International Journal of Electronics and Electrical Engineering
ISSN : 2277-7040 Volume 1 Issue 1
http://www.ijecee.com/ https://sites.google.com/site/ijeceejournal/

Modified TCP Peach Protocol for Satellite based Networks

Mohanchur Sarkar[1], K.K.Shukla[2], K.S.Dasgupta[3]

[1]Satellite Communication Technology Division, Space Applications Centre (ISRO), Ahmedabad, India
[2]Department of Computer Engineering, Institute Of Technology, BHU, Varanasi, India
[3]Indian Institute of Space Science & Technology, IIST, Thiruvananthapuram, India
Email: mohanchur@yahoo.com|msarkar@sac.isro.gov.in, kkshukla.cse@itbhu.ac.in, ksd@iist.ac.in

Abstract. TCP has become the de-facto protocol standard for congestion control in the existing terrestrial Internet. But it was designed keeping in mind low Round Trip Time and low channel error rates typically experienced in a wired network. In this paper we have considered TCP Protocol variants like Tahoe, Reno, New Reno, SACK, FACK, Vegas, Westwood and Peach. TCP Peach is found to be better than the other TCP Protocol variants in case of satellite based networks but its performance also degrades when the packet error probability is high and in cases where there are multiple packet losses in the same window. In this paper a modification has been suggested to the existing PEACH protocol and the modified PEACH Protocol is tested to provide performance improvement especially in the cases where the packet error rate is very high. The modified Peach Protocol has been implemented in the ns2 Simulator and evaluated considering a Geo Satellite Network with varying channel conditions with all other TCP variants.

Keywords: Tahoe, Reno, New Reno, PEACH, Vegas, Westwood, SACK, FACK, RTT, ns2 Simulator

1 Introduction

Though TCP has been greatly successful in the terrestrial Internet and provides a robust reliable service for the transfer of data from one part of the earth to the other using an unreliable network layer it performs quite poorly in wireless and satellite based networks. In a satellite based network where the RTT is more than that of the terrestrial counterpart the use of a proper Transport Protocol becomes very

important [35], [36], [37]. In the terrestrial internet the main problem faced by the congestion control protocols is the uncertainty of the traffic condition in the network and the fear of overloading the network with excessive traffic. For this whenever the TCP sender doesn't receive any acknowledgement for the data it sent, it considers that the packet was dropped by the router because of congestion in the network and thereby reduce the congestion window. One of the major problems in a satellite-based network is the random packet errors, which are not common in the wired counterpart [30], [31], [32]. TCP protocols react to the lack of arrival of acknowledgements or duplicate ACK as a sign of congestion [12]. Therefore, the congestion window is reduced which leads to unnecessary throughput degradation. It is a challenge for the network researchers and protocol developers to find means to differentiate the cause of the DUP ACK arrival.

Moreover TCP injects a new packet into the network only after it receives an acknowledgement of the previously sent packets. This works fine as long as the RTT is moderate by keeping the network load within tolerable limits and maintaining the reliability of the data transmitted. But when the RTT increases this mechanism creates the bottleneck in the performance of the protocol. The RTT is constrained by the speed of light and the total amount of data that needs to be sent in one RTT is given by the bandwidth-delay product of the link concerned and is not really achieved by the acknowledgement driven logic of TCP [1], [3], [6]. Moreover there are problems because of bandwidth asymmetry and intermittency of the link.

Generally, probing is done in protocols like Peach [26], Peach+ [21], TP-Planet [20], and RCS [10], [16]. Other approaches to transport protocol design are found in [1], [2], [4], [5], [15], [17], [22], [23], [25], [28], [29], [33], [34], [38], [40], [41], and [42] where the transport protocol stack in the sender and receiver are only modified. Transport protocols with network-assisted operation are given in [8], [9], [11], [13], [14], [18], [19], [24] and [27]. Thus it can be seen that many TCP variants have come up to address these issues and in this paper we have simulated the performance of some of these TCP variants in ns2 simulator to see their applicability for satellite based networks and proposed a modified Peach Protocol for satellite based networks.

2 TCP Protocol Variants

In this section the main characteristics of all the TCP Protocol variants which are analyzed in the simulation are discussed. An overview is provided about their main mechanism of action with the basic advantages and disadvantages.

2.1 TCP Tahoe

TCP Tahoe involved a few new algorithms in early TCP implementations like Slow-Start, Congestion Avoidance, and Fast Retransmit [42]. Among these the fast retransmit algorithm [42] is of special interest as it has been retained in its basic form in subsequent versions of TCP. In Fast Retransmit, after receiving a small number of duplicate acknowledgments for the same TCP segment (*dup ACKs*), the data sender infers that the packet has been lost and retransmits the packet without waiting for a retransmission timer to expire. Generally, it has been seen that the duplicate acknowledgment threshold is fixed as three. Therefore, on receiving three successive duplicate acknowledgments the sender can infer that receiver has not received the packet, and a retransmission is triggered without waiting for timeout.

Independency on the retransmission timeout for taking retransmission decisions of the lost packet and subsequent earlier loss recovery leads to higher channel utilization and connection throughput. The protocol returns to slow start and sets slow start threshold to one-half of the congestion window. This strategy does not change even if the number of packets dropped is more.

Resetting of congestion window to one irrespective of degree of congestion on the network drastically reduces TCP flow has some implication on the network performance and adversely affects it. For connections with a larger congestion window, the delay in slow-start because of the logic of setting the slow start threshold to half the

previous congestion window, can have a significant impact on overall performance of a Tahoe connection.

2.2 TCP RENO

TCP Reno [41] variant of TCP consists of the slow start, fast retransmit, fast recovery and congestion avoidance algorithms. During fast retransmit sender transmits what appears to be the missing segment and congestion avoidance begins. This is the fast recovery algorithm [41], which assumes that each dupack represents a single packet having left the network. The additional incoming duplicate acknowledgments clock the subsequent outgoing packets. The cwnd is set to ssthresh plus three packets. This inflates the cwnd by the number of segments that have left the network and which the other ends has cached [41]. Each time another dupack is received, cwnd is incremented by one. When the next ACK arrives that acknowledges new data, cwnd is set to S*sthresh* and the sender exits fast recovery. This scheme works optimally as compared to Tahoe, when there is a single loss within window as it halves the cwnd instead of resetting to one.

When multiple packet losses occur however, Reno's exit from fast recovery immediately after recovering from the first loss does not allow it to attempt recovery for remaining losses within the same window. Therefore, it has to depend on timeout for recovery. This creates a longer delay at the source, which dramatically reduces the throughput. The throughput is not suddenly decreased in Tahoe as, it always enters into slow start following fast retransmit, and does not depend on timeout causing larger delays at the source. One important point is that, Tahoe will retransmit all packets from the highest acknowledgment seen after fast retransmit. Because of this automatically Tahoe [42] is able to recover from multiple packet losses without reverting to timeout.

2.3 TCP NEW-RENO

TCP New-Reno [33] avoids many of the retransmit timeouts experienced by Reno. The New-Reno TCP includes a small change to

the Reno algorithm at the sender side that eliminates Reno's wait for a retransmit timer when multiple packets are lost from a window [33]. The change concerns the sender's behavior during Fast Recovery when a partial ACK is received, that acknowledges some but not all of the packets that were outstanding at the start of that Fast Recovery period. In New-Reno, partial ACKs do not end Fast Recovery. Instead, partial ACKs received during Fast Recovery indicate that the packet immediately following the acknowledged packet in the sequence space has been lost and has to be retransmitted. The delayed exit from fast recovery enables it to handle multiple retransmissions without reducing the window again. This in turn confirms the throughput is maintained. Thus, inside a single window when multiple packets are lost, New-Reno can recover without a retransmission timeout. One lost packet is retransmitted per round-trip time until all of the lost packets from that window have been retransmitted. New-Reno remains in Fast Recovery until all of the data outstanding when Fast Recovery was initiated, has been acknowledged.

Both Reno and New-Reno senders can retransmit at most one dropped packet per round-trip time, even if senders recover from multiple drops in a window of data without waiting for a retransmit timeout. Tahoe TCP does not exhibit these characteristics, which is not limited to retransmitting at most one dropped packet per round-trip time. This is an outcome of a fundamental inadequacy, in handling the recovery of lost packets and necessitates the use of selective loss recovery. In the absence of selective loss recovery handling, either retransmit at most one dropped packet per round-trip time, or retransmit packets that might have already been successfully delivered.

Reno and New-Reno use the first strategy, and Tahoe use the second. Therefore, a New-Reno sender is still unable to protect network completely from unnecessary retransmissions wasting bandwidth. Further, Dup ACK can also be caused by replication of ACK or data segments by the network. In these circumstances, the sender will attempt fast retransmission for a false dupack threshold.

2.4 TCP SACK

Cumulative acknowledgment scheme is generally used by transport protocols in which received segments that are not at the left edge of the receiver window are not acknowledged. Either this force the sender to wait a full roundtrip time to find out each lost packet or to retransmit segments that have been correctly received. TCP sender can only learn about a single lost packet per round trip time with the limited information available from cumulative acknowledgments. An aggressive sender could choose to retransmit packets early, but such retransmitted segments may have already been successfully received as in Tahoe [42]. Moreover, with the cumulative acknowledgment scheme, multiple dropped segments generally cause TCP to lose its ACK-based clock, reducing overall throughput.

The basic requirements of flow control and reliability are handled by TCP's 20-bytes header. Using the option field there is a provision for extending it further to make it more efficient. Window scaling already uses this facility. The same provision is exploited to acknowledge the segments, a receiver has received out of sequence and cached without forwarding them to application [38]. This acknowledgment is in addition to the regular acknowledgment, given by a receiver for the last correctly received segment in order. Multiple option bytes can be used by a receiver, to indicate noncontiguous blocks or Selectively Acknowledged (SACK blocks) of data, received beyond the expected sequence number [38]. The format followed by SACK option is given in [38]. The SACK option field contains a number of SACK blocks, where each SACK block reports a non-contiguous set of data that has been received and queued. In a SACK option, the first block is required to report the data receiver's most recently received segment, and the additional SACK blocks repeat the most recently reported SACK blocks. When the Timestamp option is used in the SACK option specified for TCP Extensions for High Performance, then the SACK option has room for only three SACK blocks [38]. Two TCP options are used in selective acknowledgment extension. The first is an enabling option SACK-permitted, which may be sent in a SYN segment to indicate that the SACK option can be used once the connection is established. The other is the SACK option itself, which

may be sent over an established connection once permission has been given by SACK-permitted. It is expected that SACK will often be used in conjunction with the Timestamp option, which takes an additional 10 bytes; thus, a maximum of three SACK blocks will be allowed in this case. SACK options should be included in all ACKs, which do not ACK the highest sequence number in the data receiver's queue.

Over an error prone high-delay internet path, simple experiments have shown that disabling the selective acknowledgment facility greatly increases the number of retransmitted segments. Simulation based study by Kevin Fall and Sally Floyd, demonstrates the strength of TCP with SACK over the non-SACK Tahoe and Reno TCP implementations. A sender has a better idea of exactly which packets have been successfully delivered when SACK is used as compared with comparable protocols lacking SACK. Having such information, a sender can avoid unnecessary delays and retransmissions, resulting in faster recovery and improved throughput. The use of TCP SACK is one of the most important changes that should be made to TCP to improve its performance [38]. The SACK TCP implementation preserves the properties of Tahoe and Reno TCP of being robust in the presence of out-of order packets, and uses retransmit timeouts as the recovery method of last resort. The main difference between the SACK TCP implementation and the Reno TCP implementation is in the behavior when multiple packets are dropped from one window of data.

SACK TCP maintains a variable called pipe during Fast Recovery, which is not present in Reno implementation that represents the estimated number of packets outstanding in the path. The sender only sends new or retransmitted data when the estimated number of packets in the path is less than the cwnd. The variable pipe is incremented by one when the sender either sends a new packet or retransmits an old packet. It is decremented by one when the sender receives a dupack with a SACK option reporting that new data has been received at the receiver. Use of *pipe* decouples the decision of when to send a packet from the decision of which packet to send. The sender maintains a data structure, which remembers acknowledgments from previous SACK options. When the sender is allowed to send a packet, it retransmits the next packet from the list of packets inferred to be missing at the

receiver. If there are no such packets and the receiver's advertised window is sufficiently large, the sender sends a new packet.

2.5 TCP FACK

Reno works well in a single packet loss scenario but degrades in case of multiple segment loss as the protocol is not able to estimate the amount of data outstanding in the network. The requisite network state information can be obtained with accurate knowledge about the forward most data held by the receiver. The forward most data means the correctly received data with the highest sequence number. The goal of the FACK [45] is to perform precise congestion control during data recovery by keeping an accurate estimate of the amount of data outstanding in the network by using the SACK option and by periodically updating some state variables.

2.6 TCP Vegas

TCP Vegas [40] incorporates a couple of modifications over Reno, especially in the way the Fast Retransmit is handled and the congestion avoidance algorithm. A new re-transmission strategy is used where retransmissions may occur before the reception of three duplicate acknowledgments. This strategy is taken to avoid the time wastage, in waiting for the three duplicate ACKs. TCP Vegas Congestion Avoidance looks at the change in the throughput rate or more specifically the sending rate. It compares the measured throughput rate with an expected throughput rate. The basic idea used by Vegas is to measure and control the amount of extra data the connection has in transit. By extra data, we mean data that would not have been sent if the bandwidth used by the connection exactly matches the available bandwidth of the connection. Mainly the goal of Vegas is to maintain the right amount of extra data in the network. Vegas has also recommended by the CCSDS [8] for use in the SCTS-TP protocol standard as the congestion control algorithm. Even though TCP Vegas is very successful, a lot of research is going on regarding the fairness [7][43] provided by TCP Vegas with its linear increase/decrease mechanism of congestion control. The problem with TCP Vegas is

observed when it is interoperated with other TCP 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.

2.7 TCP WESTWOOD

TCP congestion control algorithm is modified by TCP Westwood (TCPW) [23] to improve its performance, especially over error prone wireless links [39] such as satellite links. It incorporates only a sender-side modification and does not require any inspection and/or interception of TCP packets at routers. TCPW continuously monitors the arrival rate of acknowledgments to measure the bandwidth used by the connection at the sender side. The bandwidth is estimated by dividing the amount of data confirmed by an ACK; by the acknowledgment inter arrival time. Using a low pass filter, this estimate is smoothed over time and then used to compute congestion window size and slow start threshold after a congestion event. In the case of link congestion, TCPW selects a new congestion window size by taking into account the network capacity at the time of congestion instead of directly halving the congestion window size as TCP does. TCP Westwood based on the philosophy of bandwidth estimation from an estimate of the ACK received is good in wireless environments and very effectively utilizes the bandwidth mainly when used with other connections. It has good friendliness property, well adapts to New Reno flows, and can very effectively utilize the residual bandwidth when used in a heterogeneous environment. Evaluation results from the protocol developers show that TCPW outperforms TCP in cases of very high packet error rate. However, it is probably not an effective transport protocol in space Internet, because it does not solve the long propagation delay problem that is the major issue for performance enhancement in space-based networks.

3 Peach Protocol Implementation and Modification

The problem of large bandwidth-delay-product (BDP) and high BER over Satellite channels is addressed by TCP Peach [26]. It replaces slow start and fast recovery with sudden start and rapid recovery, along with the direct adaptation of classical congestion avoidance and fast recovery of TCP. The new mechanisms like sudden start and rapid recovery use a probing technique to estimate the availability of network resources by the use of low priority dummy packets. Sudden start addresses the degradation caused by a slow pace and long duration of the TCP slow start. TCP's misinterpretation of packet corruption as link congestion is addressed by Rapid Recovery because on a satellite link packet losses are more likely due to channel-error corruption. In Peach [26] successful acknowledgments of the received dummy packets are interpreted as an indication of bandwidth availability.

Dummy packets and regular packets are differentiated using attached control bits. Dummy packets are generally the first to be discarded by routers in case of link congestion because of their low priority. The routers for discarding low priority packets in the event of congestion require the priority information of packets. TCP Peach is considered an innovative way to address transmission control in TCP.

However, the capability to support priority drops in the routers in needed, which is presently unavailable. In addition to this, TCP Peach still allows fluctuation of congestion windows, leading to complicated implementations. Moreover, it could be potentially unfair if congestion causes many connections to send dummy packets at the same time, which may create wastage of the satellite bandwidth.

3.1 TCP Peach Slow Start Mechanism

The working of TCP Peach Protocol is shown in the flowchart of Fig 1. Instead of slow start algorithm TCP Peach uses Sudden Start where the congestion window starts with one but dummy packets are sent every RTT/rwnd seconds to probe the availability of the network resources. The idea is that if there are unused resources in the network, the

dummy packets will be acknowledged back after one RTT and with the arrival of the ACK for each dummy packet the congestion window can be likewise increased. Thus, the congestion window reaches to the optimum level after 2 RTT time instead of a binary slow increase of the congestion window. This significantly increases the throughput of TCP Peach. After the Sudden Start phase Peach moves to the Congestion Avoidance phase.

3.2 Peach Congestion Control

In the Congestion Avoidance phase on the reception of three duplicate acknowledgements the unacknowledged packet is sent by Fast Retransmit and Rapid Recovery is entered. In the Rapid Recovery phase the congestion window is halved and protocol will not let the transmission of data packets unless all the dummy packets probing the network are transmitted. Therefore, the network load is reduced as the dummy packets are low priority packets and they do not create load on the router. After the dummy transmission is over the data packets are transmitted up to the allowed congestion window value which is half the value of the window immediately before the Fast Retransmit took place. One RTT after the Fast Retransmit is done the ACK for the retransmitted packet should come back which ends the Rapid Recovery Phase. If the loss of the packet is because of congestion then the dummy packets will not be acknowledged and the congestion window will not be increased. If the packet loss is because of error in the channel then the dummy packets will be acknowledged and the congestion window will come to its original level after two RTT from the time when the problem started and only half the value of congestion window data will be not transmitted. This works fine giving a significant performance benefit compared to other TCP variants especially in the case of networks where the probability of error is more. The detail algorithm is elaborated in [26].

3.3 Modified TCP Peach

However, when the error becomes quite high and there are multiple packet losses inside the same window the Peach protocol goes on

decreasing the congestion window each time it receives three duplicate ACK. This recursive reduction in the value of congestion window leads to performance degradation mainly in cases of high channel error rate. Moreover, in satellite based networks where the main problem is not congestion but channel error this algorithm has a devastating effect on the value of the congestion window. So a modification of the protocol is done in this paper where the state of the protocol is remembered using a state variable. Once the protocol is inside the Rapid Recovery phase it will not react to the duplicate ACKs and no further reduction will be done in the congestion window. If the retransmitted packet does not come back after the Retransmission Timer is over then again Sudden Start algorithm is started as per the Peach Algorithm. This modification to the Peach protocol has been implemented in ns2 simulator [44] and the performance of the protocol is evaluated as explained in detail in Section 4 and found to be quite suitable for use in Geo Satellite based networks and mainly when the error is more.

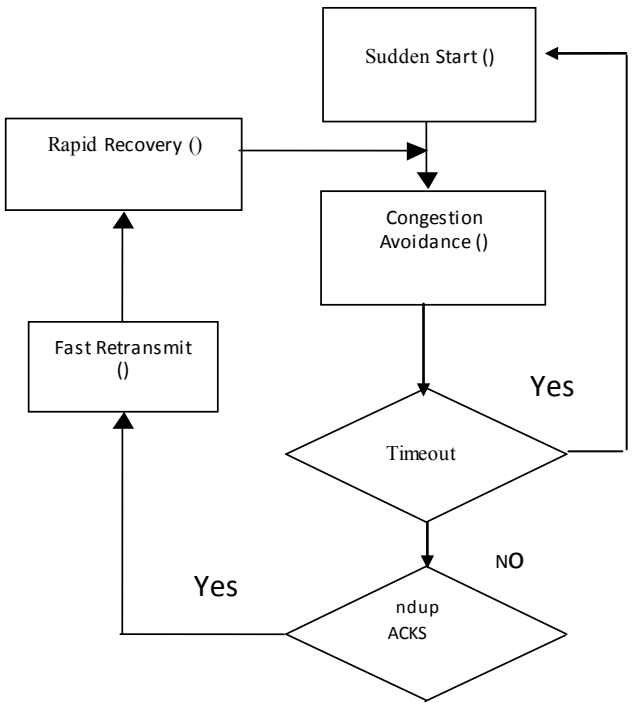

Fig. 1. Flow chart of Peach

The logic behind this decision is that when the Peach protocol has already taken some preventive action in response to the reception of duplicate ACK by reducing the congestion window to half, the network load is automatically decreased. So time has to be given to the network to adjust with this reduced transmission rate. It has been seen that in case of satellite networks, with more errors due to this recursive decrease the PEACH protocol performance degrades. In Section 4.0 we have simulated with the modified PEACH.

4 TEST AND Evaluation

We evaluate the performance of the proposed TCP in terms of goodput and fairness [7],[43] through simulations when several connections share the same link. We simulate the system as in Fig2 below where N senders transmit data to N receivers through a satellite channel. The N streams are multiplexed in Earth Station A, whose buffer can accommodate K segments. The segments may get lost with a packet error rate PER. In this experiment all the N senders are each connected to the Earth station A with a link of bandwidth 500kbps and RTT of 10ms. All the N receivers are connected to Earth station B with a 500kbps link with RTT 10ms. We have taken N = 10, K = 25 segments, rwnd = 64 segments the link between Earth Station A to B via satellite to be 5Mb and the RTT between the two stations as 550ms. All the results obtained in this section have been obtained by considering the system behavior for T_Simulation = 550s which is 1000 times the round trip time value.

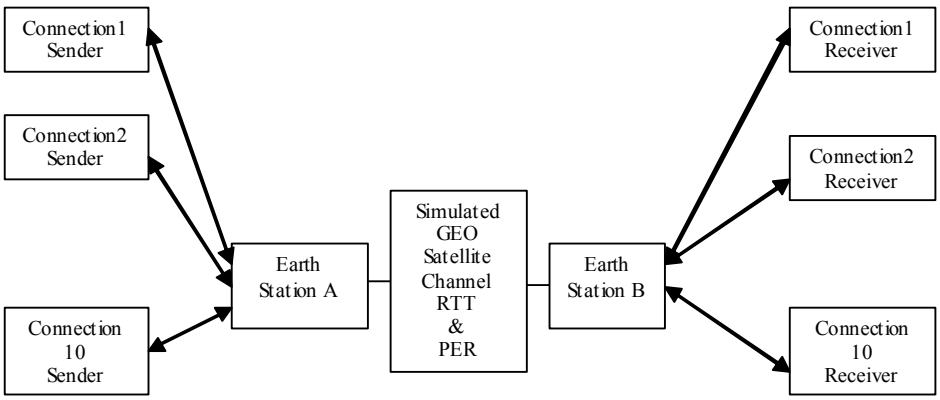

Fig. 2. Simulation Scenario

With each node is associated a FTP sender and receiver which tries to send certain amount of data and the maximum segment number reached after a specified point of time is used to calculate the average transmission rate. TCP variants like Tahoe [42], Reno [41], New Reno [33], SACK [38], FACK [45], Vegas [40], and Westwood [23] are simulated and the results are as shown in the Figures below. Here the average throughput obtained by the 10 connections is used for analysis. The modified peach protocol is also evaluated for its fairness property [7][43] and it is seen that all the connections get a fair share of the bandwidth The modified Peach Protocol is also simulated and it is seen to give a much better performance when the packet error rate is like 1 out of 100 packets in error or 1 out of 10 packets in error. This is because that the recursive decrease of the congestion window is restricted in the case of modified Peach. As a result better bandwidth utilization and performance of Peach can be seen when out of every 10 packets one packet is dropped. Note that although we consider only a GEO satellite system, we can obtain similar results for LEO and MEO satellite systems as well.

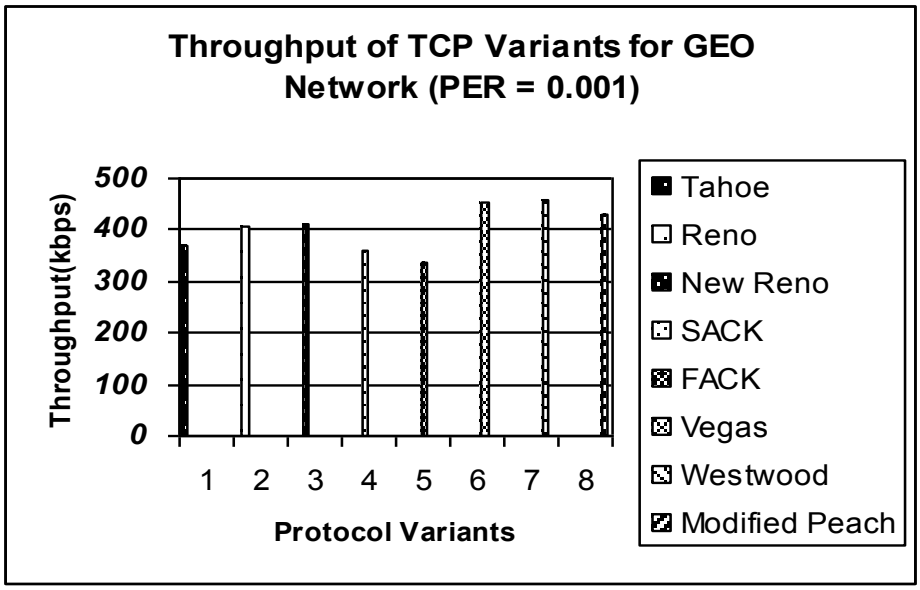

Fig. 3. TCP Throughput for PER = 0.001

For the evaluation of the performance of the modified peach protocol for a satellite based network using a GEO satellite we have considered three different error conditions. In Fig 3, the case considered is that of PER of 0.001 and it can be seen from the throughput obtained by the different TCP variants that Vegas, Westwood and Modified Peach performance is somewhat comparable. This is because of the fact that Peach uses dummy segments to figure out the cause of duplicate ACK arrival and when the error is not that much in the channel the overhead caused leads to performance degradation. In Fig.4, the case for more error where 1 out of 100 packets lost is considered and in this case it can be seen that the modified Peach protocol gives the best performance among all the eight TCP variants considered in the simulation. This is because the use of dummy segments gives the best estimate of the condition in the network.

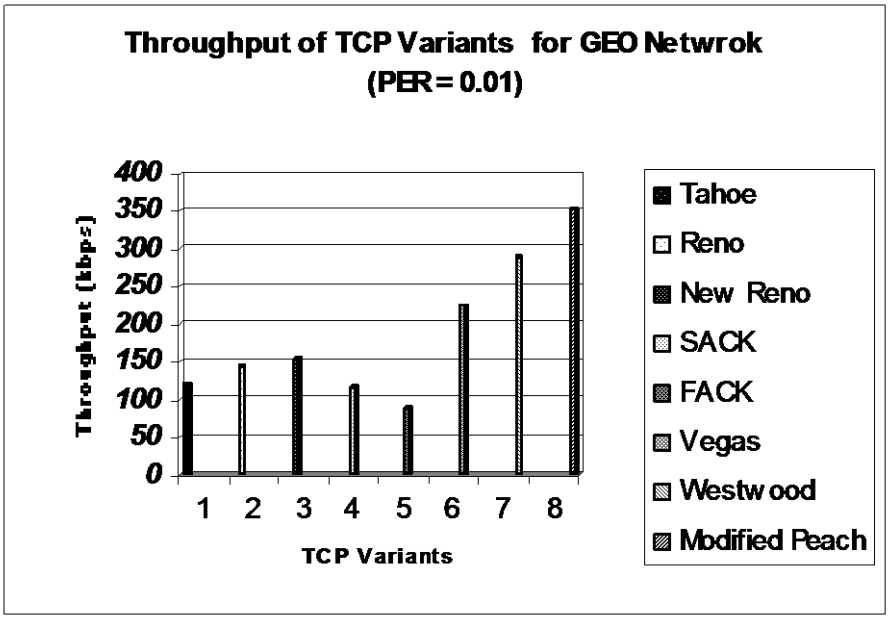

Fig.4 TCP Throughput for PER = 0.01

In Fig.5, the case where the channel error is very high is considered and 1 out of 10 packets is assumed to get dropped in this case. It can be seen from the plot that in this case the modified Peach Protocol outperforms all its peers by almost 100%. This is because of the fact that the recursive degradation of the congestion window is restricted in modified Peach and in this case multiple losses happen in the same window which is very well taken care.

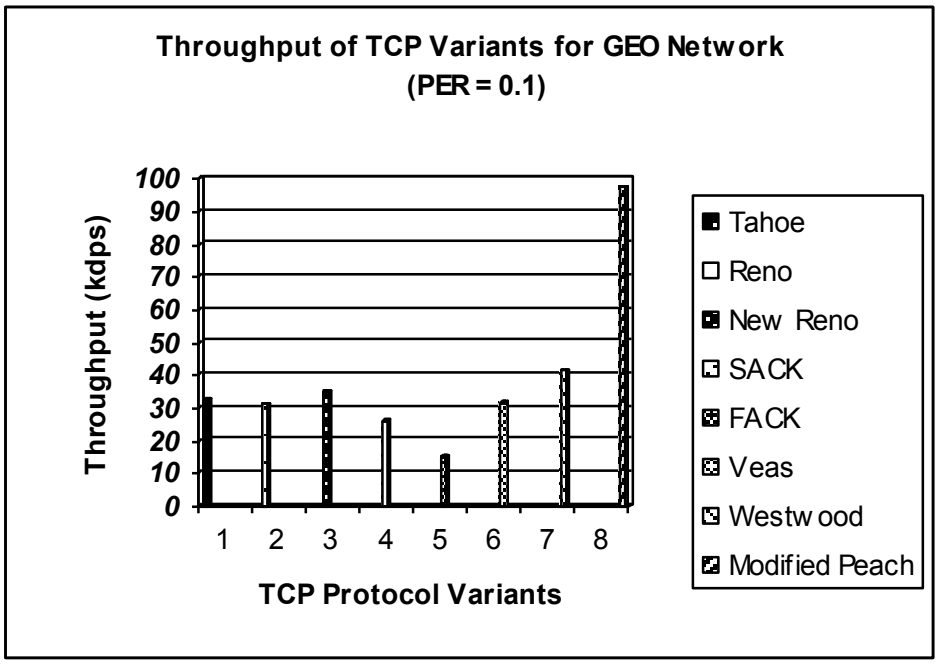

Fig. 5 TCP Throughput for PER = 0.1

5 Conclusion

We have augmented the NS-2 software with the support for Peach protocol and the implementation has been tested for various scenarios including a typical satellite scenario with varying packet error rates and the results are exactly the same as expected. Modified Peach has been found to outperform the other TCP protocol variants mainly in high channel error condition. The Modified Peach Protocol becomes a very good candidate to be used in Satellite Based Networks.

Acknowledgment
Authors would like to thank Mr.Deval Mehta, Head SCTD, Mr. A.P.Shukla, Group Head, DCTG, and Mr.K.S.Parikh, Deputy Director, SNAA of Space Applications Centre (ISRO) for their constant encouragement towards the realization of this work.

References

[1] Mohanchur Sarkar, K.K.Shukla and K.S.Dasgupta, Delay Resistant Transport Protocol for Deep Space Communication, International Journal of Communications, Network and System Sciences (IJCNS), Vol. 4, No. 2, February 2011, pp. 122-132.

[2] Mohanchur Sarkar, K.K.Shukla and K.S.Dasgupta, Network State Classification based on the Statistical Properties of RTT for an Adaptive Multi State Proactive Transport Protocol for Satellite based Networks, International Journal of Computer Networks and Communication (IJCNC), Vol. 2, No. 6, November 2010, pp. 155-174.

[3] Mohanchur Sarkar, K.K.Shukla, and K.S.Dasgupta, Performance Analysis of Proactive and Reactive Transport Protocols using a Proactive Congestion Avoidance Model, International Journal of Computer Applications (IJCA), Vol. 6, No. 5, September 2010, pp. 10-17.

[4] Mohanchur Sarkar, K.K.Shukla, and K.S.Dasgupta, A Proactive Transport Protocol for Performance Enhancement of Satellite based Network, International Journal of Computer Applications (IJCA), Vol. 1, No. 16, Feb 2010, pp. 114-121.

[5] L. Xu and I. Rhee, **C**UBIC: a new TCP-friendly high-speed TCP variant**,** ACM SIGOPS Operating Systems Review, Vol. 42 Issue 5, July 2008.

[6] R.H. Wang, W.T. Hsu, X. Wu, T. T. Wang and X. B. Wang, Experimental and Comparative Analysis of Channel Delay Impact on Rate-Based and Window-Based Transmission Mechanisms over Space-Internet Links, Proceedings of IEEE International Conference on Communications, Beijing, May 2008, pp. 2990-2994.

[7] S.C. Tsao, Y.C. Lai, and Y.D. Lin, Taxonomy and Evaluation of TCP Friendly Congestion Control Schemes on Fairness, Aggressiveness, and Responsiveness, IEEE Network, Vol. 21, Issue 6, November 2007, pp. 6-15.

[8] Space Communications Protocol Specification (SCPS) - Transport Protocol (SCPS-TP), Recommended Standard CCSDS 714.0-B-2 Blue Book, October 2006.

[9] SatLabs Groups, Interoperable PEP (I-PEP) transport extensions and session framework for satellite communications: air interface specifications, Techinicala Report, Issue 1a-27, October 2005.

[10] I. F. Akyildiz, O. B. Akan, and G. Morabito, A Rate Control Scheme for Adaptive Real-Time Applications in IP Networks with lossy links and long round trip times, IEEE/ACM Transactions on Networking, Vol. 13, Issue 3, June 2005, pp. 554-567.

[11] T. Taleb, N. Kato, and Y. Nemoto, REFWA Plus: Enhancement of REFWA to Combat Link Errors in LEO Satellite Networks, in Proc.2005 IEEE Workshop on High Performance Switching and Routing, Hong Kong, P. R. China, May 2005, pp. 172-176.

[12] C. Liu and E. Modiano, On the performance of additive increase multiplicative decrease (AIMD) protocols in hybrid space terrestrial networks, Elsevier Computer Networks, Vol. 47, Issue 5, April 2005, pp. 661–678.

[13] Kai Xu, Ye Tian and Nirwan Ansari, TCP performance in integrated wireless communications networks, The International Journal of Computer and Telecommunications Networking, Vol. 47, Issue 2, February 2005.

[14] K. Zhou, K. Yeung and V. O. K. Li, P-XCP: A-transport layer protocol for satellite IP network, In Proc. IEEE Globecom, November 2004, pp. 2707-2711.

[15] C. Caini and R. Firrincieli, TCP Hybla: a TCP enhancement for heterogeneous networks, International Journal of Satellite Communication and Networking, Vol. 22, No. 5, September 2004, pp. 547–566.

[16] Jian Fang and Ozgur B. Akan, Performance of Multimedia Rate Control Protocols in InterPlaNetary Internet, IEEE Communications Letters, Vol. 8, No. 8, August 2004, pp. 488-490.

[17] L. Xu, K. Harfoush and I. Rhee, Binary increase congestion control for fast, long distance networks, In Proceedings of IEEE INFOCOM, Hong Kong, Vol. 4, March 2004, pp. 2514-2524.

[18] M. Luglio, M. Sanadidi, M. Gerla, and J. Stepanek, On-board satellite split TCP proxy, IEEE Journal on Selected Areas in Communication (JSAC), Vol. 22, No. 2, February 2004, pp. 362-370.

[19] M. Marchese, M. Rosei and G. Morabito, PETRA: Performance enhancing transport architecture for satellite communications, IEEE Journal on Selected Areas in Communication, Vol. 22, No. 2, February 2004, pp. 320-332.

[20] O. B. Akan, J.Fang and I. F. Akyildiz, TP-Planet: A Reliable Transport Protocol for InterPlaNetary Internet, IEEE/SAC, Vol. 22, No. 2, February 2004, pp 348-61.

[21] K. Fall, W.Hong, and S. Madden, Custody Transfer for Reliable Delivery in Delay Tolerant Networks, Technical Report IRB-TR-03-030, Intel Research, Berkeley, July 2003.

[22] C.P Fu and S.C.Liew, TCP Veno: TCP enhancement for transmission over wireless access networks, IEEE Journal on Selected Areas in Communications, Vol. 21, Issue 2, Feb 2003, pp. 216-228.

[23] C. Casetti, M. Gerla, S. Mascolo, M. Sanadidi, and R. Wang, TCP Westwood: End-to-end congestion control for wired/wireless networks, ACM/Kluwer Wireless Networks (WINET) Journal, Vol. 8, Issue 5, September 2002, pp. 467-479.

[24] D. Katabi, M. Handley, and C. Rohrs, Congestion control for high bandwidth-delay product networks, In Proceedings of ACM SIGCOMM02, Pittsburg, August 2002, pp. 89-102.

[25] Ian F. Akyildiz, Xin Zhang and Jian Fang, TCP-Peach+: Enhancement of TCP-Peach for Satellite IP Networks, IEEE COMMUNICATIONS LETTERS, Vol. 6, No. 7, July 2002.

[26] Ian F. Akyildiz, Giacomo Morabito and Sergio Palazzo, TCP-Peach: A New Congestion Control Scheme for Satellite IP Networks, IEEE/ACM TRANSACTIONS ON NETWORKING, Vol. 9, No. 3, June 2001, pp. 307-321.

[27] J. Border, M. Kojo, J. Griner, G. Montenegro, and Z. Shelby, Performance enhancing proxies intended to mitigate link-related degradation, RFC 3135, June 2001.

[28] R. Steward et al., Stream Control Transmission Protocol, Internet Society, Reston, VA, RFC 2960, October 2000.

[29] S. Floyd, M. Handley, J. Padhye, and J. Widmer, Equation-Based Congestion Control for Unicast Applications, ACM SIGCOMM Computer Communication Review, Vol 30, Issue 4, October 2000, pp. 45-56.

[30] M. Allman, S.Dawkinks, et al., Ongoing TCP research related to satellites, RFC 2760, Feb 2000.

[31] C. Barakat, E. Altman, and W. Dabbous, On TCP performance in a heterogeneous network: a survey, IEEE Communications Magazine, Vol. 38, Issue 1, Jan 2000, pp. 40-46.

[32] N. Ghani and S. Dixit, TCP/IP enhancement for satellite networks, IEEE Communication Magazine, Vol. 37, No. 7, July 1999, pp. 64-72.

[33] S. Floyd and T. Henderson, The New Reno Modification to TCP's Fast Recovery Algorithm, RFC 2585, April 1999.

[34] T. Henderson and R. Katz, Transport protocols for Internet-compatible satellite networks, IEEE Journal Selected Areas in Communication, Vol. 17, No. 2, February 1999, pp. 326-344.

[35] C. Metz, TCP over satellite the final frontier, IEEE Internet Computing, Vol. 3, Issue 1, January 1999, pp. 76–80.
[36] C. Partridge and T. J. Shepard, TCP/IP performance over satellite links, IEEE Network Magazine, Vol 11, Issue 5, October 1997, pp. 44–49.
[37] T.V.Lakshman and U.Madhow, The performance of TCP/IP for networks with high bandwidth-delay products and random loss, IEEE/ACM Trans. Networking, Vol. 5, Issue 3, June 1997, pp. 336-350.
[38] Mathis, M., J. Mahdavi, S.Floyd, and A.Romanow, TCP Selective Acknowledgment Options, RFC 2018, April 1996.
[39] H. Balakrishnan, S. Seshan, E. Amir and R. H. Katz, Improving TCP/IP Performance over Wireless Networks, Proc. ACM MOBICOM, November 1995, pp. 2-11.
[40] L. S. Brakmo, S. O'Malley and L. L. Peterson, TCP Vegas: New Techniques for Congestion Detection and Avoidance, ACM SIGCOMM Computer Communication Review, Vol. 24, Issue 4, October 1994, pp. 24-35.
[41] V. Jacobson, Berkeley TCP Evolution from 4.3-Tahoe to 4.3-Reno, Proc. British Columbia Internet Engineering Task Force, July 1990.
[42] V. Jacobson, Congestion avoidance and control , ACM SIGCOMM Computer Communication Review, Vol. 18, Issue 4, August 1988, pp. 314-329.
[43] R. Jain, D. Chiu and W.Hawe, A Quantitative Measure of Fairness and Discrimination for Resource Allocation in Shared Computer Systems, DEC, Research Report TR-301, September1984.
[44]UCB/LBNL/VINT Network Simulator. http://www.isi.edu/nsnam/ns/
[45] M. Mathis and J. Mahdavi, "Forward Acknowledgment: Refining TCP Congestion Control", ACM SIGCOMM Computer Communication Review, Vol.26, Issue 4, pp. 281-291, October 1996.