TCP Cross Layer Adaptive Policy

Throughput Optimization over Wireless Links

Purvang D. Dalal *(Author)*
Electronics & Communication Department
Dharmsinh Desai University
Nadiad, Gujarat, INDIA.
pur_dalal@yahoo.com , ec@ddu.ac.in

K. S. Dasgupta *(Author)*
Director, SITTA
Space Application Center, ISRO
Ahmedabad, Gujarat, INDIA
ksd@sac.isro.gov.in

***Abstract*—TCP is the most common transport layer protocol used in Internet. It was designed primarily for wired networks, assuming reliability at lower layers and packet losses are considered as an indication of congestion. The characteristic of wireless links is very different from wired links, particularly in terms of loss behavior. In wireless networks, most packet losses are due to poor link quality and intermittent connectivity, which TCP may falsely assume as congestion. These wrongly trigger the congestion control mechanism of TCP, resulting into end-to-end throughput degradation. The problem is further compounded by the large propagation delay common in such environments. TCP-ACC proposed earlier attempts to combat this problem by adopting a dynamic value of congestion window for the slow start. In this paper, we extend TCP-ACC with the help of a cross layer solution to differentiate between a loss due to congestion and a loss due to noise in the wireless link based on the number of attempts of MAC layer retransmissions.**

Keywords-slow start, congestion control, retry limit, retry count, RTT, duplicate acknowledgment, cwnd, ssthresh.

I. INTRODUCTION

TCP interpret packet loss as an indication of network congestion [RFC2581]. As a policy, TCP reduces its transmission rate to reduce the network congestion as a correcting action against packet loss. This policy dramatically reduces TCP performance over wireless links. This is mainly due to the unpredictable characteristics of a wireless network with (a) *High Bit Error Rate* and (b) *higher delay*.

In wireless links value of *RTT* is very high and highly varying because of changing network and link conditions. As a result during a TCP connection, it usually takes more time to come out of the slow start phase. This will restrict TCP to utilize maximum of the available bandwidth, which in turns, pull down the throughput [3]. A scheme referred as *FastTCP* was proposed to address the issue, [RFC2414] particularly for the wireless networks. However the Fast TCP has the following drawbacks; (a) It assumes some static value (other than 1) for *cwnd* in the beginning of a new TCP connection. However considering the unpredictable characteristic of the wireless networks the same value may not suite best to all network conditions. (b) The Fast TCP scheme attempts slow start phase with higher value of *cwnd* only in the beginning of a new TCP connection, however it is not using the same strategy for the other slow start phases then after.

In conventional cellular wireless networks, non-negligible random wireless channel error rates also contribute to losses [2][8]. In wireless networks, in addition to congestion and wireless errors, mobility serve as another primary contributor to losses perceived by connections. By assuming network congestion as the cause of these errors, TCP does the wrong thing; it drastically reduces it's transmit window and deploys the slow start algorithm. This results in underutilized network bandwidth as discussed above and applications experiencing increased network latency.[7] In the networks other than wired, usually the reasons of packet losses are other than congestion, leading to an inappropriate triggering of TCP congestion control algorithm; results into unnecessary reduction in throughput and, hence the performance degradation.

The proposed scheme by the author attempts to assign dynamic value to the cwnd , for all of the slow start phases during the entire TCP connection. At the same time it also uses feedback from MAC layer to inform TCP about link failure. This will prevent TCP to trigger congestion control algorithm inappropriately. also reduced to certain extent.

II. RELATED WORK

Because of its inability to distinguish between packet losses due to congestion and due to transmission errors, TCP performs poorly on wireless links. Maximum throughput conceived by the TCP connection, when the TCP congestion window is equal to the *Bandwidth Delay product* of the channel, at which the maximum capacity of the channel is utilized [5] Several interesting approaches have already been proposed to improve TCP performance over wireless networks.[5][6]9][10][11] However, many of these schemes require TCP-specific actions on the part of intermediate nodes on the path from TCP sender to TCP receiver. Such schemes are sometimes referred as being TCP-aware [8].

In the split connection approach, a TCP connection is broken into two TCP connections – one from the sender to the base station, another from the base station to the receiver [11]. Thus, wireless errors can be handled locally, by means of retransmissions from the base station. This approach, however, violates the end to end reliability semantics of TCP.

The proposal described in [12] employs TCP Westwood. It lies on bandwidth estimate at the sender side and is based on the received sequences of acknowledgements. The estimate is realized in order to improve the TCP congestion control mechanism and to minimize the effect of packet losses due to the wireless context. The proposal describes in [5] improves the bandwidth estimate at the TCP layer, particularly the performance is effected by the random losses. The TCP jersey version described in [6] has been specifically developed to detect TCP congestions in a wireless network and to react consequently. Two mechanisms are thus installed: an estimate of the available bandwidth allowing the sender to adjust its flow throughput in the case of congestion, and a congestion warning carried out by intermediate nodes and based on marking of packets towards the sender.

All these solutions are proposed in the general context of wireless networks and do not take into accounts the MAC layer specificities and the interactions with it. These mechanisms are thus not optimized for 802.11 wireless networks where a first level of error recovery carried out at the MAC layer.

III. ADAPTIVE CONGESTION CONTROL

Considering merits of TCP NewReno over other TCP flavors, namely Tahoe, Reno and SACK [1], it is adopted as a base TCP protocol for modifications. In general SACK TCP needs one RTT to recover all losses in a single transmission window, where as NewReno TCP requires L*RTT to finish, where L is the number of lost segments. NewReno TCP however, overcomes multiple segment loss without any overhead, and it can support existing receiver side equipment without any modifications [4]. However, few weaknesses of to TCP NewReno, while implemented over wireless links, are highlighted as follows.

- TCP NewReno uses slow start mechanism when data transfer is initiated; same as TCP [RFC2582]. If network links are slow, it takes long time to grow congestion window & hence results into an excessive delay, while probing network capacity. In this phase throughput will be degraded due to relatively less transmission rate [9].
- TCP New Reno modifies *cwnd = cwnd/2* after successfully recovering from last fast retransmit phase [RFC2001]. Which unnecessarily slowdowns the transmission rate of sender by 2. Some estimation based on *ACK* filtering [12] or analysis of *ssthresh* variable during previous data transfer interval may be useful in tuning transmission rate of sender more appropriately particularly in case of packet losses due to reasons other than congestion.
- TCP New Reno does not have the capability to distinguish and to isolate congestion loss from wireless loss. Under these circumstances, it reacts to wireless loss with drastic reduction of the sender transmission rate, whereas the best strategy would be to increase or maintain the retransmission rate [8]. In erroneous channel it is obvious that discovering the loss and quick retransmission is desirable. However with disconnection quick retransmissions are not useful until the link is re-established, as it is guaranteed to be a waste of network bandwidth [4][8].

A. Algorithm

To overcome the first two weaknesses of TCP NewReno, an adaptive congestion control algorithm is developed. This proposed algorithm modifies TCP NewReno congestion control, with more emphasis on its performance improvement over wireless links; ensuring its rated performance over wired links. The following session summarize the logic behind development of TCP– Adaptive Congestion Control (referred as TCP ACC in rest of the paper).

During slow start phase of a new TCP connection, TCP limits its capacity by sending one packet per *RTT,* assuming that the sender is unaware of network condition and hence it starts with a very basic value .This is done by setting it's *cwnd* to one [1]. For higher value of *RTT*, this results into an excessive delay, in slow start phase.

Consider a TCP sender, having multiple TCP connections over a point to point wireless link. All TCP connections using the common interface are going to suffer from similar kind of environment. Under these circumstances the link estimation based on ongoing connection can be used as reference for new connection to provide approximate idea of the network. As *ssthresh* level gives partial approximation of link behavior [1][3], we take *ssthresh* to calculate value of parameter reflecting approximation of link. All TCP connections are going to update one global variable *nvar0*, which basically reflects link condition. Whenever any ongoing connection detects a packet loss on the network, its *ssthresh* value is used to recalculate *nvar0.* Whenever a new connection is established, instead of starting from *cwnd = 1*, we start using *cwnd = nvar0.* This will allow higher value of *nvar0,* under better link conditions, whereas for poor link conditions *nvar0* is set to its minimum value 1. This will not only reduce $t_{slowstart}$ time, but also facilitate initialization of cwnd dynamically. Reflections of link characteristics, from peer TCP connections will also improvise stability during slow start phase by not allowing higher value for *cwnd* in the beginning; particularly under poor link conditions. The basic coding can be understood as shown below.

```
Global variable nvar0
Modifynvar0 IN SLOWDOWN( )
     if (loss is reported) then
          nvar0= int ((0.4*ssthresh_) + (0.6*nvar0));
Implementation IN SETWNDINIT ( )
/* modification is variable to indicate modified new
Reno */
 if(new TCP connection establishes and modification
= 1) then
            cwnd = nvar0
else        cwnd = 1;
```

Figure 1. Modifications in slow start phase of TCP NewReno

Wireless links frequently suffer from link disconnection [2]. It is reflected by timeout in NewReno TCP. Simulation results of first modification shows that modified NewReno TCP behaves in a same way as original NewReno TCP after timeout, and no advantage of modification is gained. Timeout causes significant fall in throughput as TCP reduces its rate to one packet per *RTT.* As in wireless links cause of timeout is mostly link disconnection not congestion [7][8] and as value of *RTT* is much higher, TCP unnecessarily wastes its time in probing network after timeout. It may also happen that link will be down again within some time after link is up. That is why we need to pump more data as soon as link is up [8].

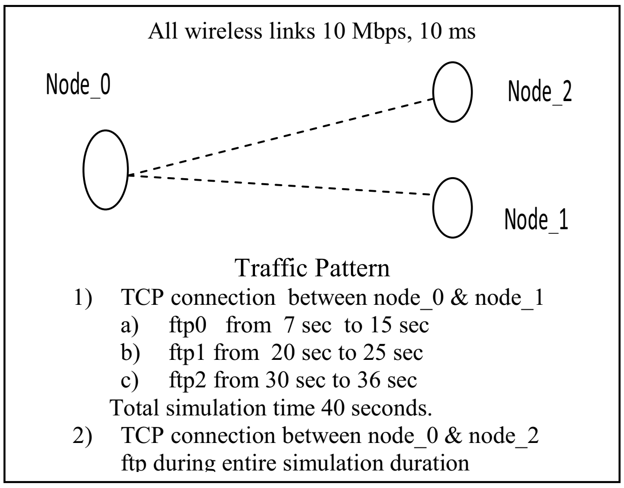
With all above considerations we again modify the source code of NewReno TCP to achieve advantages of modified slow start even after of timeout. When timeout occurs the cwnd is initialized with *nvar0/2*, instead of *wnd_restart*. Rest of implementation in modified New Reno is taken as it is. The basic coding can be understood as shown below.

```
Timeout MODIFICATIONS FOR SLOWDOWN( )
/*
wnd_restart_ is a variable used to initialize cwnd
after timeout.
*/
 if (timeout is detected and modification = 1) then
       wnd_restart_ = nvar0/2;
       cwnd = int(wnd_restart_);
```

Figure 2. Modifications in timeout phase of TCP NewReno

B. *Related Work*

The complete TCP NewReno with suggested modifications is referred as TCP-ACC in rest of the paper. The performance of TCP ACC is compared with TCP NewReno using NS2 based simulations over different network scenarios. The performance comparison on the simple wireless link is as shown below. The figure3 represents the network scenario with all relevant parameters. The figure 4 represents the *cwnd vs. time* plot for TCP-ACC connection over the simulated network. Analysis showing performance comparison is summarized in table 1.



Traffic Pattern

1) TCP connection between node_0 & node_1
   a) ftp0 from 7 sec to 15 sec
   b) ftp1 from 20 sec to 25 sec
   c) ftp2 from 30 sec to 36 sec

   Total simulation time 40 seconds.
2) TCP connection between node_0 & node_2
   ftp during entire simulation duration

Figure 3. Network Topology for Simulation

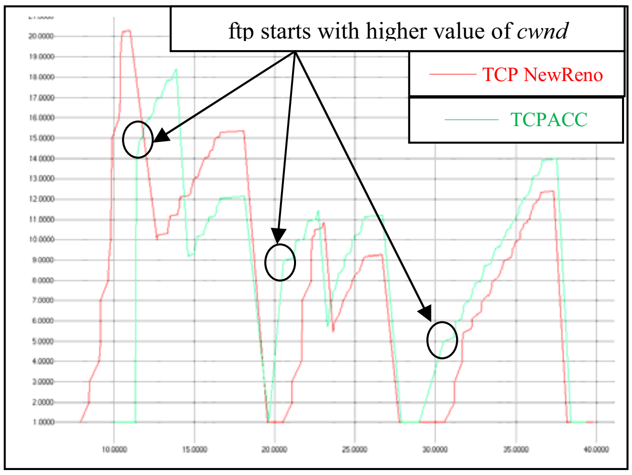


Figure 4. *cwnd vs Time* for TCP NewReno and TCPACC

The congestion control policy of TCP ACC allows TCP to conceive more throughput in presence of interference as mentioned below in table. That demonstrates the strength of the suggested algorithm. The interference introduced during simulation will lead to network congestion and resulted into packet loss. However if the packet losses are due to the reason other than congestion, inability of this algorithm to differentiate between type of packet loss is surfaced out. This was evaluated by introducing losses at link layer, during simulations in NS2. This demands presence of some Loss Differentiation algorithm at TCP.

TABLE I. THROUGHPUT COMPARISON

Interference Packets/sec	**Throughput**		
	NewReno	***TCP ACC***	***Improvement in %***
0	254.418	263.689	3.52
200	254.418	263.689	3.52
700	241.312	245.679	1.78
1500	239.187	244.261	2.08
3000	229.067	235.376	2.68

Measured in Kilobits / second at receiver end.

TABLE II. THROUGHPUT COMPARISON

MAC BER in %	**Throughput**		
	NewReno	***TCP ACC***	***Impact on throughput in %***
0	254.418	263.689	3.52
0.05	216.504	213.992	-1.17
0.07	197.757	197.319	-0.22
0.1	155.453	168.948	7.99

Measured in Kilobits / second at receiver end.

The readers can refer to the reference [10] for better understanding and the evaluation of the TCP-ACC using NS-2 simulations.

IV. A CROSS LAYER ADAPTIVE POLICY

A retransmission at the MAC layer occurs, when the 802.11 acknowledgement is not received by the transmitter within the specified delay, is indicated in the frame through the retry bit of the MAC header. For each retransmission a counter is incremented until a threshold, named retry limit, is reached. This is the basic loss differentiation Algorithm [2][11] used at the MAC layer of the proposed scheme. The idea is to count the number of MAC retransmissions for each of the *n* segments composing the current TCP window. When the TCP layer is alerted by the reception of three duplicated acknowledgments, the above count will be very useful. For one of these not acknowledged segments, if the number of MAC retransmissions (Retry Count) is equal to the threshold (Retry Limit), we consider that the loss is due to interferences and not due to network congestion.. This information is made available to TCP-ACC for triggering its congestion control algorithm appropriately as described below.

A. *Algorithm*

When the source detects a segment loss, i.e. when 3 duplicate acknowledgments are received, the Loss

Differentiation Algorithm is asked to know the cause of the packet loss.

- If the loss is classified as due to congestion, a normal TCP-ACC reaction is triggered.
- If the loss is classified as due to interferences (short signal loss), *cwnd* is not reduced. This allows the source to achieve higher transmission rates in the event of short successive signal losses, if compared to the blind reduction of the throughput performed by the legacy operations of TCP [2].

The basic coding can be understood as shown below.

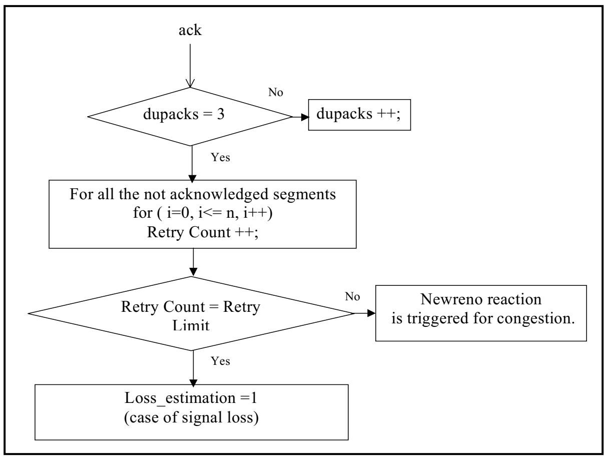


Figure 5. *Flowchart for TCPCLAP*

Pseudo codes for the implementations.

```
/* at MAC layer */
If (segment is retransmitted)
    \\ For each retransmission counter is incremented.
    Retry count ++;
    If (Retry Count = Retry Limit)
  /* In this case the segment loss is mainly due to link
            failure rather than congestion. */
                LDA_Estimator = 1;
    end if
end if
```

Figure 6. MAC Layer implementation of LDA

```
/* at Transport Layer */
if (no. of duplicate acks = 3)
        for( i = 0; i < = n; i ++)
                if (LDA_Estimator = 1)
                        /* Calculated at MAC Layer */
                                Flag = 1;
    /* Flag =1, prevents triggering of congestion control
            algorithm in TCP ACC */
                end if;
        end if;
end if;
```

Figure 7. Transport Layer Implementation for TCPCLAP

Here *n* is the maximum seq. no. in a window at time of reception of 3 duplicate acks. For one of these segments, if number MAC Retransmissions (*Retry Count*) is equal to threshold (*Retry Limit*) we consider that loss is due to link failures and not to TCP congestion, as explained before.

The next section describes the performance evaluation of the proposed scheme over a wireless network using a simulation in NS-2.

V. Simulations and Results

In order to analyze the performance improvements brought by the TCP-CLAP, simulation is carried out using NS-2. The simulation is carried out over a network scenario as described in TCP-ACC for 100 sec of period. The NewReno at a Transport Layer and 802.11g at MAC layer are used as base protocols for all modifications. Initially the simulation is carried out for different interferences introduced in TCP connection during the simulation duration. The P1 traffic pattern is followed. (please refer to table IV).

Here Interference is generated by allowing CBR/UDP traffic on the same wireless (802.11g) channel between node_1 and node_2. Variations in the interference are achieved by changing the transmission rate of the CBR traffic. During the simulations the interference rate is varied from 200 packets/sec to 4000 packets/sec. Here interference rate is considered to be the key parameter. The following table summarizes the performance comparison between TCP NewReno and TCP CLAP, in terms of throughput.

TABLE III. Throughput Comparison for Interferences

Interference Packets/sec	**Throughput***		
	NewReno	***TCP CLAP***	***Improvement in %***
0	254.418	263.689	3.52
200	254.418	263.689	3.52
500	243.182	244.261	0.44
3000	229.067	235.376	2.68

** Indicates throughput measured in Kilobits / second at receiver end*

The above comparison clearly indicates the strength of TCP CLAP over TCP NewReno in the presence of interferences at TCP, which are responsible for causing network congestion, resulted into the packet loss. This demonstrates effectiveness of TCP ACC even in the presence of LDA. The performance of TCP CLAP in presence of link failures has been evaluated. A BER variation at link layer is considered to be the key parameter.

The simulation results shows that with the introduction of LDA scheme in TCP ACC, the performance is much improved. This is certainly due to the ability of TCP to differentiate between different types of losses as mentioned earlier and its appropriate triggering of congestion control algorithm based on the estimate from MAC layer. A performance comparison between various schemes is again carried out using simulations under NS2. The results of above comparisons are as tabulate below.

TABLE IV. Throughput Comparison For BER

MAC BER	**Throughput***		
	NewReno	***TCP CLAP***	***Improvement in %***
0	254.418	263.689	3.52
0.05	216.504	241.312	10.28
0.07	197.757	214.85	7.96
0.1	155.453	196.34	20.82

** Indicates throughput measured in Kilobits / second at receiver end.*

Comparisons (as summarized in table III and IV) based on the simulation results, indicate performance improvement in TCP CLAP over TCP NewReno. It is also observed that the percentage increase in improvement is higher with higher value of BER and Interference Rate. This is the case mainly with wireless networks. At the same time it is also observed that the performance of TCP CLAP is not tainted while operated with lower value of BER or Interference in the network

Performance of TCP CLAP with different traffic pattern is also evaluated, considering the busty traffic in wireless networks. For that a single TCP connection between two nodes over a wireless link is simulated. To observe impact of the traffic pattern on the efficiency of TCP CLAP, 3 ftp connections were simulated one after another for different time intervals, as shown in table V below.

TABLE V. DIFFERENT TRAFFIC PATTERNS FOR SIMULATIONS

Duration in Sec → **Pattern**	**ftp1**	**ftp2**	**ftp3**
P1	1-30	35- 65	70-100
P2	1-10	15-45	50-100
P3	1-50	55-85	90-100
P4	1-70	75-85	90-100

The comparison between TCP NewReno and TCP CLAP for different traffic pattern is shown in table below. The simulation results clearly expose the efficiency of TCP CLAP over TCP NewReno under changing traffic patterns in the network.

TABLE VI. THROUGHPUT COMPARISON

BER	P1		P2	
	TCP * NewReno	TCP * CLAP	TCP * NewReno	TCP * CLAP
0.01	249.433	250.729	254.418	263.689
0.05	216.504	241.312	230.442	240.502
0.1	197.757	214.85	201.851	221.754
0.5	8.97	9.5	9.746	9.869
BER	P3		P4	
	TCP * NewReno	TCP * CLAP	TCP * NewReno	TCP * CLAP
0.01	243.428	247.714	254.418	263.689
0.05	206. 439	221. 172	224.412	225. 022
0.1	190. 572	204. 582	191. 516	198. 463
0.5	6.999	7.721	7.023	7.721

** Indicates throughput measured in Kilobits / second at receiver end*

VI. CONCLUSIONS

The wireless links are suffering a lot mainly by link errors and large propagation delays. Both the factors cause decrease in the acceleration of TCP transmission rate and subsequently, in the overall link utilization. This paper illustrates performance optimization in TCP with the help of cross layer mechanism between TCP and MAC layer protocols. The algorithm differentiates between packet loss due to network congestion and the same due to error in the transmission link. Identification of the reason for the packet loss helps TCP to trigger its congestion control mechanism appropriately, which prevents transmission ceases and reduction in congestion window unnecessarily, in absence of congestion. Subsequently, an approach to make TCP congestion control more efficient using adaptive approach based on parallel connection is also evaluated.

The modified TCP, referred as TCP CLAP, attempts to differentiate between network congestion and link failures during packet transmission and triggers TCP congestion Control whenever packet loss is because of network congestion. This in turn helps the TCP CLAP to avoid negative impact of reduced transmission rate in absence of congestion. However, the algorithm does not degrade performance of TCP in a congested network, since the modified TCP, referred as TCP ACC reverts back to the conventional congestion control, with an adaptive policy. Analysis of the results of simulations carried out indicates improvement in the overall performance of TCP CLAP in presence of the modifications. Performance enhancement of the proposed scheme is higher in case of higher value of BER. With lower value of BER (Generally with wired network) no degradation in the TCP performance observed which reveal portability and reliability of the TCP CLAP on heterogeneous networks.

The proposed TCP CLAP outperforms TCP NewReno, in case of interference caused by the parallel traffic and even in case of traffic pattern variations. Considering all the abilities of TCP CLAP, it may be accepted for reliable data transfer using TCP, over wired as well as wireless networks.

REFERENCES

[1] G. K. Fall, and S. Floyd, " Simulation – Based comparison of Tahoe, Reno and SACK TCP", Computer Communications Review ACM-SIGCOMM, Vol.26, No.3 , July 1996.

[2] N. K. G. Samaraweera, "Non-Congestion Packet Loss Detection for TCP error recovery using wireless links", IEE Proc. on Communications, Vol 146, No.4, Aug 1999, pp 222-230.

[3] W. Stevens, " TCP Slow Start, Congestion Avoidance , Fast Retransmit and Fast Recovery Algorithms", January 1997, RFC 2001.

[4] Motoharu Miyake and Hiroshi Inamura, "TCP Enhancement Using Rcovery of Lost Retransmissions for NewReno TCP", IPSJ Digital Courier, Vol 1 , Sep. 2005. pp 370-381.

[5] A Capone, L. Fratta, and F. Martignon, " Bandwidth Estimation Schemes for TCP over Wireless Networks", IEEE transactions on Mobile Computing, vol.3, no. 2, 2004.

[6] K. Xu, Y.Tian, and N. Ansari, " TCP-Jersey for Wireless IP Communications.", IEEE Journal on Selected Areas in Communications, vol. 22, no. 4, May 2004.

[7] A. Bakre and B. Badrinath, "I-TCP: Indirect TCP for mobilehosts," in Proc. 15th International Conf. on Distributed Computing Systems (ICDCS), May 1995.

[8] H. Balakrishnan, V. Padmanabhan, S. Seshan, and R. Katz, "Acomparison of mechanisms for improving TCP performance over wireless links," in ACM SIGCOMM, Stanford, CA, August 1996.

[9] A. DeSimone, M. Chuah, and O. Yue, "Throughput performance of transport-layer protocols over wireless LANs," in Proc. Globecom '93, December 1993.

[10] Prof. Purvang Dalal, "Adaptive TCP: Enhancing Performance over Heterogeneous Networks". ICOICT 2009, SCT College of Engineering, Trivandrum, Kerala, INDIA. February 2009.

[11] N. Vaidya, M. Mehta, C. Perkins, and G. Montenegro.: "Delayed duplicate acknowledgements: A TCP-unaware approach to improve performance of TCP over wireless". Technical Report, Computer Science Dept., Texas A&M University, February 1999.

[12] S. Mascolo, M.Y. Sanadidi, C. Casetti and R. Wang, " TCp Westwood: End to End Congestion Control for Wired/Wireless Networks", Wireless Networks Journal, vol.8, pp. 467-479, 2002.