Comparative Analysis of Compound TCP with Various End-to-End Congestion Control Mechanisms

Subhra Priyadarshini Biswal, Sheshank Gupta, Arun Kumar, and Sanjeev Patel
Department of Computer Science and Engineering
National Institute of Technology Rourkela, Odisha-769008, India

***Abstract*—Congestion control is a fundamental mechanism offered by the Transmission Control Protocol (TCP) to make sure stable data flow and efficient bandwidth allocations in the network. It mainly focuses on the adjustment of the incoming sending rate with respect to the receiving rate. The reliable TCP helps to manage the transmission rate by avoiding the state of network congestion. In recent years, different variants of TCP congestion control protocols have been proposed to enhance TCP reliability and performance under different network scenarios. Moreover, it is associated with several types of network states such as latency, throughput, and packet loss. In this paper, we have performed a comparative study on the performance analysis of various TCP congestion control protocols on NS2 simulator. The results show that CUBIC gives the highest average throughput, Vegas gives the lowest packet loss ratio, and Veno gives the lowest average end-to-end delay among other TCP congestion control techniques that are considered in this paper.**

***Index Terms*—TCP, Latency, Bandwidth, RTT, Throughput, Congestion Control**

I. Introduction

The Internet has facilitated an environment for fast and timely communication over both clients and servers. The transport layer supports host-to-host packet delivery between various application programs that are running in the end nodes. Transmission Control Protocol (TCP) is the most widely used transport layer protocol that facilitates connection-oriented, byte stream, end to end data stream reliable services to the applications that are communicating through an Internet Protocol (IP) network. It is a type of communications standard that helps to transmit packets across the Internet and guarantees the successful delivery of messages. Moreover, it is used by many application layer protocols such as FTP (File Transfer Protocol), Telnet, HTTP(Hypertext Transfer Protocol), HTTPs (HTTP Secure), SMTP (Simple mail transfer protocol) etc. Initially, the design of TCP was only for the wired base network [1]. But with the increasing demand for huge amount of data transmission over both wired and wireless communication network requires up-gradation or enhancement of the existing protocol suite to improve the performance [2].

Now a days, an effective and fairly resource allocation among various competing users is a key requirement of many applications. The sharing of resources among network nodes requires huge bandwidth links as well as buffers present on the switches and routers at which the packets are queued and waiting for the transmission. When more number of packets are waiting for the transmission by the same link, then the queue gets overflow. As a result, the router starts to drop the packet that is subject to the network congestion [3]. In addition, when the sender makes overflows situation over the network with enormous number of packets, the problem of network bottleneck or congestion situation arises. As a consequence, when the data traffic is not handled properly may leads to the problem of degradation of quality of service (QoS).

To avoid such situation, TCP provides congestion control (CC) mechanisms which were first introduced in the end of 1980s by Van Jacobson. The main goal of TCP congestion control is to know how many number of packets a source can safely transmit by determining the capacity of the link over the network. After transmitting these many number of packets, the source wait for the arrival of ACKs (acknowledgements) based on which it can adjust its window size. So that it is also known as self-clocking protocol. After this, it can safely insert new packets in to the network without of any congestion [4]. It adjusts the rate of transmission by seeing the network bandwidth to deal with the level of congestion. The development of a proper congestion control algorithm is more difficult as it needs a keen awareness of all available network resources. Moreover it also focuses on how these resources can be used in fair and efficient way. It is required to find out the most relevant and effective solution that is desirable for most of the users and applications. There are so many existing solutions have been proposed that are not exactly perfect for all types of situations depending on the different network scenarios.

When TCP sends huge amount of data without worrying about congestion state of the network and bottleneck queue, as a result it leads to excessive packet drop. Moreover, it degrades the network performance and increases the delay. To avoid such situation, it is required to develop and implement the various end to end CC protocol to monitor network's current congestion state. TCP congestion control relies on four different mechanisms, i.e. a) congestion avoidance (additive increase), b) slow start, c) fast retransmit and d) fast recovery. In congestion avoidance phase, for every received acknowledgement, the size of congestion window ($cwnd$) is increased linearly i.e. roughly by one segment. Similarly, in slow start approach, it increases the cwnd in an exponential

order until it will reach to a certain threshold. When the packets start to get drop, it will leads to multiplicative decrease of the $cwnd$ accordingly. In fast retransmit, it will retransmit the missing or dropped packets immediately after receiving three duplicate acknowledgements (ACK). In fast recovery phase, it helps to immediate recover of the lost packets by removing the slow start phase. The main reason behind this is, after receiving duplicate acknowledgements; the sending side drastically reduced its $cwnd$ into 1 and starts the slow start phase. So instead of this, in fast recovery phase, the sender can increase the $cwnd$ by 1 after receiving a duplicate ACK and keep doing this until it has received a non-duplicate ACK. The $cwnd$ value is increased by one MSS for every duplicate ACK that has been obtained for the lost segments.

Section II deals with related work. We have presented various popular TCP congestion control mechanism in section III. The results analysis is discussed in section IV. The concluding remarks are depicted in the last section.

II. RELATED WORKS

Many research work has been performed on design of different congestion control schemes with proper analysis of their behaviour and performance. Hasegawa and Murata [5] have summarised different types of TCP congestion control mechanisms, especially TCP Reno and Vegas. However, the author has analysed different types of fairness issues due to heavy traffic and their possible solutions to overcome these types of problem. To overcome these issues, several packet buffering and scheduling algorithms like Per–flow Queueing and RED Variants are mentioned properly. Alrshah et al. [6] have analysed performance of different congestion control algorithm based on $cwnd$, fairness, and loss rate under wired network. The outcome shows that, TCP CUBIC and TCP (YeAH) Yet Another High-speed get better result over other congestion control algorithm. Using NS2 simulation, the author has analysed performance of different TCP variant by taking high BDP (Bandwidth Delay Product) networks. Ros and Welzl [7] have performed survey of different TCP variant congestion control mechanisms used by Linux Kernel 2.6.x. The author has analysed behaviour of these protocols based on their variations in window size behaviour.

Lukaseder et al. [8] have studied of various congestion control algorithms in wired networks and calculated the variations in throughput, loss rate, fairness and stability. The author has explained various congestion control algorithm like Reno, Scalable TCP, HighSpeed TCP, BIC (Binary Increase Congestion Control) and CUBIC. The author has also evaluated various performance like fairness, link utilization, convergence time, spread of the converged flows, efficiency and responsiveness using a standard dumbbell topology using HP NC523SFP Dual 10 Gbps NICs using Ubuntu 14.04.1 or Linux 3.16. As a result, TCP CUBIC shows same behaviour as Reno in terms of packet loss and TCP BIC behaves best in lossy network. Afanasyev et al. [9] have done study of classification of different host-to-host congestion control principles, packets reordering, performance in wireless network with traffic prioritizing features. The author has explained the concept of congestion collapse and congestion control algorithms like Tahoe, Reno, Dual, New Reno, SACK (Selective Acknowledgments), FACK, Vegas, Vegas, Vegas+ and Veno. Moreover, the concept of TCP variants that help to solve the problem of packet reordering such as TD-FR (Time-delayed fast-recovery), Eifel, TCP DOOR (Detection of Out of Order and Response), DSACK (Duplicate Selective Acknowledgement) and RR-TCP (Reordering-Robust TCP) are explained properly. Sun et al. [10] have visualized TCP congestion control problems using classification techniques. Different network parameters are considered as input and the current status of the network that is analyzed by observing changes in the existing congestion control algorithms. That is considered as output. It makes the congestion control algorithm to learn from the network-produced data.

Pokhrel and Williamson [11] study the performance of Compound TCP in IoT (Internet of Things) based home infrastructure network. Compound TCP is fusion of both loss and delay based approach that acts as a central component in infrastructure based WiFi enabled devices in home network. It provides a good bandwidth scalability and increases the data rate when the network link is under-utilized. It is mainly designed to achieve high efficiency and requirement of TCP friendliness. The author has identified various- performance and behaviour of compound TCP over infrastructure based WiFi network with respect to buffer overflow and wireless transmission impairments. Claypool et al. [12] have analyzed different TCP congestion control algorithms over a satellite-based network. The author has considered TCP CUBIC, Hybla, PCC (Performance-oriented Congestion Control), and BBR (Bottleneck Bandwidth and Round-trip time) for the power analysis by calculating the throughput and latency. Here PCC gives higher throughput in comparison to other algorithms.

III. TCP CONGESTION CONTROL ALGORITHMS

TCP congestion control protocols are categorized using various congestion conditions. These congestion control protocols are categorised in to three different types based on their feedback. These are loss-based, delay-based and loss-delay-based (hybrid-based). The loss based mechanism considers packet loss as a sign of network congestion. For these types of protocols, both packet loss as well as random error are taken as signal for network congestion. Similarly, delay based approach mainly focuses on high link utilization with in short bottleneck queue by considering the information related to RTT (round-trip time) samples. It mainly considers the increase in RTT as a predictor of network congestion. Loss based CC mainly focuses on satisfaction of bandwidth requirement which leads to the issues of both RTT and TCP unfairness. Delay based CC facilitates only the RTT fairness. To overcome these issues, some of the TCP variants use delay based approach in addition to get loss feedback. This type of CC becomes more scalable in case of long distance and fast network architecture. It doesn't

take much stress in the rapid growth of $cwnd$ when the queue is short and easily switch to slow start phase in case of large queue. Moreover, it always tries to maintain RTT fairness over the communication network.

A. TCP BIC

It stands for binary increase congestion control protocol. It is a type of loss based approach which is more appropriate for high speed long fat network having high latency. It mainly focuses on improving the underutilized bandwidth. The $cwnd$ depends on two types of control policies which are called additive increase and binary search increase [13]. It always tries to find maximum $cwnd$ by searching in 3 ways as i.e. binary search increase, additive increase, and slow start.

Binary Search Increase: It considers congestion control as a type of searching problem where the system provides yes or no feedback by the help of packet loss. It repeatedly tries to compute the midpoint (targetin) between W_{min} and W_{max}. It updates the midpoint as the current $cwnd$ and observed feedback in terms of packet loss. Based on this feedback, if there is a packet loss then the mid-point is considered as new W_{max}. If there is no packet loss, then the mid-point is considered as new W_{min}.

Additive Increase: BIC uses another reference variable known as S_{max} (maximum increment). If the difference between targetin and current $cwnd$ is too large, then increase the $cwnd$ by S_{max} until the distance from the current $cwnd$ will be less than S_{max}. To provide RTT fairness, the binary increase strategy has been combined with additive increase.

Slow Start: The slow start mechanism used in TCP BIC is different from TCP Reno. When the value of current $cwnd$ is greater than W_{max}, then the binary search algorithm starts of searching new W_{max} which is unknown. At this time, it runs a slow start mechanism if the value of W_{max} is lesser then the value of (W_{max} + S_{max}).

B. TCP CUBIC

It is a loss based approach and an enhanced version of TCP BIC which is also the default congestion control protocol for TCP in Linux. It uses two function concave and convex after the last congestion event. In concave function, the window size rapidly ramps up before the last congestion event [13]. In the convex portion, first, it slowly grows the window size and later it quickly does the same task if more bandwidth is available. The main benefit of using this protocol is that the growth of the congestion window doesn't depend on the RTT. It depends on the two consecutive congestion events. During packet transmission, if any loss occurs, then it behaves same as TCP BIC. It uses multiplicative decrease factor β for reduction of $cwnd$. If the $cwnd$ is less than particular threshold value, then it uses normal TCP. If the value is greater, then it uses CUBIC mechanism to decide the concave and convex region. If the current window size is less then the threshold value, then it follow normal TCP approach. Similarly, if W is less then W_{max}, then it continues in concave region else convex region.

C. TCP Vegas

It considers packet delay as a sign for detecting congestion instead of packet drop and according to this it adjusts its sending rate. It updates $cwnd$ by evaluating the congestion status of the network using RTT measurement. When congestion is about to occur in the network, it instantly minimizes the window size to avoid the condition of packet loss [9]. It mainly focuses on maximizing throughput and minimizes the packet drop factor. It uses slow start phase same as the slow start phase of TCP Reno. TCP Vegas uses the difference between actual throughput and expected throughput to compute the level of congestion in the network for deciding the window growth function during congestion avoidance phase. The interested one may refer to [9] for detailed discussion. Vegas uses the difference between expected throughput and actual throughput to decide the window growth function. In case of congestion avoidance phase, window size is updated by comparing this difference with two threshold values α and β. The difference between expected throughput and actual throughput is expressed as follows:

$$Diff = Expected\,Throughput - Actual\,Throughput \quad (1)$$

The $cwnd$ in congestion avoidance phase is updated as follows:

$$cwnd = \begin{cases} cwnd\ +\ 1, if\ Diff{<}\alpha \\ cwnd, if\ \alpha{<}Diff{<}\beta \\ cwnd\ -\ 1, if\ Diff{>}\beta \end{cases} \quad (2)$$

If RTT >Base RTT, then there will be a bottleneck link and the backlog at the queue can be calculated as follows:

$$N = Diff * BaseRTT \quad (3)$$

BaseRTT is the minimum RTT value of all measured RTT, alternatively we can consider as the measured RTT of first segment that has been sent over the network. TCP Vegas always tries to keep the value of N as small as possible to avoid packet drop due to buffer overflow.

D. TCP Veno

In recent years, wireless communication technology has been building significant growth to access network. However in wireless medium, the event of packet loss may occur due to noise like error which is known as random loss. In TCP Reno, an issue was encountered by misinterpreting random loss as a signal of packet loss that causes unnecessary reduction of $cwnd$. As a result, it leads to performance degradation. TCP Veno helps to distinguish between packet drop due to congestion and random loss. We can say, it is a combination of both TCP Reno and Vegas algorithm. It predicts congestion before the occurrence of any random packet loss. It continuously monitors the level of network congestion level and according to that it decides the cause of packet loss due to

random bit error or network congestion. Basically, it rectifies the multiplicative decrease and linear increase algorithm to fully utilize the network bandwidth. It behaves same as TCP's Reno with some certain modifications.

Slow Start: In this phase, initially Reno sets the $cwnd$ to one MSS. After each ACK, it increases the $cwnd$ exponentially and so on until there is an occurrence of packet loss. Veno also applies the same initial slow start phase as Reno without any modifications.

Additive Increase: In TCP Veno, the additive increase phase has been modified as follows:

$$cwnd = \begin{cases} \text{(for each new ACK is received)} \\ cwnd + \dfrac{1}{cwnd(t)}, \ if \ N{<}\beta \\ \text{(for every other new ACK is received)} \\ cwnd + \dfrac{1}{cwnd(t)}, \ if \ N \geq \beta \end{cases} \quad (4)$$

Multiplicative Decrease: This is same as TCP Reno. It only changes the setting of slow start threshold parameter ($ssth$) based on some certain condition. If value of N is less then β, then the $ssth$ value will be updated as $cwnd{*}4/5$. If not, then it will be half of the $cwnd$ value.

E. Compound TCP

The Compound TCP (CTCP) adds a scalable delay based component in the standard TCP CC protocol. By adding this component, it provides a good bandwidth scalability and increases the data rate when the network link is under-utilized [14]. It also gives early reaction to congestion in case of changes in RTT and according to that it changes its $cwnd$. CTCP is mainly designed to achieve high efficiency and to achieve the requirement of TCP friendliness. To deploy compound TCP, it maintains some state variables $cwnd$, $dwnd$ (delay window), $awnd$ (receiver advertised window). The interested one may refer to [15] for detailed implementation of CTCP. Receiver's advertised window shows that the amount of data packets that the receiver side can receive.

IV. Simulation and Results

A. Simulation Setup

We have performed analysis of various TCP congestion control algorithm based on the network topology as given in Fig. 1 and its parameter values is taken according to values depicted in Table I. The orange nodes are the TCP source nodes followed by sinks. Similarly UDP source nodes are indicated with yellow colour. Each source and sink node has bandwidth of 1Gbps with delay of 2 ms. All nodes are connected through three routers having bandwidth 1Gbps with delay of 20 ms. The source node starts sending data through router R_1 at time = 1 ms. Then it will pass to router R_2 which has one TCP and one UDP source node attached to it. Finally these data are reached to destination through router R_3. By the help of the following topology, we have analysed performance of various congestion control algorithm. To implement the following, NS-2 network simulator has been used. It is mainly helpful for implementing any existing or new TCP congestion control algorithm.

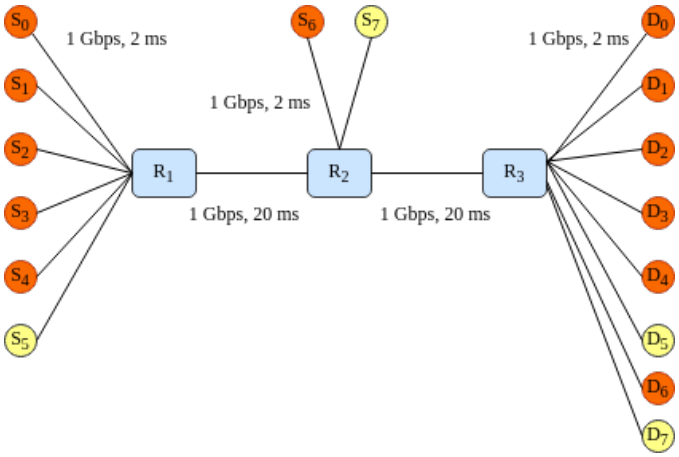


Fig. 1. Network Topology

TABLE I
Parameter Settings

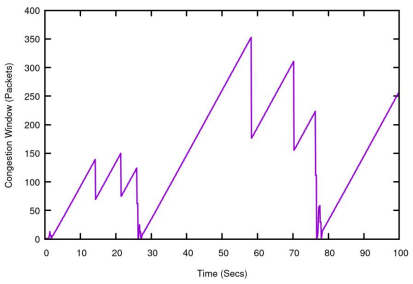
Parameter	Values
TCP CCAs	Reno, Newreno, BIC, CUBIC, Vegas, Veno, CTCP
Total number of souce node	8
Total number of destination node	8
All link Speed	1Gbps
Delay(Source to R1)	2ms
Bottleneck Delay	20ms
Link Management	Droptail
Simulation Time	100 sec

B. Performance Analysis

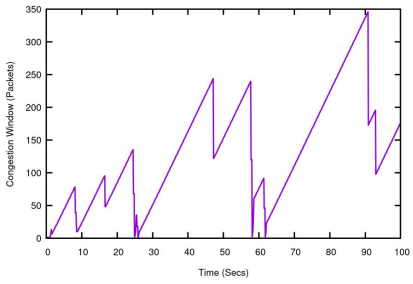
1) Variations in Congestion Window Size: Here, we have analysed variations in congestion window of different congestion control algorithms as depicted in Fig. 2. It has been observed from the Fig. 2 that Veno is more robust than other existing schemes. The cubic behaviour has been observed in TCP CUBIC mechanism.

2) Throughput Analysis: The sum of total average throughput has been calculated by amount of data that has been successfully transferred at a given point of time. In our experiment, we have calculated average throughput by tracing the path from R_2 to R_3. We observe from the results depicted in Table II that CUBIC achieves the highest average throughput while Vegas achieves the lowest average throughput. Furthermore, to show the variations of throughput after every 1 second, we have depicted the throughput analysis in Fig. 3. It has been observed that Reno shows least stability in terms of throughput.

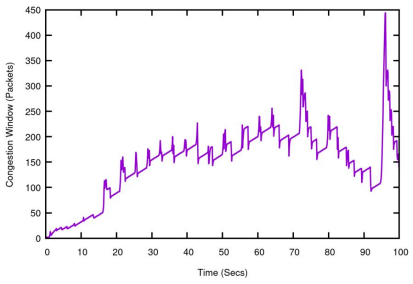
3) End-to-End Delay: The end-to-end delay indicates the time required to travel the data packet from the source to destination. Here we have calculated average end-to-end delay by considering the total sum of delay calculated by recording start time at the sender and end time at the destination. From



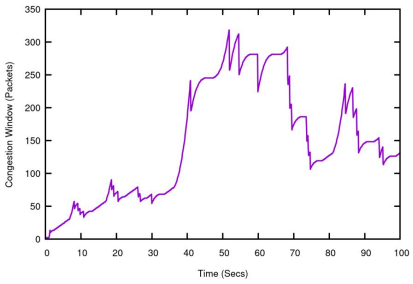
(a) Reno



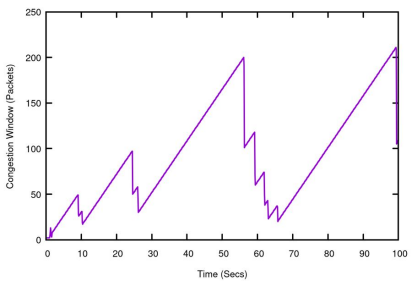
(b) Newreno



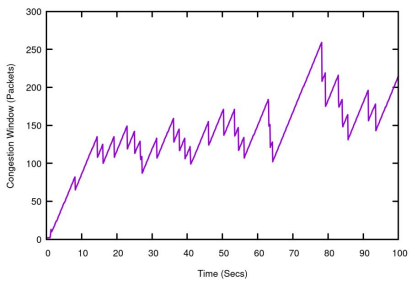
(c) BIC



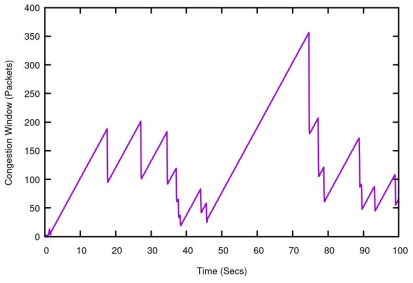
(d) CUBIC



(e) Vegas



(f) Veno



(g) CTCP

Fig. 2. Analysis of Congestion Window

TABLE II
COMPARISON OF AVERAGE THROUGHPUT(MBPS)

Type of CCA	Average Throughput(Mbps)
Reno	108.68
Newreno	106.23
BIC	107.23
CUBIC	118.02
Vegas	83.41
Veno	113.67
CTCP	108.68

TABLE III
COMPARISON OF END-TO-END DELAY

Type of CCA	Average End-to-End Delay (ms)
Reno	44.0319
Newreno	44.0322
BIC	44.0326
CUBIC	44.0320
Vegas	44.0315
Veno	44.0298
CTCP	44.0324

this, we have observed that average end-to-end delay of TCP BIC is higher while it is lower for Veno as compared to other mechanisms as shown in Table III. Veno is one of the latest TCP scheme among these TCP congestion control algorithms that is based on delay and loss based. The results shows that Veno achieves its objective to reduce the delay.

4) Packet Loss Ratio: The analysis of packet loss ratio of different algorithms is illustrated in Table IV. It is calculated by the total number of packets dropped with respect to total number of packets sent at the sender side. In our analysis, Vegas has low packet drop ratio as compared to other algorithms. It was proposed to reduce the loss and delay by considering the parameter delay for deciding the window growth function

V. CONCLUSION

In this paper, we have performed analysis of various TCP congestion control protocols based on different performance metrics. TCP CUBIC achieves highest average throughput of 118.02 Mbps while highest maximum throughput is achieved

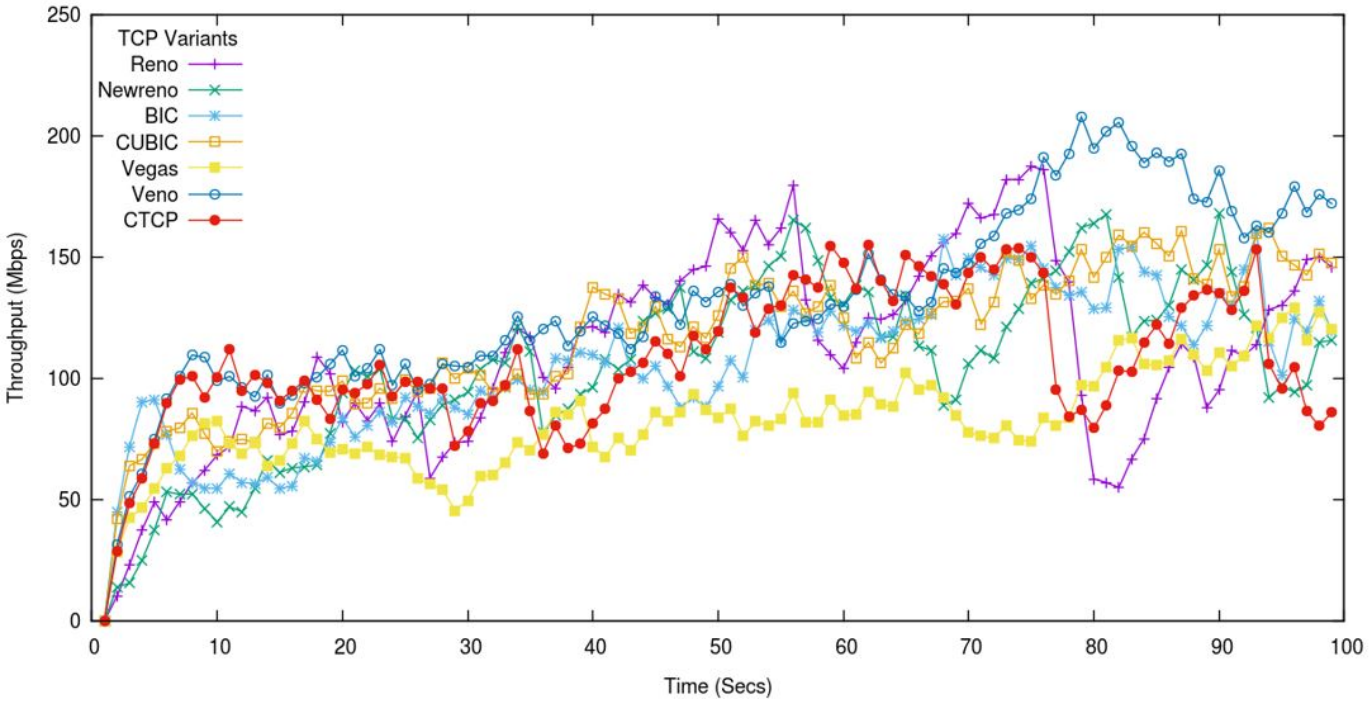


Fig. 3. Comparison of Throughput(Mbps) in every 1 Second

TABLE IV
COMPARISON OF PACKET LOSS RATIO

Type of CCA	Packet Loss Ratio (%)
Reno	0.0102
Newreno	0.0108
BIC	0.0582
CUBIC	0.0126
Vegas	0.0081
Veno	0.0122
CTCP	0.0138

by Veno as depicted in Fig. 3. Similarly, TCP Veno gives lowest end-to-end delay of 44.0298 ms and Vegas has lowest packet drop ratio of 0.0081% among all other TCP mechanisms. Furthermore, we summarized these algorithm using variation of congestion window size. In future work, we will try to examine all these protocol under IoT based home infrastructure network scenario using both wired and wireless medium. Moreover, we will attempt to use an optimized parameter settings to improve the above algorithm performance.

REFERENCES

[1] K. Sasaki, M. Hanai, K. Miyazawa, A. Kobayashi, N. Oda, and S. Yamaguchi, "Tcp fairness among modern tcp congestion control algorithms including tcp bbr," in *2018 IEEE 7th international conference on cloud networking (CloudNet)*, pp. 1–4, IEEE, 2018.

[2] R. Al-Saadi, G. Armitage, J. But, and P. Branch, "A survey of delay-based and hybrid tcp congestion control algorithms," *IEEE Communications Surveys & Tutorials*, vol. 21, no. 4, pp. 3609–3638, 2019.

[3] K. Sasaki, M. Hanai, K. Miyazawa, A. Kobayashi, N. Oda, and S. Yamaguchi, "Tcp fairness among modern tcp congestion control algorithms including tcp bbr," in *2018 IEEE 7th international conference on cloud networking (CloudNet)*, pp. 1–4, IEEE, 2018.

[4] W. Li, H. Zhang, S. Gao, C. Xue, X. Wang, and S. Lu, "Smartcc: A reinforcement learning approach for multipath tcp congestion control in heterogeneous networks," *IEEE Journal on Selected Areas in Communications*, vol. 37, no. 11, pp. 2621–2633, 2019.

[5] G. Hasegawa and M. Murata, "Survey on fairness issues in tcp congestion control mechanisms," *IEICE Transactions on Communications*, vol. 84, no. 6, pp. 1461–1472, 2001.

[6] M. A. Alrshah, M. Othman, B. Ali, and Z. M. Hanapi, "Comparative study of high-speed linux tcp variants over high-bdp networks," *Journal of Network and Computer Applications*, vol. 43, pp. 66–75, 2014.

[7] D. Ros and M. Welzl, "Less-than-best-effort service: A survey of end-to-end approaches," *IEEE Communications Surveys & Tutorials*, vol. 15, no. 2, pp. 898–908, 2012.

[8] T. Lukaseder, L. Bradatsch, B. Erb, R. W. Van Der Heijden, and F. Kargl, "A comparison of tcp congestion control algorithms in 10g networks," in *2016 IEEE 41st Conference on Local Computer Networks (LCN)*, pp. 706–714, IEEE, 2016.

[9] A. Afanasyev, N. Tilley, P. Reiher, and L. Kleinrock, "Host-to-host congestion control for tcp," *IEEE Communications surveys & tutorials*, vol. 12, no. 3, pp. 304–342, 2010.

[10] G. Sun, C. Li, Y. Ma, S. Li, and J. Qiu, "End-to-end tcp congestion control as a classification problem," *IEEE Transactions on Reliability*, 2022.

[11] S. R. Pokhrel and C. Williamson, "Modeling compound tcp over wifi for iot," *IEEE/ACM transactions on networking*, vol. 26, no. 2, pp. 864–878, 2018.

[12] S. Claypool, J. Chung, and M. Claypool, "Comparison of tcp congestion control performance over a satellite network," in *International Conference on Passive and Active Network Measurement*, pp. 499–512, Springer, 2021.

[13] 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 *ICSC 2020*, pp. 23–28, IEEE, 2020.

[14] D. Ghosh, K. Jagannathan, and G. Raina, "Right buffer sizing matters: some dynamical and statistical studies on compound tcp," *Performance Evaluation*, vol. 139, p. 102095, 2020.

[15] S. R. Pokhrel and C. Williamson, "Modeling compound tcp over wifi for iot," *IEEE/ACM Transactions on Networking*, vol. 26, no. 2, pp. 864–878, 2018.