# Analysis of TCP BBR Congestion Control Algorithm

**Muppiri Abhishek**

Department of Computer Science and Engineering
**National Institute of Technology Rourkela**

# Analysis of TCP BBR Congestion Control Algorithm

*Dissertation submitted in partial fulfillment*

*of the requirements for the degree of*

***Bachelor of Technology***

*in*

***Computer Science and Engineering***

*by*

***Muppiri Abhishek***

(Roll Number: 119CS0505)

*based on research carried out*

*under the supervision of*

***Prof. Sanjeev Patel***

May, 2023

Department of Computer Science and Engineering

**National Institute of Technology Rourkela**

Department of Computer Science and Engineering
**National Institute of Technology Rourkela**

May 11, 2023

# Certificate of Examination

Roll Number: *119CS0505*
Name: *Muppiri Abhishek*
Title of Dissertation: *Analysis of TCP BBR Congestion Control Algorithm*

We the below signed, after checking the dissertation mentioned above and the official record book (s) of the student, hereby state our approval of the dissertation submitted in partial fulfillment of the requirements of the degree of *Bachelor of Technology* in *Computer Science and Engineering* at *National Institute of Technology Rourkela*. We are satisfied with the volume, quality, correctness, and originality of the work.

Sanjeev Patel
Principal Supervisor

Head of the Department

Department of Computer Science and Engineering
**National Institute of Technology Rourkela**

**Prof. Sanjeev Patel**
Professor

May 11, 2023

## Supervisors’ Certificate

This is to certify that the work presented in the dissertation entitled *Analysis of TCP BBR Congestion Control Algorithm* submitted by *Muppiri Abhishek*, Roll Number 119CS0505, is a record of original research carried out by him under our supervision and guidance in partial fulfillment of the requirements of the degree of *Bachelor of Technology* in *Computer Science and Engineering*. Neither this dissertation nor any part of it has been submitted earlier for any degree or diploma to any institute or university in India or abroad.

Sanjeev Patel
Professor

# Dedication

I dedicate this work to my supervisor, [Dr. Sanjeev Patel], for his valuable guidance and support throughout this project. His expertise in the field of Transmission control protocol(TCP) has been instrumental in shaping my research and refining my skills.
Furthermore, I dedicate this thesis to all the researchers and scientists who have contributed to the advancement of Transmission control protocol(TCP). Their groundbreaking work has paved the way for innovations such as TCP BBR Congestion control algorithm, and I am honored to be a part of this community.
I finally Dedicate this work to all the pioneers of computer networking and internet technology, whose dedication and hard work have made the world a more connected and accessible place. In particular, this work is dedicated to the developers of the TCP BBR congestion control algorithm, whose innovation and ingenuity have paved the way for faster, more efficient, and more reliable communication over the Internet. May this work contribute in some small way to the continued progress and evolution of computer networking and its positive impact on society.

*Signature*

# Declaration of Originality

I, *Muppiri Abhishek*, Roll Number *119CS0505* hereby declare that this dissertation entitled *Analysis of TCP BBR Congestion Control Algorithm* presents my original work carried out as a doctoral student of NIT Rourkela and, to the best of my knowledge, contains no material previously published or written by another person, nor any material presented by me for the award of any degree or diploma of NIT Rourkela or any other institution. Any contribution made to this research by others, with whom I have worked at NIT Rourkela or elsewhere, is explicitly acknowledged in the dissertation. Works of other authors cited in this dissertation have been duly acknowledged under the sections “Reference” or “Bibliography”. I have also submitted my original research records to the scrutiny committee for evaluation of my dissertation.

I am fully aware that in case of any non-compliance detected in future, the Senate of NIT Rourkela may withdraw the degree awarded to me on the basis of the present dissertation.

May 11, 2023
NIT Rourkela

*Muppiri Abhishek*

# Acknowledgment

- My journey in research began during my undergraduate studies. I approached Dr.Sanjeev Patel and discussed about my area of interest. He suggested me to work on Transmision Control Protocol(TCP) and introduced me to a PhD student, Subhra Biswal who helped me a lot in my work. I worked on a topic called ”Analysis of TCP BBR Congestion control algorithm”. With the support of my guide Subhra Biswal, I was able to complete the project.
- Overall my journey in research has been a fulfilling and enriching experience. I have gained valuable knowledge and skills that have prepared me for a future career in research. I am grateful for the support and guidance of my supervisor.

May 11, 2023
NIT Rourkela

*Muppiri Abhishek*
Roll Number: 119CS0505

# Abstract

As the complexity of the network communication environment increases. Investigating the TCP congestion management mechanism is one of the best ways to lessen network congestion. Google introduced the bottleneck bandwidth and round-trip time (BBR) congestion control method in 2016. BBR regulates its pacing rate using an estimate of the bottleneck link's available bandwidth and RTT, in contrast to many other loss-based congestion management algorithms (such as BIC) currently in use. BBR receives a lot of attention and performs well on the Google internal network. Using NS3, we create a number of network communication models in this research. We find from the simulation results that BBR outperforms BIC on high-latency, high-bandwidth networks, that BBR has a fairness problem with regard to other TCP protocol versions, and that BBR periodically lowers the pacing rate to deal with the bufferfloat problem. Additionally, we show that BBR flows have a problem with RTT equity since they outperform competition flows with shorter RTTs. Finally, we investigate the degree of adaptability of the BBR and AQM algorithms.The network simulator-3 (ns-3) is a popular tool for evaluating the current TCP congestion management methods and learning more about them. This facilitates the development of a fresh algorithm. We have compared research window size for several congestion management methods, including New Reno and CUBIC. We have also investigated the throughput of these algorithms using ns-3.

***Keywords*: *BBR*; *TCP*; *Newreno*; *CUBIC*; *Congestion Control*; *RTT*.**

# Contents

# List of Figures

# Chapter 1

# Introduction

Over the last several years, computer networks have grown rapidly, and with that expansion has come serious congestion issues [1]. Research on congestion reduction is currently underway [2]. But in the present network context, the traditional congestion management methods are no longer effective.On networks with high capacity and latency, TCP based on the AIMD algorithm (Additive Increase Multiplicative Decrease) performs badly. The reason is because TCP continuously seeks to progressively extend the transmission window during the congestion avoidance phase. The congestion window quickly contracts when a packet is lost, making it difficult to use the whole effective bandwidth. The bufferbloat [1] produced by the congestion management approach based on packet loss also affects network performance when the cache size of the core router continues expanding.In 2016 [4], Google's Bottleneck Bandwidth and RTT (BBR) congestion management method was initially proposed. It is neither delay-based nor loss-based and ignores packet loss as a sign of congestion. It builds a model of the network route made up of the bottleneck bandwidth and RTT to adjust its pacing rate [5]. BBR is focused on solving two problems: reducing network buffer occupancy and making optimal use of the network's capacity even in the face of considerable packet loss. Google has implemented BBR on its own production platforms, such as the servers for YouTube and the B4 wide-area network at Google.com, in order to develop and test it. Additionally, Google provided quick BBR integration with the Linux kernel (starting with version 4.9).

## 1.1 Objective

1. Analysis of TCP BBR algorithm using an opensource network simulator for high BDP networks
2. BBR's performance should be compared to that of other congestion control protocols, such as TCP Cubic, TCP Reno, or TCP Vegas, under various network scenarios (such as those with differing link capacities, round-trip times, and network topologies).
3. Analyze the behavior of TCP variants in a common networking environment.

4. To investigate the impact of different BBR parameters, such as the pacing gain, congestion window gain, and round-trip time filter, on the protocol's performance under various network conditions.

5. To evaluate the robustness of BBR to network dynamics, such as link failures, congestion, and traffic variations, and compare it with other protocols under similar conditions.

6. To identify the limitations of BBR and areas for improvement in the protocol's design and implementation.

## 1.2 Motivation

1. Gain a deeper understanding of computer networking: TCP BBR is a complex algorithm that involves concepts like congestion control, bandwidth estimation, and packet loss recovery. By analyzing this algorithm, we can gain a deeper understanding of how computer networks operate and how different protocols work together to ensure network stability and eliability.

2. Emerging technology: TCP BBR is a relatively new congestion control algorithm that has gained a lot of attention in recent years. By analyzing this algorithm, we can stay current with emerging technology in the field of networking and gain insights into how it is being used to optimize network performance.

3. Optimization of network performance: TCP BBR is designed to optimize network throughput and reduce latency in high-bandwidth, long-distance networks.By analyzing its performance, we can identify ways to improve network performance for their organizations or other applications.

4. Comparison with other algorithms: TCP BBR is not the only congestion control algorithm out there. By comparing its performance with other algorithms, such as TCP Cubic or TCP Vegas, we can identify the strengths and weaknesses of different congestion control algorithms and understand which algorithm may be best suited for different network environments.

# Chapter 2

# Literature Review

1. ”BBR: Congestion-Based Congestion Control” by Mohammad Alizadeh et al. This paper introduces the BBR congestion control algorithm and provides a detailed analysis of its key features, such as its congestion window estimator and pacing rate control mechanism. It also presents experimental results demonstrating its performance in different network environments.

2. The article ”Experimental Evaluation of TCP BBR Congestion Control” was written by Nandita Dukkipati and others. In both wired and wireless networks, the TCP BBR congestion control method is experimentally assessed in this research. It analyses how it behaves in various network scenarios and evaluates how it performs in relation to other TCP variations.

3. ”Performance Evaluation of TCP BBR in Mobile Networks” by Yu Li et al. This paper evaluates the performance of the TCP BBR congestion control algorithm in mobile networks, which have different characteristics than traditional wired networks. It analyzes its behavior under different network conditions and compares its performance with other TCP variants.

4. ”TCP BBR vs. TCP CUBIC: A Comparative Study” by Vijay Kumar et al. This paper provides a comparative study of the TCP BBR and TCP Cubic congestion control algorithms in different network environments. It examines their performance under different traffic patterns and network conditions and identifies their strengths and weaknesses.

5. ”Performance Analysis of TCP BBR and TCP New Reno in Data Center Networks” by Lijuan Xie et al. This paper compares the performance of the TCP BBR and TCP New Reno congestion control algorithms in data center networks. It evaluates their performance under different traffic loads and network conditions and identifies the factors that affect their performance.

Chapter 3

# Classification of TCP Variants

Over the years, various TCP variants have been developed to address different network conditions and requirements. Here are some common classifications of TCP variants:

1. Congestion Control-Based TCP Variants: These TCP variants are designed to improve the performance of TCP congestion control mechanisms in various network environments. Examples of congestion control-based TCP variants include TCP Reno, TCP Vegas, TCP Cubic, and TCP BBR.

2. TCP Variants for Wireless Networks: These TCP variants are designed to handle the unique characteristics of wireless networks, such as high bit error rates and variable link quality. Examples of TCP variants for wireless networks include TCP Westwood, TCP New Reno, and TCP Hybla.

3. High-Speed TCP Variants: These TCP variants are designed to handle high-speed networks that have high bandwidth and long delay times. Examples of high-speed TCP variants include TCP HighSpeed and TCP Scalable.

4. TCP Variants for Real-Time Applications: These TCP variants are designed to handle real-time applications, such as video streaming or voice over IP (VoIP). Examples of TCP variants for real-time applications include TCP Friendly Rate Control (TFRC) and TCP Proportional Rate Reduction (PRR).

5 Low-Latency TCP Variants: These TCP variants are designed to reduce latency and improve the responsiveness of TCP connections. Examples of low-latency TCP variants include TCP Fast Open and TCP Quick-Start.

Overall, these classifications of TCP variants reflect the different network conditions and requirements that TCP must handle in order to provide reliable, efficient data transmission over the internet.

## 3.1 Loss Based

Loss-based TCP variations reacts to packet loss events brought on by network congestion. These versions seek to avoid worsening network congestion, maintain equitable allocation of network resources among various flows, and enhance overall network performance by detecting and responding to packet loss. Examples of loss-based TCP variations include TCP Reno, TCP New Reno, and TCP Cubic protocols. These variations apply different congestion control algorithms [2]to modify their transmission behaviour in response to observed loss events and use packet loss as a primary congestion indication. These TCP variations, namely TCP Reno, TCP New Reno, and TCP Cubic, have been widely deployed and studied in networks characterized by moderate BDP. They are specifically designed to effectively manage congestion in such network environments

### 3.1.1 TCP Reno

The TCP Reno congestion control method consists of the following components:

1. Slow Start: TCP Reno starts the slow start phase when a TCP connection is made or after a period of idleness. The sender gradually increases its transmission rate during this phase by progressively extending the congestion window size. Through exploring the network's capacity, TCP Reno is able to investigate and identify the ideal transmission rate.

2. Congestion Avoidance: When the size of the congestion window exceeds a certain [1]threshold, TCP Reno switches from the slow start phase to the congestion avoidance phase. In this stage, the sender linearly and steadily expands the congestion window size. Maintaining a constant transmission rate while preventing network congestion is the goal.

3. Fast Retransmit/Fast Recovery: When TCP Reno detects a packet loss, it determines network congestion. It responds by immediately retransmitting the missing packet without waiting for the retransmission timeout to expire[3]. Furthermore, TCP Reno switches into a mode known as quick recovery to prevent future packet losses. In order to minimise congestion and preserve network stability, it reduces the congestion window size in this condition by 50

4. Timeout-based Recovery: When TCP Reno does not get an acknowledgement (ACK) for a sent packet within a certain timeout interval, it recognises a packet loss. It responds by sending the missing packet again. To further minimise congestion, TCP Reno also dramatically shrinks the congestion window size.

### 3.1.2 TCP-New Reno

By altering the quick recovery method and introducing a mechanism for reacting to incomplete ACKs, Newreno is a variation of the Reno approach [8].With the intention of improving its functionality and correcting some of its shortcomings, the TCP New Reno congestion management algorithm is an extension of the TCP Reno algorithm. In the case of packet loss, it enhances recovery, allowing for speedier and more successful retransmissions. One of the most important enhancements in TCP New Reno is partial acknowledgments (ACKs). In the event of packet loss, TCP New Reno uses partial acknowledgments (ACKs) to pinpoint individual packets that the recipient has successfully received. TCP New Reno, in contrast to TCP Reno, acknowledges[1] individual packets, enabling it to identify and resend only the lost packets as opposed to the full congestion window. This focused strategy improves TCP New Reno's retransmission's effectiveness. By reducing needless retransmissions, the TCP New Reno selective retransmission mechanism improves network efficiency. It facilitates speedier recovery from repeated packet losses and lessens performance degradation induced on by[4] congestion occurrences. In TCP New Reno, the speedy recovery technique from TCP Reno has been modified. When TCP Reno gets three duplicate ACKs, it uses a speedy retransmit method and goes into a fast recovery stage. With a feature known as ”partial ACK congestion avoidance,” TCP New Reno improves on this strategy by allowing it to manage situations in which many packets are dropped sequentially. With this improvement, TCP New Reno is better equipped to react to these circumstances and modify its congestion management strategies as necessary. In some situations, TCP New Reno decreases the congestion window size more cautiously to avoid responding excessively to incomplete ACKs.

### 3.1.3 TCP-Cubic

To improve speed and fairness in these network settings, this protocol uses a cubic increment approach. One of TCP Cubic's key characteristics is how it responds to network congestion[4-7]. Instead of the linear increase and multiplicative reduction methods used by previous TCP versions, TCP Cubic uses a cubic increase function. This technique allows TCP Cubic to aggressively widen[4] its congestion window and achieve a more stable equilibrium state during times of low congestion. The TCP Cubic programme determines the network capacity and congestion level by counting the number of Acknowledgment (ACK) packets received. RTT data and the cubic function are used to alter the congestion window size. The cubic function used by TCP Cubic uses the amount of time that has passed since the [1]last time there was congestion to decide how quickly the congestion window should grow or shrink. The key benefit of TCP Cubic is the efficiency with which it performs on networks with high BDP levels. It successfully tackles the difficulties of sluggish convergence and underutilization of network capacity that are frequently present in such network circumstances for standard TCP versions

### 3.1.4 TCP-West Wood

TCP Westwood proposes a fully TCP Reno-based end-to-end bandwidth estimate method. TCP Westwood is a Newreno variant that only works on the sender side [3]. The expected rate, sometimes referred to as the eligible rate, is determined (see reference ref1). When a loss indication occurs, the sender uses this rate to adjust the slow start threshold (SSTHRESH) and congestion window size (CWND). [**?** ]The agile probing phase has replaced the well-known slow start phase [10]. To identify a prolonged lack of congestion, a continuous non-congestion detecting method has also been put out.TCP Westwood produces delayed start and congestion avoidance phases similarly to TCP Reno. TCP Westwood adjusts to link usage by figuring out the appropriate bandwidth and adjusting the slow start threshold, in contrast to TCP Reno, which cuts the size of the congestion window in half when congestion occurs.

## 3.2 Delay based

Delay-based TCP is a subset of TCP that emphasises the use of delay measurements as a crucial element in congestion management. Delay-based TCP variants assess network congestion levels by taking into account RTT or other delay measures, in contrast to conventional loss-based TCP variants that largely depend on detecting packet loss as a signal of congestion. By responding to changes in delay rather than relying simply on packet loss as[4] a congestion indicator, delay-based TCP versions seek to maintain a stable network. These variations optimise performance and prevent unneeded congestion by monitoring the RTT and making appropriate adjustments to the transmission rate and congestion window size. They are especially helpful in high-speed networks where performance might be negatively impacted by lengthy delays, such as those brought on by long-distance communication or significant BDP

### 3.2.1 TCP-Vegas

TCP-Vegas was developed to minimise TCP network congestion, which impacts packet loss and packet delay. TCP Vegas is a TCP congestion management strategy that, like TCP-Reno and TCP-New Reno, focuses on reducing network delay rather than using just packet loss as a symptom of congestion in order to enhance performance. network functionality. It was developed to address issues with loss-based TCP variants on high-speed networks. Unlike conventional TCP versions, TCP Vegas determines the available bandwidth by accounting for the RTT and monitoring the queuing time. The majority of traditional TCP variants respond to packet loss. Using both RTT and queue delay information, TCP Vegas can precisely assess the network environment and adjust its transmission rate. It aims to keep the routers' queuing delays within a reasonable range in order to reduce latency. The congestion window size is gradually raised as the Round-Trip Time (RTT), which monitors bandwidth availability, drops below the acceptable level. On the other side, the congestion window size is decreased if the RTT exceeds the necessary value. TCP Vegas employs an iterative process to dynamically change the congestion window size in order to improve performance and manage network congestion. TCP Vegas prioritises delay reduction to provide low-latency communication for applications that need real-time response.

### 3.2.2 FAST-TCP

High latency and long-distance lines are where FAST-TCP [11] is most often employed.The California Institute of Technology's Net Lab created it, but it is a commercial product[5]. Existing algorithms are also supported by the FAST-TCP. Based on the internal delay-based assessment of the network status, it defines a recurring congestion window update The congestion management approach TCP FAST (Fast Active Queue Management Scalable Transmission management Protocol)[12] was created to solve the shortcomings of conventional TCP versions in high-speed networks. To improve performance, it incorporates changes to the TCP sender and router. TCP FAST takes a preventative stance by keeping an eye on and managing [4] the queue length at the network routers. To keep the network stable and effective, it modifies the sender's congestion window based on the predicted queue length. In high-speed network situations, this proactive congestion control aids in lowering packet loss, raising throughput, and enhancing fairness. The utility of TCP FAST in enhancing performance in settings with significant bandwidth-delay products and fast network rates has been investigated.

## 3.3 Hybrid TCP

By accounting for both delay and loss signals, hybrid TCP may effectively adapt to varied congestion situations while increasing throughput, latency, and fairness. The implementation of hybrid TCP may vary depending on the variant. Compound TCP is a well-known hybrid TCP variant. In the CTCP sluggish start phase, a delay-based approach is utilised, whereas the CTCP congestion avoidance phase uses a loss-based techniqu[4]e. Hybrid TCP variants can be used to manage a wide range of network conditions, including high-speed networks with large delays and fluctuating levels of congestion. By using delay- and loss-based approaches, these versions aim to boost flexibility, stability, and efficiency in managing network congestion. The development of hybrid TCP variants addresses the challenges that traditional TCP variations have in [1]adapting to shifting network dynamics and a variety of application requirements. By combining the advantages of delay-based and loss-based approaches, hybrid TCP provides a workable solution for enhancing performance in modern network situations.

### 3.3.1 Compound TCP

The design of Compound TCP seeks to strike a compromise between TCP friendliness and efficiency. The crucial point is that the high-speed protocol must aggressively raise the transmitting price if the link isn't being used to its fullest capacity. A common TCP congestion prevention method is included in compound TCP. The CTCP manages crowded areas using a two-phase technique.It employs a delay-based strategy to progressively raise the congestion window size during the slow start phase in order to assess the network capacity. The queuing delay and actual RTT are used to calculate the optimal transmission rate. The CTCP begins the congestion avoidance phase when the size of the congestion window reaches a predetermined threshold. It now uses a loss-based congestion control technique similar to TCP Reno cite ref1. CTCP monitors the instances of packet loss and modifies the congestion window size using the conventional additive increase/multiplicative reduction mechanism. What sets CTCP apart from other protocols is its ability to switch between delay-based and loss-based operations dynamically. CTCP can function effectively and reliably in a variety of network circumstances, including high-speed networks with large delays and various degrees of congestionciteref19. It has been shown that CTCP outperforms traditional TCP versions in terms of performance, latency, and fairness.

### 3.3.2 TCP YeAH

A TCP congestion control method called TCP YeAH (Yet Another High-Speed TCP) was created specifically to enhance performance on high-speed networks. It focuses on speeding up the network and decreasing wait times. TCP YeAH uses the observed RTT[4] and the anticipated volume of competing traffic to dynamically alter the congestion window in this manner. TCP YeAH employs an AIMD algorithm in a similar way to TCP Reno, but with modified parameters for faster convergence and improved performance. A self-clocked system additionally adjusts the congestion window based on the measured RTT and the amount of competing flows in the network. Since TCP YeAH responds more quickly to changes in network[1] conditions, it is able to use available bandwidth more efficiently while ensuring fairness. According to test results, TCP YeAH may operate on high-speed networks with a speedier throughput and lessened queuing latency than conventional TCP versions.

### 3.3.3 TCP Veno

Congestion-control module TCP Veno enhances TCP's performance on wireless networks. The capacity of the accessible channels is determined by TCP Veno. To determine the congestion factor, the actual RTT is compared to a predicted RTT. The size of the congestion window may be altered using this congestion factor. TCP Veno considers RTT variations in addition to packet loss, unlike conventional TCP versions that just use packet loss as a congestion indication. TCP Veno seeks to improve fairness and responsiveness by leveraging both delay and loss information. TCP Veno has done better in testing than other TCP variations, especially on networks with large bandwidths and long latency. Compared to TCP variations like TCP Reno or TCP Cubic, which are more extensively used, their adoption and utilisation may be limited.Being shrewd, TCP Veno can tell them apart. By avoiding the base RTT from fluctuating, the connection will be operational for a longer period of time in the operating zone.

## 3.4 BDP-Based

### 3.4.1 BBR

The BBR method is unique from the majority of others in that it gives packet loss very little consideration. Instead, the actual bandwidth of data transmitted to the far end serves as its key indicator[6]. The BBR changes the amount of data supplied every time an acknowledgement packet is received. Since the connection has clearly given that bandwidth recently, the total amount of data transmitted over a period of time is a pretty good indicator of the bandwidth the connection is capable of providing. BBR will be in the ”startup” stage when a connection first establishes itself. In this mode, it operates similarly to most conventional[2] congestion-control algorithms in that it ramps up transmission speed fast after a sluggish start in an effort to gauge the available bandwidth. While most algorithms ramp up until a packet is discarded, BBR monitors the bandwidth measurement as mentioned before. It specifically checks to determine if the actual supplied bandwidth has changed during the previous three round-trip periods. BBR believes it has discovered the connection’s effective capacity after it stops increasing the bandwidth; this is likely to occur far before packet loss would start[7].

# Chapter 4

# BBR Algorithm

The BBR method is unique from the majority of others in that it gives packet loss very little consideration. Instead, the actual bandwidth of data transmitted to the far end serves as its key indicator[4]. The BBR changes the amount of data supplied every time an acknowledgement packet is received. Since the connection has clearly given that bandwidth recently, the total amount of data transmitted over a period of time is a pretty good indicator of the bandwidth the connection is capable of providing. BBR will be in the ”startup” stage when a connection first establishes[1] itself. In this mode, it operates similarly to most conventional congestion-control algorithms in that it ramps up transmission speed fast[3] after a sluggish start in an effort to gauge the available bandwidth. While most algorithms ramp up until a packet is discarded, BBR [8] monitors the bandwidth measurement as mentioned before. It specifically checks to determine if the actual supplied bandwidth has changed during the previous three round-trip periods. BBR believes it has discovered the connection’s effective capacity after it stops increasing the bandwidth; this is likely to occur far before packet[1] loss would start. The pace at which packets should be transmitted over the link is then determined by the observed bandwidth. However, while measuring that rate, BBR likely sent packets at a faster rate for a period; some of them will be awaiting delivery in queues[9]. BBR will enter a ”drain” mode where it will transmit below the measured bandwidth until it has made up for the extra packets transmitted earlier in an effort to drain those packets out of the buffers where they are sitting. BBR enters the steady-state mode when the drain phase is complete and transmits at about the estimated bandwidth. That is ”more-or-less” true since a network connection’s properties change over time, requiring constant monitoring of the actual provided bandwidth. Also, BBR will scale the rate up by 25[4]around 1/8 of the time because a gain in effective bandwidth can only be recognised by seldom attempting to transmit at a higher rate. That probe will be followed by a drain phase to restore balance if the bandwidth has not risen (transmitting at a faster rate does not result in data being delivered at a higher[10] rate, in other words). One intriguing feature of BBR is that, in contrast to most other algorithms, it does not primarily manage outgoing traffic using the congestion window [3]. A wave of packets will often use up the additional bandwidth if the congestion window is expanded, which limits the amount of data that may be in flight at any one time. Instead, BBR uses the tc-fq packet scheduler to transfer data at the right pace. Although it is no longer the primary regulatory tool, the congestion window is nevertheless adjusted to make sure there is never too much data in flight. One further issue[4]: many network connections are controlled by ”policers,” middleboxes that set a maximum data rate cap on each connection. There is no purpose in attempting to go faster than the rate that the box will permit if one exists. The BBR algorithm searches for periods with suspiciously stable bandwidth (within 4Kb/sec) and a high packet loss rate; if this occurs, it deduces that there is a policer in the loop and sets the bandwidth limit to a value that will prevent the[1] policer from beginning to lose packets. Neal Cardwell published the BBR patch set’s source code, which also has the signatures of Van Jacobson and Eric Dumazet. When just one side of the connection is utilising BBR, it performs fairly well, thus each deployment should, if it fulfils its promises, improve the internet even more. Google [3] has reportedly been utilising BBR for some time and is reportedly happy with the outcomes. By the time we learn more, networking maintainer David Miller should have already applied the patches, making BBR part of the 4.9 kernel.[8] Due to its essential nature, BBR’s advantages reach beyond Google and YouTube. A few manufactured microbenchmarks demonstrate the benefits (albeit maybe not to their full extent). By routinely calculating the bottleneck bandwidth (BtlBw) and round-trip propagation delay (RTprop), BBR builds a network model. A product called the Bandwidth Delay Product (BDP) controls how quickly packets enter the network and how much data may be delivered at once (reference ref 18). Pacing rate and congestion window (Cwnd) are the two control parameters that BBR uses to regulate its transmitting behaviour.gain factors (G). [3].

$$BDP = BtlBW * RTprop \tag{1}$$

$$Cwnd = G * BDP \tag{2}$$

$$Pacing\,rate = G * BtlBW \tag{3}$$

## 4.1 Higher Throughput:

On the high-speed, long-haul networks, BBR enables large throughput enhancements.this Considers the typical server-class computer that transmits data over a 10 Gigabit Ethernet connection from Chicago to Berlin with a 100 ms round-trip time and a 1% packet loss rate. The throughput of CUBIC, the most efficient loss-based congestion control currently in use, is 2700 times lower in this situation than that of BBR. (BBR reaches speeds of nearly 9,100 Mbps, whilst CUBIC only reaches speeds of 3.3 Mbps.)[4]. A single BBR connection may fully use a route with packet loss because of its loss resistance. Given that there is just one connection required, HTTP/2 is the best option.[3] As a consequence, users may avoid using workarounds like establishing several TCP connections to enhance consumption. higher capacity, higher traffic on today's high-speed backbones, and faster download times for websites, movies, and other data are the end results.

## 4.2 Lower Latency:

In last-mile of networks that link consumers to the internet, BBR enables a significant decrease in latency [7]. Take into account a typical last connection with 10 Megabits of bandwidth, a bottleneck buffer of 1000 packets, and a 40 ms round-trip latency .[6]. According to reference 15, in such a case, BBR maintains a queuing latency that is 25 times lower that mile CUBIC (CUBIC has a median round-trip length of 1090 ms, while BBR has only 43 ms). BBR shortens last-mile lines, which lowers waiting times when streaming movies or downloading applications. Additionally, it quickens internet surfing and improves the responsiveness of video conferences and games. One may argue that BBR could stand for the BufferBloat Resilience but in addition to Bottleneck Bandwidth and Round-trip propagation time because of its capacity to reduce bufferbloat.

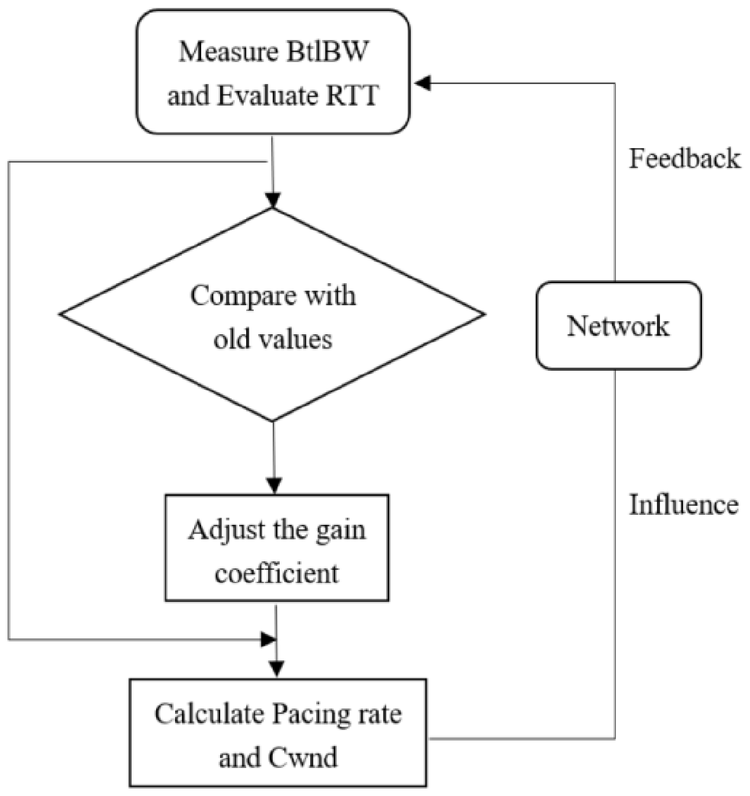
Measure BtlBW and Evaluate RTT
Feedback
Compare with old values
Network
Adjust the gain coefficient
Influence
Calculate Pacing rate and Cwnd

Figure 4.1: The working principle of the TCP-BBR.

# Chapter 5
# Results and Discussions

## 5.1 Simulation Setup

Figure shows the topology design used for performing various comparisons and evaluations.

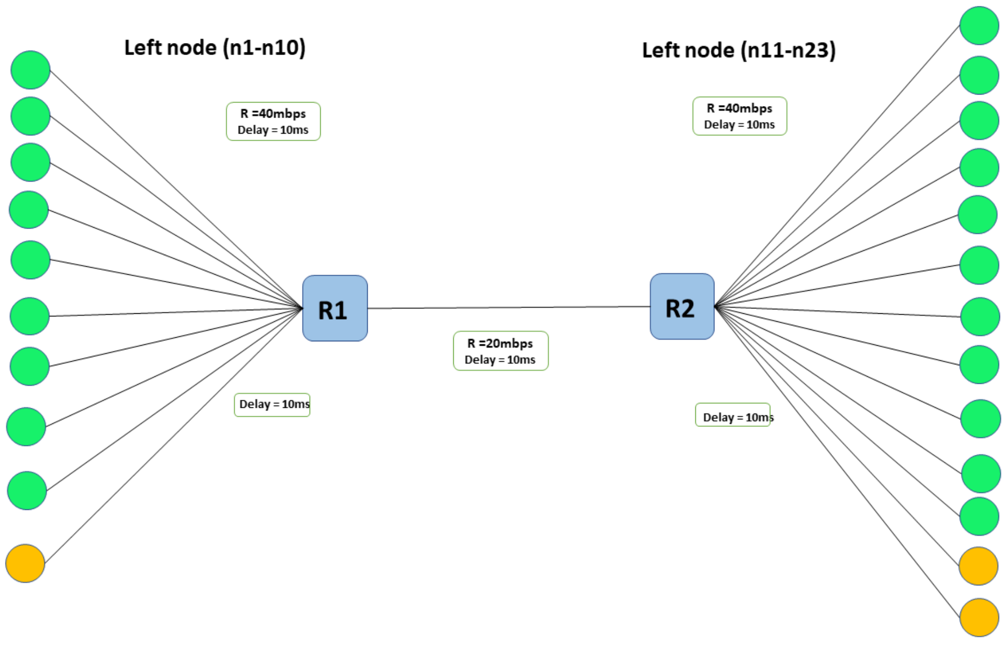

Figure 5.1: Topology for a network simulation experiment with the green nodes using TCP and the yellow ones using UDP.

In a network environment, we have looked at TCP congestion management techniques. The yellow nodes are UDP sources, whereas the green nodes are TCP sources and sinks. Peer to Peer architecture is used to link one node to the other nodes. Nodes to routers have a 40 Mbps bandwidth and a 10 ms latency. Two routers, R1 and R2, are the blue boxes. At time 1 ms, here left nodes (9 TCP and 1 UDP) begin transmitting data packets to router R1. These packets are sent to R2 by the router of R1 [3]. There are 2 UDP receivers and 11 TCP sinks among the 13 receiving nodes. Congestion occurs at the routers as a result of the source nodes' and R1-R2's lower bandwidths. We made an effort to examine the various algorithms.We conducted our investigation using the ns-3 network simulator, citing source 15. A discrete-event network simulator is the network simulator ns-3 (reference ref 9). It is one of the techniques often utilised by individuals seeking for a novel method of TCP congestion management in an unfamiliar environment.

## 5.2 Window Size Comparison:

The congestion window size can be set to different values depending on the network conditions and the application requirements[4]. In general, a larger congestion window size can lead to higher throughput, but it can also increase the risk of congestion and packet loss.

Here is a comparison of different congestion window sizes for TCP BBR:

1. Default Window Size: The default congestion window size for TCP BBR is 10 packets. This is a relatively conservative value that is designed to avoid congestion and minimize packet loss.

2. Large Window Size: Some applications may benefit from a larger congestion window size, especially if they are transferring large amounts of data over a long period of time. In such cases, the congestion window size can be increased to 100 or even 1000 packets, which can lead to higher throughput and faster data transfer.

3. Small Window Size: In some cases, it may be desirable to use a smaller [5]congestion window size, such as 2 or 4 packets. This can be useful for applications that require low latency and fast response times, such as online gaming or real-time video streaming.

Overall, the choice of congestion window size for TCP BBR depends on the specific requirements of the application and the network conditions. It is important to carefully evaluate the trade-offs between throughput, latency, and packet loss when selecting a congestion window size for TCP BBR

## 5.3 Congestion window of BBR

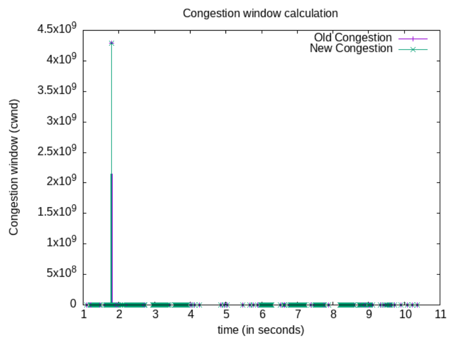

Figure 5.2: Congestion window variation for BBR.

## 5.4 Congestion window of NewReno

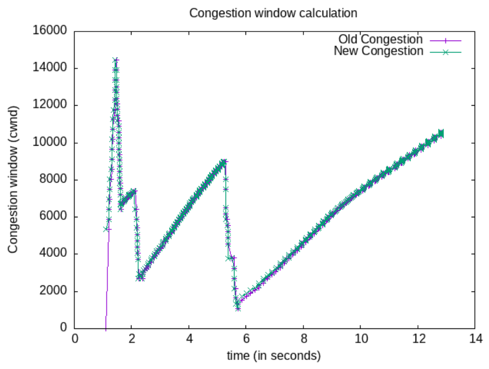

Figure 5.3: Congestion window variation for NewReno.

## 5.5 Congestion window of Cubic

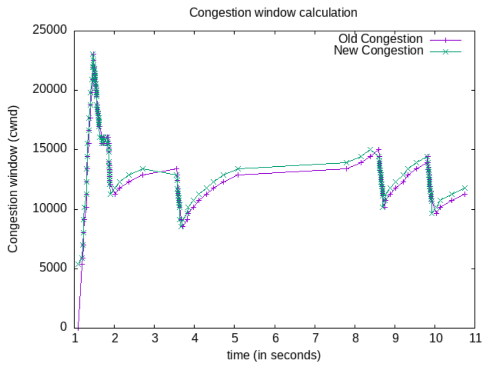

Figure 5.4: Congestion window variation for Cubic.

## 5.6 Comparison of Throughput:

The total quantity of data that is successfully transferred to every network terminal is known as throughput.[11]. It is a significant performance indicator, and the connection ends (sender and receiver) should use the whole link capacity for the optimum throughput.

Throughput = sum of all bits received / connection time

For Data Rate = 40Mbps, bt1 bandwidth = 20 Mbps and bt2 , It performed 0.64% better than New Reno, 0.28% better than variant with wwf parameter 1/3, 0.50% better than variant with wwf parameter 1/3.

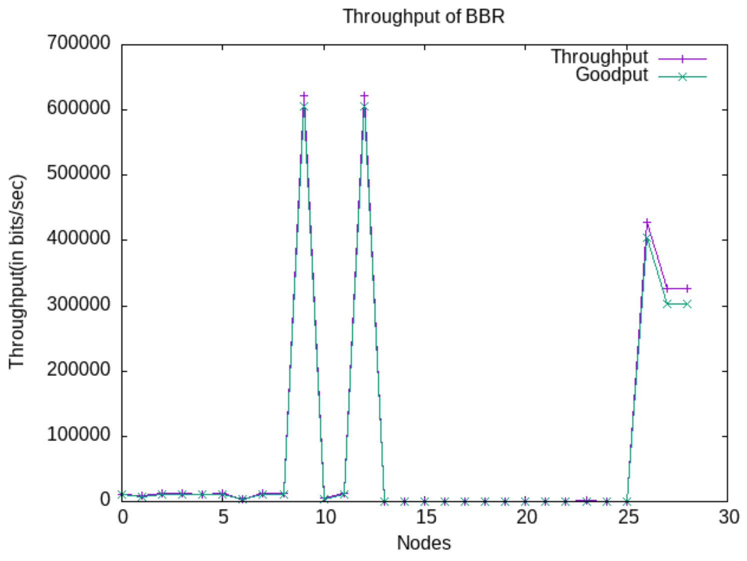

Figure 5.5: Throughput(in bits/second) of TCP BBR for Varying Load.

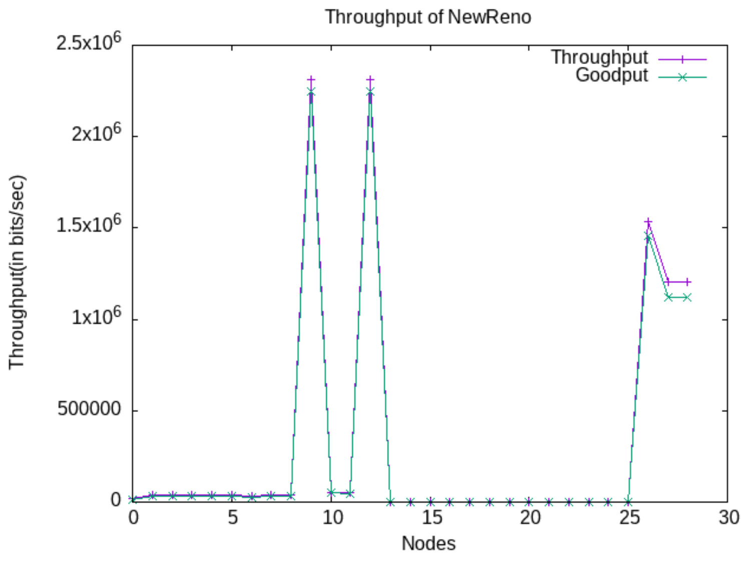

Figure 5.6: Throughput(in bits/second) of TCP NewReno for Nodes.

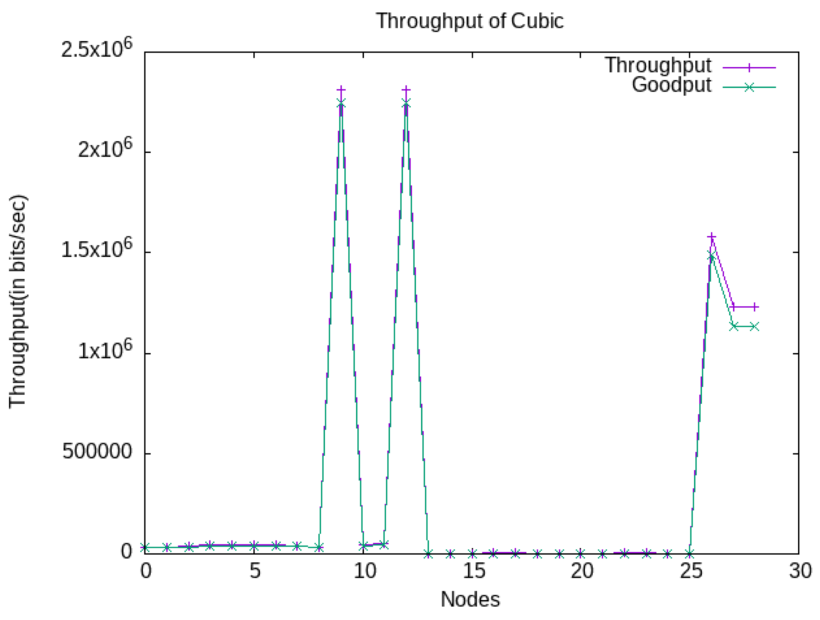

Figure 5.7: Throughput(in bits/second) of TCP Cubic for Nodes.

## 5.7 Comparison of cwnd and RTT for various TCP Algorithms

The following figures represent the plots of Congestion window, and RTT against time for different TCP variants generated from the simulation.

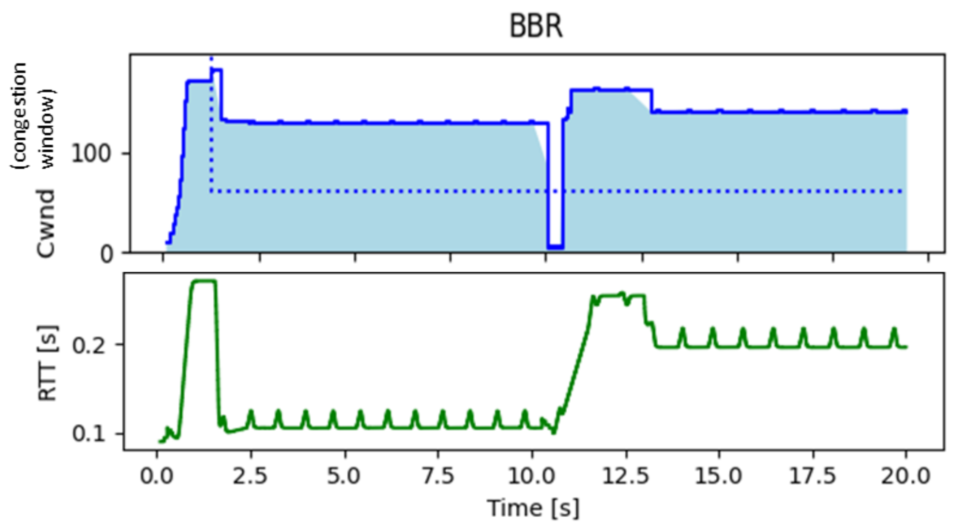

Figure 5.8: Variation of Congestion window, and RTT against time for Bbr.

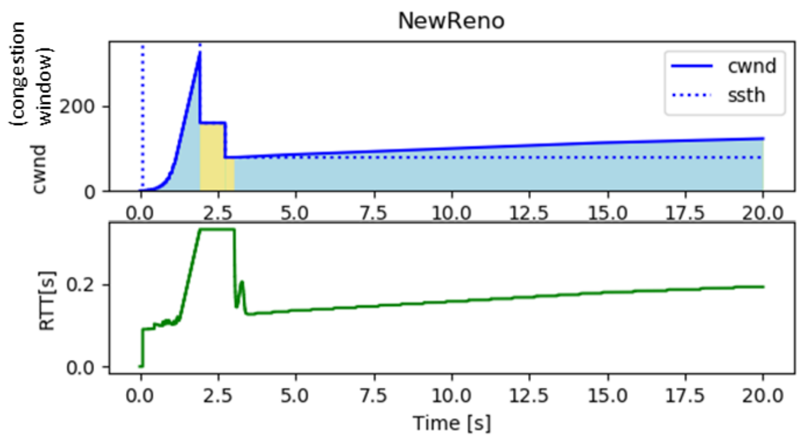

Figure 5.9: Variation of Congestion window, and RTT against time for Newreno.

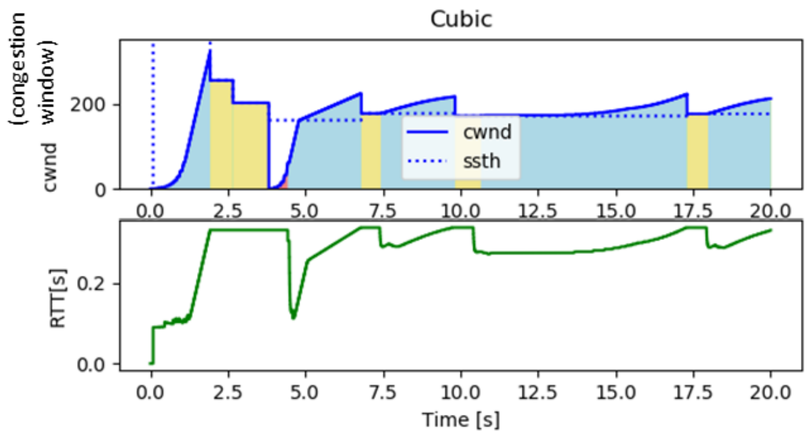

Figure 5.10: Variation of Congestion window, and RTT against time for Cubic.

# Chapter 6

# Conclusion and Future work

With an emphasis on flexibility, portability, repeatability, and automation, we proposed a framework for TCP congestion control techniques. We simulated a number of user-configured flows using Mininet. Without any user input, experiments provide a report with graphs showing 14 metrics.To prove the effectiveness of our method via emulation, we replicated relevant work citeref18. We further expanded recent results in a number of areas and detailed the current state of the art for TCP BBR research. We have shown that the mechanism used to calculate the duration of the Probe RTT phase is faulty, and thus BBR and CUBIC generally do not divide bandwidth equitably. An experimental investigation of the synchronisation process is our last contribution. Two significant issues were found. The fairness of bandwidth sharing will depend on when new flows join existing flows in reference to their Probe RTT phase. The duration it takes for a bandwidth equilibrium to recover is the second issue. According to reference 1, this may last up to 30 seconds, which is problematic for short-lived flows, which characterise the Internet of today.We discovered that this is connected to the synchronisation trigger, which is when the queues are emptied during the Probe RTT phase (reference ref 16). TCP BBR may not perform at its best in networks with high levels of dynamicity and quick changes to these characteristics, despite the fact that it is designed to adapt to changing network circumstances like bandwidth and latency [6].

Future work may focus on developing flexible algorithms that can better handle such changing environments. In future, we would identify potential areas for future research on the TCP BBR algorithm, such as integration with emerging network technologies or investigation of its behavior in other network environments.

# References

[1] Song, Y.-J., Kim, G.-H., Mahmud, I., Seo, W.-K., and Cho, Y.-Z., 2021. "Understanding of bbrv2: Evaluation and comparison with bbrv1 congestion control algorithm". *IEEE Access,* **9**, pp. 37131–37145.

[2] Brakmo, L., and Peterson, L., 1995. "Tcp vegas: end to end congestion avoidance on a global internet". *IEEE Journal on Selected Areas in Communications,* **13**(8), pp. 1465–1480.

[3] Wei, W., Xue, K., Han, J., Xing, Y., Wei, D. S. L., and Hong, P., 2021. "Bbr-based congestion control and packet scheduling for bottleneck fairness considered multipath tcp in heterogeneous wireless networks". *IEEE Transactions on Vehicular Technology,* **70**(1), pp. 914–927.

[4] Poorzare, R., and Waldhorst, O. P., 2023. "Toward the implementation of mptcp over mmwave 5g and beyond: Analysis, challenges, and solutions". *IEEE Access,* **11**, pp. 19534–19566.

[5] Ha, P., Vu, M., Le, T.-A., and Xu, L., 2021. "Tcp bbr in cloud networks: Challenges, analysis, and solutions". In 2021 IEEE 41st International Conference on Distributed Computing Systems (ICDCS), pp. 943–953.

[6] Su, B., Jiang, X., Jin, G., and Ma, A., 2019. "Dvptcp: A delay-driven virtual parallel tcp for high-speed and lossy networks". *IEEE Access,* **7**, pp. 99746–99753.

[7] Wu, Z., Guo, J., and Cui, W., 2021. "Analysis of bbr's non-queuing optimal model". In 2021 IEEE Intl Conf on Parallel  Distributed Processing with Applications, Big Data  Cloud Computing, Sustainable Computing Communications, Social Computing Networking (ISPA/BDCloud/SocialCom/SustainCom), pp. 626–631.

[8] Alrshah, M. A., Al-Maqri, M. A., and Othman, M., 2019. "Elastic-tcp: Flexible congestion control algorithm to adapt for high-bdp networks". *IEEE Systems Journal,* **13**(2), pp. 1336–1346.

[9] Song, Y.-J., Kim, G.-H., Mahmud, I., Seo, W.-K., and Cho, Y.-Z., 2021. "Understanding of bbrv2: Evaluation and comparison with bbrv1 congestion control algorithm". *IEEE Access,* **9**, pp. 37131–37145.

[10] Utsumi, S., and Hasegawa, G., 2021. "Refining calculation algorithm for packet pacing rate of bbr". In 2021 IEEE International Workshop Technical Committee on Communications Quality and Reliability (CQR 2021), pp. 1–6.

[11] Han, J., Xue, K., Xing, Y., Li, J., Wei, W., Wei, D. S. L., and Xue, G., 2021. "Leveraging coupled bbr and adaptive packet scheduling to boost mptcp". *IEEE Transactions on Wireless Communications,* **20**(11), pp. 7555–7567.