VFAST Transactions on Software Engineering
https://vfast.org/journals/index.php/VTSE@ 2024, ISSN(e): 2309-3978, ISSN(p): 2411-6246
Volume 12, Number 2, April-June 2024 *pp:155-163*

Evaluating Congestion Control Methods for enhanced Throughput

Imtiaz Ali Halepoto [1]*, **Fayaz Ahmed Memon** [1], **Sajida Parveen** [1], **Mohammad Asif Khan** [2], **Ali Raza Bhangwar** [1], **Shafaq Rafique** [3]

[1]Department of Software Engineering Department, QUEST Nawabshah, Pakistan; [2]Department of Computer Science, IBA Sukkur University, Pakistan; [3]Department of Computer Systems Engineering, QUEST Nawabshah, Pakistan

Keywords: *Congestion Control methods, TCP, BIC, CUBIC, throughput.*

Journal Info:
Submitted:
May 15, 2024
Accepted:
June 26, 2024
Published:
June 30, 2024

Abstract

With easy access and many services such as social networks, online shopping, video streaming the data traffic over the Internet is increasing. On the other side, the traditional congestion control strategies of TCP due to the huge data are not sufficient. The TCP protocol uses such traditional techniques to minimize the network congestion. Moreover, handling applications with smartphones is challenging in terms of congestion due to the long delay networks such as 4G. Many TCP variants have been proposed for the network congestion particularly for long delay networks such as TCP (Binary increase congestion control) BIC and CUBIC. These proposed techniques for TCP are also not perfect and they require extensive experimentation over long delay networks. In this research, the performance evaluation is accomplished between three variants of TCP for network congestion i.e., TCP, BIC and CUBIC. Particularly the simulations are proposed in order to trace the throughput and fairness of these congestion control techniques. Over a number of simulations it is concluded that under the proposed topology TCP CUBIC improves the throughput and increases the bandwidth fairness approximately by 38% when compared with the basic TCP.

***Correspondence author email address:** halepoto@quest.edu.pk
DOI: 10.21015/vtse.v12i2.1843

1 Introduction

The environment of network traffic has changed significantly as a result of the widespread use of Internet services and applications. The growth of social media, e-commerce, and video streaming services has led to an unparalleled surge in data traffic over the Internet. In order to preserve network performance and guarantee user satisfaction, this spike of huge data necessitates the implementation of effective and efficient congestion control systems. However, the exponential expansion in data makes traditional Transmission Control Protocol (TCP) [1] [2] [3] congestion control solutions unmanageable. For that, the authors are motivated to examine congestion

control methods through simulations, with a focus on high-latency networks like 4G. The effectiveness of three TCP variations is assessed, conventional TCP, Binary Increase Congestion Control (BIC) [5], and TCP CUBIC [4]. Conventional methods are employed by the TCP protocol, which is the backbone of Internet communications, to control network congestion. These methods use algorithms to regulate the speed at which data packets are transmitted in order to prevent network overload. The congestion window size is modified by traditional TCP congestion control algorithms, such TCP Reno and TCP NewReno [6], in response to packet loss signals, which signify network congestion. But these antiquated techniques frequently fall short of managing the enormous and fluctuating volume of data flow in contemporary networks.

In order to overcome the constraints of conventional TCP in settings with high latency and high bandwidth, multiple TCP variations have been suggested. TCP Binary Increase Congestion Control (TCP BIC) and TCP CUBIC are two of these that have drawn interest due to their potential to improve network performance. In order to more effectively modify the congestion window size and stabilize the network while increasing throughput, TCP BIC includes a binary search method. BIC is designed to improve performance in high-speed and long-distance networks. It aims to achieve better utilization of available bandwidth and fairness among competing flows. In contrast, TCP CUBIC uses a cubic function to expand the congestion window size, which improves scalability and performance in long-distance and high-speed networks [7]. It simplifies the congestion control mechanism while improving scalability and fairness.

The objective of this paper is to evaluate the performance of three TCP variants—standard TCP, BIC, and CUBIC—specifically in the context of network congestion in high-latency environments. All of the three algorithm use the traditional slow start and congestion avoidance methods. Through a series of simulations, the study aims to trace and compare the throughput and fairness of these congestion control techniques. The ultimate goal is to determine which TCP variant provides the most significant improvements in network performance under the proposed topology. Network administrators and engineers that want to maximize data transfer over the Internet must comprehend the performance variations between different TCP variants. Choosing the right congestion control strategy can significantly increase network efficiency and user experience as data traffic grows. The study's conclusions, which show that CUBIC is superior to regular TCP in terms of improving throughput and bandwidth fairness by about 38% over long propagation delay networks, offer insightful information for upcoming network optimization initiatives.

The remainder of this paper is structured as follows: Section 2 provides a detailed review of related work in the field of congestion control. Section 3 outlines the methodology used for the simulations, including the network topology and performance metrics. Section 4 presents the results of the simulations, with a discussion of the implications and significance of the findings. Finally, Section 5 concludes the paper.

2 Related Work

The research in [8] presented a novel approach to throughput control called SP-MAC, by leveraging the SP-MAC's ability to modify obtained throughput through parameter adjustments, provides fairness among groups of terminals using different TCP versions. The authors also validated the efficacy of the suggested approach by the simulation assessment in NS2. Authors in [13] proposed variable changes in congestion window based on the current and previous status and history information, which is different from the additive and multiplicative increase and decrease method. The authors conducted experiments using NS3. The experiments demonstrate that our algorithm surpasses legacy congestion control algorithms such as CUBIC, RENO, and traditional TCP in terms of throughput and fairness. Moreover the algorithm also reaches and maintains such parameters for a large scale network. Authors in paper [9] evaluated several key metrics, including throughput, queue size, packet delay rate, and end-to-end delay, across different TCP variants to determine the optimal performer in

various congestion scenarios. The study analyzes four TCP variants: TCP Reno, TCP New Reno, TCP Tahoe, and TCP Vegas. In terms of throughput and packet delivery, TCP Vegas demonstrates superior performance compared to other TCP variants in the proposed scenarios. The authors suggested the use of vegas variant in the data traffic over the internet. The study in [10] aims to assess and compare the performance of recent TCP implementations deployed across popular and new operating systems. They selected specific scenarios to evaluate the goodput and fairness of these TCP variants. The findings indicate that CUBIC, when used on wired links, outperforms Compound and New Reno in the presence of the background data traffic. However, their performance differs on wireless links, where they exhibit lower goodput with minimal variation. Furthermore, all three variants demonstrate a very good level of fairness in terms of fairness. All the variants are recommended as emerging in terms of usage in order to increase communication reliability. Many similar studies also conclude that the role of congestion control variants is a key for the enhanced and fast communication such as over packed switched networks, data centres and cloud computing environments, and large scale live-streaming applications [11] [12] [18]. The study on TCP BIC with parameter of packet loss, network utilization and queuing delay also suggest the effective transmission of data over mininets [15]. BIC and CUBIC are also analyzed over NS3 in [16] simulator and authors found performance improvement over TCP. Authors in [17] carried out experiments on multipath TCP with new congestion control algorithms and found that CUBIC improves the throughput as well as fairness.

3 Basic Congestion Control Methods

3.1 TCP

TCP begins with the slow start phase for handling the congestion. The congestion window (CWND) is initially set to one segment during this phase. The CWND doubles with each round-trip time (RTT), resulting in exponential growth (1, 2, 4, 8, etc.). This speedy rise in CWND aids in making quick use of the network's available capacity. But only up until the slow start threshold (SSTHRESH) is crossed will this exponential growth stop. TCP enters the congestion avoidance phase when the SSTHRESH is reached. The CWND grows more conservatively during the congestion avoidance phase. The CWND increases linearly, by one segment every RTT, as opposed to doubling. By doing this, congestion and overcrowding of data on the network are prevented. During this phase, TCP intervenes immediately if it detects packet loss, which is often indicated by the receipt of three duplicate acknowledgments. Without waiting for a timeout, it sends the missing data again quickly.

TCP moves into the fast retransmission phase after the quick retransmit. During this phase, the CWND is cut in half as opposed to being reduced to one segment. This permits the transmission to proceed, albeit at a little slower pace. This helps to keep the flow of data consistent and is more efficient than stopping at a single segment. The CWND grows linearly throughout the congestion avoidance phase until there is another packet loss. The principle of additive increase/multiplicative decrease (AIMD), governs this entire process. Reacting to congestion indicators and increasing network usage are two requirements that AIMD helps balance. This strategy reduces the possibility of creating congestion collapse while guaranteeing that TCP can use network resources efficiently. These phases enable TCP to adjust to changing network conditions and continue to provide dependable data transfer.

3.2 BIC

The BIC process starts with the slow start phase like typical TCP connection. When TCP reaches the SSTHRESH, it moves on to the BIC congestion management phase. Compared to ordinary congestion avoidance of TCP, the formation of the CWND is more intricate and sophisticated during the BIC congestion management phase. To control the congestion window, BIC combines additive increase techniques with binary search. To swiftly explore the available bandwidth, the CWND expands logarithmically when it falls below a predetermined threshold. The increase becomes less rapid and more linear as it gets closer to this threshold. Like TCP, BIC intervenes right away if it

detects packet loss during this phase, which is usually indicated by the receipt of three duplicate acknowledgments. Without waiting for a timeout, it sends the missing piece again quickly. After the quick retransmit, a modified recovery phase is entered by BIC. The CWND is decreased during this phase, although not by half. Rather, BIC employs a binary search strategy to choose a new congestion window size that strikes a compromise between the requirement to prevent congestion and the goal to make the most use of available bandwidth. As a result, the transmission can carry on at its best rate, constantly changing according to network conditions. Until another packet loss happens, the CWND keeps adjusting throughout this phase in accordance with the binary search and additive increase procedures. The binary increase and multiplicative decrease principle governs this entire process. More effectively than normal AIMD of TCP, BIC helps strike a compromise between the requirement to react to congestion indicators and the desire to boost network utilization. By taking this method, BIC may make better use of the network resources and reduces the possibility of congestion collapse.

3.3 CUBIC

Thc CUBIC also utilizes the same slow start phase as like to TCP and BIC. Similar to TCP and BIC it enters the CUBIC congestion control phase one the CWND is equal to the SSTHRESH value.

The expansion of the CWND in the CUBIC congestion control phase is controlled by a cubic function, which is very different from the linear increase in conventional congestion avoidance. When the network is underutilized, the cubic function permits a more aggressive growth, and when it gets closer to the previous congestion window size before the loss event, it permits a more cautious increase. This indicates that the CWND grows quickly at first, slows down as it gets closer to the prior congestion site, and then picks up speed once more after passing it. CUBIC intervenes right away if it detects packet loss during this phase as it retransmits the missing data immediately as like TCP and BIC. After the quick retransmit, a modified recovery phase is entered by CUBIC. The cubic function is used to direct the reduction of the CWND in order to establish the new congestion window size during this phase. This preserves network conditions while enabling the transmission to proceed at a rate that smoothly recovers from loss. The cubic function makes sure that following the reduction, the CWND will rise smoothly, aiding in the speedy recovery of available bandwidth. The CWND keeps adjusting in accordance with the cubic function throughout this phase until another packet loss is detected.

The CUBIC concept, which substitutes a cubic growth function for the AIMD method, leads the entire process. Compared to regular AIMD, CUBIC is more efficient and effective in balancing the requirement to react to the network congestion with the need to expand network utilization. With this method, there is less chance of network congestion and failure and CUBIC can use network resources more efficiently.

4 Proposed Methodology

The research method is divided into four steps as presented in Figure 1.

4.1 Network Scenario and Simulator

In this scenario there are 7 nodes are used. For the data transmission we selected node 6 and node 0 as the sending nods. The node under monitoring is node 1, which is the receiving node for both the senders. We established two connections simultaneously. One connection is from node 6 to node 1 and other is from node 0 to 1. The proposed scenario is shown in Figure 2. The simulation tool used for the creation of the network scenario is NS2 [14]. It provides a comprehensive platform for modeling the dynamics of wired and wireless networks [19]. It enables comprehensive and varied network protocol simulations by supporting a large number of network protocols, such as TCP, UDP, HTTP, FTP, and others. NAM (Network Animator), one of the network visualization tools included with NS2, enables users to see and observe network simulations, which facilitates the analysis of network behavior and outcomes. Its extensive library of network components, which includes various link types, traffic sources, error models, and more, makes it easier to create intricate network situations. A complete tool

for network research, NS2 allows simulations to span several tiers of the network stack, from the physical layer to the application layer. Because the simulator is highly extendable and modular, users may quickly add new protocols, change current ones, and construct bespoke network models. In this simulator the code for the network scenario is written and links and nodes are connected through configuration as shown in Figure 2. The scenario is designed in such a way so that it could reflect real life traffic due the configuration of the background traffic.

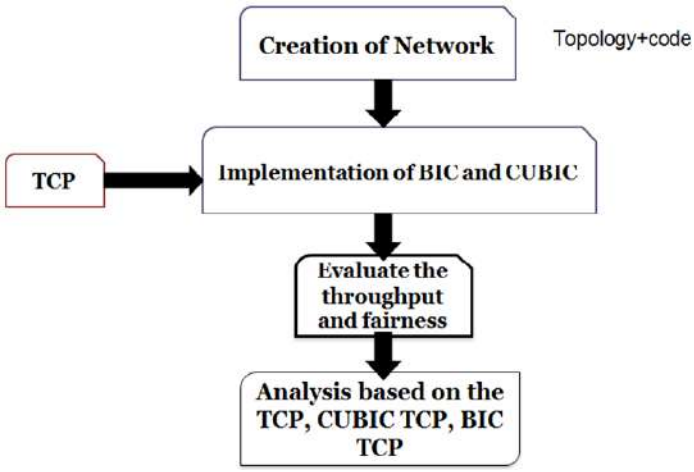

Figure 1. Proposed Methodology

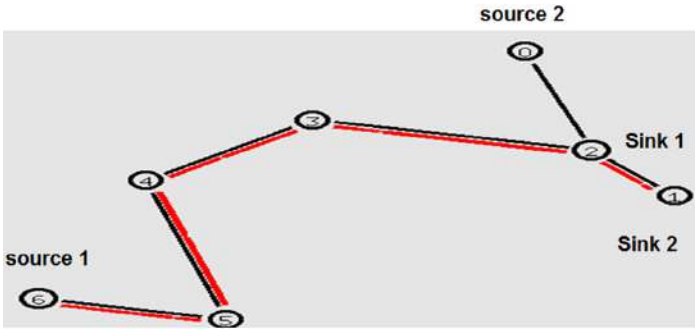

Figure 2. Network Scenario, Data travels through two links. From node 0 to 1 and node 6 to 1.

4.2 Implementation of variants

On the above network scenario we have implemented CUBIC TCP and BIC TCP and TCP as default protocol in the NS2. The sending and receiving nodes are configured accordingly. The code for the three protocols is open source and it is implemented in the NS2. After the implementation of the network scenario all the variant are executed through TCL scripts. One simulation of TCL is repeated for TCP, BIC and CUBIC. For the one simulation all the parameters of the simulations are fixed to the same value for TCP, BIC and CUBIC.

4.3 Evaluation of throughput and fairness

TCP fairness refers to the equitable distribution of network resources, particularly bandwidth, among multiple TCP connections sharing the same network link. Fairness ensures that no single connection monopolizes the available bandwidth, allowing for a balanced and efficient network operation. In the proposed network scenario fairness refers to balanced is the throughput among the connections from node 6 to 1 and from node 0 to 1. Throughput is the amount of data successfully received by the receiver in 1 second and it is measure in mbps.

4.4 Analysis

The analysis is carried out on the output generated by the simulator. Normally NS2 creates traces of the events through out the simulation. Such traces contains a lot of data that needs analysis. For the the data extraction and analysis purpose AWK scripting is used. After the data extraction the summarised data is presented in plots using GNU plot. AWK is a powerful and versatile scripting language primarily used for pattern scanning and processing. AWK is typically used for manipulating data and generating reports. It is particularly well-suited for processing text files and extracting information from structured data like log files, CSV files, and other delimited text formats normally generated through simulations. GNUPLOT is a popular and flexible open-source plotting tool that is mostly used for scientific graph creation and data display. Using scripting, users can produce intricate graphs that are repeatable and automated. Plot styles, labels, data sources, and other features can all be defined with GNUPLOT scripts. In addition to performing data fitting using a variety of algorithms, including linear and nonlinear least squares fitting, it allows the direct display of mathematical functions. It is

appropriate for tracking and visualizing real-time data streams since it allows for real-time data graphing.

5 Results and Discussions

5.1 Fairness of TCP, BIC and CUBIC

In this scenario, there are 7 nodes, and we established two connections that send and receive data simultaneously. Node 0 and node 6 are the source nodes, while node 1 is configured as the sink node for both source nodes. One connection is from node 6 to node 1, with a propagation delay set to 50ms, bandwidth set to 5Mbps, and cwnd set to 512KB. Propagation delay is the time it takes for a signal to travel from the sender to the receiver through a communication medium. The second connection is from node 0 to node 1, with a propagation delay set to 0ms, bandwidth set to 5Mbps, and cwnd set to 512KB. The simulation was repeated 10 times to measure the throughput, focusing on TCP fairness among the connections. Both sending devices were configured to send traffic for a simulation time of 100 seconds. The simulation was repeated for TCP, BIC, and CUBIC, and the results were observed through the generated trace file from NS2. In terms of throughput the connection with 0ms produces on average 2mpbs for both the TCP (Figure 3) and BIC (Figure 4), whereas the connection configured to 50ms produces less throughput nearly 0.1mbps for both the TCP and BIC. The tiny difference is seen in the throughput of BIC, which reflects a slight improvement in terms of fairness among the connections. On the same configuration and topology the CUBIC reaches to the throughput of 2.5mbps on average for 0ms connection and on average 1.7mpbs for connection with 50ms delay (Figure 5).

In the given scenario, TCP CUBIC achieves higher throughput for both connections (from node 0 to node 1 and from node 6 to node 1) compared to TCP and BIC due to its advanced congestion control mechanisms. CUBIC Uses a cubic function for congestion window growth. This cubic function allows for more aggressive growth when the network is underutilized and more conservative growth when the network is near capacity. This approach ensures that CUBIC can quickly utilize available bandwidth while avoiding congestion collapse, leading to higher overall throughput. Moreover, CUBIC is designed to perform well in networks with high bandwidth-delay products. In this scenario, the difference in propagation delays (0ms vs. 50ms) is effectively managed by CUBIC's window growth algorithm, which adapts more smoothly to varying conditions and maintains higher throughput.

5.2 Instantaneous Throughput

This experiment was done to measure the throughput of TCP, BIC TCP and CUBIC TCP on only the connection from node 6 to node 1. The node 0 in this simulation is set as a background traffic generator and it is not used in the calculation of the throughput.

In the scenario where all connections have a delay of 50ms, TCP CUBIC produces more throughput than TCP and BIC due to its advanced congestion control mechanisms (Figure 6), while the throughput of TCP and BIC is very similar because they handle congestion in a comparable manner. CUBIC Utilizes a cubic function for the growth of the congestion window, which allows for more aggressive utilization of available bandwidth when network conditions are good. This cubic growth is particularly effective in high-bandwidth and high-delay environments (like the 50ms delay in your scenario), enabling faster window expansion and better use of available network resources. TCP and BIC Both use less sophisticated window growth algorithms. TCP increases the congestion window linearly during congestion avoidance, which is much slower compared to CUBIC's cubic function. BIC, while more aggressive than TCP, still does not match the efficiency of the cubic function used by CUBIC.

During the initial phase of the simulation (up to 45 seconds) CUBIC begins with a slow start phase and quickly ramps up its congestion window (Figure 7). During this phase, the throughput differences between CUBIC, BIC, and TCP may be minimal because all algorithms are ramping up their windows and stabilizing. TCP and TCP BIC also ramp up their congestion windows but more slowly (in the case of TCP) or less efficiently (in the case of BIC), leading to slightly lower throughput compared to CUBIC. From 46 seconds to 100 seconds, the throughput of CUBIC becomes significantly larger than that of TCP and BIC

(Figure 7). The CUBIC algorithm's ability to quickly increase the congestion window when there is available bandwidth and to back off smoothly when congestion is detected leads to higher sustained throughput. TCP in the steady-state phase, the linear congestion window growth is too slow to fully utilize the available bandwidth, especially in a high-delay environment. This results in lower throughput. BIC although BIC can be more aggressive than New Reno, its binary search approach can lead to oscillations in the congestion window size. This oscillation can cause periods of under utilization of the available bandwidth, resulting in lower overall throughput compared to CUBIC. In the longer delay networks of the proposed topology the throughput of CUBIC is approximately 38% greater than the other two.

In the scenario with a 1ms delay configuration, the throughput of all three protocols, TCP CUBIC, BIC, and TCP is very similar with only a slight improvement in CUBIC (Figure 8). In a low-latency network with a 1ms delay, the round-trip time (RTT) is very short. This minimizes the impact of congestion control algorithms since the feedback loop (i.e., the time it takes for acknowledgments to travel back to the sender) is fast. Consequently, the window growth and adjustments happen more quickly, and all protocols can respond to changes in network conditions almost immediately. With such a low delay, the available bandwidth is more easily and fully utilized by all three protocols. The differences in how they increase and decrease their congestion windows become less significant because the network conditions are consistently optimal.

6 Conclusions

In this paper, we evaluated the performance of congestion control techniques of TCP, BIC, and CUBIC across different network scenarios characterized by varying propagation delays. Our findings highlight the strengths and limitations of each protocol under specific conditions, providing insights into their suitability for different network environments. On monitoring the individual network connections working in parallel it was observed that the CUBIC is more fair in terms of connection utilization. Both the TCP and BIC mainly

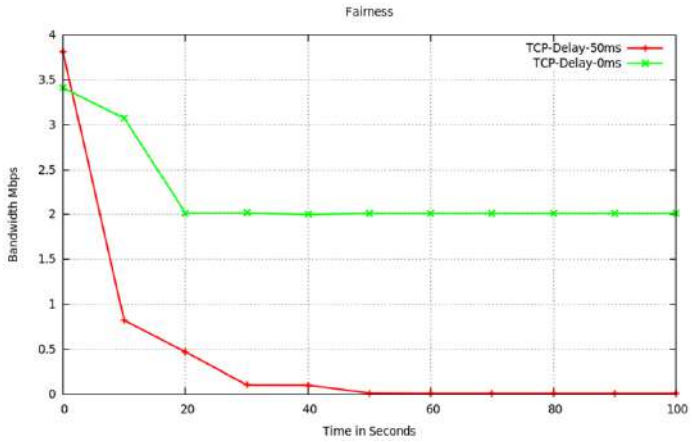

Figure 3. Fairness of TCP

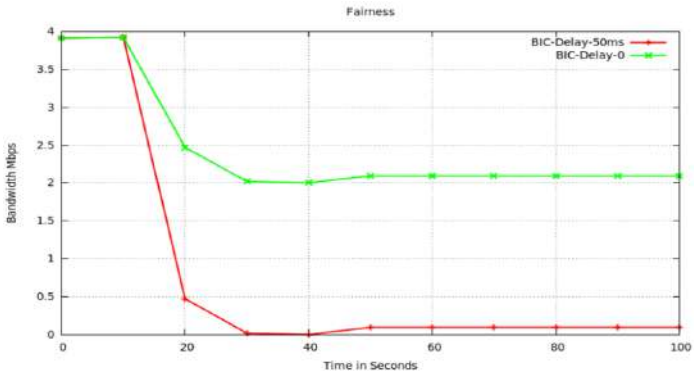

Figure 4. Fairness of BIC

utilize and occupy the Congestion window for a single connection while producing less throughput over second connection. In terms of of single connection throughput measurement, when propagation delay is too short (1ms) all the protocols reach approximately the similar throughput. When the delay is longer (300ms) the CUBIC outperform the TCP and BIC by 38% in terms of throughput by efficiently managing the network congestion,

The following are the directions for the future:

- Fairness of CUBIC can be further improved by utilizing the benefits of proper queuing management.
- It would be interesting to see the comparative analysis of congestion control variants with routing protocols [20]
- Multipath protocols such as Multipath TCP and

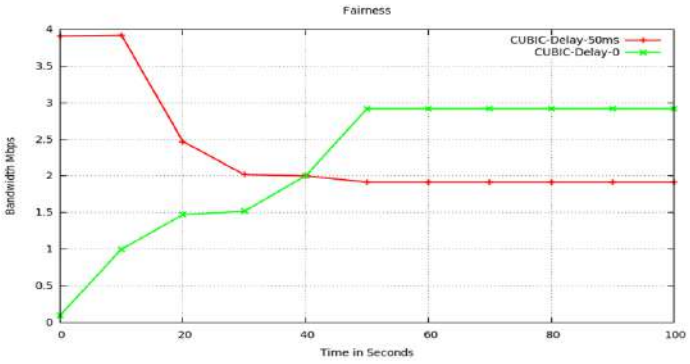

Figure 5. Fairness of TCP CUBIC

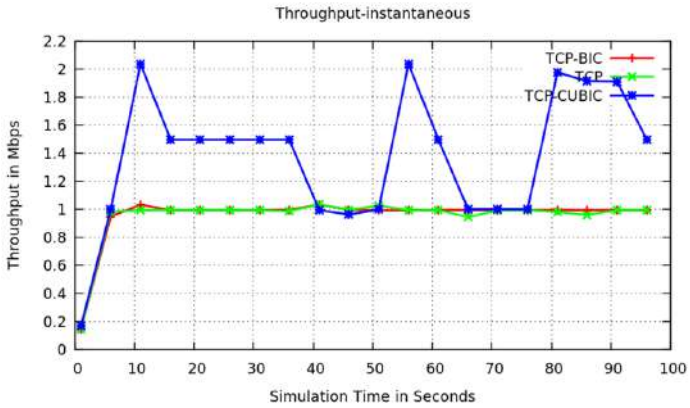

Figure 6. Throughput

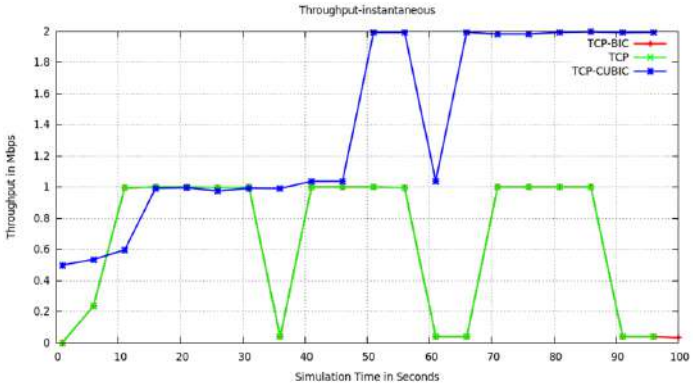

Figure 7. Throughput over longer delay networks

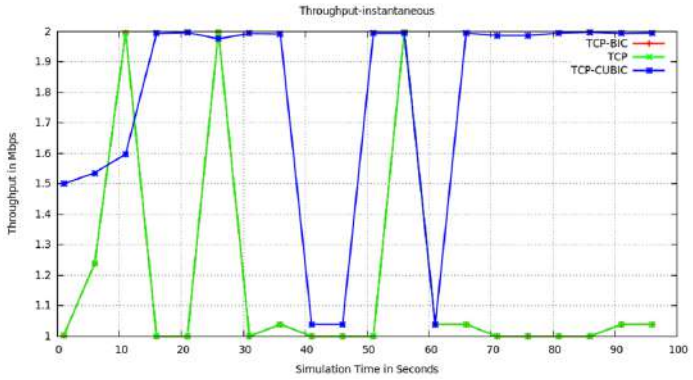

Figure 8. Throughput over short delay networks

SCTP [21] which divides the data among different paths could be examined through simulations with CUBIC.

Author Contributions

Imtiaz: Conceptualization and supervision. **Fayaz**: Visualization, Investigation. **Sajida**: software. **Asif**:Reviewing and Editing. **Ali**: validation. **Shafaq**: Data collection, methodology and results.

Compliance with Ethical Standards

It is declare that all authors don't have any conflict of interest. It is also declare that this article does not contain any studies with human participants or animals performed by any of the authors. Furthermore, informed consent was obtained from all individual participants included in the study.

References

[1] H. Shi and J. Wang, "Intelligent TCP congestion control policy optimization," Applied Sciences, vol. 13, no. 11, p. 6644, 2023.

[2] A. Qureshi, A. Qamar, I. A. Halepoto, S. Ashfaque, and M. Lohana, "Analysis of Congestion Control Techniques for Time–Critical Applications," QUEST Research Journal, vol. 17, no. 2, pp. 43-49, 2019.

[3] I. A. Halepoto, N. H. Phulpoto, A. Manzoor, S. A. Memon, and Umair A. Qadi, "Effect of TCP Buffer Size on the Internet Applications." International Journal of Advanced Computer Science and Applications 9, no. 6p. 417–422, 2018.

[4] S. N. Sunder, V. R. Akathya, S. Seshadri, S. M. Rajagopal, and S. Bhaskaran, "Implementation and analysis of TCP CUBIC congestion control system in Mininet," in 2024 IEEE 9th International Conference for Convergence in Technology (I2CT), pp. 1-6, IEEE, 2024.

[5] A. Assar and K. Hofmann, "A hardware implementation of the TCP protocol applying TCP-BIC and TCP-CUBIC standards," in 2016 28th International Conference on Microelectronics (ICM), pp. 37-40, IEEE, 2016.

[6] S. Abdullah, "Enhancing the TCP Newreno fast recovery algorithm on 5G networks," Journal of Computing and Communication, vol. 3, no. 1, pp. 33-43, 2024.

[7] L. Xu, S. Ha, I. Rhee, and V. Goel, "RFC 9438: CUBIC for Fast and Long-Distance Networks," 2023.

[8] R. Tsurumi, M. Morita, H. Obata, C. Takano, and K. Ishida, "Throughput control method between different TCP variants based on SP-MAC Over WLAN," in 2018 IEEE International Conference on Consumer Electronics-Taiwan (ICCE-TW), pp. 1-2, 2018.

[9] P. Chaudhary and S. Kumar, "A review of comparative analysis of TCP variants for congestion control in network," International Journal of Computer Applications, vol. 160, no. 8, pp. 1-6, 2017.

[10] I. Abdeljaouad, H. Rachidi, S. Fernandes, and A. Karmouch, "Performance analysis of modern TCP variants: A comparison of Cubic, Compound and New Reno," in 2010 25th Biennial Symposium on Communications, pp. 80-83, IEEE, 2010.

[11] R. P. Tahiliani, M. P. Tahiliani, and K. C. Sekaran, "TCP variants for data center networks: A comparative study," in 2012 International Symposium on Cloud and Services Computing, pp. 57-62, IEEE, 2012.

[12] C. Socrates, P. M. Devamalar, and R. K. Sridharan, "Congestion control for packet switched networks: A survey," International Journal of Scientific and Research Publications, vol. 4, no. 12, pp. 1-6, 2014.

[13] M. R. Kanagarathinam, S. Singh, I. Sandeep, A. Roy, and N. Saxena, "D-TCP: Dynamic TCP congestion control algorithm for next generation mobile networks," in 2018 15th IEEE Annual Consumer Communications Networking Conference (CCNC), pp. 1-6, IEEE, 2018.

[14] I.A Halepoto, I. H. Sadhayo, M. S. Memon, A. Manzoor, and S. Bhatti. "Analysis of Retransmission Policies for Parallel Data Transmission." Engineering, Technology & Applied Science Research 8, no. 3, p.3079-3083, 2018.

[15] S. N. Sunder, V. R. Akathya, S. Seshadri, S. M. Rajagopal, and S. Bhaskaran, "Implementation and Analysis of TCP CUBIC Congestion Control System in Mininet," in 2024 IEEE 9th International Conference for Convergence in Technology (I2CT), pp. 1-6. IEEE, 2024.

[16] S. Patel, Y. Shukla, N. Kumar, T. Sharma and K. Singh, "A Comparative Performance Analysis of TCP Congestion Control Algorithms: Newreno, Westwood, Veno, BIC, and Cubic," in 2020 6th International Conference on Signal Processing and Communication (ICSC), Noida, India, 2020, pp. 23-28.

[17] J. Y. Lee, B. C. Kim, Y. Kwon, and K. Han, "Coupled CUBIC Congestion Control for MPTCP in Broadband Networks," Computer Systems Science Engineering 45, no. 1 (2023).

[18] D. Yuan, W. Zhang, Y. Qiu, H. Huang, M. Yang, P. Chen, K. Xiao, H. Yan, Y. He, and Y. Zhang, "Context-Aware Cross-Layer Congestion Control for Large-Scale Live Streaming," IEEE/ACM Transactions on Networking (2024).

[19] A. Avvaru, A. Tandon, J. J. J. Kumari, and A. S. Raja, "Performance Analysis of Routing Protocols in VANETs using OSM, SUMO, and NS2," in 2024 IEEE International Conference on Interdisciplinary Approaches in Technology and Management for Social Innovation (IATMSI), vol. 2, pp. 1-6. IEEE, 2024.

[20] N. H. Bhangwar, I. A. Halepoto, I. H. Sadhayo, S. Kumar, and A. A. Laghari, "On Routing Protocols for High Performance," Studies in Informatics and Control, vol. 26, no. 4, pp. 441-448, 2017.

[21] I. A. Halepoto, F. Halepoto, F. A. Memon, A. R. Bhangwar, B. A. Zardari, and S. Iqbal, "Enhancing SCTP Performance through the Selection of Appropriate Retransmission Policies," VFAST Transactions on Software Engineering, vol. 11, no. 2, pp. 11-16, 2023.