**Study of TCP Variants**

**Group 1**

***Gnanesh Patel (MT22030)***

***Hritika Shah (MT22101)***

***Kirti Vashishtha (MT22035)***

**Introduction:**

The TCP (Transmission Control Protocol) protocol is a fundamental component of the IP (Internet Protocol) suite used for online communication. It is a trustworthy, connection oriented protocol that allows for end-to-end data exchange between devices. As TCP ensures guaranteed data delivery which means it also needs to guarantee data delivery at the time when there is congestion in the network or even when there are packets loss, so in order to overcome these types of problems various TCP variants are being introduced to perform well in such specific type of situations. TCP must adapt and change to ensure the best speed and dependability as more people access the Internet and other networks. As an illustration, TCP Reno, the most popular TCP variation, was created to handle congestion control in conventional wired networks. TCP Reno’s performance might be better in certain conditions due to networks expanding to include wireless and rapid connections. Variants, including Tahoe, Reno, NewReno, Vegas, Cubic, Westwood, and BBR, and many others were developed to address these problems and improve TCP’s performance in various network environments. Each TCP variation went on to add one or more congestion-relieving methods. Over TCP Tahoe, which is the fundamental TCP variant, TCP Reno added Fast Recovery. Over TCP Reno, TCP New Reno employs quick retransmission. In contrast to Vegas, which uses a different congestion avoidance algorithm better suited to high-speed networks, Westwood is a congestion avoidance algorithm specifically designed for wireless networks, where packet loss may be caused by factors other than congestion. BBR, a more recent variant of TCP, is designed for high-speed, high-latency networks, which are increasingly widespread. TCP is more flexible and dependable for various applications because it is available in multiple versions that can be utilized for different network topologies.

Here, we have used NS-3 mmwave to study the behaviour to different TCP variants at high frequency range i.e. >= 22GHz where most of the variants do not perform well except TCP BBR

**Tools Used:**

* **NS-3 Simulator mmwave :-** NS-3 is a discrete-event network simulator that can be used to study the behavior of various network protocols, including TCP variants in millimeter-wave (mmWave) networks. To study TCP variants in mmWave networks using NS-3, we can create network topologies, specify various mmWave scenarios, and simulate network traffic to analyze the performance of protocols under different conditions. Here parameters of TCP variants can also be modified, such as the congestion control algorithm or the initial window size, to evaluate their impact on network performance. NS-3 provides several built-in mmWave models, such as the channel model, mobility model, and antenna model, that allows to create realistic mmWave scenarios for their simulations. The mmWave channel model in NS-3 considers factors like path loss, shadowing, and blockage to simulate the effect of signal attenuation in mmWave networks. The mobility model in NS-3 allows researchers to simulate the movement of nodes in the network, which can impact the quality of mmWave links. The antenna model in NS-3 provides various antenna models to simulate the directional transmission characteristics of mmWave networks. Using this, we can gain insights about the strengths and weaknesses of different TCP implementations and design improvements that can enhance the performance and stability of TCP in different mmWave environments.
* **Python3 :-**  Python is an open, high-level, object-oriented, and dynamic programming language that is extensively used. The data from the labelled dataset was obtained and analysed using the Python programming language

**Methodology:**

Using the NS-3 mmwave tool to simulate the network environment, we examined how various TCP changes behave throughout the network simulation. We can employ a method that calls for a number of crucial stages to analyse the behaviour of often occurring TCP changes utilizing ns3. The first step on our computer is to install ns3. The simulation must be built after choosing the network architecture, TCP variation type, and other parameters. A phased array antenna model having threeGpp antennas is used for high-frequency networks. The TCP variation we want to research must then be configured into ns3.After configuring and using the TCP variant, we can run the simulation and get performance information. Metrics such as throughput,congestion window may be included in these statistics. Finally, we must examine the simulation results to comprehend how the TCP variant functions in the high-frequency network. To do this, it could be necessary to evaluate the effects of various network characteristics on TCP performance or to compare the performance of several TCP variations. With the help of this methodology, we can examine the behaviour of TCP variations on high-frequency networks and devices for improving TCP performance in various network setups.

**Different types of TCP variants used:**

* **TCP New Reno:** TCP New Reno is a TCP congestion control algorithm that is an extension of the original TCP Reno congestion control algorithm. The main goal of TCP New Reno is to improve TCP's performance in situations where multiple packets are lost in a single round-trip time (RTT) interval, known as "fast recovery." In TCP New Reno, when a packet is lost, the sender reduces its congestion window to half of its previous value and enters "fast recovery" mode. In this mode, the sender retransmits the lost packet and sends new packets up to the new congestion window size. If another packet is lost during this period, the sender halves the congestion window again and continues fast recovery. Once all the lost packets have been successfully retransmitted, the sender exits fast recovery and enters "congestion avoidance" mode, where it slowly increases the congestion window to find the available network capacity. Overall, TCP New Reno is an improved version of the original TCP Reno algorithm that provides faster recovery from packet losses and better utilization of network capacity in congested environments. It is widely used in modern TCP implementations and has become a standard TCP congestion control algorithm.
* **TCP Vegas:** TCP Reno has been improved by TCP Vegas. It eliminates the problem of requiring enough duplicate acknowledgement to detect a packet shortfall, and it also proposes an improved slow start strategy that prevents the network from being blocked. Since packet losses occur, it identifies congestion. Vegas employed three techniques to enhance throughput while minimizing losses.

1. Slow Start approach with a twist

2. An improved Congestion Avoidance method

3. A re-transmission technology that has been updated.

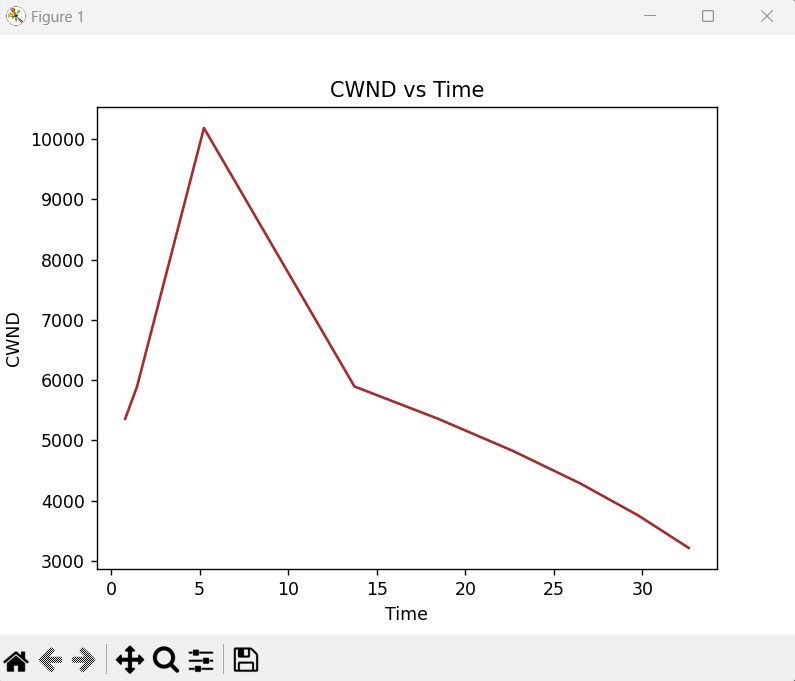
The congestion window in TCP Vegas has been upgraded from the following regulation.

* Expected rate = CWND (t)/Base RTT
* Actual rate = CWND (t)/RTT
* Diff = Expected rate - Actual rate
* Using the value of Diff the CWND is adapted as:
* If Diff < α TCP Vegas increases CWND linearly
* If Diff > β, TCP Vegas decreases the CWND linearly.
* If α < Diff
* **TCP Cubic:** TCP CUBIC (Congestion Control for TCP) is a TCP congestion control algorithm that is designed to improve network throughput in high-speed and long-distance networks. TCP CUBIC uses a cubic function to determine the sender's congestion window size. The congestion window size is increased during the "slow start" phase and "congestion avoidance" phase in a way that is proportional to the cubic function of the time since the last congestion event. This helps TCP CUBIC to achieve a better balance between TCP fairness and high throughput, even in high-speed networks. TCP CUBIC also includes a mechanism for detecting and reacting to congestion. When congestion is detected, the sender reduces its congestion window size based on the estimated congestion window and the current congestion window. This helps to reduce the amount of traffic sent over the network, preventing further congestion and improving network stability. One of the unique features of TCP CUBIC is its ability to adapt to the network conditions quickly. Overall, TCP CUBIC is a congestion control algorithm that is optimized for high-speed and long-distance networks.

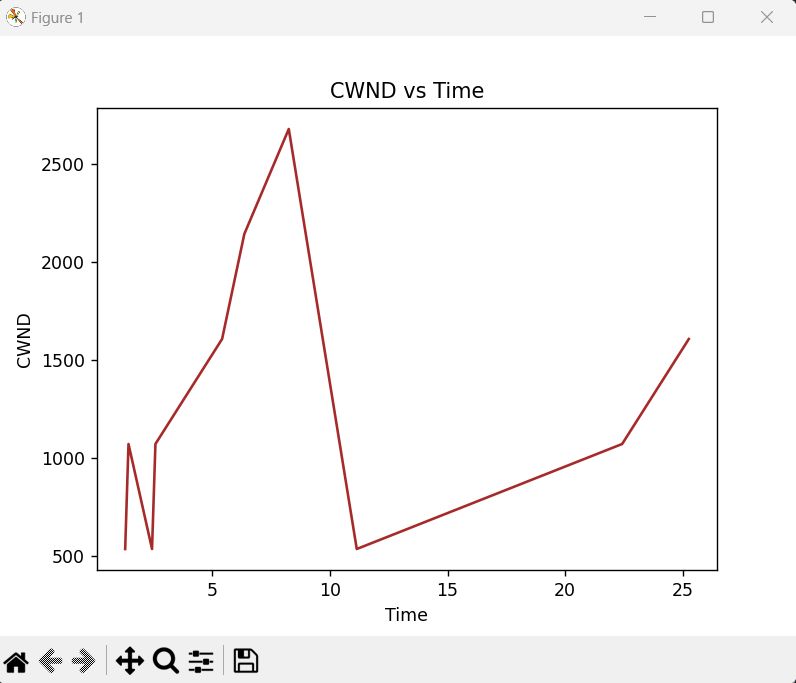
**RESULTS:**

**CONGESTION WINDOW VS TIME**

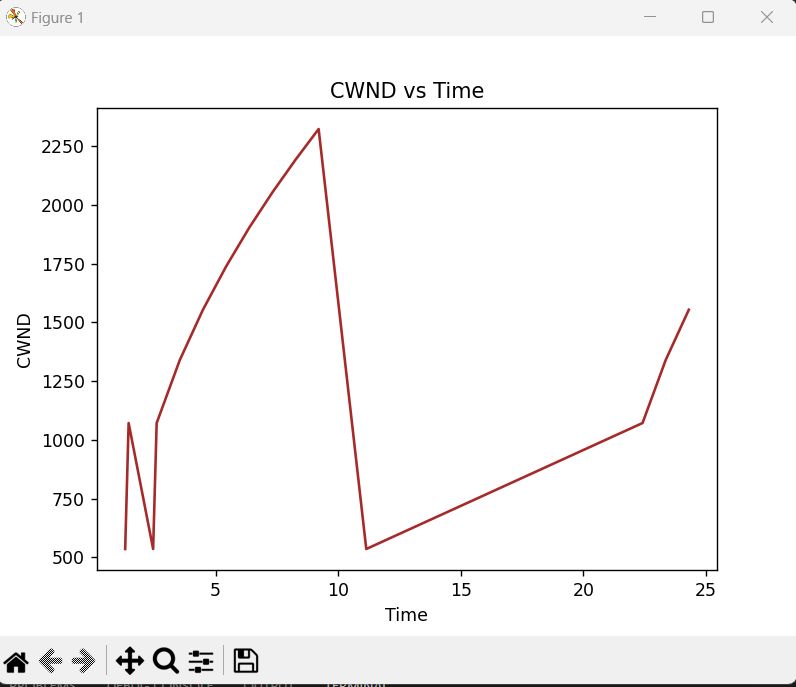
**1.TCP VEGAS**

****

**2.TCP CUBIC**

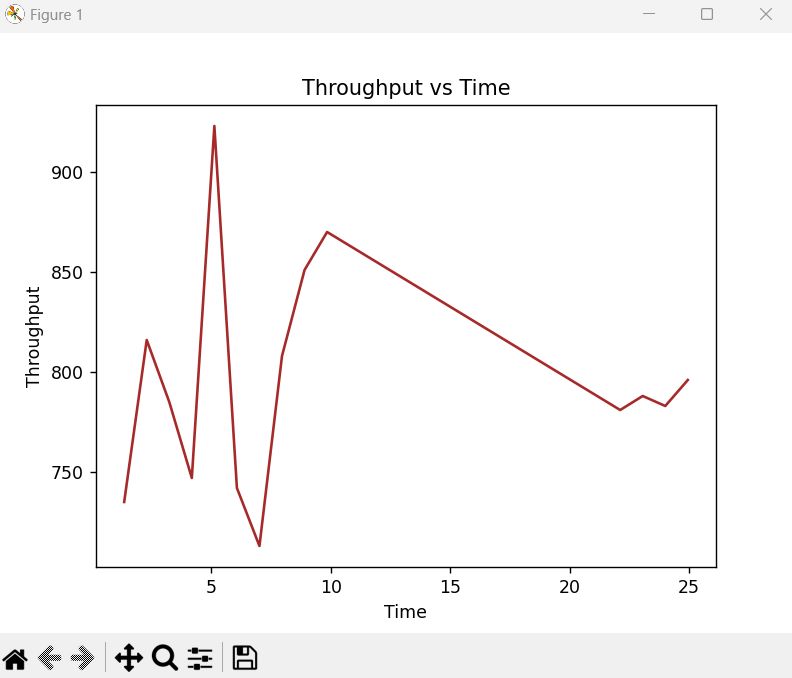
****

**3.TCP NEW RENO**

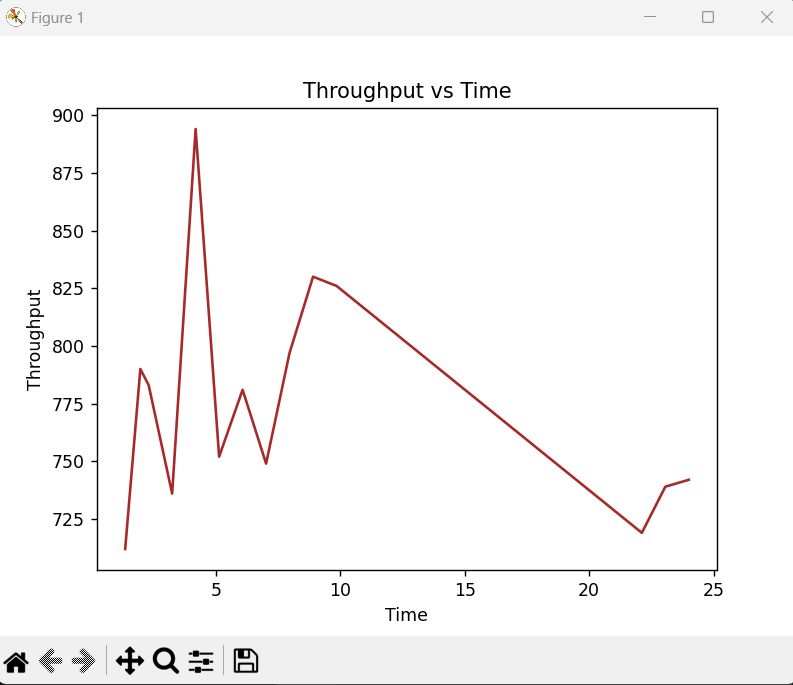
****

**THROUGHPUT VS TIME FOR ONE OBSTACLE BUILDING:**

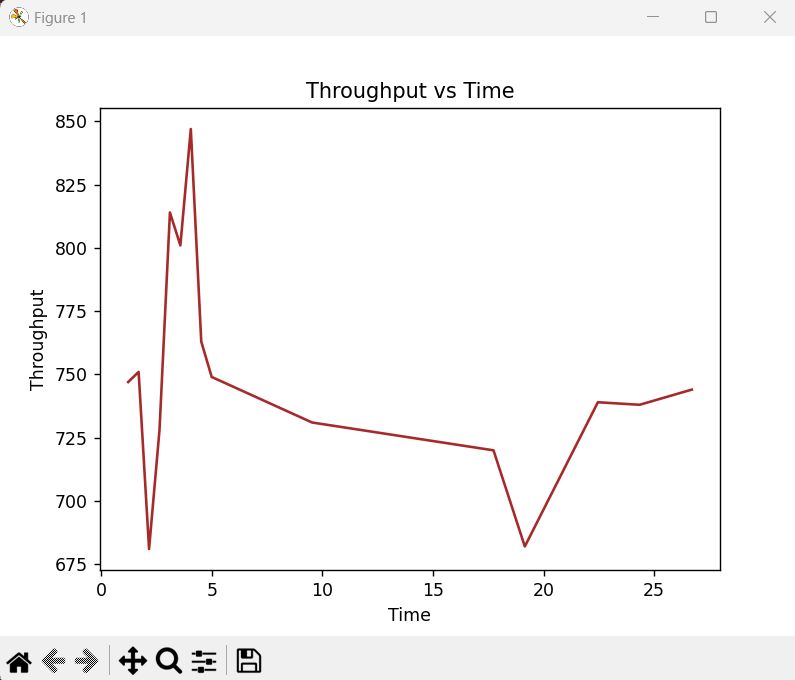
1. **TCP CUBIC:**



1. **TCP NEWRENO:**

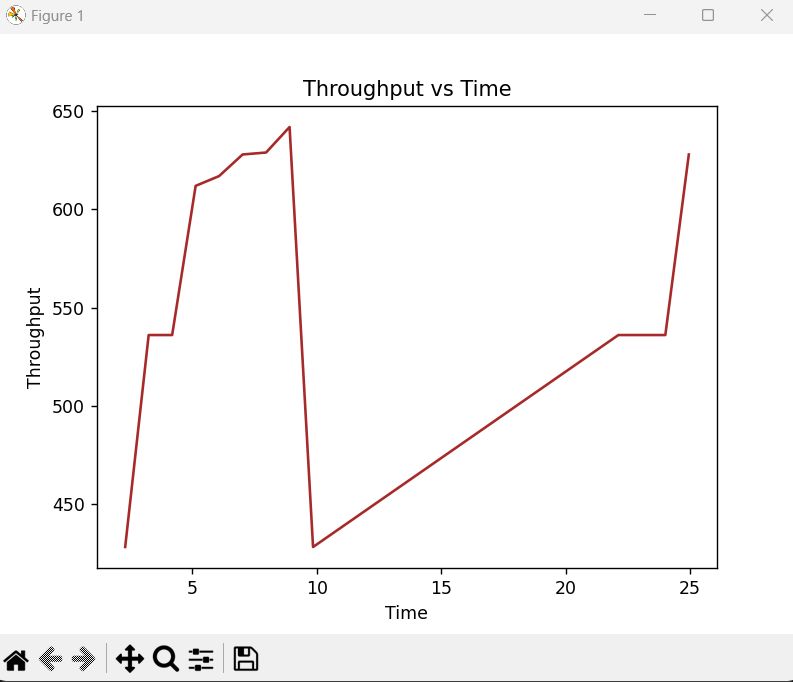


1. **TCP VEGAS:**

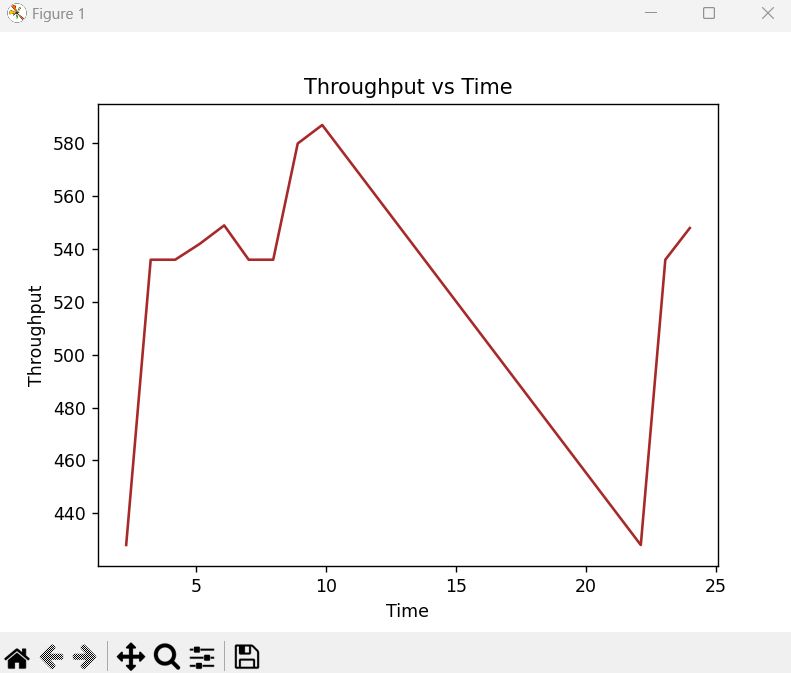


**THROUGHPUT VS TIME FOR MULTIPLE BUILDINGS,TREES AND HUMAN BLOACKADGE :**

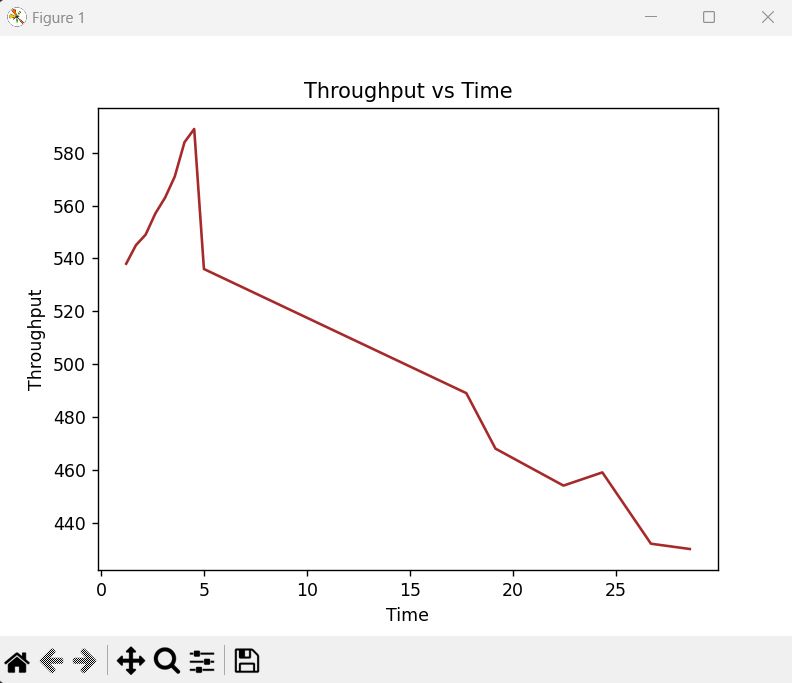
**1.TCP CUBIC:**

****

**2.TCP NEW RENO:**

****

**3.TCP VEGAS:**



**CONCLUSION:**

TCP congestion control algorithms, such as TCP Cubic, New Reno, and Vegas, can struggle to perform well in high-frequency networks due to the variable and unpredictable nature of the channel conditions. These algorithms rely heavily on packet loss as a congestion signal, which can lead to inefficient use of network resources and increased latency.

The graphs of TCP Vegas cwnd over time in a high-frequency network might show more frequent and rapid changes in cwnd compared to a lower frequency network. This can be due to the higher variability in channel conditions in high-frequency networks, which can lead to more frequent and sudden changes in RTT.

Studies have shown that TCP BBR can perform well in high-frequency networks by maintaining a high throughput and low latency, even in the presence of congestion. This is because TCP BBR is able to detect the available network capacity more accurately and adjust its sending rate accordingly, without relying on packet loss as a congestion signal.