar, Ali Kalakech, Mohamad Smaili Robust, "TCP: An Improvement on TCP Protocol", Journal of Theoretical and Applied Information Technology 2005.

[4] Huaizhong Han, Srinivas Shakkottai, C. V. Hollot, R. Srikant, and Don Towsley, " Multi-path TCP: a joint congestion control and routing scheme to exploit path diversity in the internet", IEEE/ACM Transactions on Networking (TON), Vol. 14, Issue 6, December, 2006, PP. 1260-1271. DOI=10.1109/TNET.2006.886738 http://dx.doi.org/10.1109/TNET.2006.886738

[5] Nga J.H.C., Iu H.H.C., Ling S.H., Lam H.K. "Comparative study of stability in different TCP/RED models", Chaos, Solitons and Fractals, the interdisciplinary journal of Nonlinear Science, and Nonequilibrium and Complex Phenomena, Vol. 37, Issue 4, August 2008, pp. 977-987.

[6] P. Kuusela, P. Lassila, J. Virtamo and P. Key, "Modeling RED with Idealized TCP Sources", 9th IFIP Conference on Performance Modeling and evaluation of ATM & IP networks, 2001.

[7] S. H. Low, F. Paganini, and J. C. Doyle, ”Internet Congestion Control”, IEEE Control Systems Magazine, FEB 2002, pp:28-43.

[8] L. Yao-Nan, and H. Ho-Cheng, ”A New TCP Congestion Control Mechanism over Wireless Ad Hoc Networks by Router-Assisted Approach”, International Conference on Distributed Computing Systems, JUN 2007, pp:84-84.

[9] S. Ryu, C. Rump, and C. Qiao, ”Advances In Internet Congestion Control”, IEEE Communications Surveys & Tutorials, Third Quarter, vol.5(1), 2003, pp:28-39.

[10] C. Wanxiang, S. Peixin, and L. Zhenming, ”Network-assisted congestion control”, Info-tech&Info-net International Conferences, vol.2, JUN 2001, pp:28-32.

[11] K Fang-Chun., and X. Fu, ”Probe-Aided MulTCP: an aggregate congestion control mechanism”, ACM SIGCOMM Computer Communication Review, Vol.38 (1), JAN 2008, PP: 17-28.

[12] Andrzej Chydzinski, Agnieszka Brachman, "Performance of AQM Routers in the Presence of New TCP Variants," Advances in Future Internet, International Conference on, pp. 88-93, 2010 Second International Conference on Advances in Future Internet, 2010.

[13] Hsiuyuan Chu, Kuohui Tsai, and Wenjer Chang, " Fuzzy control of active queue management routers for transmission control protocol networks via time-delay affine Takagi-Sugeno fuzzy models", International Journal of

Innovative Computing, Information and Control, Volume 4, Number 2, February, 2008.

[14] Luigi A. Grieco and Saverio Mascolo, "Performance evaluation and comparison of Westwood+, New Reno, and Vegas TCP congestion control", SIGCOMM Computer Communication Review, Vol. 34, Issue 2, April, 2004. PP. 25-38. DOI=10.1145/997150.997155 http://doi.acm.org/10.1145/997150.997155

[15] W. Boulevard, and A. Way," Transmission Control Protocol", RFC 793, September 1981.

[16] V. Jacobson and M. J. Karels, "Congestion avoidance and control", In ACM Computer Communication Review; Proceedings of the Sigcomm'88 Symposium, volume 18, pages 314–329, Stanford, CA, USA, August 1988.

[17] Hanaa A. Torkey, Gamal M. Attiya and I. Z. Morsi, "Performance Evaluation of End-to-End Congestion Control Protocols", Menoufia journal of Electronic Engineering Research (MJEER), Vol. 18, no. 2, pp. 99-118, July 2008.

[18] Dirceu Cavendish, Kazumi Kumazoe, Masato Tsuru, Yuji Oie, and Mario Gerla, "CapStart: An Adaptive TCP Slow Start for High Speed Networks", In Proceedings of the 2009 First International Conference on Evolving Internet (INTERNET '09). IEEE Computer Society, Washington, DC, USA, 15-20. DOI=10.1109/INTERNET.2009.10 http://dx.doi.org/10.1109/INTERNET.2009.10

[19] N. Parvez, A. Mahanti, and C. Williamson, "TCP NewReno: Slowbut- Steady or Impatient?", IEEE International Communications Conference, ICC '06, vol. 2, June 2006, pp: 716-722.

[20] M. Allman, V. Paxson, and W. Stevens. RFC 2581 - TCP Congestion Control. The Internet Society, 1999.

[21] Hanaa A. Torkey, Gamal M. Attiya and I. Z. Morsi, "Enhanced Fast Recovery Mechanism for improving TCP NewReno", Proceedings of the 18th International Conference on Computer Theory and Applications (ICCTA08), pp. 52-58, Alexandria, Egypt, 11-13 October 2008.

[22] V. Jacobson and R. Braden. RFC 1072 - TCP Extensions for Long Delay Paths. October 1988.

[23] Beomjoon Kim, Dongmin Kim, and Jaiyong Lee, "Lost Retransmission Detection for TCP SACK", IEEE COMMUNICATIONS LETTERS, VOL. 8, NO. 9, September 2004.

[24] V. Jacobson, "Modified TCP congestion avoidance algorithm",url: ftp://ftp.ee.lbl.gov/email/vanj.90apr30.txt.

[25] L. S. Brakmo and L. L. Peterson, "TCP vegas: End to end congestion avoidance on a global internet", IEEE Journal on Selected Areas in Communications, 13(8):1465–1480, 1995.

[26] K.N. Srijith, Lillykutty Jacob1, and A.L. Ananda, "TCP Vegas-A: Improving the Performance of TCP Vegas" Computer communications 28 (2005), pp. 429-440.

[27] S. H. Low, L. L. Peterson, and L. Wang, "Understanding TCP Vegas: A Duality Model", Journal of the ACM, Vol.49 (2) , March 2002, pp:207-235.

[28] L. S. Brakmo, and L. L. Peterson, "TCP Vegas: End to End Congestion Avoidance on a Global Internet", IEEE Journal on Selected Areas in Communications, vol.13 no.8, October 1995, pp: 1465-1480.

[29] M. Gerla, M.Y. Sanadidi, RenWang, A. Zanella, C. Casetti, and S. Mascolo, "TCP westwood: congestion window control using bandwidth estimation", In IEEE Global Telecommunications Conference, GLOBECOM '01, volume 3, pages 1698 – 1702, November 2001.

[30] Gerla, B.K.F. Ng, M.Y. Sanadidi, M. Valla, R. Wang, "TCP Westwood with adaptive bandwidth estimation to improve efficiency/friendliness tradeoffs", Computer Communications 27 (2003) pp. 41-58.

[31] Ayman EL-SAYED, Nawal EL-FESHAWY and Shimaa HAGAG "A Survey of Mechanisms for TCP Congestion Control", International Journal of Research and Reviews in Computer Science (IJRRCS), Vol. 2, No. 3, June 2011.

[32] Laxmi Subedi, Mohamadreza Najiminaini, and Ljiljana Trajkovi "Performance Evaluation of TCP Tahoe, Reno, Reno with SACK, and NewReno Using OPNET Modeler" Communication Networks Laboratory http://www.ensc.sfu.ca/research/cnl OPNET technologies, 2008

[33] Maxim Podlesny and Carey Williamson "Providing Fairness Between TCP NewReno and TCP Vegas with RD Network Services" Department of Computer Science, University of Calgary, 2010.

[34] Damon Wischik, Costin Raiciu, Adam Greenhalgh, and Mark Handley, "Design, implementation and evaluation of congestion control for multipath TCP", In Proceedings of the 8th USENIX conference on Networked systems design and implementation (NSDI'11), USENIX Association, Berkeley, CA, USA, PP.8-8, 2011.

[35] Nadim Parvez, Anirban Mahanti, and Carey Williamson, "An Analytic Throughput Model for TCP NewReno", IEEE/ACM TRANSACTIONS ON NETWORKING, Vol. 18, No. 2, April 2010.

[36] Saverio Mascolo and Francesco Vacircay, "The effect of reverse traffic on the performance of new TCP congestion control algorithms " University of Rome "La Sapienza" 2006.

[37] Prof.K.Srinivas, Dr.A.A.Chari and N.Kasiviswanath "Updated Congestion Control Algorithm for TCP Throughput improvement in Wired and Wireless Network", Vol. 9 Issue 5 (Ver. 2.0), January 2010.

[38] Salem Belhaj, and Moncef Tagina, "VFAST TCP: A delay-based enhanced version of FAST TCP" International Journal of Computer and Information Science and Engineering 2;2 2008.

[39] Nandita Dukkipati, Tiziana Refice and Yuchung Cheng "An Argument for Increasing TCP's Initial Congestion Window" Google Inc. 2010

[40] Bhavika Gambhava, N. J. Kothari and Dr. K. S. Dasgupta, "Analysis of RTO Caused by Retransmission Loss to Combat Channel Noise", International Journal of Computer Applications (0975 – 8887) Vol. 1– No. 8, 2010.

[41] Ns-2 network simulator (ver. 2). LBL, URL: http://wwwmash.cs.berkeley.edu/ns.