Comparison and performance evaluation of TCP congestion control protocols.

Preetkumar Patel
Student Id: 1107843
Lakehead University

Alizar Marchawala
Student Id: 1103258
Lakehead University

Simranjeet Singh
Student Id: 1093604
Lakehead University

Ankit Kundal
Student Id: 1116536
Lakehead University

***Abstract*—The only reason why we can utilize the internet in such an easy manner is the use of various congestion control algorithms which deal with reducing the traffic of the network. TCP congestion control has been designed to ensure the reliability of the Internet, along with equal and effective network bandwidth allocation. Numerous congestion control algorithms have been proposed during the last decade. We are focusing on few of them mainly Reno, Vegas and Tahoe and comparing them from various viewpoints. This paper also discusses about three main phases of the congestion control techniques. These phases includes slow start phase, congestion avoidance phase and fast recovery phase. Its shows that Vegas performed better as compared to other implemented algorithms in terms of performance. To reproduce these techniques, NS2 has been utilized.**

Keywords–TCP, Vegas, Reno, Tahoe, Congestion window, slow start, congestion avoidance, fast retransmit, fast recovery.

I. INTRODUCTION

Congestion is a condition that occurs in network layer when the message traffic is so heavy it reduces the response time for the network. Due to increase in delay, the performance also decreases. Moreover, congestion in the network occurs because load on the network is greater than the capacity of the network. Congestion control is the mechanism and methods which is used to keep the load below the capacity of the network. Congestion can be prevented before it has happened or it can be removed after it has happened. According to this, it can be classified into two categories: open loop congestion control and closed loop congestion control. Basically, open loop congestion control is prevention of congestion control whereas closed loop congestion control means removal or reduction of congestion. Open loop congestion control includes different policies which are needed in order to prevent it before it happens. These policies are retransmission policy, window policy, acknowledgement policy, discarding policy and admission policy. In closed loop congestion control, it tries to reduce or alleviate the congestion which has already happened. In this type, different mechanisms have been used such backpressure, choke packet, implicit signaling and explicit signaling.

All the strategies mention above are discussed below:

Open loop Congestion controls: In this, congestion control is handled by source or the destination. This includes following policies:

- Retransmission Policy: It can be sometimes unavoidable. If the sender feels that the packet which was sent is corrupted or lost, it needs to be retransmitted again. In general, continuous retransmission may increase congestion. In order to prevent congestion, the retransmission timers must be configured to prevent congestion and to maximize performance.
- Window policy: The type of window on the sender side can also impact congestion. Several packets in the Go-back-n window are resentful, although some packets can be received successfully from the receiver side. This duplication could increase and worsening congestion in the network. Thus, the Selective Repeat Window should be followed because it sends out a particular packet that might have been lost.
- Acknowledgement policy: Because the acknowledgment is also part of the network load, congestion may also be caused by the acknowledgment policy enforced by the receiver. Several methods may be used to avoid acknowledgment-related congestion. The receiver can send a N packet acknowledgement rather than a single packet acknowledgement. The receiver will give an acknowledgement only if the receiver has to give an acknowledgement.
- Discarding policy: A router should avoid congestion while at the same time partially discarding the less secure package and still preserving consistency of message. Routers will discard less sensitive packets when transmitting audio files to avoid congestion and also preserve the quality of the audio file.
- Admission policy: A mechanism to prevent congestion should be used by admission policy. Flow switches should first check the need for a network flow resource before transferring it further. If there is a chance of congestion or congestion in the network, the router should refuse to establish a virtual network connection to prevent further congestion.

Closed loop congestion control: Here, different mechanisms have been used by different protocols. It includes following:

- Backpressure:Backpressure is a method in which the congested node stops receiving packets from the upstream node. This can also cause the upstream node or nodes to become congested and to refuse to receive data from the above nodes. Backpressure is a node-to-node congestion control methodology that propagates in the opposite direction of data flow. This approach can be applied to a virtual circuit where each node has information about its upstream node.

- Choke packet method: Choke packet method is applicable to both virtual networks and datagram subnets. It is a packet sent by a node to the source to inform the node of congestion. Each router monitors its resources and uses at each of its output lines.Whenever resource utilisation exceeds the threshold value set by the administrator, the router starts sending a choke packet straight to the source giving it feedback to ease congestion. The intermediate nodes by which the packets travelled are not warned of congestion.
- Implicit Signaling: There is no communication between the congested nodes and the source in implicit signalling. The source assumes that there is congestion on the network. For instance, if the sender sends a few packets and there is no acknowledgement for a while, it assumes that there is congestion.
- Explicit Signalling: If a node encounters congestion, it may directly send a packet to the source or destination to warn about congestion while explicitly signalling it. The difference between choke packets and explicit signalling is that the signal should be included in packets that take data rather than having different packets, as seen in the case of choke packet technique. There are two types of explicit signalling: forward or backward.
  - Forward Signalling: In this type of signal, the signal will be sent in the direction of congestion. Destination is being warned of congestion. In this case the receiver adopts strategies to reduce more congestion.
  - Backward Signalling: In this type of signal, the signal will be sent in the opposite direction to the congestion. The source is advised of congestion and needs to slow down.

TCP is a protocol for transport layers used by applications which require guaranteed delivery. It is a protocol with a sliding window that provides:

- It provides Intermediate level communication between an application system and the Internet Protocol.
- It is also used for handling of retransmissions and timeouts.
- It also provides full duplex virtual communication between endpoints where each end point is specified by TCP port number and IP address.

TCP is widely used internet protocol. TCP's key advantage is that it's ability to tackle congestion. When the rate of packet send is greater than the rate of received then congestion occurs. Here, it provides connection oriented network. TCP provides numerous variants for dealing with congestion. These are TCP Vegas, TCP Tahoe, TCP Reno, TCP RBP, Full TCP, TCP New Reno, TCP SACK, TCP FACK, TCP Asym, TCP RBP, Full TCP, and CUBIC. Three important algorithm used in this variants are slow start, congestion avoidance and Fast Retransmit. New Reno has additional algorithm which is called fast recovery. Furthermore, in the slow start phase, the sender starts the transmission with very slow rate but it increases the transmission rate rapidly until it reaches the threshold. When it reaches the threshold, the data rate is reduced to avoid congestion. Finally, when congestion is detected it goes back to slow start or congestion avoidances depending upon how the congestion is detected.

Slow start: In this phase, cwnd starts with one maximum segment size (MSS). The size of the window increases one MSS every time the acknowledgement is received by the sender. This is shown in the Fig 1. It shows that it starts slowly but increases exponentially. Assumption over here is rwnd is much higher than cwnd and each segment is acknowledged individually.

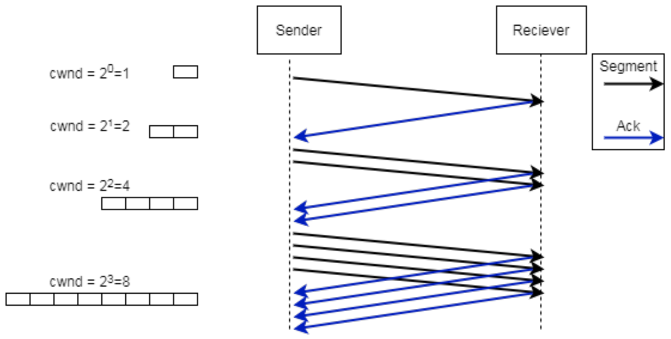


Fig. 1. Slow start phase

Congestion avoidance: To avoid congestion before it happens one must slow down the growth from exponential to linear. In slow start algorithm, the size of congestion window increased exponentially but here the growth increases linearly. So, when the size of congestion window in slow start reaches the slow start threshold then it enters the additive phase which is called as congestion avoidance phase. Moreover, after each round the size of congestion window is increased by one as shown in the Fig 2. In this phase, congestion window increases by one until the congestion is detected.

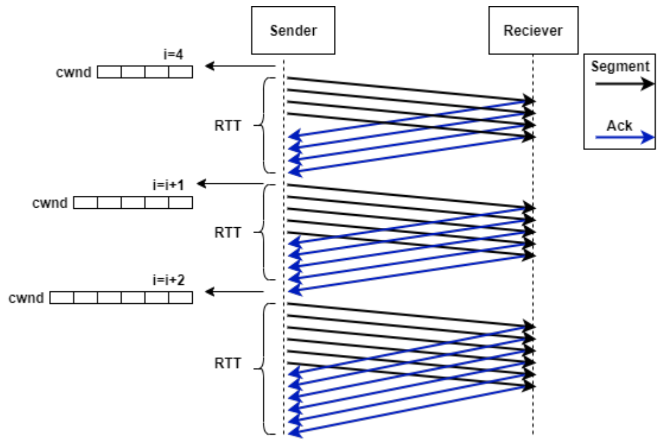


Fig. 2. Congestion avoidance phase

Basically, there are two signs of congestion:

- Timeout: This case is also called severe congestion. If time out occurs, there is strong possibility of getting congestion. It is considered as severe because segment has probably been dropped in the network and there is no news of the segment. So in this case TCP does following things: First it set the threshold to half of congestion window and set congestion window to one. Secondly, it starts the slow start phase again.
- 3 Duplicate Acknowledgement: This is also called weak congestion. In this case, only some segments have been lost but there are some segments which reached safely because three duplicate acknowledgements are received. This is called fast transmission and fast recovery phase. In this case, TCP does following things: Firstly, it set the threshold to half of congestion window and assigns value previous threshold to congestion window. Secondly, it starts the congestion avoidance phase.

According to all this phases, in the proposed method, we have implemented Tahoe, Vegas, Reno and New Reno. All the three algorithms are compared according to various view points. In this paper, we have used NS-2 (Network Simulator) for the implementation of congestion control protocols.

II. LITERATURE REVIEW

In the first paper [2] they have explained different TCP variants like TCP Tahoe, TCP Reno, TCP New Reno, TCP Sack, TCP Asymmetry, TCP Rate Based Pacing and Slow Start. A brief knowledge about congestion control is also mentioned in this paper. Although we have many congestion control algorithms that are used in reducing the network traffic but choosing the best is a tough part.The TCP sender utilizes a congestion window (cwnd) in directing its transmission rate dependent on the International Journal of Advanced Research in Computer Engineering and Technology input it gets from the system. The clog window is the TCP sender's gauge of how a lot of information can outstand in the system bundles without being lost. In the wake of introducing cwnd to a couple of fragments, the TCP senders permitted to expand the congestion window either as indicated by a moderate beginning calculation, that is, by one portion for every approaching affirmation (ACK), or as per congestion shirking, at a pace of one section in a full circle time. The moderate beginning limit (ssthresh) is utilized to decide if to utilize moderate beginning or congestion avoidance algorithm. They used [1] fast retransmit algorithm in TCP Tahoe in which the sender conclude that the packet was lost when it receive a small number of acknowledgements and retransmit it without waiting for the expiry of retransmission time. The duplicate acknowledgment threshold is fixed at three.Notwithstanding these upgrades Tahoe additionally incorporates Fast Retransmit, better RTT change estimation, and an exponential retransmit back-off timer.

The next paper [1] focussed on TCP RENO where the congestion control window size is W which gets decreased by 1/W each time an Acknowledgement is received. However, when a lost packet is detected, the size of that window gets decreased as well. The decrease in window size depends upon the detection of a duplicate packet of Timeout. In their model the congestion avoidance behavior of TCP in terms of "rounds." A round begins with transmission of W packets, where W is the present size of the TCP congestion window. When all packets falling inside the congestion window have been sent, no different packets are sent until the first ACK is received for one of these W packets. Here, duration of a round is equivalent to the RTT and is thought to be free of the window size. Their idea of rounds is like the idea of "mini-cycles". They have additionally expected that the time expected to send all the packets in a window is smaller than the RTT. They also considered "b" as the number of packets. If W packets are sent in the first round and are all received and acknowledged correctly, then W/b acknowledgments will be received. Since every acknowledgment increases the window size by l/W, the window size at the beginning of the second round is then W' = W + 1 lb. That is, during congestion avoidance and without loss, the window size increments linearly in time, with a slope of lib packets per RTT.In their model[1], they detect packet loss into two ways. First with "Triple Duplicate Acknowledgements" which means 4 acknowledgements with the same sequence number. Secondly with Timeouts.

In one of papers, they have [3] explained about different phases of congestion control methods and also how TCP works over a heterogeneous network. Also, they have provided and compared the throughput of all the different congestion control methods and how is there a performance. All TCP variations are considered as the characteristics and attributes of wired systems, which are not reliant on the lower layers. In case of heterogeneous systems, the congestion control of TCP ends up being deficient with regards to execution. TCP additionally experiences terrible showing in high data transmission links too. Another difficulty to the TCP variations is its powerlessness to completely alter itself to its restricted assets and its lacking capacity to perceive congested or irregular lost packets.

III. PROPOSED METHOD

Tahoe: The two main phases of Tahoe is slow start and congestion avoidance. The FSM diagram of Tahoe is shown in the Fig 3. The problem with Tahoe is that a packet loss can be observed within a full timeout, and it does also take much longer due to coarse grain timeout in most implementations. Also since it doesn't transmit immediate ACK's, it sends cumulative acknowledgements, therefore it follows a 'go back n' approach. Hence, every time a packet is lost it waits for a timeout and the pipeline is emptied. This may have major cost in high-bandwidth delay product links.

Reno: This Reno retains Tahoe's basic concept, such as slow start and retransmit timer. It does however add some knowledge to it so that missing packets are detected sooner

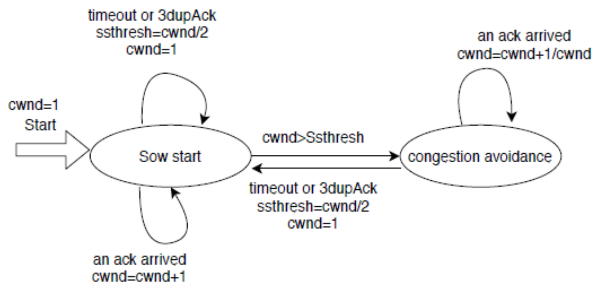


Fig. 3. FSM diagram of Tahoe

and the pipeline is not drained every time a packet is missing. The FSM diagram of the Reno is shown in the Fig 4.

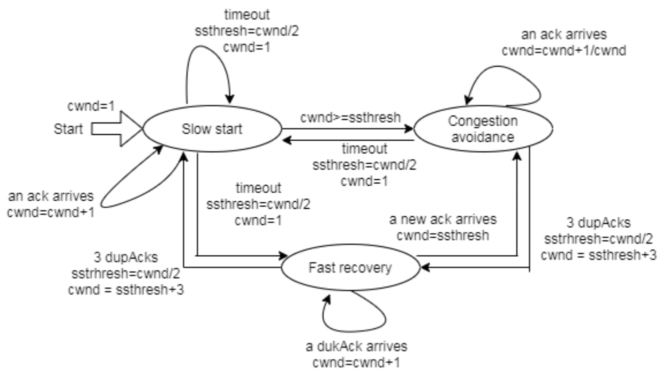


Fig. 4. FSM diagram of Reno

Algorithm of Reno:

- Here, window size continuous to increase until it packet loss occurs.
- It performs the process in two phases:
  - Slow start phase
  - Congestion avoidance phase
  - Fast recovery/transmission
- The slow start method is used at the beginning of the transmission. Upon receiving each response Acknowledgement (ACK) segment, the general formula is used to obtain a Congestion Window during a slow start which is given by :
  - cwnd = cwnd+1 (if cwnd<ssthresh)
- From a slow start, when the congestion window reaches slow start threshold (ssthresh), the sender moves into congestion avoidance. The equation for this is:
  - cwnd = cwnd +1/cwnd (if cwnd>=ssthresh)
- When packet loss is detected by retransmission timeout expiration, cwnd and ssthresh are modified as:
  - cnwd =1 and ssthresh=cwnd/2
- On the other hand, when TCP detects packet loss by a fast retransmit method, it updates the values as:
  - ssthresh=cwnd/2 and cwnd=ssthresh
- Next, the fast retransmission method sends out what seems to be missing part. The fast method of recovery governs the transmission of a new data until the non-duplicate ACK comes in. The flow diagram of reno is shown in the Fig 5.

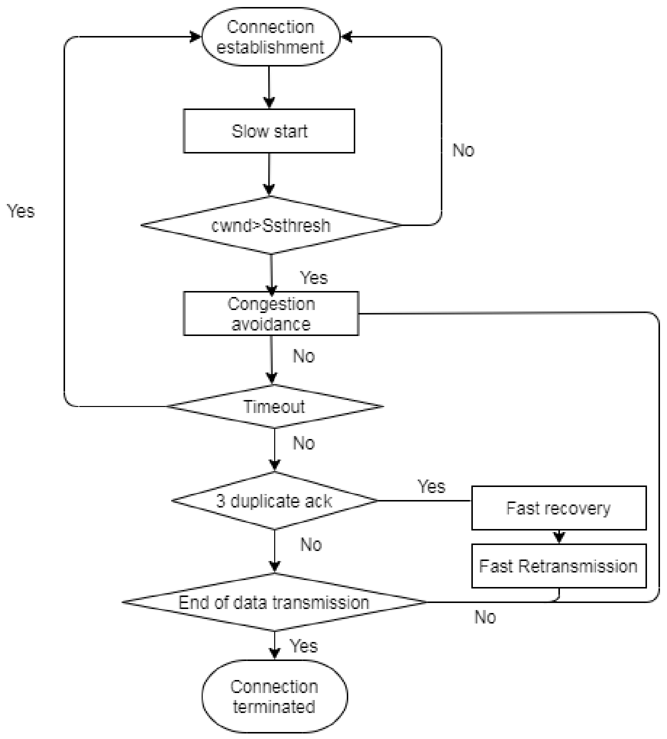


Fig. 5. Flow diagram of TCP Reno

The Fig 6 shows the congestion window verse RTT of TCP Reno. It can be seen from the graph that initially in slow start phase it increses the size of congestion window exponentially whereas in congestion avoidance phase it increases linearly.

Disadvantages of Reno: TCP New Reno requires one RTT to detect the loss of any packet. To determine which segment is missing, Ack must be obtained from the previous transmitted segment.

Vegas: Vegas is a TCP implementation that is a modification of Reno. As a sign of congestion it does not actually depend on the loss of the packets. It detects congestion before the loss of the packet occurs. It overcomes the problem of needing more duplicate ACKs to detect a packet loss, and suggests a modified slow start algorithm that prevents the network from congesting. The three major changes made are:

- A new retransmission mechanism: Here, for lost packets, a new retransmission mechanism is developed. The new system for retransmission in Vegas extends over RENO's retransmission system. This maintains track by calculation of RTT estimates.
- Modification in congestion avoidance phase: It keeps track of when each segment was sent and also calculates the RTT estimate by keeping track of how long it takes

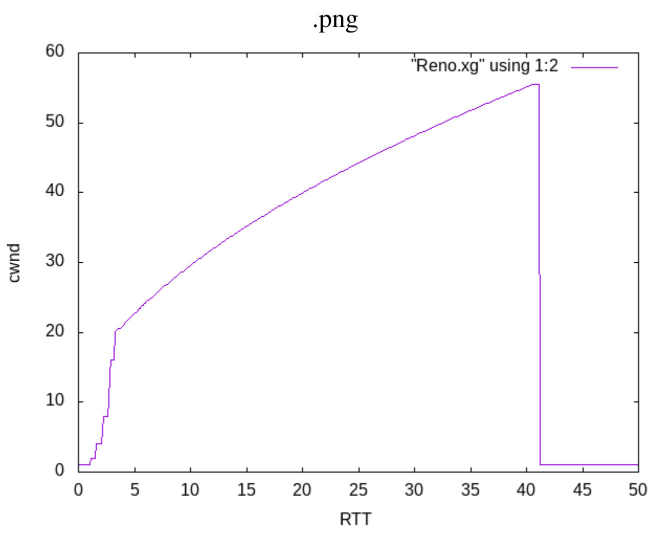
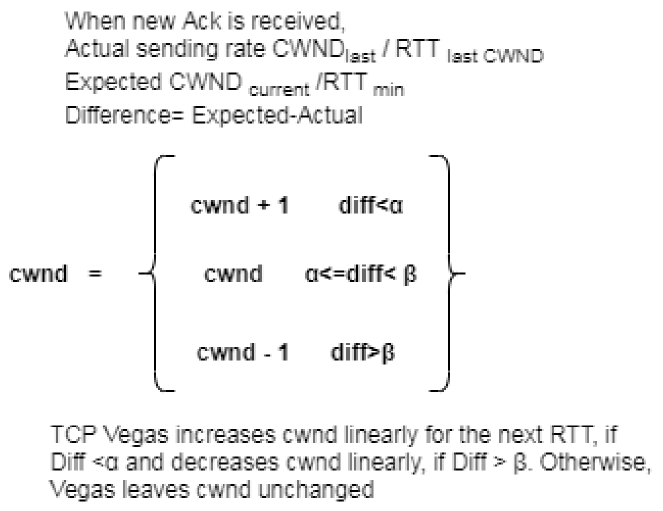


Fig. 6. cwnd vs RTT diagram of Reno

for the acknowledgement to be returned. TCP Vegas in its behavior during congestion avoidance varies from all other implementations. It defines congestion by a decrease of the sending rate compared to the expected rate as shown in the Fig 7.



When new Ack is received,
Actual sending rate $CWND_{last}$ / $RTT_{last\ CWND}$
Expected $CWND_{current}$ / RTT_{min}
Difference= Expected-Actual

$$cwnd = \begin{cases} cwnd + 1 & diff < \alpha \\ cwnd & \alpha <= diff < \beta \\ cwnd - 1 & diff > \beta \end{cases}$$

TCP Vegas increases cwnd linearly for the next RTT, if Diff <α and decreases cwnd linearly, if Diff > β. Otherwise, Vegas leaves cwnd unchanged

Fig. 7. Congestion avoidance in Vegas

- Modification in slow start phase: The purpose for this change is that when a connection begins first it has no idea of the usable bandwidth and it is likely that it may increase over shoots the bandwidth by a significant amount during exponential increases and therefore causes congestion. To avoid this, Vegas only increases every other RTT exponentially, between calculating the actual sending throughput to the expected one and exiting slow start when the difference reaches a certain threshold and entering the congestion avoidance process.

The Fig 8 shows congestion window vs RTT of Vegas. It is seen from the graph that it increases congestion window exponentially every other RTT. In congestion avoidance phase it increases linearly as per the equations.

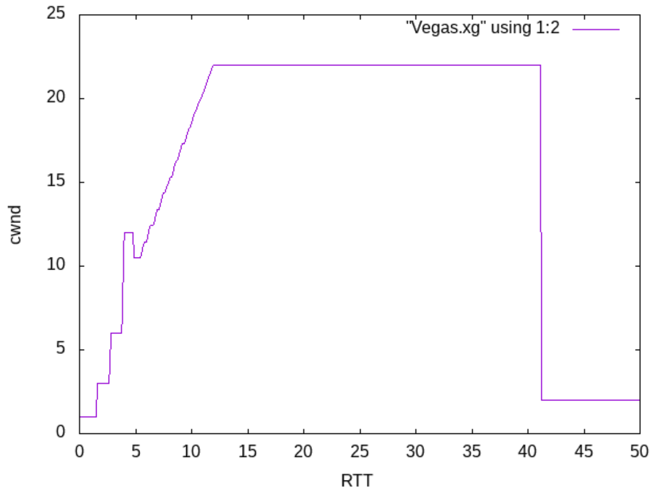


Fig. 8. cwnd vs RTT diagram of TCP Vegas

Disadvantages: In the case of incorrect RTT estimation, TCP-VEGAS efficiency decreases as it is dependent on the RTT value. The efficiency of TCP VEGAS also decreases when buffers decrease at routers.

IV. Experimental Analysis

TCP variants will be efficient on the basis of the parameters to be taken into account. The following table shows the contrast between TCP Tahoe,TCP Reno, and TCP Vegas:

Parameters	Tahoe	Reno	Vegas
Slow start	Yes	Yes	Modified version
Congestion avoidance	Yes	Yes	Modified Version
Fast re-transmit	Yes	Yes	Yes
Fast recovery	No	Yes	Yes
Re-transmit	Simple	Simple	New mechanism
Congestion control	Simple	Simple	New mechanism
Packet loss	Single	Single packet loss	Multiple

TABLE I
"Comparison based on different parameters"

Network topology: The model that we have used is shown in the Fig 9. This network topology consists of two source (N1 and N5), two intermediate nodes (N2 and N3), and two destination nodes (N4 and N6) and inter connecting links. Each link in the topology is full duplex and has a bandwidth of 10 Mbps and default delay of 10 ms. For analysis of the performance of TCP protocols is evaluated in presence of constant bit rate (CBR rate). Add a source of CBR at N2 and a sink at N3, and then add a single stream of TCP from N1 to N4 sink. Analyze the TCP stream's throughput as a function of the CBR flow bandwidth used. In this experiment, the bandwidth of the CBR flow is the parameter that needs to vary. Here, CBR is increased linearly from 1Mbps to 10Mbps to induce congestion and to observe the behavior of the tcp protocols.

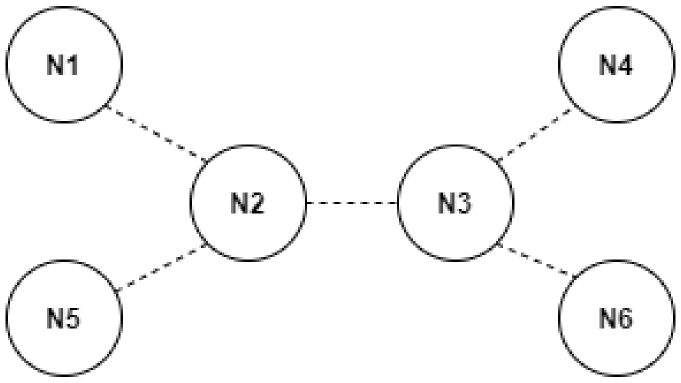


Fig. 9. Network Topology

Throughput: The TCP packets that were successfully received were observed at TCP sink. The total numbers of bytes received at that node were determined until the last packet received. The result is shown in the figure. As the bandwidth of CBR increases, Tahoe's throughput decreases as a result of packet drops and it reaches the slow start stage and reduces the congestion window to 1 whose value does not increase unless the TCP receives ACK's for all the packets sent, and therefore it will stay in the slow start for a longer time.TCP Reno enters and exists a fast recovery mode for each packet in case of multiple drops in a single window, which reduces its overall throughput.

It is seen in the Fig 10 that the initial throughput values of TCP Vegas are lower than the other variant's throughput value, because Vegas first calculates the base RTT which helps it figure out how much bandwidth is available to it and predicts congestion well in advance, thus showing better performance.

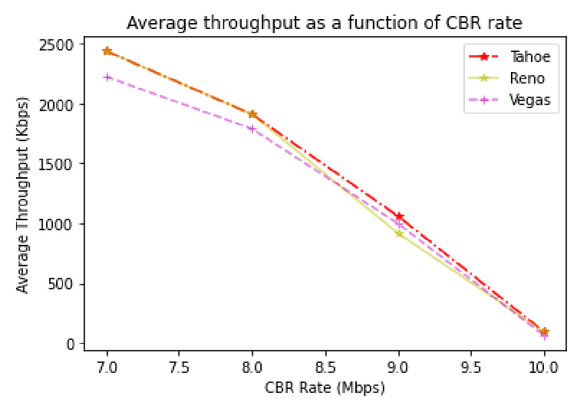


Fig. 10. Throughput vs Bandwidth

V. CONCLUSION

This paper aimed at the use of NS-2 to analyze and compare various TCP variants by examining the throughput. In this research, all the different phases of TCP Tahoe, TCP Reno and TCP Vegas are discussed in detail. Moreover, all the working of the TCP protocols for controlling the congestion is analyzed. All in all, it can be concluded that TCP Vegas performs best in terms of throughput for most of the time, particularly under high congestion conditions in the network.

REFERENCES

[1] Padhye, Jitendra Firoiu, Victor Towsley, Donald Kurose, James. (2000). Modeling TCP Reno performance: a simple model and its empirical validation. IEEE/ACM Transactions on Networking, 8(2):133-145. Networking, IEEE/ACM Transactions on. 8. 133-145. 10.1109/90.842137.

[2] Kamboj, Ravi Singh, Gurpreet. (2015). VARIOUS TCP OPTIONS FOR CONGESTION EVASION. 4. 1534-1539.

[3] Abed, Dr.Ghassan Ismail, Mahamod Jumari, Kasmiran. (2012). Exploration and evaluation of traditional TCP congestion control techniques. Journal of King Saud University - Computer and Information Sciences. 24. 145–155. 10.1016/j.jksuci.2012.03.002.

[4] Awdeh, Ra'ed. (2008). Compatibility of TCP Reno and TCP Vegas in wireless ad hoc networks. Communications, IET. 1. 1187 - 1194. 10.1049/iet-com:20060592.