SSRG International Journal of Electronics and Communication Engineering
ISSN: 2348-8549/ https://doi.org/10.14445/23488549/IJECE-V11I7P110

Volume 11 Issue 7, 98-107, July 2024

Original Article

A Novel Approach to Enhance the Performance of TCP SACK on Wireless Links

Rahul Dhirendrabhai Mehta[1], Hitesh Thakarshibhai Loriya[2], Divyesh Rudabhai Keraliya[3]

[1,3]*Electronics and Communication Engineering Department, Government Engineering College - Rajkot, Gujarat Technological University, Gujarat, India.*
[2]*Electronics and Communication Engineering Department, L E College - Morbi, Gujarat Technological University, Gujarat, India.*

[1]*Corresponding Author : rdmehta@hotmail.com*

Received: 08 May 2024 Revised: 07 June 2024 Accepted: 07 July 2024 Published: 27 July 2024

Abstract *- TCP has become one of the most prominent contributors to the TCP/IP protocol stack that drives the Internet. TCP is known for offering reliable, connection-oriented services to Internet users. TCP carries around 85% of the Internet traffic, which puts an immense responsibility on its shoulders to offer the utmost reliability and faster services. Unfortunately, the protocol that was initially developed for wired networks gradually became unable to cope with the issues offered by growing wireless networks and hence felt incompetent in providing quality services to competing Internet traffic. The researchers have shown keen interest in making TCP equally adaptive and competent to the growing networks; hence, revisions in basic TCP have started taking place. Over the decades of research, today's TCP has become completely different from its original version and competent enough to perform far better in the presence of various wireless links. However, the area where TCP is still lacking is the absence of intelligence to discriminate between the losses due to random errors and the losses due to congestion. This research paper focuses on adding adequate intelligence to differentiate between the types of losses and act accordingly. The proposed algorithm integrates unique modifications into an existing variant, TCP SACK, which passes the testing phase and performs quite superior to the original and older versions. The simulations are performed on a standard Network Simulator (ns-2), and plots are drawn from the traces of observations. Overall, the work is useful in enhancing the performance of SACK TCP on wireless erroneous links by adding the intelligence to differentiate between types of losses and act accordingly.*

Keywords *- Congestion control, Fast recovery, Random loss, RTT, Selective ACKnowledgement (SACK), TCP.*

1. Introduction

The Internet has grown massively in the last decades. This tremendous growth is mainly due to the heterogeneous networks, also known as mixed-mode networks. The heterogeneous networks consist of wired and wireless networks and links. Data reliability has become an essential issue in these complex networks.

The reliability issues in the TCP/IP protocol stack are handled by the most reliable transport layer protocol, Transmission Control Protocol (TCP), which sits between the application layer and the underlying connectionless and unreliable IP layer. Due to the growth in various wireless links, the up-gradation of TCP was also required. The reason is that the TCP was initially designed for wired links, which had the only cause, congestion, for the packet loss. However, as the count of wireless networks has grown, there comes an additional factor, Bit Error Rate (BER), causing the error and, hence, packet loss over and above the existing reason for packet loss, called congestion.

However, as we know, loss due to congestion is entirely different from loss due to random errors, and remedies to come out of each type of loss must be different. To deal with this problem, researchers have shown a great interest in modifying and revising TCP to adapt it to the growing modern network requirements [1].

Various TCP revisions like TCP Tahoe, Reno, New Reno, and many more came into existence to improve the performance of existing TCP on heterogeneous networks. The TCP SACK (Selective ACKnowledgement), one of the most profound revisions of TCP, has impressed users and developers with its unique policies to deal with retransmissions, resulting in an excellent performance rise compared to other TCP versions.

However, even after decades of extensive research, unfortunately, the TCP has yet to find adequate intelligence to find the cause of packet losses and act accordingly. This research focuses on discriminating packet losses occurring

due to BERs, which cause random losses from a loss due to congestion and then act accordingly [2]. Section 2 explains the baseline TCP and major standard revisions of TCP. Section 3 describes the proposed modified scheme based on the base algorithm. Section 4 discusses the topologies and parameters used to test the proposed algorithm. Section 5 summarizes the observations in tabular form with analysis and discussion. Section 6 concludes the research work.

2. TCP Versions

TCP is a transport layer connection-oriented protocol that provides reliable services to Internet users. It also has flow and error control to recover from the losses. It uses process-to-process communication for reliable data delivery [3]. A few significant revisions done on the baseline TCP are discussed below:

2.1. TCP Tahoe

TCP Tahoe has built-in Slow Start, Congestion Avoidance, and Fast Retransmit mechanisms. The algorithm performs well in the initial state. However, a slow start and multiplicative decrease restrict the transmission rate in the presence of congestion, which drops the congestion window to one when the loss is detected, reducing overall transmission flow [4].

2.2. TCP Reno

TCP Reno has an additional feature called Fast Recovery. This TCP variant maintains the channel with packets after a retransmission in case of one packet loss. Hence, a slow start is avoided. TCP executes a fast recovery phase once three duplicate acknowledgements are received. The lost packet is retransmitted, and the congestion window is set to half of the current congestion window instead of one, as was done in the case of TCP Tahoe. This improves performance significantly but suffers when multiple packet losses occur in a single data window.

2.3. TCP New Reno

TCP New Reno has an additional integration of modified fast recovery to handle multiple packet losses within a single data window. It leaves a Fast Recovery phase once the last packet is retransmitted. However, this needs to be more efficient regarding bandwidth while retransmitting during fast recovery [5].

2.4. TCP SACK (Selective ACKnowledgement)

High bandwidth data networks have a greater probability of multiple packet loss in a single data window. Despite recovering from losses without waiting for the expiration of the retransmission timer, it still sometimes becomes hard to stay away from the issues, reducing overall efficiency. In the Selective ACKnowledgement scheme, the receiver sends an acknowledgment with the SACK option enabled, which contains the information about which data were received correctly and in the correct order. So, the transmitter has to resend only the missing data after getting the three duplicate acknowledgements. The information helps calculate new congestion windows and pump data effectively on the channel to gain overall throughput [6].

3. Proposed Modification Scheme

The widespread literature survey depicts that various proposed modifications have been implemented and tested in integration with the baseline TCP on various network and link parameters to gain performance. It is evident that only fast retransmission is required in case of a random loss without altering the current transmission flow, but unfortunately, SACK TCP also adapts the strategy of retransmission and transmission flow reduction. A loss due to congestion reflects the current data jamming conditions of the network, which have to be addressed with transmission flow reduction to reduce severe congestion. It gives the network time and support to release the highly occupied network resources. Hence, there is a scope of research on adding intelligence that can distinguish between causes of packet loss. The search for a reference algorithm ended with discovering an algorithm called 'Delayed Fast Recovery' (DFR), later considered a reference algorithm. Further modifications are done to improve the performance of TCP SACK on wireless networks [7, 8].

3.1. Proposed Modification Scheme - Modified TCP SACK

The literature survey confirms that the random BERs cause single scattered random packet loss without any more frequent successive loss in the same data window. The Round Trip Time (RTT) is considered one of the most fascinating network parameters, which, if observed minutely, can reveal helpful information about the current state of the network. In this research, the RTT is observed with other network dynamics. Efforts have been made to modify the existing TCP SACK to perform even better in average throughput and delay. In case of an increment in error rate, RTT also increases because of the loss recovery procedures. Also, RTT increases in case of congestion due to bottlenecks at intermediate routers. However, the rate of change of RTT, i.e., deviation in RTT, is entirely different in each case. It takes a much longer time to release severe congestion compared to the recovery from random arbitrary losses.

This insight helps differentiate between the causes of packet losses and required subsequent actions. To identify the severity of the congestion and random losses, RTT deviations are categorized into specific tolerance levels, and decisions of counter actions are triggered accordingly. In addition, the second loss in a single data window is also treated as a random loss if the RTT deviation is within the tolerance limit. This action is applied to all the subsequent losses in a single data window in case of transmission errors, which will also help counter-attack bursty errors [9]. The extended flow chart accommodating this strategy is shown in Figure 1.

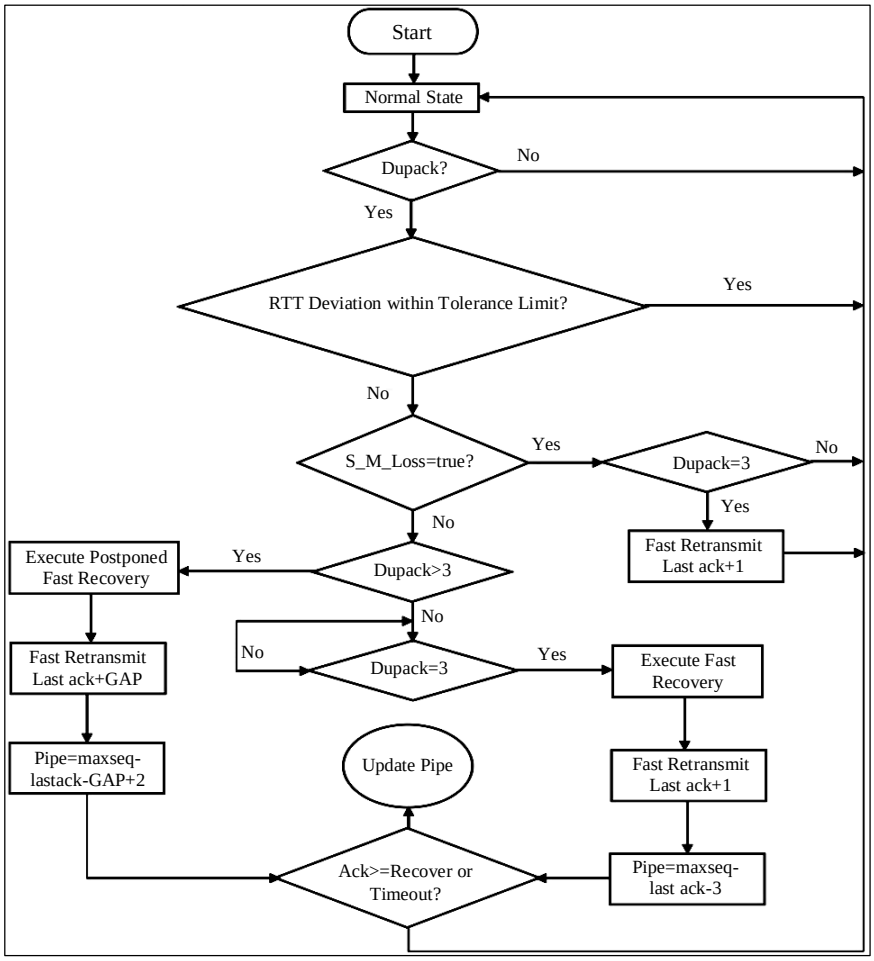

Fig. 1 Proposed modification algorithm flow chart

As shown in Figure 1, the proposed modification algorithm flow chart explains the new strategy. The RTT is continuously recorded, averaged out, and stored to get the current status of the network. When a duplicate acknowledgement is seen on the sender side, the RTT tolerance level check is performed to determine whether it is within the tolerance limit. If RTT is within the tolerance limit, the protocol maintains its normal state. However, if the RTT deviation is above the tolerance limit, flag S_M_LOSS is checked, indicating Single or Multiple loss.

If the flag is enabled and three duplicate acknowledgements are received, then it will consider this loss a random loss due to BERs, and only fast retransmission of the lost packet will be done based on the SACK information received. However, suppose S_M_LOSS is disabled (which happens in case of consecutive loss), and the duplicate count is greater than 3. In that case, the fast recovery, which was postponed, will be executed to recover from the loss, considering this as a congestion loss, and the parameters will be updated accordingly. The recovery will always be delayed for all the losses for which RTT is within the tolerance limit by considering all as random losses. However, if a loss occurs with the RTT value above the tolerance limit, fast recovery is immediately executed to help the network escape severe congestion.

4. Simulation Topologies and Parameters

4.1. Simulation Topologies

Two different realistic scenarios are considered to test the performance of modified TCP SACK. As shown in Figure 2, the topology simulates the erroneous environment with no congestion. In contrast, Figure 3 shows the topology for simulating the effects of error and congestion, which is the most realistic case.

All simulations are performed on Network Simulator (ns-2) for 10 seconds to measure an initial response and for 100 seconds to observe the initial and sustained response of the network and the protocol under test, i.e., to observe the effects of small and large file size transfer scenarios. File Transfer Protocol (FTP) is run on the application layer to generate the required variable bit rate traffic that chooses TCP from the transport layer [10]. The performance is measured in terms of the number of packets delivered and the time taken to do the

same. Observations are tabulated and plotted for comparative analysis.

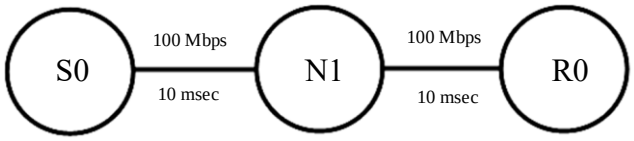

Fig. 2 Simulation scenario – I (only error, no congestion)

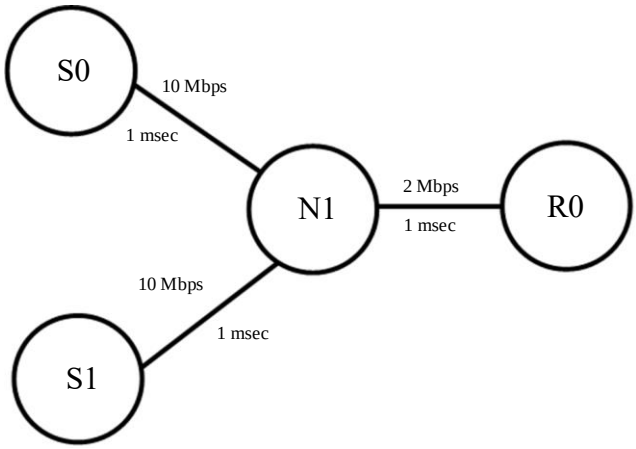

Fig. 3 Simulation scenario – II (error and congestion)

4.2. Simulation Parameters

- Topologies: Topology-I - Simulating Errors only, no Congestion. Topology-II - Simulating Errors and Congestion all together.
- Simulation Time: 10 seconds for initial response and 100 seconds for sustained response.
- Packet Error Rates: 0.0, 0.00001, 0.0001, 0.001, 0.01
- TCP Agent: Original SACK TCP and Modified SACK TCP (with RTT tolerance deviations 0%, 2%, 5%, 10% and 20%).
- Performance Measurement: Instantaneous Throughput vs. Time, Congestion Window vs. Time and Number of Packets delivered.

5. Simulation Result Analysis and Discussion

Simulation outcomes of the Original TCP and Proposed modifications for various RTT tolerance upper bounds are listed in Tables 1 to 4.

The first column lists various error probabilities of Packet Error Rates (PERs) observed on various standard wireless links. The next column lists several packets transmitted for the Original TCP SACK, and the following subsequent columns tabulate the performance of the proposed modified TCP SACK with various percentage RTT tolerances.

The parameter, number of fresh packets delivered to the network, shows the condition of the sender, whether the sender is busy resending and recovering from losses or sending fresh packets, by discriminating between the type of loss and resource demands of the respective types of losses. The congestion window, the prime parameter, is observed and controls the overall transmission flow [11].

5.1. Initial Response – Topology – I

An in-depth analysis of simulation outcomes and plots reveals the following facts about data transfer quality: Table 1 shows the observations for the simulations performed on Topology-I for original and modified SACK TCP with various RTT tolerances. The presence of only errors and absence of congestion with a simulation time of 10 seconds is considered in this scenario.

As can be seen from the observations, in the cases of PER 0, 0.00001, and 0.0001, which show the cases of no error and very few errors, respectively, there is no significant change in the number of packets sent. Hence, original and modified protocols perform equally well. However, as PER increases to 0.001 and 0.01, the significant change in the number of packets sent drastically reduces by 62% and 226% for respective PERs in the case of the original TCP SACK compared to the modified one.

It is also observed that the reduction in performance in the case of original TCP SACK reduces by 158% between 0.001 to 0.01 PERs. Modified SACK TCP also has the same problem of performance reduction by 29% between 0.001 and 0.01 PERs. This shows that when there is no congestion, the errors occurring causing PERs are very well recognized as a loss due to random errors, and the treatment to fight with the same, i.e., retransmission, is performed only. There is no sign of transmission flow reduction, which saves overall network performance and sustains it faster than the original SACK TCP. The performance reduction is mainly due to the number of retransmissions that restrict the transmission of new packets.

The above result analysis is equally supported by the plot shown in Figure 4. A 3-D plot is chosen to represent the effects of various error rates on several packets delivered in the presence of original and modified TCP SACK and in the presence of various RTT tolerances. Variations of TCP SACK are plotted on the X-axis, the number of packets delivered to the network is on the Y-axis, and on the Z-axis, all error rates are marked.

Observations and plots discussed above show the average throughput of the protocol under test. In addition, the instantaneous throughput variants and congestion window alterations play an essential role in understanding and justifying the overall performance of the original and modified protocols and providing a platform for graphical comparisons. Figure 5 shows a group of figures that compare the congestion window and throughput patterns for original and modified TCP SACK for a simulation time of 10 seconds and 0.001

error rates. The upper portion of the figure shows the congestion window alterations, whereas the bottom half shows throughput variations of respective original and modified TCP SACK protocols. As can be seen from the figures, the congestion window alterations are more frequent for OTCP (Original TCP), and hence, its throughput is not steady and is gradually falling. On the contrary, MTCP (Modified TCP) shows a very rare congestion window alteration, which causes a rise and smoother, steadier throughput response [12-14].

Table 1. Packets delivered (only errors – 10 seconds)

Packet Error Rates (PERs)	Original SACK TCP	Modified SACK TCP / RTT Deviation 0	Modified SACKTCP / RTT Deviation 2	Modified SACKTCP / RTT Deviation 5	Modified SACK TCP / RTT Deviation 10	Modified SACK TCP / RTT Deviation 20
0.00	12213	12213	12213	12213	12213	12213
0.00001	12213	12213	12213	12213	12213	12213
0.0001	12213	12213	12213	12213	12213	12213
0.001	7182	11330	11680	11680	11680	11680
0.01	2775	7672	9046	9046	9046	9046

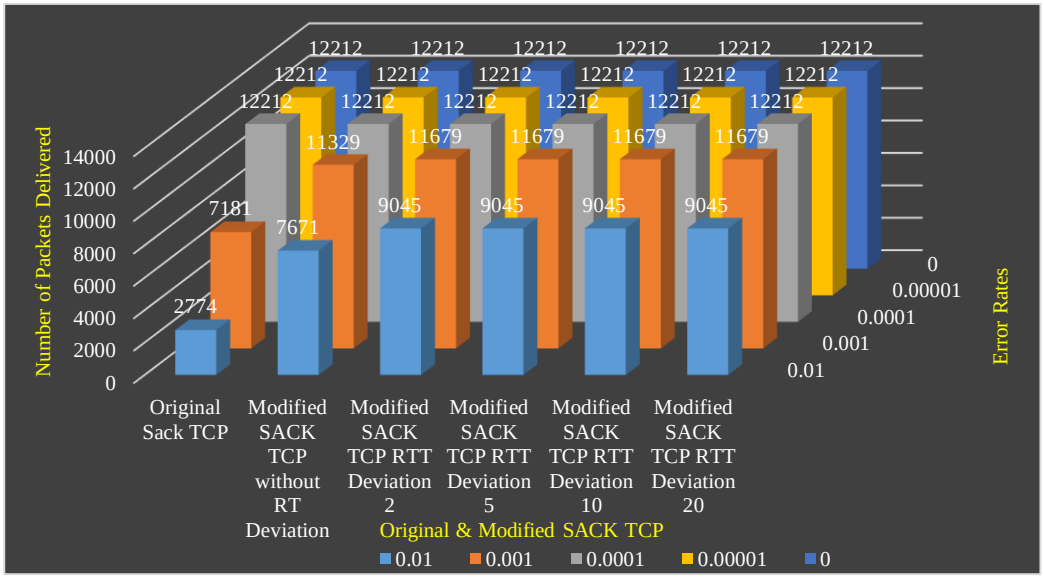

Fig. 4 Number of packets delivered (only errors – 10 seconds)

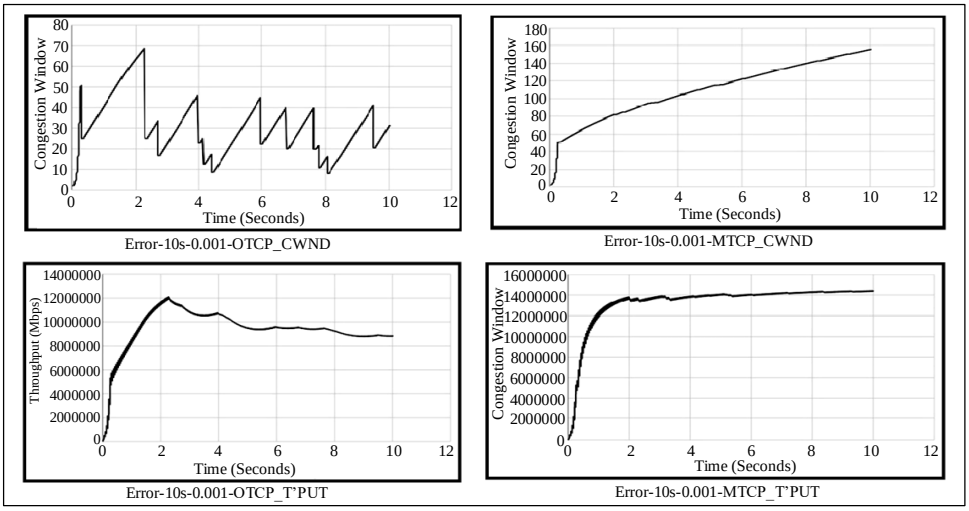

Fig. 5 CWND and throughput – 10S – 0.001 PER

5.2. Sustained Response - Topology - I

The initial response has the effects of connection setup and slow start phase of TCP connection setup phase. Once the connection is established and sustained, it will maintain a higher data rate as it will be running at the highest values of its parameters. Table 2 shows the observations for a simulation period of 100 seconds, which covers the effects of the initial and sustained response of the protocol. The analysis reflects the same observations as in the previous case but with higher data rates. There is no distinct difference in the performance for error rates of 0, 0.00001, and 0.0001. However, the modified protocol performs better in the case of 0.001 PER by 38% and continues to rise higher with higher RTT tolerance. The effect on performance can be higher with a higher error rate of 0.01, which increases by 241% compared to the original TCP SACK. Figure 6 graphically supports the results discussed. Figure 7 shows the same response and pattern as shown and discussed earlier for a 0.001 error rate with just an addition of more frequent congestion window alterations and throughput variations due to higher error rates and transmissions. Still, in this scenario, MTCP also has fewer congestion window alterations and sustains a higher throughput than OTCP, confirming that modified TCP performs better.

Table 2. Packets delivered (only errors – 100 seconds)

Packet Error Rates (PERs)	Original SACK TCP	Modified SACK TCP / RTT Deviation 0	Modified SACK TCP / RTT Deviation 2	Modified SACK TCP / RTT Deviation 5	Modified SACK TCP / RTT Deviation 10	Modified SACK TCP / RTT Deviation 20
0.00	124013	124013	124013	124013	124013	124013
0.00001	123593	123593	123593	123593	123593	123593
0.0001	120097	120097	120097	120097	120097	120097
0.001	86883	119833	120033	120183	120183	120183
0.01	28312	91848	96609	96609	96609	96609

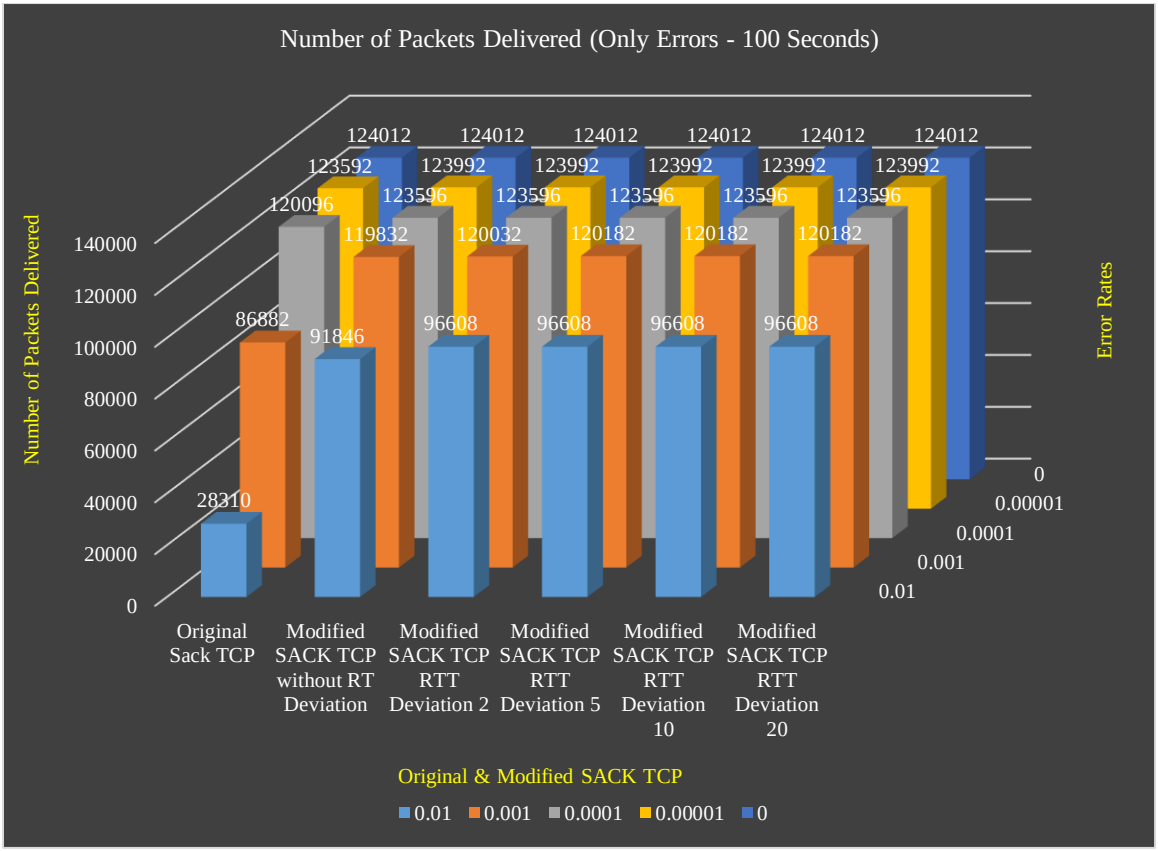

Fig. 6 Number of packets delivered (only errors – 100 seconds)

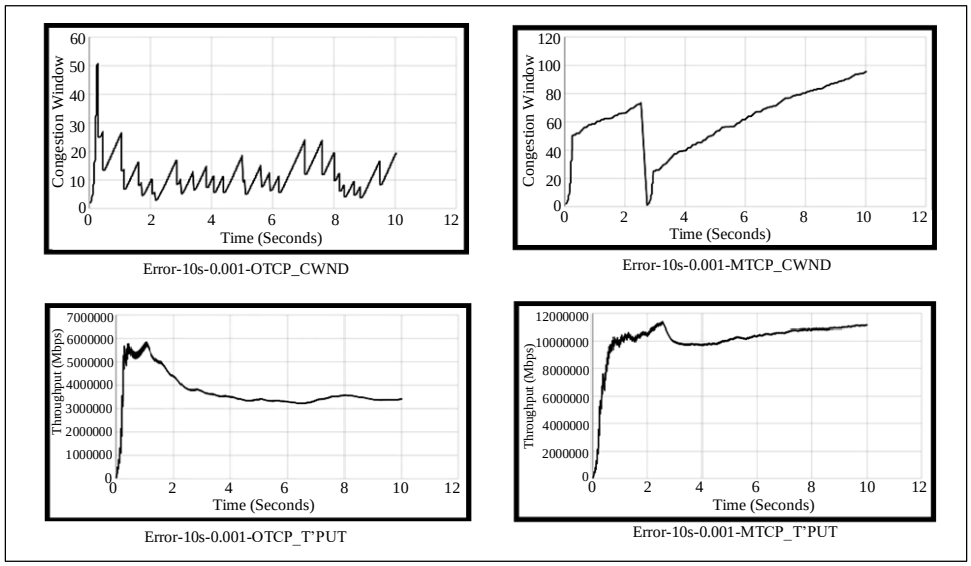

Fig. 7 CWND and throughput – 10S – 0.01 PER

5.3. Initial Response - Topology - II

In the case of Topology – II, which simulates the effects of congestion and errors together, there is no significant difference between the overall performance in the original and the modified TCP SACK as there exists the loss due to congestion with the loss due to error rates.

Hence, due to the integrated intelligence of discriminating between a loss due to congestion and a loss due to random errors, the effects of congestion take over the effects of random loss, and fast recovery is immediately executed to deal with the severe effects of congestion.

This shows the backward compatibility of the modified algorithm, which turns off the modifications in case of congestion loss and treats them accordingly. So, there is no impact of increment in PER or RTT tolerances as the network is dealing with actual congestion. One thing to be marked here is that the modified TCP SACK performs equal to the original TCP SCK in case of congestion.

In the case of 10 seconds of simulation time for a network with congestion and errors acting together, there is only a rise or fall in performance by just 0.13%, which is negligible. This is because the detection of congestion immediately calls for fast recovery, which suddenly reduces the transmission flow. Hence, a modified TCP SACK acts similarly to the original TCP SACK, so a performance rise cannot be expected. The plot shown in Figure 8 compares the observations listed in Table 3.

Figure 9 shows the higher congestion window alterations and lower throughput in the case of 0.001 for the original TCP, simulated for 100 seconds, showing the protocols' sustained response. In contrast, modified TCP shows a better throughput performance and rare congestion window reduction. In this case, the congestion window reaches approximately 110 as it takes more time to grow. In contrast, the congestion window reaches a maximum value of around 70 due to staying in the initial response period only. The same logic applies to the maximum throughput capacity.

Table 3. Packets delivered (errors & congestion – 10 seconds)

Packet Error Rates (PERs)	Original SACK TCP	Modified SACK TCP / RTT Deviation 0	Modified SACK TCP / RTT Deviation 2	Modified SACK TCP / RTT Deviation 5	Modified SACK TCP / RTT Deviation 10	Modified SACK TCP / RTT Deviation 20
0.00	1648	1668	1665	1665	1669	1668
0.00001	1648	1668	1665	1665	1669	1667
0.0001	1648	1668	1665	1665	1669	1668
0.001	1667	1631	1661	1661	1661	1653
0.01	1632	1664	1664	1664	1664	1664

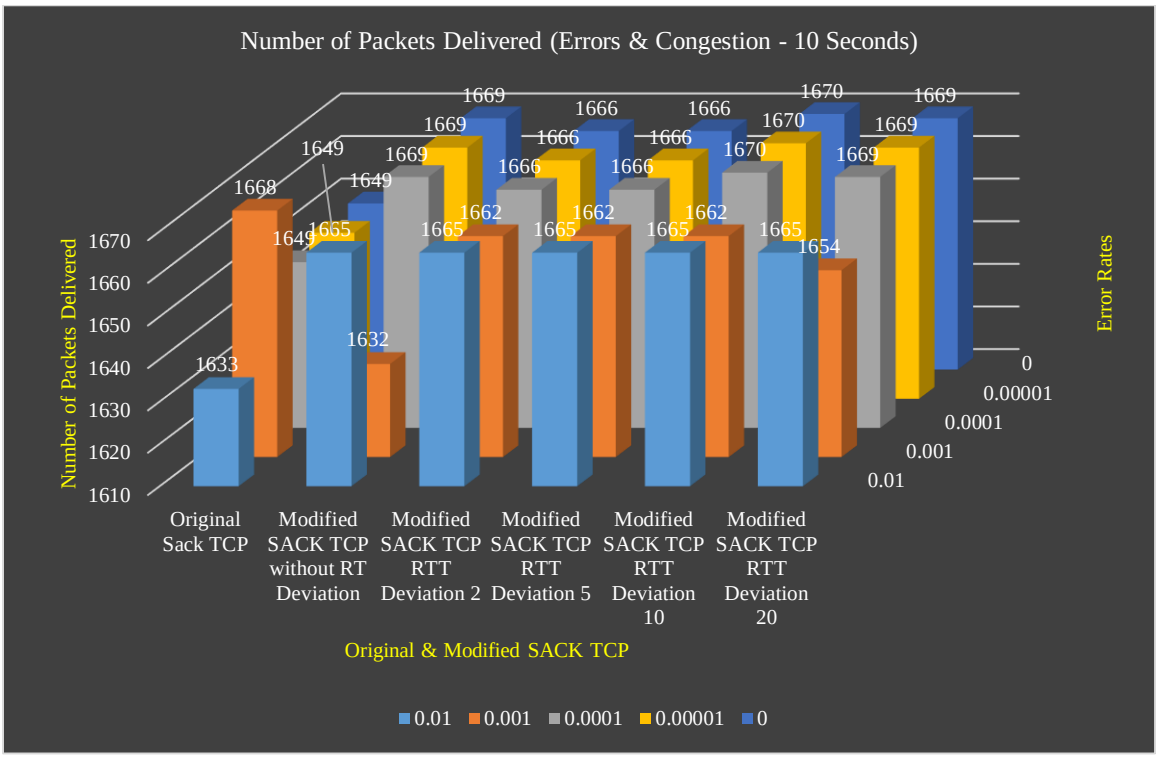

Fig. 8 Number of packets delivered (errors & congestion – 10 seconds)

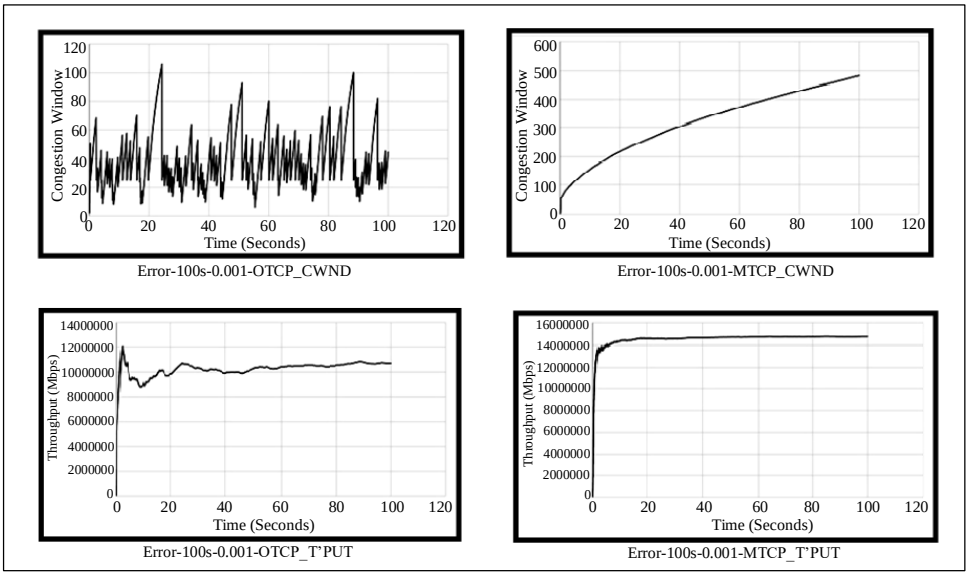

Fig. 9 CWND and throughput – 100S – 0.001 PER

5.4. Sustained Response - Topology - II

The analysis of the observations listed in Table 4 reflects the same thing as discussed earlier in the case of Table 3, irrespective of simulation time. As seen, there is no significant change in performance in any case, which concludes that the congestion takes complete control over transmission flow. All the actions taken will be to support the congestion clearance policy. The plots shown in Figure 10 support the analysis. Figure 11 shows the performance of the protocols in case of a higher error rate of 0.01, simulated for 100 seconds. In the case of OTCP, congestion window alterations are persistent, as in previous cases, and hence, performance gets degraded. The plots show that the maximum congestion window reaches around 50 once and then maintains around 30 due to higher error rates, and performance decreases drastically. In the case of MTCP, congestion window reduction is rare and reaches a maximum value of just above 200. Still, in all cases, MTCP performs better than OTCP.

Table 4. Packets delivered (errors & congestion – 100 seconds)

Packet Error Rates (PERs)	Original SACK TCP	Modified SACK TCP / RTT Deviation 0	Modified SACK TCP / RTT Deviation 2	Modified SACK TCP / RTT Deviation 5	Modified SACK TCP / RTT Deviation 10	Modified SACK TCP / RTT Deviation 20
0.00	16282	16245	16242	16246	16255	16227
0.00001	16282	16245	16242	16246	16255	16227
0.0001	16282	16245	16242	16246	16255	16227
0.001	16258	16245	16238	16249	16232	16241
0.01	16251	16251	16250	16242	16235	16211

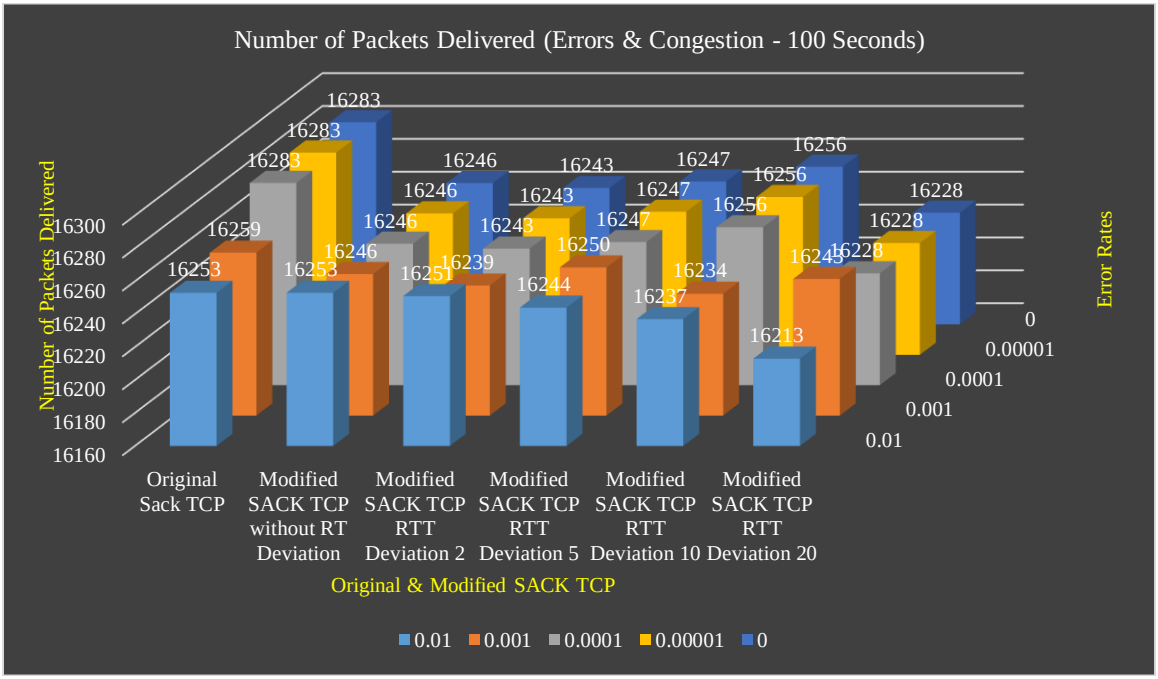

Fig. 10 Number of packets delivered (errors & congestion – 100 seconds)

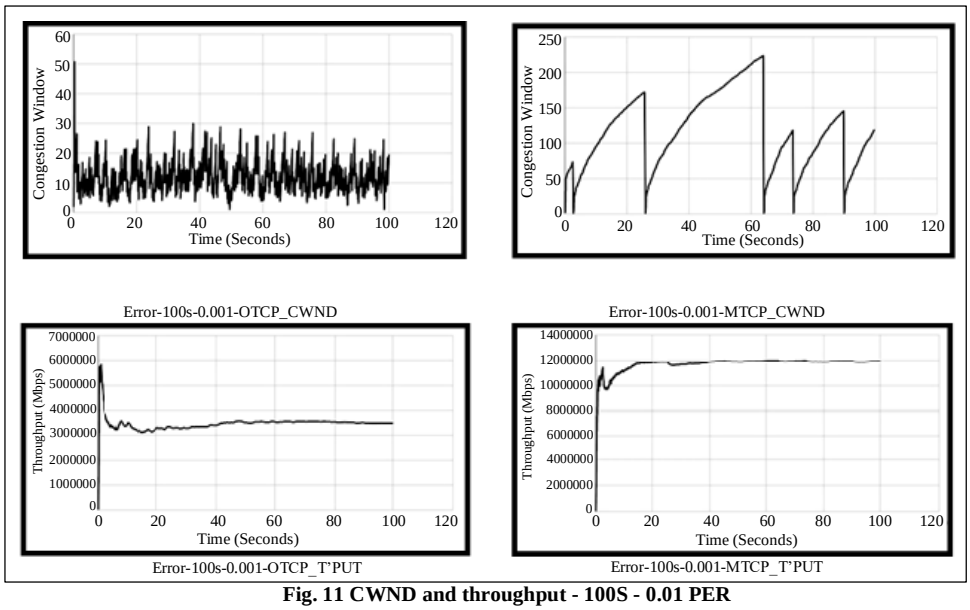

Fig. 11 CWND and throughput - 100S - 0.01 PER

6. Conclusion

The Network Simulator (ns-2) tests and compares original and modified SACK TCP performance under the effect of significant network performance affecting parameters. As the result analysis reflects, there is a high gain in the performance in the case of modified SACK TCP compared to the original one in case of only errors, as the modified algorithm accurately detects the cause of packet loss and acts accordingly. Whereas, in case of severe congestion, the only action that dominantly leads the communication is Fast Recovery which causes transmission flow reduction to fight against the effects of congestion to release it as soon as possible. So, in case of congestion with errors, when congestion dominates, both algorithms perform equally well. The modified algorithm only shows dominance in case of losses due to random errors, whereas it shows equality with the existing SACK TCP to support backward compatibility. The proposed algorithm has the additional advantage of its simplicity and requirement of modifications on the sender side only and has no dependency on the receiver side. This concludes that the proposed modified SACK TCP performs better in case of random losses caused by BERs.

Acknowledgments

We would like to convey our sincere thanks to Dr. Nikhil Kothari and Dr. C. H. Vithalani for their guidance and support.

References

[1] Maulin Patel et al., "*TCP over Wireless Networks: Issues, Challenges and Survey of Solutions*," University of Texas, Dallas, pp. 1-23, 2001. [Google Scholar]

[2] C. Barakat, E. Altman, and W. Dabbous, "On TCP Performance in a Heterogeneous Network: A Survey," *IEEE Communications Magazine*, vol. 38, no. 1, pp. 40-46, 2000. [CrossRef] [Google Scholar] [Publisher Link]

[3] J. Postel, "*Transmission Control Protocol*," Information Sciences Institute, University of Southern California, 1981. [Google Scholar] [Publisher Link]

[4] Kevin Fall, and Sally Floyd, "Simulation-Based Comparisons of Tahoe, Reno and SACK TCP," *ACM SIGCOMM Computer Communication Review*, vol. 26, no. 3, pp. 5-21, 1996. [CrossRef] [Google Scholar] [Publisher Link]

[5] Andrei Gurtov et al., "*The NewReno Modification to TCP's Fast Recovery Algorithm*," RFC 2582, 1999. [Google Scholar] [Publisher Link]

[6] M. Mathis et al., "*TCP Selective Acknowledgment Options*," RFC 2018, 1996. [Google Scholar] [Publisher Link]

[7] N.J. Kothari, and K.S. Dasgupta, "Performance Enhancement of SACK TCP Protocol for a Wireless Network by Delaying Fast Recovery," *2006 IFIP International Conference on Wireless and Optical Communications Networks*, Bangalore, India, pp. 1-5, 2006. [CrossRef] [Google Scholar] [Publisher Link]

[8] Nikhil J. Kothari, Bhavika M. Gambhava, and K.S. Dasgupta, "Adaptive Flow Control: An Extension to Delayed Fast Recovery," *15th International Conference on Advanced Computing and Communications (ADCOM 2007)*, Guwahati, India, pp. 620-625, 2007. [CrossRef] [Google Scholar] [Publisher Link]

[9] Andrei Gurtov, and Sally Floyd, "Modeling Wireless Links for Transport Protocols," *ACM SIGCOMM Computer Communication Review*, vol. 34, no. 2, pp. 85-96, 2004. [CrossRef] [Google Scholar] [Publisher Link]

[10] Teerawat Issariyakul, and Ekram Hossain, *Introduction to Network Simulator NS2*, Springer, 2011. [CrossRef] [Google Scholar] [Publisher Link]

[11] Matthew Mathis et al., "The Macroscopic Behavior of the TCP Congestion Avoidance Algorithm," *ACM SIGCOMM Computer Communication Review*, vol. 27, no. 3, pp. 67-82, 1997. [CrossRef] [Google Scholar] [Publisher Link]

[12] M. Allman, S. Floyd, and C. Partridge, "*Increasing TCP's Initial Window*," RFC 3390, 1998. [Google Scholar] [Publisher Link]

[13] W. Richard Stevens, "*TCP Slow Start, Congestion Avoidance, Fast Retransmit, and Fast Recovery Algorithms*," RFC 2001, 1997. [Google Scholar] [Publisher Link]

[14] M. Allman, V. Paxson, and W. Stevens, "*TCP Congestion Control*," RFC 2581, 1999. [Google Scholar] [Publisher Link]

Appendix

Specific skill sets needed to be acquired to implement simulation setup and to obtain observations for this research are listed in brief in this section: Linux OS installation and Setup, UNIX Commands, NS-2 Installation, Trace File Handling, TCL Scripting, Code Modifications, and plotting various graphs.