Performance Improvement of TCP New Reno using one way delay Measurement

Sanjesh S. Pawale, Sandeep B. Vanjale

Abstract: ***Well known TCP NewReno protocol when used with wire-cum-wireless network its performance decreases, due to the inability to discriminate wireless loss with congestion loss. One way delay can give better network characteristics than approximate RTT used in TCP congestion control algorithms. In this paper, we have proposed a modified version of TCP NewReno based on one-way delay measurement. We called it as WNewReno. Our proposed algorithm detects wireless loss and improves performance. Simulations are carried out using ns-2.34.***

Keywords: one-way delay (OWD Round Trip time (RTT), Transmission control protocol (TCP), TCP NewReno, TCP Reno.

I. INTRODUCTION

The Wireless networks have become popular from the last decade. Combination of wired and wireless is predominant and used in a variety of applications. Various versions of TCP proposed in the past. Some are standard while most are experimental [1]. Standard TCP variants Reno, NewReno are extensively used in real networks.

NewReno is the most widely deployed transmission control protocol on the internet, however NewReno does not achieve the best possible throughput due to the inability to differentiate congestion loss with the wireless loss.

Congestion loss occurs due to buffer overflow at a router. Wireless loss is mainly due to inherent characteristics of wireless medium [2] like signal fading, mobility of devices, etc. In TCP NewReno when sender detects packet loss, by default assume it is due to congestion. After the detection of packet loss sender reduces its transmission rate and resend lost packets. These results decrease in throughput. Even if the loss is due to wireless, TCP NewReno takes congestion control action, which is not appropriate. If TCP sender can be made clever to discriminate wireless loss with congestion and take appropriate action after the wireless loss then overall performance can be improved.

Method proposed in this paper using one-way delay will help to discriminate loss type and improves TCP NewReno performance. For the proposed method, we have performed extensive simulation using ns-2 for varieties of scenarios, results demonstrate TCP NewReno performance improvement.

This paper is organized as follows: Section II describes brief analysis of some work related to the method proposed. The description of the one way delay measurement models is carried out in Section III; Section IV describes the proposed WNewReno algorithm; Section V contains the experimental results; and finally, Section VI summarizes our work.

Revised Manuscript Received on August 05, 2019.

Sanjesh S. Pawale, Research Scholar, Department of Computer Engineering, Bharati Vidyapeeth (Deemed to be University), College of Engineering, Pune, India.

Sandeep B. Vanjale, Professor, Department of Computer Engineering, Bharati Vidyapeeth (Deemed to be University), College of Engineering, Pune, India.

II. RELATED WORK

Balakrishnan [3] summarizes all flavors of TCP for wireless into three categories end-to-end, link-layer and split connection. Link-layer approach manages packet loss from link-layer but has a complication to maintain TCP connection. Split approach divides TCP connection into two parts at the wireless gateway. Split connection violets end-to-end TCP semantic, also wireless link face significant overhead as packet must undergo TCP processing twice. A reliable transport protocol must provide end-to-end semantic, the acknowledgment must absolutely certify that packet have reached the destination safely. Our main motivation is to preserve end-to-end semantics of TCP.

Protocols used in preserve end-to-end semantic are Tahoe, Reno, NewReno and rest based on this. Tahoe is a very old TCP algorithm, obsoleted now. Tahoe has very poor performance. This is due to its feature of every three duplicate acknowledgments or timeout it reset the slow start threshold (ssthresh) and set congestion window (cwnd) to one.

TCP Reno [1] [4] adds new state in congestion control algorithm on Tahoe, called fast recovery state. Features of this protocol are it treats two signals of congestion three duplicate ACKs and timeout differently. If a timeout occurs TCP moves to slow start state on the other hand if three duplicate ACK arrived TCP moves to fast recovery state and remain there as long as duplicate ACK arrived. Problem with TCP Reno is that it cannot handle multiple packet loss from the same window.

TCP NewReno [1] [5] has a fast retransmit phase same as Reno. It has different fast recovery phase due to which it is able to detect multiple packet loss. Whenever NewReno enters the fast recovery phase, it records a maximum segment sent. As fast recovery proceeds, NewReno receives a fresh ACK. If ACK is partial it deduces next segment in line was a loss, it retransmits that segment and remains in the fast recovery state. When it receives ACK of all packets sent it exits fast recovery by setting cwnd to ssthresh.

TCP NewReno suffers from the fact that it takes one RTT to detect packet loss. When ACK from the first retransmitted segment received only then we can deduce other segments were lost. All end to end TCP protocols discuss above are designed mainly to use with the wired network. We will discuss now wireless medium characteristics and modification proposed by researchers in end to end TCP, to optimized for the wireless environment.

Wireless environment characteristics

Victor [2] gives a description of major types of wireless network, mainly infrastructure and Ad-hoc network. Reasons for packet loss in a wireless network are signal fading, mobility, channel contention, and energy consideration. Apart from packet loss, other problems are associated with TCP in a wireless network like packet reordering, i.e. receiving order is not the same as sending order, etc. Survey of all problem and available solutions are mentioned in [2].

This section presents an overview of some research related to one-way-delay measurement and queuing delay points of view.

III. ONE-WAY DELAY

One way delay is becoming a great interest in research in recent. One-way delay comprises of difference in the time interval between injection of the first bit of packet in the link and reception of last bit of the packet at receiver [8]. Therefore One-way delay is the sum of transmission delay, propagation delay and queuing delay.

Queuing delay analyzed with queuing theory and simulations. Average time packet spent in a queue is calculated by little's theorem as follows.

$$N=\lambda*T$$

Where N is the average number of the packet in the buffer, λ is effective arrival rate, T is the average time that packet store in the queue, T_t is considered as queuing delay and is given by.

$$T_t = \frac{\sum_{i=0}^{\alpha(t)} T_i}{\alpha(t)}$$

Using Little's theorem we can say, as queue size (N) increases, queuing delay (T) in the network also increase. If queuing delay is small and the sender receives duplicate ACK, intuitively loss would not be due to congestion. On the other hand if queues size in the large, i.e. queue is about to get full, queuing delay will be more, subsequently, the one-way delay(OWD) will also be increased. In such a situation if any packet loss in the network, more probability it is lost due to congestion.

Thus to discriminate wireless loss with congestion, a threshold can be set based on OWD.

Measurement of one-way delay is motivated due to asymmetric queuing from source to destination [7]. If there is an asymmetric path then ACK may come from different longer path, will lead to an increase in RTT, the rise in RTT can be misunderstood as congestion in the forwarding path. Even paths are symmetric there may have asymmetric queuing, e.g. forward path buffers are empty and reverse path buffers are full. Therefore one-way delay can give accurate information of forwarding path.

Kind =8 |len=10|TS Value(Tsval)|TS Echo Replay(Tsecr)
1 byte 1 byte 4 byte 4 byte

Fig. 1. TCP Timestamp option

One-way delay can be calculated using option field provided in TCP header. As shown in fig.1, the option field has time stamp value (TS Value), which is used to store packet sent time by the sender. When the receiver receives a packet it calculates one way delay simply by subtracting receiving time with a timestamp. Receiver creates ACK packet, adds OWD value to TCP header and sends to the sender.

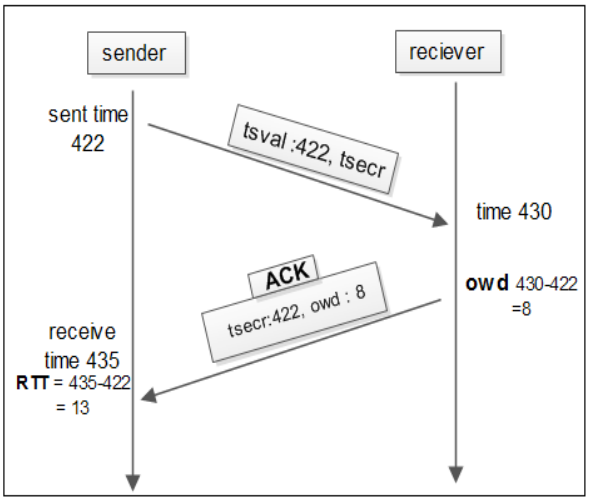

Fig. 2. One Way Dealy Calculation

Fig. 2 explains the sender insert sent time (422) of the packet in a *tsval* field of the header. The receiver calculates one-way delay by subtracting *tsval* from the received time (430) and sends one-way delay (430-422 = 8ms) to the sender in ACK packet. The timestamp value is used in fig. 2 are random for the sake of understanding. We calculate one- way delay with the assumption that clocks are synchronized.

Various TCP algorithms had been proposed based on one-way delay. TCP-LP [9] uses a one-way delay threshold as an early congestion indication as compare to traditional packet loss as a sign of congestion used by TCP. It aims to measure unused bandwidth using one-way delay and utilize it for low priority traffic applications. The scenario considered in TCP-LP is only in the wired domain.

Based on the principle of one-way delay threshold TCP-E2E [10] is proposed, which is based on TCP Reno. As stated in [8] E2E works well only for a wire-cum-wireless network with a small variation in the end to end delay. TCP E2E performance can be checked with a better version of TCP like NewReno, etc.

One-way-delay and RTT

RTT is one of the key parameters which TCP often uses as an approximation of delay. Jin-Hee-Choke [8] suggests RTT is the estimation of forwarding delay and reverse delay.

One way delay often also called a forward delay and misunderstood as RTT/2. Since forward and reverse condition in the network may not be same, one-way-delay should not be approximated as RTT/2 rather it has to be calculated separately.

IV. PROPOSED ALGORITHM: WNEWRENO

WNewReno is a modification to TCP NewReno. To implement WNewReno one-way delay calculation capability is added in NewReno.

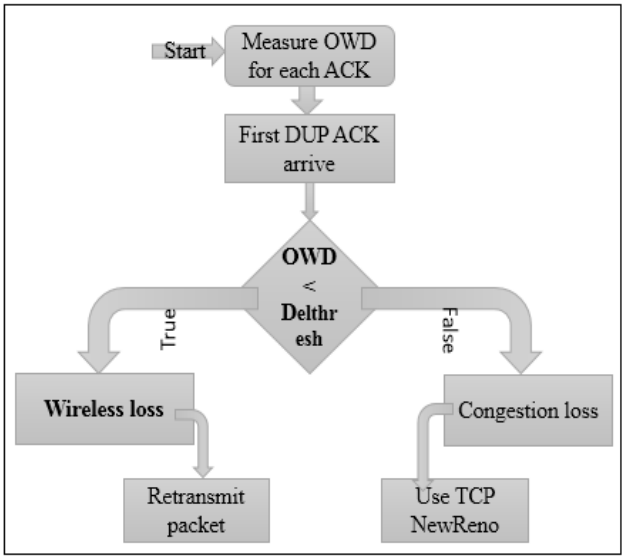

Fig. 3. TCP WNewReno for wireless loss detection

Sender's side

ON sending segment to receiver

1. tcph.timestamp = current_time

i. **ON** receiving ACK

2. Calculate minimum OWD(min_owd)
3. **IF**(OWD < min_owd){
4. wireless_margin = 0.15*min_owd;
5. min_owd = OWD;
6. delay_threshold = min_owd + wireless_margin;
7. }
8. **END**
9. **ON** receiving every duplicate ACK
10. **IF**(first dup ACK)
11. {
12. extract OWD from ACK header
13. **IF**(OWD < delay_threshold)

/* *Wireless loss* */

Retransmit the packet

14. **Else**

/* *Congestion loss* */

15. Congestion control as NewReno
16. }

Receiver's side

ON receiving TCP segment receiver calculate OWD and Update in ACK packet

1. tcph.owd = current_time – tcph.timestamp;
2. Send this OWD with ACK packet.

TCP uses three duplicate ACK as a sign of congestion. Our schema on receiving first duplicate ACK compares instantaneous OWD with delay threshold. For OWD lesser than delay threshold, it indicate a wireless loss, therefore, it instantly retransmits packet without waiting for further duplicate ACK. This increases the throughput of WNewReno unlike NewReno waiting for further two more duplicate ACK to come or retransmission time out to happen.

On every new ACK, sender extract OWD and calculate minimum OWD. Delay threshold is set as the sum of minimum OWD and wireless margin [10]. The wireless margin is due to the mobility of node and data transmission by variable path in a wireless environment. Wireless margin can be selected between 5-15% of minimum OWD. Based on the experiment we found out 5-15 % of min_owd is suitable. Setting the wireless margin above 15% performance gets degraded since the probability of misclassification of losses increases.

If OWD greater than delay threshold then probability of congestion is more, thus WNewReno goes to congestion control state. After receiving three duplicate acknowledgment or time out it assumes losses is due to congestion. To recover from congestion loss WNewReno uses NewReno procedure.

V. EXPERIMENTS

In this section, we present the simulation methodology used to evaluate the performance of WNewReno. Simulation results are compared with existing TCP to understand TCP performance.

A. Simulation Setup

Two different scenarios are used and many simulations are performed to evaluate performance. Throughput is chosen as a performance parameter for comparison. Throughput is the average amount of data received by the receiver per unit time.

*Throughput = (Total_Packet * Packet_Size) / Time*

The proposed algorithm is simulated for wire-cum-wireless network topology with one sender and one receiver, second wire-cum-wireless network topology with three senders and three receivers. We have use standard WLAN 802.11a at the base station with bandwidth at wireless link 2 Mbps and the bandwidth of wired link 10Mbps, and 10msec delay as used in [3].

Traffic source generates FTP traffic at the sender. The packet size is 512 bytes. Drop tail queue policy is used. Destination Sequenced Destination Vector (DSDV) is used for routing. Throughput is calculated by varying packet loss rate in a wireless network. The loss rate varies from 1 to 15%. Duration of simulation is kept 120 sec. Traffic is sent between time 10 to 110 sec.

Various simulations are performed by changing the delay threshold to understand the impact of delay threshold on performance. Result shown here are by considering wireless_margin = 0.15*min_owd.

B. Simulation Comparison

1. Simple Wire-cum-wireless topology

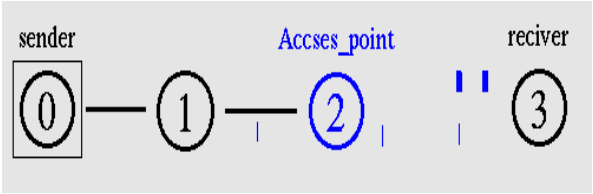

Fig. 4. Topology 1 used in simulation

In this experiment, we compare performance for simple wire-cum-wireless topology. The network consists of one sender, intermediate node, Access point, and one mobile receiver as shown in fig. 4. We evaluate throughput with respect to the packet loss

rate.

Case1.1: Receiver is mobile

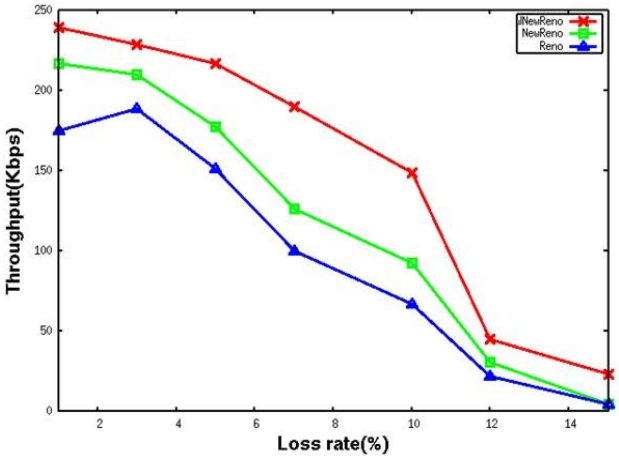

Fig. 5. Throughput (Kbps) vs. Packet loss rate

As anticipated, when loss rate increases throughput decrease due to more number of packets gets corrupted due to the wireless environment. This is shown in fig.5. Throughput vs. packet loss relationship is given by [12] as

$$\text{Avg\_tpt} = \frac{1.22}{\sqrt{p_{loss}}}\frac{\text{MSS}}{\text{RTT}}$$

This is also called the square root formula. It shows average throughput achieved is inversely proportional to the square root of loss rate. MSS is the maximum segment size, and RTT is the average round trip time.

However, compare to other TCP variant WNewReno achieve better throughput due to its immediate retransmission method. For this case, WNewReno gains 50% throughput over NewReno and 90% gain over Reno at 7% packet loss rate.

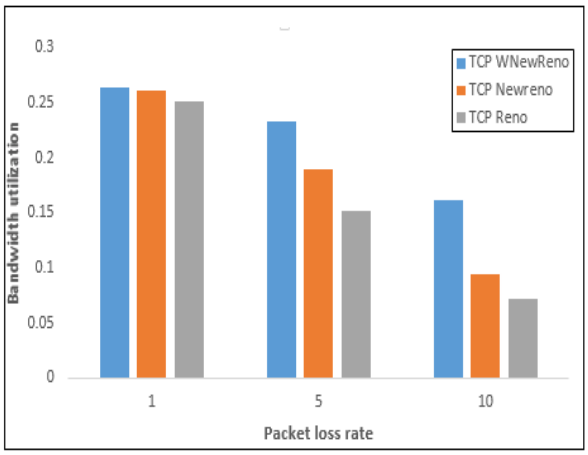

Fig. 6. Bandwidth utilization vs. packet loss rate

Bandwidth utilization graph shown in fig. 6 indicates bottleneck bandwidth used by the TCP connection. It is proportional to the throughput achieved by TCP connection.

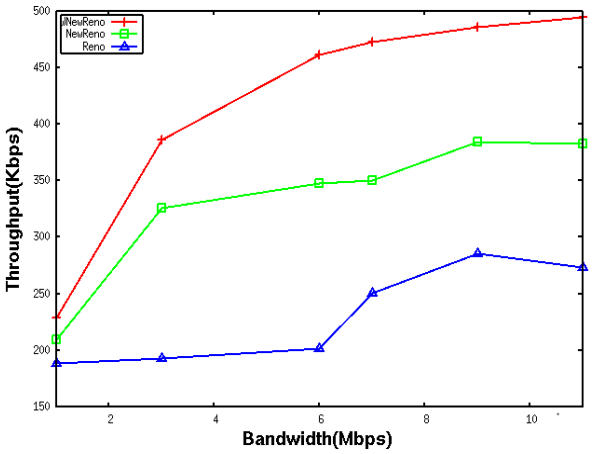

Fig. 7. Throughput vs. base station bandwidth

Fig. 7 shows base station's bandwidth influences the throughput of TCP connection. Throughput is calculated by varying bandwidth between 1 to 11 Mbps. This throughput is calculated by considering the packet loss rate of 3%. Higher bandwidth allows the base station to transmit the number of packets. As shown in fig. 7 throughput of TCP increases by increasing bandwidth, regardless of any protocol. However, All TCP's eventually reaches a point where the increase in bottleneck link's bandwidth has negligible influence on throughput. This observation can be understood using the previously discussed square root formula.

The formula shows random loss enforces an upper limit on the throughput of TCP connections. Thus all the TCP's shown in fig. 8 can use only a fraction of available bandwidth. However WNewReno throughput limit is much higher than other, due to its mechanism for distinguishing random loss. Keeping bandwidth is 3 Mbps, WNewReno gains 18%, 41% throughput improvement over TCP NewReno and Reno respectively.

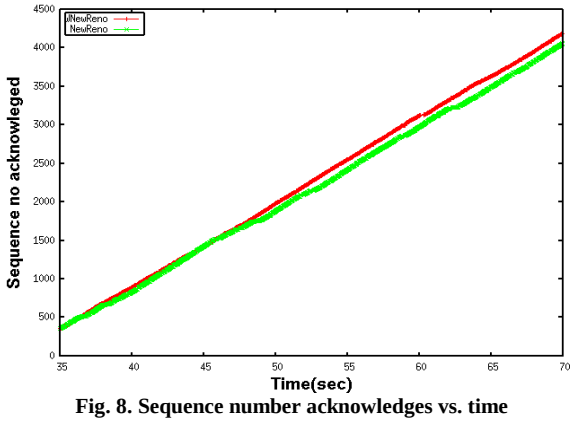

Fig. 8. Sequence number acknowledges vs. time

Fig. 8 indicates that the sequence number acknowledgements for the packets sent by WNewReno and NewReno.

Table 1. Packet received at 3% loss rate

Protocol	**TCP WNewReno**	TCP NewReno
Number of packet ACK	4265	4128

Case 1.2: Receiver is fixed

Experiments are performed by keeping the receiver at a fixed position

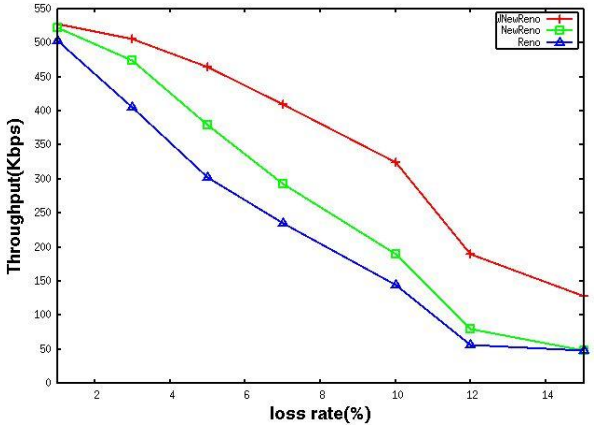

Fig. 9. Throughput (Kbps) vs. loss rates

Simulation results show in this case for non-mobile receiver. Throughput achieved by TCP connection is more as compared to the previous case where the receiver was mobile. It is due to the fact that mobility affects TCP performance. Throughput depends on other factors, like the distance between link and receiver at a movement. However, it is not necessary that if the receiver is mobile throughput will decrease as compared to the case when receivers are at a fixed location. It depends on the type of scenario selection. If in the scenario receiver is more close to base station throughput probably may increase. If receiver moves far away from base station overall throughput may decrease.

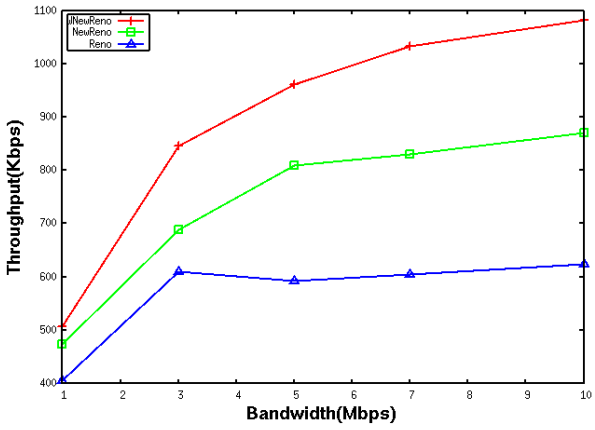

Fig. 10. Throughput vs. base station bandwidth

2. Wireless topology with multiple senders and receiver

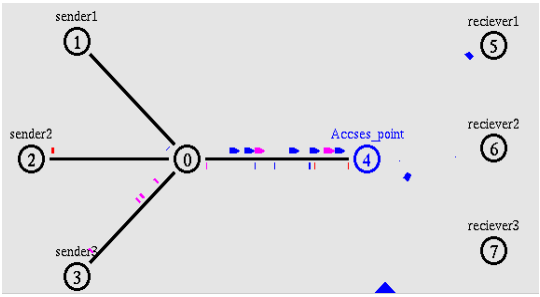

Fig. 11. Topology2 used in simulation

In this topology, for simplicity, here it is considered that the connection is in between sender node 1 and receiver 1, sender node 2 and receiver 2, similarly sender node 3 and receiver 3. Parameters of topology are taken from [3]. The topology has three TCP connections and sharing one bottleneck link, thus there are fair chances of getting congestion at the link.

All senders are connected to 10Mbps link through a router (node 0). Routers and base station are connected with 10 Mbps link. Base station bandwidth is set as 2 Mbps. WNewReno results are compared with NewReno and Reno.

Case 2.1:

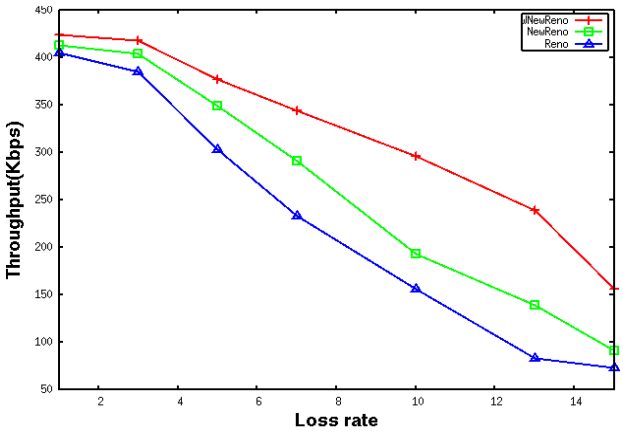

Fig. 12. Throughput vs. packet loss rate

As packet loss increases the throughput of all TCP's decreases. The throughput of all TCP is upto 3%. But when the loss rate is greater than 3%, TCP WNewReno begins to increase its throughput as compared to other TCP's. This is because other TCP can't detect packet loss in wireless medium via duplicate acknowledgment, while WNewReno can detect it. Overall throughput achieved, in this case is higher than scenario used in case1.1, due to more number of TCP connections.

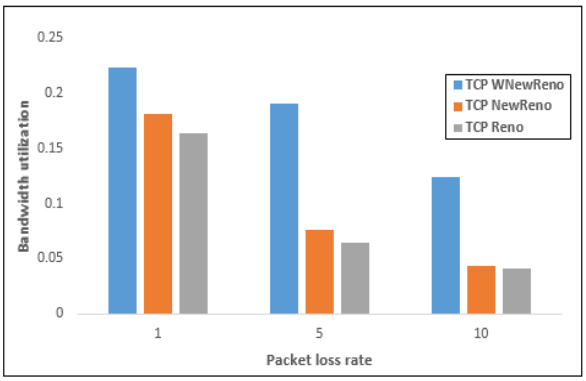

Fig. 13. Bandwidth utilization vs. packet loss rate

Fig.13 shows a comparison of wireless bandwidth utilization by WNewReno, NewReno, and Reno at 1%, 5% and 10% packet loss rate respectively. Bandwidth utilization is calculated by Throughput/bottleneck link bandwidth. With an increase in packet loss rate, throughput decreases thus bandwidth utilization also decreases. Result shows WNewReno comparatively utilizes more bandwidth even at high error rate e.g. 10 %.

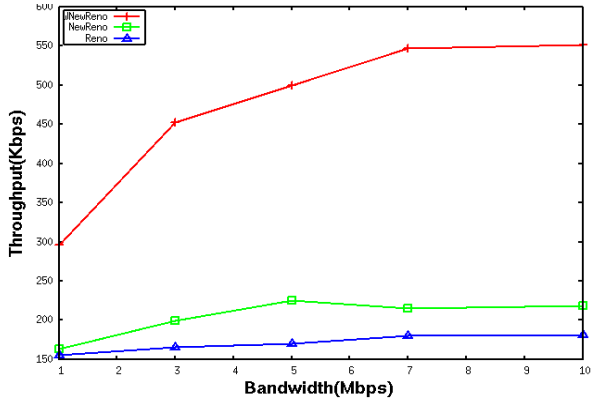

Fig. 14. Throughput vs. base station bandwidth

A throughput of TCP connections is calculated by increasing bandwidth of base station from 1 Mbps to 10 Mbps and imposing 10% packet loss rate. Throughput increases with an increase in bandwidth of the wireless link. However, after a certain bandwidth, throughput remains approximately constant.

Due to loss distinguishing mechanism and number of TCP connection, the throughput achieved by WNewReno is much higher compared to NewReno and Reno.

Case 2.2 : Recivers are fixed

In this case all receiver nodes in wireless envirnoment are at fixed position.

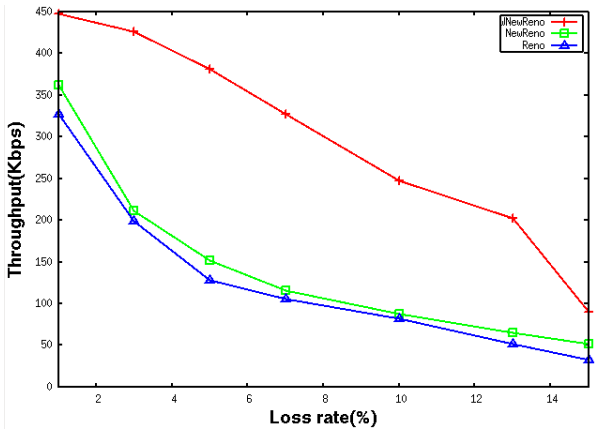

Fig. 15. Throughput vs. packet loss rate

When all receiver nodes in a wireless environment are at fixed position throughput achieved is shown in fig. 15. TCP WNewReno gains 23%, 36% throughput over NewReno and Reno respectively at 1% packet loss rate.

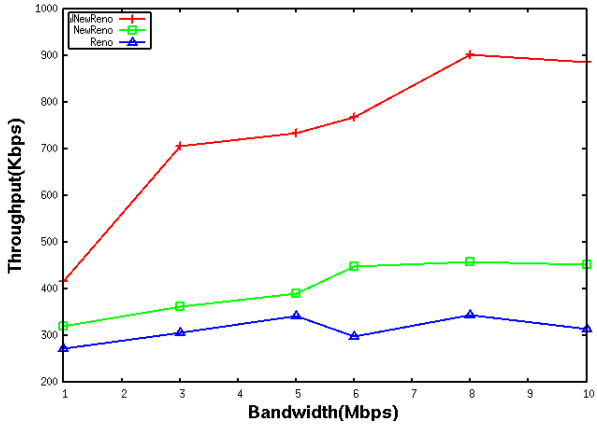

Fig. 16. Throughput vs. base station bandwidth

Throughput is calculated by varying bottleneck link bandwidth from 1 Mbps to 10 Mbps. Throughput increases with an increase in bandwidth as shown in fig. 16.

Case 2.3: Mobile node are sender, fixed nodes are receiver

For the topology shown in fig. 4, senders and receivers are interchanged. Mobile nodes are the sender and fixed nodes are receivers.

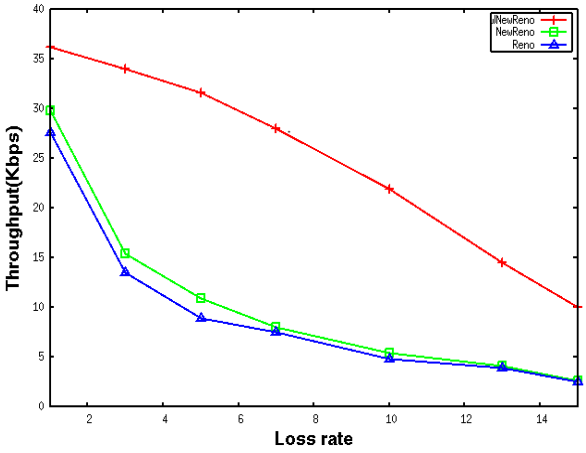

Fig. 17. Throughput vs. packet loss rate

Overall throughput observes, in this case shown in fig 17, is very less compare to other cases. It is due to underutilization of network bandwidth. When senders send data from low bandwidth network pipe (2Mbps in this case) to higher bandwidth network pipe (10Mbps), higher bandwidth network is not utilized to the fullest capacity since data rates are very low.

VI. CONCLUSION

This paper proposes TCP variant WNewReno, which is able to distinguish wireless loss from congestion loss and react appropriately. WNewReno uses one-way delay as a parameter to set the threshold for determining packet loss due to the wireless medium. A modification is mainly at sender side and small modification at receiver to calculate one-way delay. Proposed variant adds the additional capability of loss discrimination in NewReno. It gives better results for a case where routers do not have a very small buffer size. For very small buffer it's difficult to distinguish random loss from congestion loss. Simulations are performed using ns-2.34 shows WNewReno significantly gain throughput over NewReno for varieties of realistic topology.

In the future this research has scope to focus on improving the performance of TCP Veno. TCP Veno is a combination of Vegas and Reno. TCP Veno uses a parameter beta for loss discrimination. This value of beta is static thus it restricts cwnd increment. One way delay concept as applied for NewReno can be used in Veno for improving loss discrimination.

REFERENCES

1. Alexander Afanasyev, Neil Tilley, Peter Reiher, and Leonard Kleinrock, "Host-to-Host Congestion Control for TCP," IEEE COMMUNICATIONS SURVEYS & TUTORIALS, VOL. 12, NO. 3, THIRD QUARTER 2010.
2. Ka-Cheong Leung and Victor O.K. Li "TCP in Wireless Networks: Issues, Approaches, and Challenges" IEEE communication surverys 4TH QUARTER 2006, VOLUME 8, NO. 4.
3. H. Balakrishnan, V. Padmanabhan, S. Seshan, and Y. Katz, "A Comparison of Mechanisms for Improving TCP Performance over Wireless Links," *IEEE/ACM Transactions on Networking*, pp. 756–769, 1997.
4. Network M. Allamn, V. Paxson, W. Stevens "TCP Congestion Control" RFC 2581.
5. T. Henderson, S. Floyd "The NewReno Modification to TCP's Fast Recovery Algorithm" RFC 2582, April 1999.
6. Li Yun, Zhang Rui, Liu Zhanjun, Liu Qilie "An Improved TCP Congestion Control Algorithm Over Mixed wired/wireless networks" Proceedings of IC-BNMT2009.
7. G. Almes, S. Kalidindi, M. Zekauskas "A One-way Delay Metric for IPPM" RFC 2679.
8. Jin-Hee Choi, Chuck Yoo "One-way delay estimation and its application" ELSEVIER Computer Communications 28 (2005) 819–828.
9. Aleksandar Kuzmanovic and Edward W. Knightly "TCP-LP: Distributed Algorithm for Low Priority Data Transfer" IEEE INFOCOM 2003.
10. Parag Kulkarni, Mahesh Sooriyabandara, Lu Li, "Improving TCP Performance in Wireless Networks by classifying causes of packet losses" IEEE/WCNC.2009.
11. K. Stangherlin, R. Costa Filho, W. Lautenschlager, V. Guadagnin, L. Balbinot, R. Balbinot, V. Roesler "One-Way Delay Measurement in Wired and Wireless Mobile Full-mesh Networks" IEEE WCNC 2011.
12. M. Mathis, J. Semske, J. Mahdavi, and T. Ott. "The macroscopic behavior of the TCP congestion avoidance algorithm" Computer Communication Review, 27(3), July 1997.
13. Network Simulator ns2, http://www.isi.edu/nsnam/ns.

AUTHORS PROFILE

Sanjesh S. Pawale is working as a Research Scholar in Computer Engineering Department at Bharati Vidyapeeth (Deemed to be University) College of Engineering, Pune, Maharashtra, India. He received his M.E. (Computer Science and Engineering) from Pune University and Perusing Ph.D. (Computer Engineering) from Bharati Vidyapeeth (Deemed to be University) College of Engineering, Pune. His research interests include Computer Network, Network Security, WLAN Security. He attended more than 10 National and International Conferences and published 14 papers in International Conference and Journals.

Sandeep B. Vanjale is working as a Professor in Computer Engineering Department at Bharati Vidyapeeth (Deemed to be University) College of Engineering, Pune, Maharashtra, India. He received his M.E. (Computer) and Ph.D. degrees from Bharati Vidyapeeth (Deemed to be University) College of Engineering, Pune. His research interests include Computer Network, Network Security, WLAN Security. He attended more than 40 National and International Conferences and published 50 papers in International Conference and Journals.