Composite TCP: Refining Congestion Control

Bhavika Gambhava[1†] and C.K. Bhensdadia[††],

[†] Charotar University of Science and Technology, India
[††] Dharmsinh Desai University, India

Summary
Internet users need reliable transmissions for web browsing, email, file transfer, and database access. TCP is the dominant reliable transport protocol on top of which all of these services run. Since TCP is originally developed to be used on the wired network, it safely assumes that segment losses are due to congestion. This is not true for wireless media where, due to fading channels and user mobility, transmission errors are more frequent. It unconditionally reduces its flow when a packet loss is detected, assuming that it has occurred because of congestion in the network. This incorrect assumption in case of a packet loss occurring due to channel noise, adversely affects the performance of TCP. This paper focuses on the same problem of TCP related to random errors causing packet loss. Treatment for losses should be different for corruption and congestion to improve efficiency. The proposed Composite TCP (CTCP) utilizes the SACK information available from a receiver along with round trip time to avoid flow control when the loss is because of a random error. The performance b of Composite TCP is compared and analyzed with existing variants over networks with exhaustive simulations using ns-2. Simulation results indicate that the proposed protocol significantly improves performance.

Key words:
SACK TCP, congestion control, Round Trip Time, cwnd, ssthresh

1. Introduction

TCP/IP is a well-proven and accepted protocol suite, which has successfully ensured stable and robust network operations since evolution. These properties have made TCP/IP protocol suite an inseparable part of the Internet. The ongoing ramping up demand for Internet calls for new architectures and new technologies capable of providing high quality and high speed Internet services. TCP is the dominant reliable transport protocol, used for web browsing, email, file transfer etc[1]. Thus, TCP has an imperative role in the performance of the Internet.

TCP provides reliable service along with end-to-end connectivity at the transport layer [2]. TCP increases the packets sending rate if no packet losses are encountered. Due to the inherent reliability of wired networks, there is an assumption made by TCP that any packet loss is due to congestion. To reduce congestion, TCP will start its congestion control mechanism whenever any packet loss is detected. Due to its extensive use in the Internet, it is advisable that TCP remains in use to offer reliable services for communications in wireless networks and in heterogeneous networks.

In this paper, we propose a new approach to discriminate random losses from congestion induced losses. Retransmission and flow reduction are decoupled by identification of the type of the loss. Most of the congestion induced losses are multiple in nature. A single loss from the window can be attributed to random errors. Estimated delay of the connection is examined to affirm the cause of the loss. Remedial action is decided based on the type of loss to achieve improved network performance as well as robustness. The proposed scheme is evaluated with established existing versions of TCP by various simulations.

The rest of the paper is organized as follows. We explain existing TCP variants in the next section. A new algorithm called Composite TCP is explained with state transition diagram in Section 3. Simulation environment along with topologies are described in Section 4. The simulation results of conduct simulations are analyzed in Section 5. We conclude the paper in Section 6.

2. Existing Variants of TCP

TCP has undergone many revisions in the past three decades. Wired and wireless networks are notably different in terms of bandwidth, speed, propagation delay, and channel reliability [3].

The connotation of the diversity is that packet losses are not only because of congestion, but can also result from characteristics of wireless links. While TCP performs well in wired networks, it might deteriorate performance severely in wireless networks if it wrongly considers non-congestion-related losses as a sign of congestion and consequently invokes congestion control, as reported in [3,4,5].

TCP ensures reliability by using a retransmission timer. TCP increases the congestion window (cwnd) by one per acknowledgement. Hence, it effectively doubles cwnd during every round trip time (RTT). When cwnd reaches slow start threshold (ssthresh), it enters congestion avoidance phase. cwnd is increased by one per an RTT in this phase. If data is not acknowledged prior to timer

Manuscript received March 5, 2019
Manuscript revised March 20, 2019

expiration, the TCP sender retransmits the data. TCP sets back cwnd to one when Retransmission timeout (RTO) occurs. It is due to the original design of TCP for wired networks, where congestion was the prime cause of packet losses.

A loss is detected by the arrival of three duplicate acknowledgements (dupack) in Tahoe TCP [6]. Fast retransmission of the lost packet is attempted without waiting for a timeout.

TCP Reno introduced fast recovery, which halves cwnd on the loss detection by dupacks[7]. TCP Reno can recover from a single packet loss, but it is not capable of recovering from multiple losses from the same window.

TCP New Reno improves its predecessor's performance by using the enhanced recovery procedure in the case when multiple packets are lost [8].

Selective Acknowledgement (SACK) TCP uses header options to convey SACK blocks to the transmitter[9]. Each SACK block selectively acknowledges non-contiguous segments. SACK TCP can avoid unnecessary retransmissions from the knowledge of available data at the receiver. New Reno TCP with SACK options are commonly implemented in accepted operating systems.

TCP Vegas proactively tries to overcome the impact of congestion [10]. It does not only rely on packet loss as a signal of congestion. It estimates the unacknowledged data in the buffer and senses congestion before the actual packet loss. It calculates the optimal throughput by maintaining the minimum RTT. TCP Vegas faces problems of unrelenting congestion, rerouting and discrepancy in flow rate [11].

Linux TCP [12] sender is managed by a state machine to determine the actions on arrival of acknowledgements. Linux TCP implements TCP enhancements proposed by Internet Engineering Task Force (IETF), like ECN [13] and D-SACK [14].

Forward Acknowledgements (FACK) aggressively considers the unacknowledged data between the SACK blocks as lost packets[15]. Although this approach may result in improved TCP performance at times, it is certainly belligerent if packets are reordered in the network.

TCP Fast Start [16] modifies TCP's conventional slow start. The sender caches some network parameters to shun paying the slow start penalty for subsequent connections. There is a risk involved if the cached information is stale.

AFStart TCP [17] approaches ssthresh rapidly as compared to conventional slow start because cwnd is initialized with 4 packets. An abrupt increase in cwnd obtains the available resources, which may result in to performance issues with other flows.

Novel Quick Start [18] initializes the value of cwnd to the detected network bandwidth. Error in predicting bandwidth is abridged in succeeding iterations. A sudden change in cwnd may cause congestion.

An EQF (Explicit Queue-length Feedback) [19] uses the length of the queue of the congested switch port as a potential congestion to prompt the sender to control the sending rate of the sender by invoking congestion control.

3. Composite TCP

TCP performance is determined by its congestion control mechanism, which restricts the amount of transmitted data based on the estimated network capacity [20]. Congestion control mechanisms are based on either loss or RTT. We propose an algorithm which considers both the parameters at the time of setting cwnd. Hence, we call it a Composite TCP. Composite TCP (CTCP) utilizes SACK blocks along with RTT to make a decision about the type of the loss.

In the proposed scheme, transmitter keeps track of incoming SACK blocks, reported in duplicate acknowledgements. Most of the congestive losses are neither random nor sparse in nature. If all consecutive segments are reported in SACK blocks of subsequent acknowledgements following a loss, the loss indicated by three dupacks may not be due to congestion but it may be a random loss [21]. The presumed erroneous loss is also confirmed by comparing current RTT with SRTT. SRTT stands for Smoothed Round Trip Time. For each connection, TCP maintains SRTT, which is the best estimate of the round-trip time to the destination in question. The loss is treated to be an erroneous loss if RTT is smaller than or equal to SRTT. Congestion causes queuing delay, which would result in higher RTT.

When there is a single loss in a window and RTT also does not indicate congestion, then we can safely consider the loss to be a random loss [21]. In the case of such scattered losses, cwnd reduction can be avoided and only retransmission is attempted to recover from the loss. If subsequent loss takes place in the same window, a gap in SACK information is reported by the receiver. This indicates a possibility of another loss in the same window. If there are multiple losses in the same window, congestion can not be ruled out. Then the TCP sender changes the values of ssthresh and cwnd in the fast recovery phase. If the second loss is not detected, but RTT is found to be larger than SRTT, then also conventional fast recovery is followed. CTCP is designed to be conservative at the slightest signal of congestion because network resources are shared by many users and thus should be responsibly utilized.

Slow start and congestion avoidance phases are same as SACK TCP as shown in the state diagram of fig. 1. Whenever three dupacks are obtained indicating a single loss, then the fast retransmission is attempted. Delay of the link is considered at this stage to make an important

Manuscript received March 5, 2019
Manuscript revised March 20, 2019

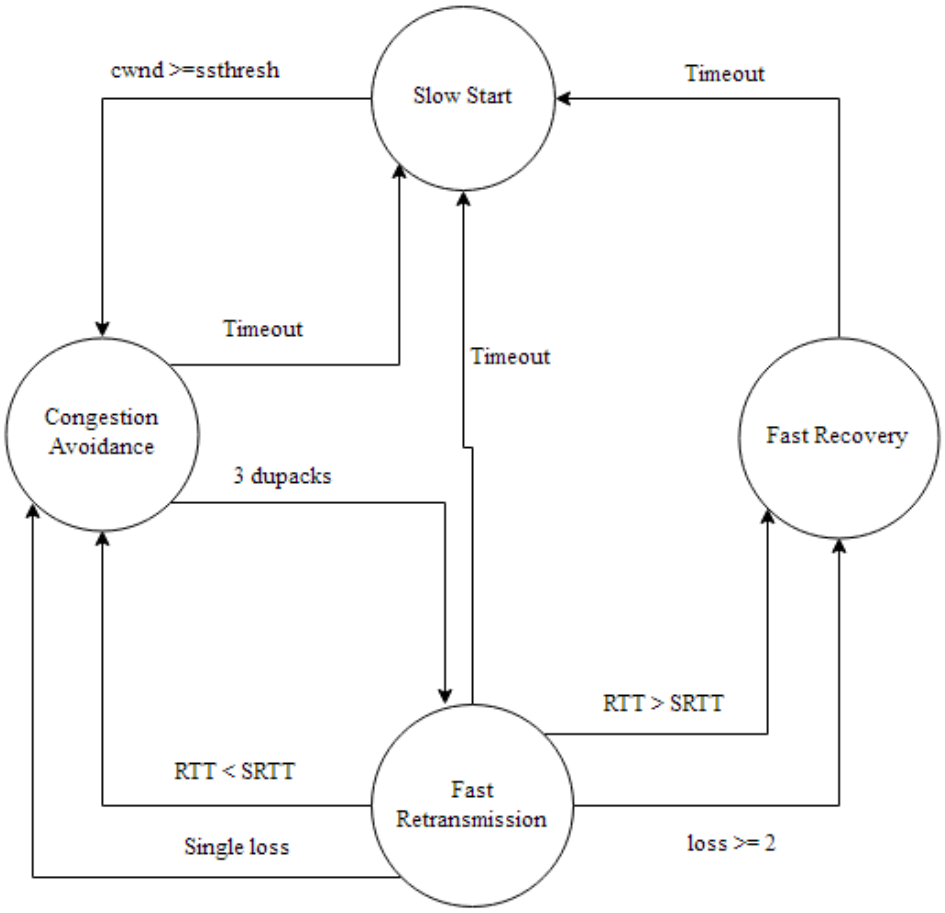

Fig. 1 State diagram of CTCP

decision of choosing the next state. If RTT is found to be lesser than SRTT, then cwnd reduction is shunned and the fast recovery phase is ignored. However, larger RTT will make the proposed algorithm revert back to the conservative behavior of TCP to protect the network against potential congestion. Timeout leads to slow start at any point of time during data transfer.

The Composite TCP remains unchanged in the following scenarios,

i) TCP sender does not detect any loss.
ii) If multiple losses are detected in a single transmission window.
iii) RTT is found to be larger than SRTT.

The fast recovery phase of the CTCP sender is same as that of SACK TCP. CTCP does not require any change at the receiver side and at the intermediate terminals. Composite TCP requires only sender side to incorporate modifications, which makes is effortlessly deployable and interoperable with senders and receivers, involving different TCP versions.

4. Simulation Environment

The performance of Composite TCP is evaluated with aforementioned TCP variants in terms of total number of successfully delivered packets to the receiver. The simulations are carried out in different network setups. The experiments were performed on ns-2[22]. Reno TCP (Reno) [7], Selective Acknowledgement TCP (SACK) [9], Vegas TCP(Vegas) [10], Linux TCP(Linux) [12], Forward acknowledgement TCP(FACK) [15] and Fast Start TCP(FS) [16] are considered for assessment.

Three different topologies are employed to test the performance of CTCP in all the possible scenarios. The real networks can be of any of these types: 1. Errors without congestion 2. Congestion without errors 3. Errors with congestion.

The simulations are carried out in the presence of random channel errors, commonly encountered in wireless links. Different error-rates (0.00, 0.001, and 0.01) in terms of

percentage of packet losses are configured during simulations to examine the impact on the performance. 0.00 is used to check the network performance in the absence of errors, which will obviously be the best performance offered by the link. Error rate 0.001 is configured to see the behavior of the network in presence of moderate errors. It is further increased to 0.01 to see how the protocol withstands when severe errors occur over the wireless links.

Application layer protocol, FTP traffic is generated for 100 seconds. Each packet is considered to be 1500 bytes for ensuring compatibility with Ethernet. All the parameters are mentioned in table 1.

Table 1: Simulation parameters

Simulator	ns2
Application	FTP
Packet size	1500 byes
Link type	Full duplex
Time	100 sec
Error rates	0, 0.001, 0.01
Error model	Random. Uniform (0-1)
Queue of router	50
Queue	Droptail

4.1 Errors without congestion

Frequent errors on wireless links hamper TCP performance by reducing data flow as discussed earlier. This topology allows errors without any scope to congestion so as to check the impact of random errors on TCP performance. Fig. 2 shows an experiment scenario which evaluates the performance of the network with errors only. Terminal 1 and terminal 3 are transmitter and destination respectively. Intermediate node 2 is configured to act as a router. All the links are duplex in nature with bandwidth 100 Mbps. The propagation delay of the transmission lines is 50 msec. The router is capable to transfer data to the receiver at the same speed as the receiving from the transmitter. Equal incoming and outgoing capacity at the router does give any chance to congestion. The packets were corrupted at the router based on the specified error rate and they were discarded. All these packet losses are due to corruption. This topology is used to illustrate a simple wireless network and impact of errors on the same.

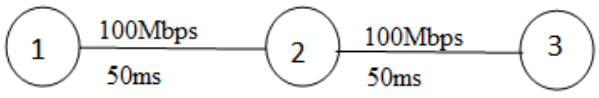

Fig. 2 Erroneous network topology

4.2 Congestion without errors

Fig. 3 describes a scenario with two senders (terminal 1 and terminal 2). As shown in the fig. 3, the bandwidth of the link is 10 Mbps and propagation delay is 1 msec. The bottleneck between intermediate node 3 and terminal 4, is configured to operate at 2 Mbps data rate with 10ms propagation delay. Intermediate node 3 is a router and terminal 4 is the receiver. The queue size of the router is 50 and it drops packets when the queue is filled. These parameters certainly create congestion at the router because of total incoming traffic from senders is 20 Mbps whereas the outgoing link is configured to be only of 2 Mbps. This environment was designed to be error free to examine how congestion deteriorates the performance of the network.

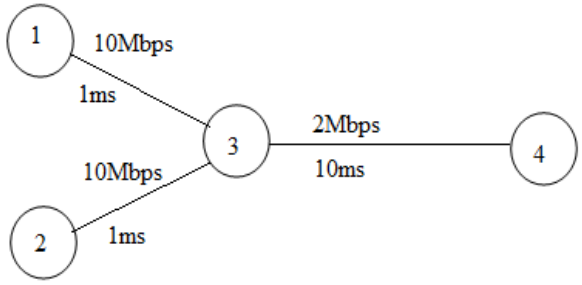

Fig. 3 Congested network topology

4.3 Errors with Congestion

Simulations are further performed on a topology shown in fig. 4. Two senders (terminal 1 and terminal 2) are connected to router1 (intermediate node 3) via 10 Mbps transmission link and 1 msec propagation delay. The bottleneck between router1 and router 2(intermediate node 4) is characterized to have 5Mbps data rate with 100ms propagation delay. Source1 is sending data to destination 1(node 5) and source2 is transmitting to destination 2(node 6). Router2 is connected to receivers by link having 10Mbps bandwidth and 1ms Delay. Error model is active at router1 to generate losses due to corruption. This topology may experience packet losses because of congestion or due to random errors. This network topology simulates a heterogeneous network having wired and wireless links together.

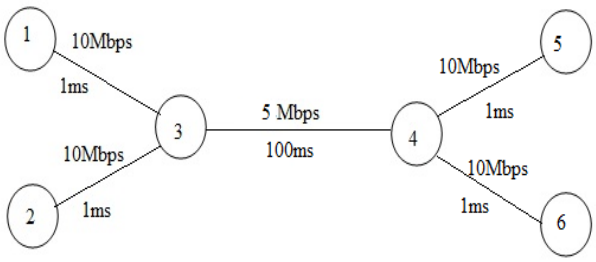

Fig. 4 Erroneous congested network topology

5. Simulation Results and Analysis

The results are scrutinized by considering the number of packets successfully delivered over the simulation period of 100 seconds. The error rate is varied from 0.00 to 0.001 and 0.01 in order to check the response of the network in the absence of errors, in presence of moderate error rate and in the presence of severe error rate. The results of three topologies are discussed in this section.

Table 2. Number of delivered packets for erroneous network

	Error=0	error=0.001	error=0.01
SACK	24762	15940	6331
Vegas	24587	19836	8456
Linux	146302	17567	6561
FACK	24762	17009	6226
FS	24762	14598	5259
Reno	24762	16225	5049
CTCP	24762	24067	12102

5.1 Errors without congestion

As the incoming and outgoing link capacity is identical in the topology of fig. 2, no packet is dropped because of congestion.

The results are analyzed on the basis of number of packets successfully delivered. The error rate was kept 0.00 initially to check the response of all variants in utopian condition. CTCP is a modified version of SACK TCP. We confirm here no actions by our proposed algorithm if no errors are encountered. The error rate was then increased moderately to 0.001. Most of these random losses were successfully identified by CTCP. The same can be verified by remarkable performance improvement as shown in table II. The error rate was further increased to 0.01 to check the response in the presence of highly error-prone links. CTCP surpasses all other protocols once again as can be observed from the last column of table II. The performance gain in the number of successfully delivered packets varies from 43.1% to 130.1% in CTCP as compared to other protocols. The rationale behind this improvement is detection of random losses by delaying fast recovery and apt usage of RTT to rule out any possibility
of congestion. Reduction in cwnd was avoided, which resulted in better utilization of the available bandwidth.

5.2 Congestion without errors

In order to examine compatibility with the existing terrestrial wired network, the next set of experiments was carried out in a congested environment without link impairments. Incoming data flow at a router, as shown in fig. 3, is much higher than the capacity of the outgoing link. Multiple losses in a single transmitted window tend to be present due to congestion. Therefore, there was no scope for CTCP to avoid a reduction in cwnd. All TCP versions perform more or less same. CTCP performs 1% better than Reno TCP, while Vegas TCP delivers 2.4% more packets as compared to CTCP. CTCP targets and tries to identify scattered error losses, commonly present in a wireless link. All TCP versions perform nearly same as seen in table III. However, it is noteworthy that DTCP does not deteriorate network performance when deployed in severely congested environment.

5.3 Errors with Congestion

Real networks are likely to suffer from errors and congestion together. Most of heterogeneous networks experience packet losses due to congestion as well as errors. The same was tested on topology shown in fig. 4 to examine all the possibilities. The error rate was increased to 0.001 to evaluate behavior in a realistic environment.

X axis displays variant of TCP and Y axis presents number of successfully delivered packets in fig. 5. Nearly the same number of packets is delivered by all variants of TCP, whenever there was no error present in the network as shown in the fig. 5. The error was increased to 0.001 to evaluate behavior in a realistic environment. CTCP transfers highest number of packets followed by Vegas TCP. CTCP delivered 6.7% and 19.3% more packets than Vegas TCP and Linux TCP, respectively. When error rate was further increased to 0.01 to observe the impact of highly noisy wireless links, CTCP outperformed all other schemes considered for experiments. CTCP's performance was found to be 42.7% and 103.5% higher than Vegas TCP and Linux TCP, respectively. It can be observed from the fig. 5 that CTCP performs better when higher error rate was encountered. It validates CTCP's ability to discriminate the kind of losses and treat them accordingly.

Table 3. Number of delivered Packets for error free congested network

	SACK	Vegas	Linux	FACK	FS	Reno	CTCP
Sender 1	7274	8334	8192	8148	8560	7366	7220
Sender 2	8983	8330	8084	8119	7610	8728	9039
Total Packets	16257	16664	16276	16267	16170	16094	16259

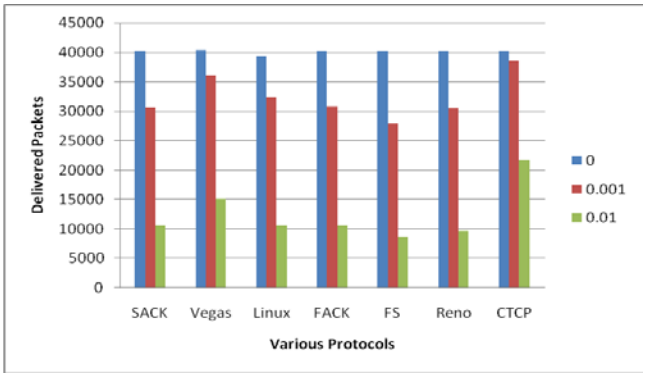

Fig. 5 Number of delivered Packets for erroneous and congested network

6. Conclusion

TCP's inherent shortcoming in distinguishing the random losses from congestive losses is addressed in this paper. Composite TCP (CTCP) uses loss based and rate based parameters to decide the cause of the loss. Random losses are recovered by fast retransmissions without cutting down cwnd. Thus, CTCP improves performance by efficient utilization of the bandwidth.

Error rate, delay and congestion are used as the parameters to evaluate the performance of CTCP over different topologies. Simulation experiments are conducted to compare CTCP with other TCP variants. Simulation results point out substantial performance enhancement from 43.1% to 130.1% in the presence of errors in CTCP as compared to other TCP variants. Behavior of CTCP is at par with other variants for congested networks.

CTCP is an alternative implementation that interoperates with any other valid implementation of TCP. CTCP proposes changes only at the sender part of TCP, while keeping header, routers and receiver implementation unchanged.

Acknowledgment

The authors would like to express their cordial thanks to Dr. Brijesh Bhatt and Dr. Nikhil Kothari for their valuable advice.

References

[1] M. Duke, E. Blanton, A. Zimmermann, R. Braden and W. Eddy, "A roadmap for transmission control protocol (TCP)", RFC 7414, 2015.

[2] Y. Tian, K. Xu and N. Ansari, "TCP in wireless environments: problems and solutions", IEEE Communications Magazine, 43(3), 2005, S27~S32.

[3] Xiaomei Yu, "A new mechanism to enhance transfer performance over wired-cum-wireless networks", IEEE International Symposium on IT in Medicine & Education, 2009.

[4] Z. Fu, P.Zerfos, H. Luo, S. Lu, L. Zhang, M. Gerla, "The Impact of Multihop Wireless Channel on TCP Throughput and Loss", IEEE INFOCOM'03, San Francisco, March 2003.

[5] Z. Fu, X. Meng, and S. Lu, "How Bad TCP can Perform in Mobile Ad-Hoc Networks", IEEE Symposium on Computers and Communications,Italy, July 2002.

[6] V. Jacobson, "Congestion avoidance and control", ACM SIGCOMM computer communication review, vol. 18, no. 4, 1988, 314~329.

[7] M. Allman, V. Paxson, and E. Blanton, "TCP congestion control", RFC 5681, 2009.

[8] T. Henderson, S. Floyd, A. Gurtov and Y. Nishida, "The NewReno modification to TCP's fast recovery algorithm", RFC 6582, 2012.

[9] S. Floyd, M. Jamshid, M. Matt and P. Matthew., "An extension to the selective acknowledgement (SACK) option for TCP", RFC 2883, 2000.

[10] U. Hengartner, B. Jürg and G. Thomas, "TCP Vegas revisited", In: Proc of Nineteenth Annual Joint Conference of the IEEE Computer and Communications Societies, vol. 3, 2000, 1546~1555.

[11] K. Srijith, J. Lillykutty and A. Ananda, "TCP Vegas-A: solving the fairness and rerouting issues of TCP Vegas", In: Proc. of IEEE conference on Performance, Computing and Communications, 2003, 309~316.

[12] P. Sarolahti, and A. Kuznetsov, "Congestion Control in Linux TCP", In: USENIX Annual Technical Conference, FREENIX Track, 2002, 49~62.

[13] K. Ramakrishnan, S. Floyd and D. Black, "The addition of explicit congestion notification (ECN) to IP", RFC 3168, 2001.

[14] E. Blanton and M. Allman, "Using TCP duplicate selective acknowledgement (DSACKs) and stream control transmission protocol (SCTP) duplicate transmission sequence numbers (TSNs) to detect spurious retransmissions", RFC 3708, 2004.

[15] M. Mathis and J. Mahdavi, "Forward acknowledgement: Refining TCP congestion control", ACM SIGCOMM Computer Communication Review, vol. 26, no. 4, 1996, 281~291.

[16] V. Padmanabhan and R. Katz, “TCP fast start: A technique for speeding up web transfers”, In: Proc. IEEE Globecom, 1998.
[17] Y. Zhang, N. Ansari, W. Mingquan and H. Yu, “AFStart: An adaptive fast TCP slow start for wide area networks”, In: Proc of IEEE International Conference on Communications, 2012, 1260~1264.
[18] D. Zhang, K. Zheng, D. Zhao, X. dong Song, and X. Wang, “Novel quick start (QS) method for optimization of TCP”, Wireless Networks, pp. 211-222, 2016.
[19] Y. Lu, X. Fan and Lei Qian, “EQF: An explicit queue-length feedback for TCP congestion control in datacenter networks”, In: Proc. of Fifth International Conference on Advanced Cloud and Big Data, 2017, 69~74.
[20] Y. Narasimha Reddy, P V S. Srinivas, “A Combined TCP-friendly Rate control with WFQ Approach for Congestion Control for MANET”, International Journal of Computer Network and Information Security(IJCNIS), Vol.10, No.6, pp.52-59, 2018.DOI: 10.5815/ijcnis.2018.06.05
[21] N. Kothari, K. Dasgupta, “Performance Enhancement of SACK TCP Protocol for wireless Network by Delaying Fast Recovery”, WOCN 2006.
[22] http://www.isi.edu/nsnam/ns/,Network Simulator ns-2.

Bhavika Gambhava has completed her B.E. from L.D.College of Engineering in 2004 and M.E. from Dharmsinh Desai University in 2006. She is a research scholar at Charotar University of Science & Technology, Changa, India.

C. K. Bhensdadia is a professor and head of computer engineering department, Dharmsinh Desai University, India. He is one of the 15 members of All India Board of Information Technology Education (AICTE). He plays a key role in curriculum design of various universities. He has contributed significantly towards projects for rural development for Department of Information Technology, Ministry of Communication and IT, Government of India and has also worked for consultancy project of a European country.