Congestion Control by using Multipath TCP for Smart Devices

Ms. Rajalaxmi Alatgi
Department of Computer Networks Engineering
VTU, Belagavi, India

Dr. S. L. Deshpande
Department of Computer Networks Engineering
VTU, Belagavi, India

Abstract

In the last few years, the nature of traffic over the network has increased adequately, in which an extensive amount of traffic is contributed by the mobile applications, web based multimedia, computation offloading like the cloud computing and the Internet of Things (IoT) and so on. There are three main requirements for the end-to-end traffic over the Internet, such as mobility of devices, the ability to upgrade through multipath end-to-end communication, and to avoid data loss when congestion arises, are not supported by the widely used transmission control protocol (TCP). TCP is a standard that defines how to build and maintain a network communication through which applications will exchange data over the network. Devices are often connected by multiple paths, but the TCP restricts the communications to a single path per transport connection. Resource usage within the network will be more productive when these paths will be used concurrently. This should boost the user experience through improved flexibility to network failures and achieve higher throughput. Therefore, MultiPath TCP (MPTCP) has been developed to address the TCP limitations. The purpose of MPTCP protocol is to make use of path diversity, so that it offers a better network connectivity, increasing flexibility to failures, enhancing load balance between paths, when more than one is available, and to enable multi-homing support without the need to modify the currently operating devices over the network. In this experiment, a network of 10 nodes is used to show the efficiency of MPTCP by comparing the results of the packet drop percentage, the packet delivered percentage and throughput with TCP for FTP communication between the end hosts. The comparison of the packet drop percentage, the packet delivered percentage and throughput between MPTCP and TCP is done using the network simulator2 and the results are analyzed using Xgraph.

Keywords: Transmission control protocol (TCP), Multipath (MPTCP), Congestion, packet drop percentage, packet delivered percentage, throughput of the system

I. INTRODUCTION

In the past years the TCP has been the basic element of Internet Protocol stack and is the most dependable and secure transmission protocol for data transmission. The TCP communicates using a single path between the source and destination. As the basic model of TCP realizes the importance of data flow control and congestion but it does not satisfy the real-time transmission in situations of critical congestion. Reliable transmission has become one of the most important features in mobile devices with pervasive and demanding applications running on these devices.

Transmission Control Protocol (TCP - RFC 793) is considered as a reliable protocol. TCP is responsible of separating the message into TCP Segments and reassembling them at the receiver side. It might not be sure that the information coming to at the receiver host is in the same order from the sending side, due to the issues in network data flows to the destination. TCP guarantees a reliable delivery by resending anything that gets lost while travelling across the system.

Recently the Internet Engineering Task Force (IETF) has created a work group to build up a standard of a multipath protocol at the transport layer that can be effectively deployable. Having this in thought an augmentation for the TCP protocol has been proposed, the Multi Path TCP (MPTCP) where each connection between two hosts can indeed use multiple parallel paths using congestion detection methods to decide the authentic path to be considered .As MPTCP can possibly enhance the throughput, reliability and adaptability in communications over the network. In reality, it is an extension of TCP and has in reverse compatibility makes it easier to be used on the Web than the other past proposals.

Multipath TCP provides the ability to simultaneously utilize multiple paths between peers. This experiment presents an algorithm that prevents data loss and congestion occurrence in TCP using MPTCP. The protocol offers the same type of service to applications as TCP (i.e., reliable byte stream), and it provides the components to establish and use multiple TCP flows across potentially disjoint paths.

Often endpoints are connected by multiple paths, but with TCP the communications are usually restricted to a single path per connection. Resource usage within the network would be more efficient were it possible for these multiple paths to be used concurrently. Multipath TCP is a proposal to achieve multipath transport in TCP.

The principle objective of this experiment is to examine new developing MPTCP with the target of reducing the data loss due to congestion in TCP. The motivation for implementing a multipath TCP protocol is to enhance throughput and execution of end-to-end connections. This arrangement permits the utilization of multiple paths by the same TCP connection to maximize resource usage, increase redundancy and flexibility.

As the enormous use of computer network spread day by day to establish network connections between sending and receiving nodes to send data packets from source to destination in such a way that the data packets would send to correct destination with no loss of data packets. In this experiment, the wired network simulation using NS2 on TCP protocol in which the simulation of the wired network design using 10-node scenario and analyze packet drop rate, packet delivered rate and throughput by varying the queue size and also analyze the packet flow scenario performance of designed wired network using Xgraph.

II. LITERATURE SURVEY

As the rapid development in handheld portable devices such as mobile phones, laptops and tablets have made it important to be always connected to users and have better throughput connection between the users and network, many applications which run on these devices share their data over cloud systems, address to data centers across the world. Simultaneously some devices have developed the ability to connect to internet with one or more interfaces so as to increase the communication resources. The data centers which are widely spread over the world use multi-homing for being connected to networks to enhance the flexibility for the services requested concurrently by the users. The major concern is that the data traffic always increases due to the increasing number of subscribers. Users always expect to have a fast and reliable network speeds which results in increase of data traffic. Some authors listed below started dedicating their work on some new and efficient ways to make the system more efficient. MPTCP which is an evolution of TCP that allows the simultaneous use of multiple interfaces for a single connection while standard TCP uses single path for each communication between the pair of hosts.

1) G. Huston, Telstra explained the TCP Performance in "The Internet protocol" journal and also gave a detailed explanation on the communication channel of TCP between processes on each host system. The author explains the TCP header and also the 3-way handshake, which initiates the TCP communication between pair of hosts. The author how the TCP channel is reliable, full-duplex, and streaming. To achieve this functionality, the author explains that the TCP drivers break up the session data stream into discrete segments, and attach a TCP header to each segment. These segments have a unique sequence number so that at the receiver the data is received in same order sent by the sender.

2) Dinamene. B, F. Silva described in their work "Multipath TCP Protocols" that MultiPath TCP aims to meet at a functional level is to improve throughput and resilience. Here the author briefly discusses about the TCP and its limitations. The author explains how to improve throughput a MultiPath TCP connection over multiple paths should not achieve a throughput worse than a single TCP connection over the best path in that group of paths. To improve resilience MultiPath TCP paths must be interchangeable and must never be less resilient than a regular single-path TCP.

3) G.Veerman in his work "MultiPath TCP: Hands-On" described that MPTCP can improve throughput and robustness when more than one path is available. Here the author explains the working of MPTCP path management. Also explains how MPTCP uses links with the same speed and little RTT difference it gets the best performance and throughput in WAN and LAN environments. Concluding that MPTCP could even get maximum throughput when both a LAN and WAN path were being used at the same time.

4) C. Raiciu from Univ. Politehnica of Bucharest and M. Handly, D.Wischik from Univ. College London put their work together in "Coupled Congestion Control for Multipath Transport Protocols". Here they have defined congestion control algorithms running on different subflows by linking their increase functions, and dynamically controls the overall aggressiveness of the multipath flow. And also explained to achieve perfect resource pooling, one must couple both increase and decrease of congestion windows across subflows.

5) Gokhan AY Neşet doğanalp Ans Mohamed A Elbousiffi Electrical and Electronic Engineering Department Eastern Mediterranean University in their work "Exploring Mobile/WiFi Handover with Multipath TCP" they gave the results about the workability of MPTCP over communication networks. They tested the following three modes of MPTCP: Full MPTCP Mode, Single-Path Mode and Backup Mode to show the power consumption of MPTCP in handheld portable devices.

6) Johanna A in TCP Performance Simulations Using Ns2 explained the instructions for the simulation exercise are given. The instructions consist of theory explaining TCP's congestion control algorithms namely Reno, Tahoe and the analytical models for TCP's throughput and description of the main features of ns2 simulator and Otcl language. The author has obtained the numerical results of packet loss in different situation of TCP congestion.

7) S.Jeneeth Subashini, D. Guna Shekar, C.Harinath Reddy and M. Manikanta worked together in "Implementation of Wired and Wireless Networks, Analysis Simulation and Result Comparison Using Ns2" and designed a six node wired network with TCL script as front end to the simulator. The TCL script uses compiled C++ hierarchy to achieve efficiency in simulation and faster execution time. After defining the topology using tool command language, agents such as FTP over TCP and CBR over UDP are attached to the respective nodes of the network. The traffic flow between the nodes allows in calculating the performance of the network. The throughput of wired six node network was been calculated. Concluding that when compared to wired network, delay and packet drop rate is higher in a wireless networks.

8) Sirwan A. Mohammed and Sattar B. Sadkhan in their work "Design and simulation of network using ns2" explained the use of network simulator (ns2) simulation for designing networks and gave a detailed explanations on C++ forming an efficient class hierarchy core of ns2 that takes care of handling packets, headers and algorithms.

III. OVERVIEW OF TCP

Transmission Control Protocol (TCP) is collection of reliable end-to-end transmission functionality in the overall Internet architecture. All the functionality required to take a clear base IP datagram delivery and expand upon this a control display that executes reliability, sequencing and flow control is implanted inside the TCP. The TCP provides a communication channel between processes on each host framework. The channel should be reliable, full-duplex, and streaming. The TCP drivers separate the session information stream into discrete fragments and append a TCP header to each portion. An IP header is joined to the TCP packet and this composite packet is then passed to the network for delivery. This TCP header has various fields that are utilized to help the TCP functionality.

TCP has the following functional attributes:

- Unicast protocol.
- Connection state.
- Reliable.
- Full duplex.
- Streaming.
- Rate adjustment

A. The TCP Protocol Header

This is the layout of TCP headers:

1) Source TCP port number (2 bytes)
2) Destination TCP port number (2 bytes)
3) Sequence number (4 bytes)
4) Acknowledgment number (4 bytes)
5) TCP data offset (4 bits)
6) Reserved data (3 bits)
7) Control flags (up to 9 bits)
8) Window size (2 bytes)
9) TCP checksum (2 bytes)
10) Urgent pointer (2 bytes)
11) TCP optional data (0-40 bytes)

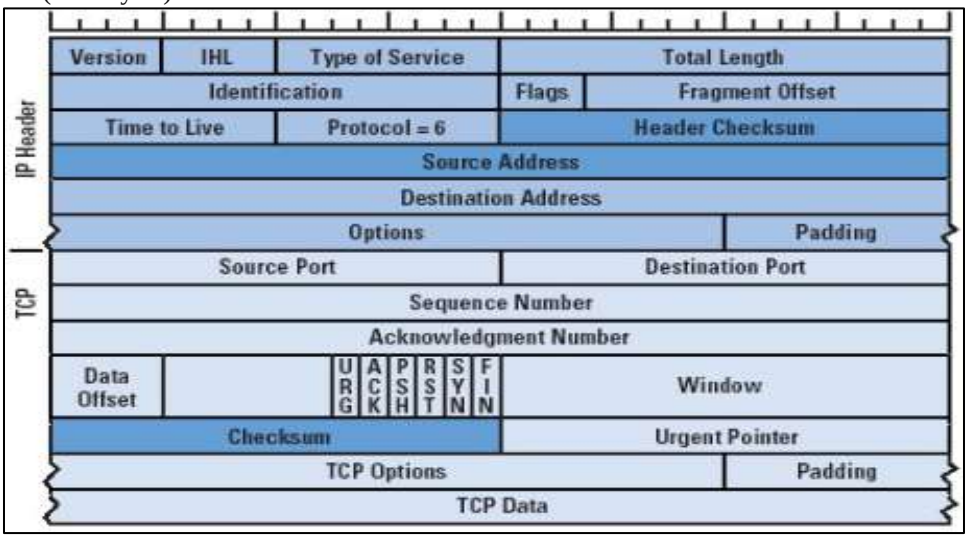

Fig. 3.1: The TCP/IP Datagram

Numerous alternatives can be carried in a TCP header. Those pertinent to TCP execution include:

- Maximum-receive-segment-size.
- Window-scale.
- SACK-permitted option and SACK.

B. TCP Sliding Window Protocol

The objective for the sliding window protocol is to maximize effectiveness of the data exchange, inferring that TCP should attempt to find the purpose of dynamic stability of most extreme system productivity, where the sending information rate is boosted only preceding the beginning of continuous packet loss.

TCP utilizes a sliding-window protocol to help large amount of data exchange (shown below in figure 3.2.1). The receiver promotes to the sender the accessible support space at the receiver. The sender can transmit up to this data before awaiting a further buffer update from the receiver. The sender will have close to this amount of information to travel in the system. The sender should likewise buffer sent information until the point when it has been ACKed by the receiver. The send window is the minimum of the sender's buffer size and the advertised receiver window. Each time an ACK is received, the trailing edge of the send window is progressed. The minimum of the sender's buffer and the promoted receiver's window is utilized to compute another leading edge. In the event that this send window incorporates unsent information, this information can be sent promptly.

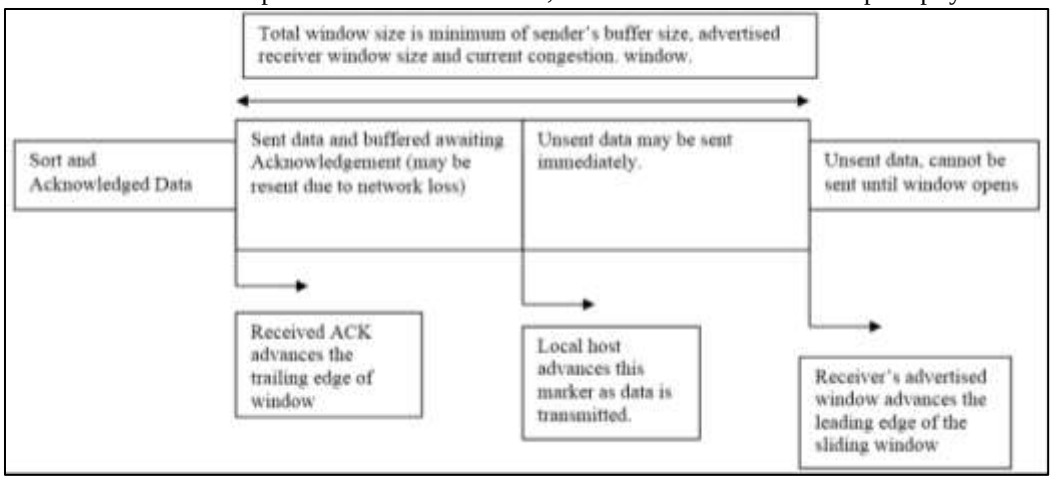

Fig. 3.2.1: TCP Sliding Window

The dynamic task of the window is a basic component of TCP performance for volume exchange. The mechanics of the protocol include an extra overriding modifier of the sender's window, the congestion window, addressed to as cwnd. The objective of the window-management algorithm is to begin transmitting at a rate that has a low chances of packet loss, then to increase the rate (by increasing the cwnd size) until the sender gets a notification, through the detection of packet loss, that the rate has exceeded the available limit of the system. The sender at that point promptly parts its sending rate by decreasing the value of cwnd, and resumes a progressive increase of the sending rate. The objective is to persistently adjust the sending rate that it fluctuates around the genuine value of available system limit. This fluctuation empowers a dynamic change that consequently observes any increase and decrease in accessible limit through the lifetime of the data flow.

C. *Packet Loss*

The beginning value of the cwnd window (the Initial Window, or IW) is set to that of the Sender Maximum Segment Size (SMSS) value. This SMSS value depends on the receiver's maximum segment size, obtained during the SYN handshake, the discovered path MTU, the MTU of the sending interface and 536 bytes. The sender at that point enters a flow control mode named Slow Start. Slow Start endeavors to begin a TCP session at a rate the system can support and after that consistently increment the rate. This Slow Start rate increment stops when the congestion window surpasses the receiver's advertised window, when the rate surpasses the value at the start of congestion as recorded in ssthresh, or when the rate is greater than the system can maintain. The appropriate response is shown by data packets being dropped by the system. For this situation, TCP needs to attempt numerous capacities:

- The packet loss has to be detected by the sender.
- The missing data has to be retransmitted.
- The sending data rate should be adjusted to reduce the probability of further packet loss.

TCP can notify packet loss in two different ways. Firstly, to start with, if a single packet is lost within a sequence of packets, successful delivery packets following the lost packet will make the receiver produce a duplicate ACK for each progressive packet. The gathering of these duplicate ACKs is a signal of such packet loss. Second, if a packet is lost at the end of a sequence of sent packets, there are no following packets to create duplicate ACKs. For this situation, there are no relating ACKs for this packet, and the sender's retransmit timer will terminate and the sender will assume packet loss.

IV. OVERVIEW MULTIPATH TCP

As the Internet advances, requests on Internet resources are ever-increasing, yet frequently these resources (specifically, transmission capacity) cannot be completely used because of protocol requirements both on the end-systems and within the network. If these resources could be utilized simultaneously, end user experience could be enormously improved. Such upgrades would decrease the resource consumption on network infrastructure.

The objectives of MPTCP are resource pooling by at the same time utilization of multiple paths over a network. The two key advantages of multipath transport are as follows:

- To boost the flexibility of the connectivity by providing multiple paths, shielding end host from the failure of one.

– To boost the efficiency of the resource use, and subsequently increment the network capacity available to end hosts.

MPTCP targets to increase wide-scale deployment by perceiving the significance of application and network compatibility objectives. The objectives relate to the feature of MPTCP to the network and to the application. The diagram shown in Figure 4.1 illustrates a typical usage scenario for Multipath TCP. Two hosts, A and B, are communicating with each other. These hosts are multihomed and multi-addressed, providing two disjoint connections to the Internet.

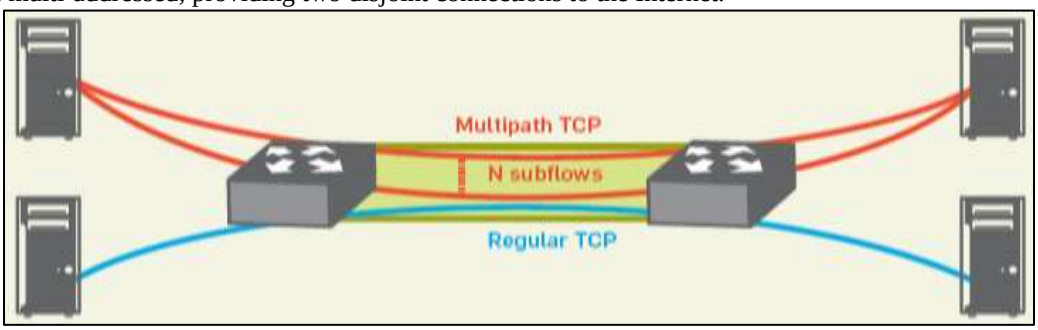

Fig. 4.1: Simple Multipath TCP Usage Scenario

The functional objective is in supporting the utilization of multiple paths, Multipath TCP has the following two functional goals.

A. Improve Throughput

Multipath TCP must help the simultaneous utilization of multiple paths. To meet the minimum performance incentives for deployment, a Multipath TCP connection over multiple paths should accomplish no worse throughput than a single TCP connection over the best constituent path.

B. Improve Resilience

Multipath TCP must help the utilization of multiple paths interchangeably for resilience purposes, by allowing segments to be sent and re-sent on any available paths.

MPTCP influences utilization of standard TCP sessions, named "subflows", to give the underlying transport per path, and as such these retain the network compatibility desired. MPTCP-particular data is conveyed in a TCP-compatible way; despite the fact that this mechanism is separate from the real data being transferred so could advance in future modifications. Figure 4.2 outlines the layered design.

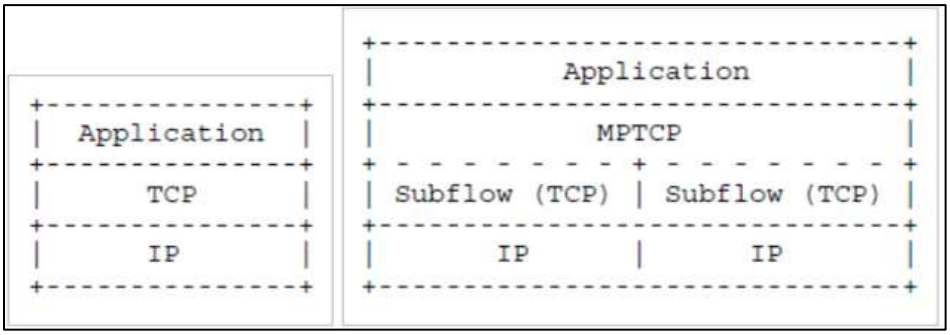

Fig. 4.2: Comparison of Standard TCP & MPTCP Protocol Stacks

Situated below the application layer, the MPTCP extension thus manages multiple TCP subflows below it. In order to do this, it must implement the following functions:

– Path Management.
– Packet Scheduling.
– Subflow (single-path TCP) Interface.
– Congestion Control.

V. EXISTING SYSTEM

During the initial data transfer phase of a TCP connection the Slow Start algorithm is utilized. In any case, there might be a point amid Slow Start that the network is compelled to drop at least one packet because of congestion. If this happens, Congestion Avoidance is utilized to slow the transmission rate.

In the Congestion Avoidance algorithm a retransmission timer terminating or the gathering of duplicate ACKs can verifiably notify the sender that a network congestion circumstance is occurred. The sender instantly sets its transmission window to 50% of the present window size, but to at least two segments. In the event that congestion was demonstrated by a timeout, the congestion

window is reset to one segment, which naturally puts the sender into Slow Start mode. If congestion is notified by duplicated ACKs, the Fast Retransmit and Fast Recovery algorithms are invoked.

The simulation parameters for this experiment are:

1) Sent packets.
2) Received packets.
3) Packet dropped percentage.
4) Packet delivered percentage.
5) Throughput of the network.

Here in this experiment wired topology for 10 nodes is being used as shown in figure 5.1.

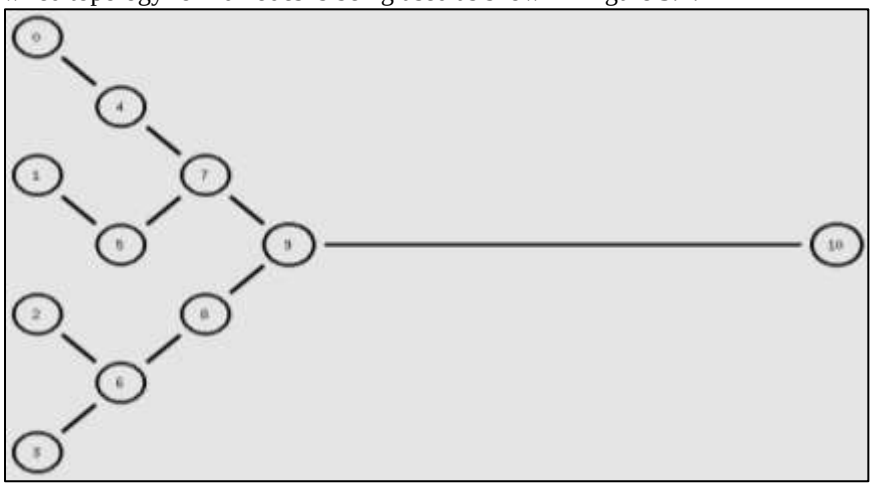

Fig. 5.1: Wired Topology of 10 for the Experiment

The sliding window size is considered here as 8 ($cwnd_i$) and the queue size at each node is varied to show how the congestion occurs. At this point, when the congestion occurs there will be significant packet drop percentage to the corresponding packet arrival rate. Due to the packet drop the ACK packets are not received and results in duplicate packets being sent again which gives rise to congestion in the network and the resources of the system are not utilized properly. This congestion situation shows the degradation percentage in throughput of the network. The results of the existing topology for 10 node using TCP for FTP communication with varying Queue size are as shown.

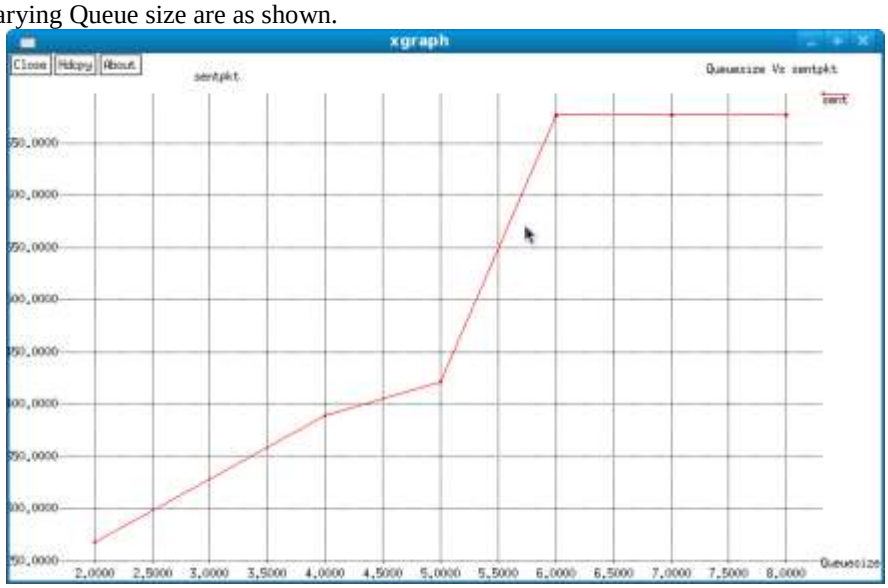

Fig. 5.2: TCP Sent packets

The above figure 5.2 shows the number of sent packets for the Queuesize 2, 4,5,6,7 and 8 for fundamental TCP.

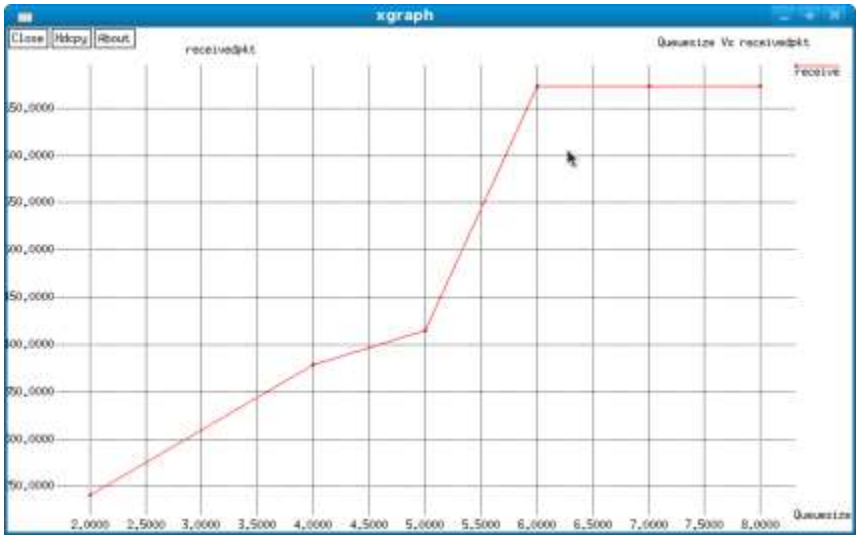

Fig. 5.3: TCP Received Packets

The above figure 5.3 shows the number of received packets for Queuesize 2,4,5,6,7 and 8 using fundamental TCP.

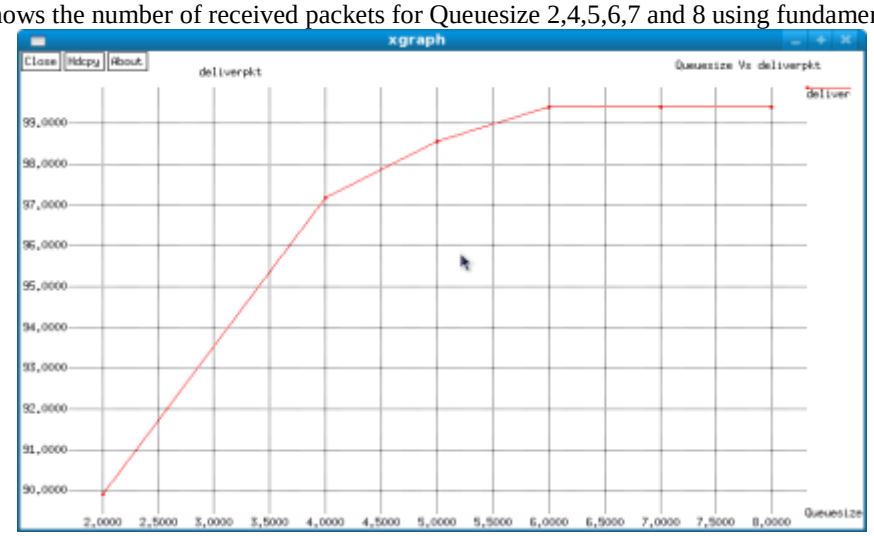

Fig. 5.4: TCP Delivered Packets

The above figure 5.4 shows the delivered packets percentage for Queuesize 2,4,5,6,7 and 8 using fundamental TCP.

Here in the above graph it is noted that at queue size 2, 4 and 5 the system suffers from congestion and so the percentage of delivered packets is not stable until the queue size is not 75% of the window size.

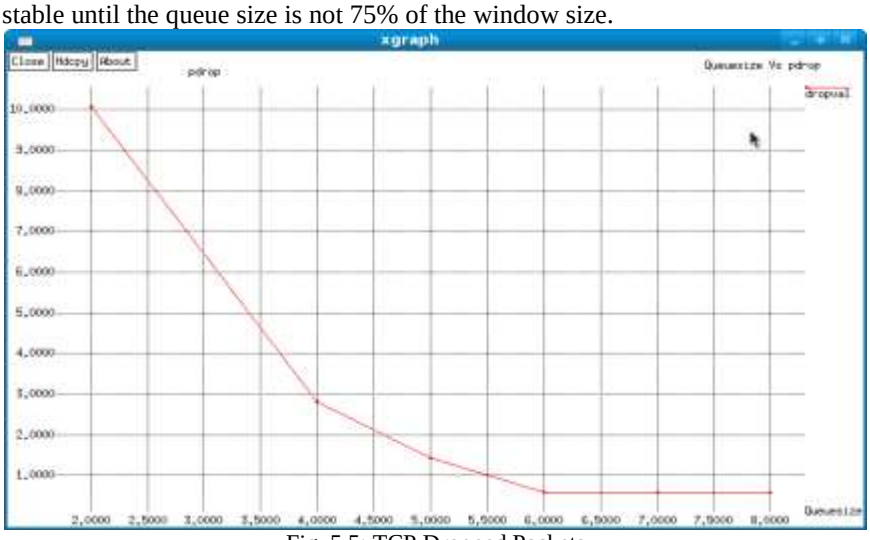

Fig. 5.5: TCP Dropped Packets

The above figure 5.5 shows the dropped packets percentage for Queuesize 2,4,5,6,7 and 8 using fundamental TCP.

In contrast with the delivered packet percentage, the dropped packet percentage is high when the Queue size is at minimum value and gradually decreases with the increasing queue size.

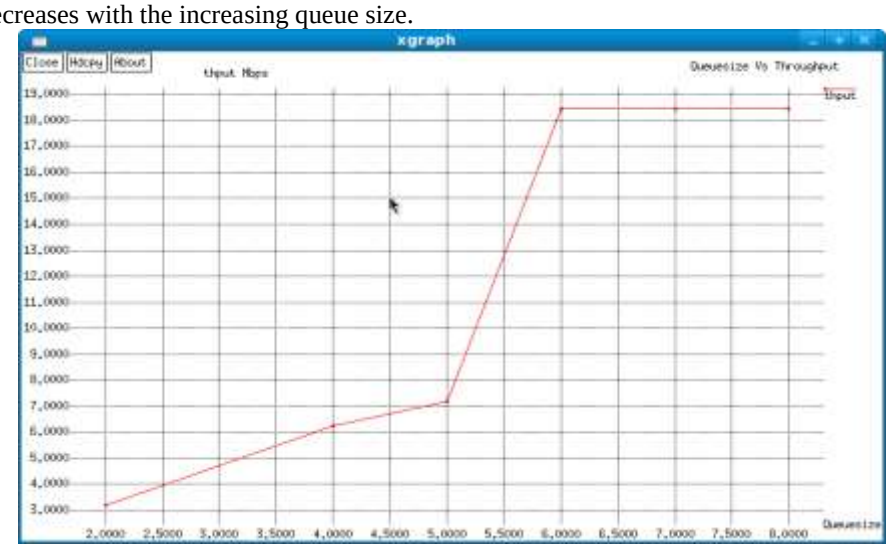

Fig. 5.6: TCP Throughput

The above figure 5.6 shows the throughput of the system (Mbps) for Queuesize 2, 4,5,6,7 and 8 using fundamental TCP.

As seen from the results the simulation parameters remain constant when the buffer is approximately 75% full and there will be no congestion further and the packet drop will decrease resulting in enhancing the throughput of the system.

VI. PROPOSED SYSTEM

The motivation behind MPTCP is the ability to utilize different ways in a system all the while. So as to avoid the congestion in the network, this results in lower packet drop rate and higher throughput of the network. Network congestion happens when an excess of data is being persisted a physical link and packets are being dropped. At the point when this happens the total sum of bandwidth that can be carried by the link gets less. There are a few methods which make it conceivable to make productively utilization of all the bandwidth available. Those procedures are actualized in the transport layer of the OSI stack and are referred to as 'congestion control' mechanism. The architecture of the proposed system is shown in figure 6.1, showing that multiple users access data using multiple paths.

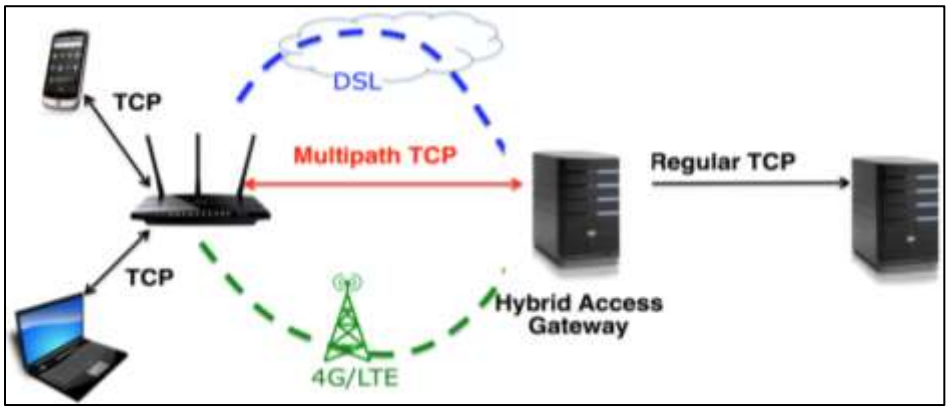

Fig. 6.1: Architectural Overview of the Experiment

Congestion control is implemented in the TCP protocol, this since TCP needs to ensure that data gets delivered, contrasted with User Datagram Protocol (UDP) which does not. Some of the congestion control algorithms used within TCP are Tahoe and New Reno, there are also specific algorithms designed for long connections which have high RTTs. A congestion control algorithm is also being utilized by MPTCP, the algorithm is relatively similar to the one from TCP however adjusted to meet MPTCP its objectives.

A. MPTCP Congestion Algorithm

All reliable transport protocol needs a congestion algorithm. Since MPTCP utilizes various connections it is not the same as TCP and utilizations a changed algorithm. TCP maintains its congestion window by increasing the window with each ACK (1/cwnd)

and dropping it to half when a ACK is dropped. For MPTCP this is no different, however now a window is kept for each subflow. The only thing changed is the increase rule for this window which is shown in equation 1. The increase rule is based on alpha, which one can see as a constant for now and is explained below. It increases the window by alpha, over the sum of the window sizes of all paths. This gives more increase to the subflows with a larger window, remember that a larger window size means less congestion. The sum of all windows part should meet goal two.

$$cwnd_i = cwnd_i + \min((alpha/ cwnd_{tot}), (1/ cwnd_i)) \quad (1)$$

Alpha is the increase parameter shown in equation 2. The first part looks at the congestion window and RTT on all different paths and checks if it is getting better or worse than normal TCP, which meets goal one. The second part of alpha moves traffic from a congested link to a uncongested link and meets goal three.

$$alpha = cwnd_{tot} * \frac{\max_i(cwnd_i * mss_i^2/rtt_i^2)}{(\sum_i(cwnd_i * mss_i/rtt_i))^2} \quad (2)$$

The decrease formula is almost the same as for TCP but is now used per subflow and is shown in equation 3.

$$cwnd_i/2 \quad (3)$$

The algorithm should make sure that MPTCP takes the best path available. Choosing the wrong paths can make MPTCP less efficient; it can for example take longer routes or use links which are congested. When MPTCP takes the most optimal paths it would get the best throughput. The buffer size needed for MPTCP is a little different than with TCP. In equation 4 is shown how the buffer size for TCP is being calculated.

$$BuffersizeTCP = RTTmax \_ LinkMaxbits \quad (4)$$

For MPTCP, this is a little bit different. Here you need to specify the total maximum bits possible of all links combined. It is noted that: "If it is required to allow all paths to keep sending while any path is fast retransmitting, the buffer must be doubled". This ends up in equation 5, shown below.

$$BuffersizeMPTCP = RTTmax \_ AllLinksMaxbits \_ 2 \quad (5)$$

For the similar topology shown in figure 5.1, the results for multipath TCP are obtained using the above algorithm for the same simulation parameters of the existing system. In MPTCP, when packet dropping is triggered. The proposed MPTCP algorithm finds the alternative path having less congestion so that the data arrives at the destination with less delay time. Thus, resulting in reduction of congestion to a reasonable extent and giving better throughput than the fundamental TCP. The detection of alternative path when the path failure is triggered in the experiment is advantageous so that the network failure is avoided and the system communicates with the end points with less time consumption.

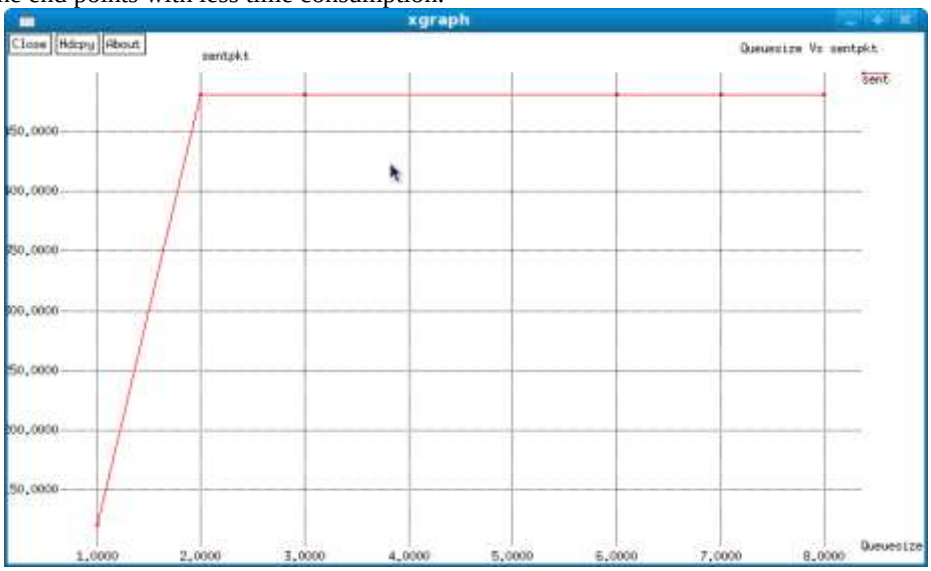

Fig. 6.1: MPTCP Sent Packets

The above figure 6.1.1 shows the number of sent packets for the Queuesize 2, 4,5,6,7 and 8.

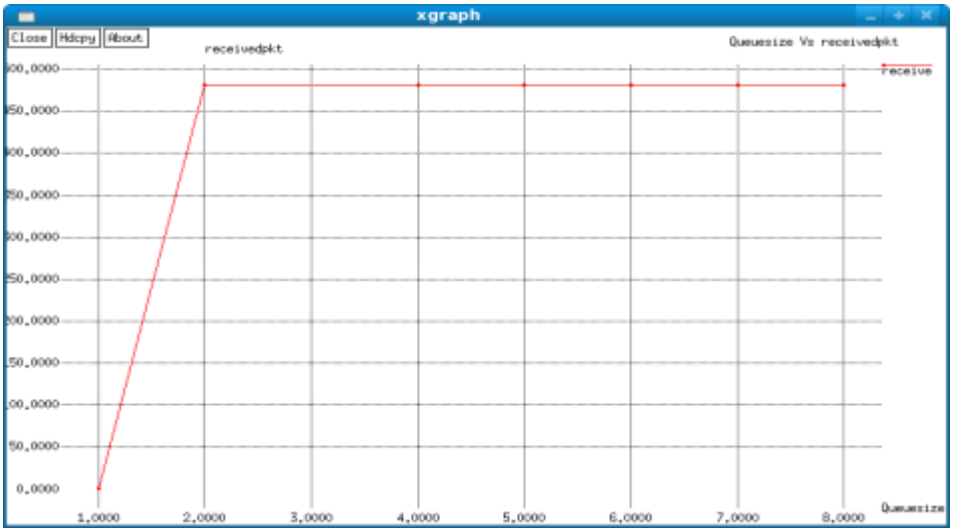

Fig. 6.1.2: MPTCP Received Packets

The above figure 6.1.2 shows the number of received packets for the Queuesize 2, 4,5,6,7 and 8.

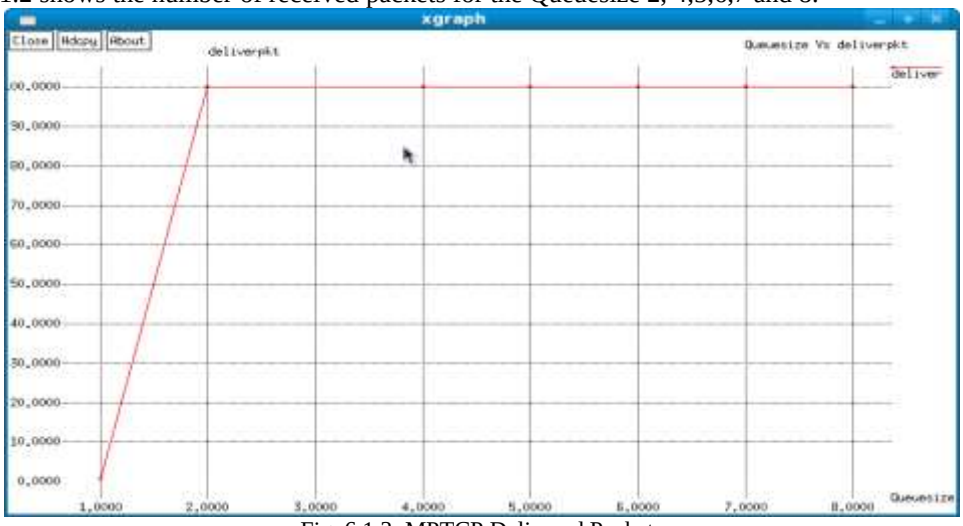

Fig. 6.1.3: MPTCP Delivered Packets

The above figure 6.1.3 shows the delivered packets percentage for Queuesize 1, 2,4,5,6,7 and 8. Here in the above graph it is noted that at queue size less than 2 the system suffers from congestion and so the percentage of delivered packets is more stable at the queue size less than 20% of the window size.

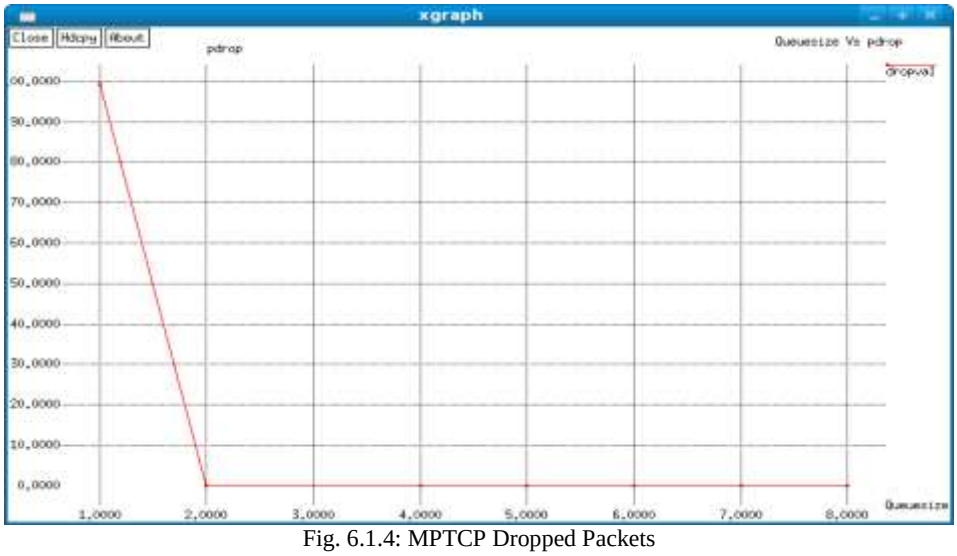

Fig. 6.1.4: MPTCP Dropped Packets

The above figure 6.1.4 shows the dropped packets percentage for Queuesize 1, 2,4,5,6,7 and 8. Here the dropped packet percentage is high when the Queue size is at minimum value and gradually decreases with the increasing queue size but here in MPTCP the packet drop rate.

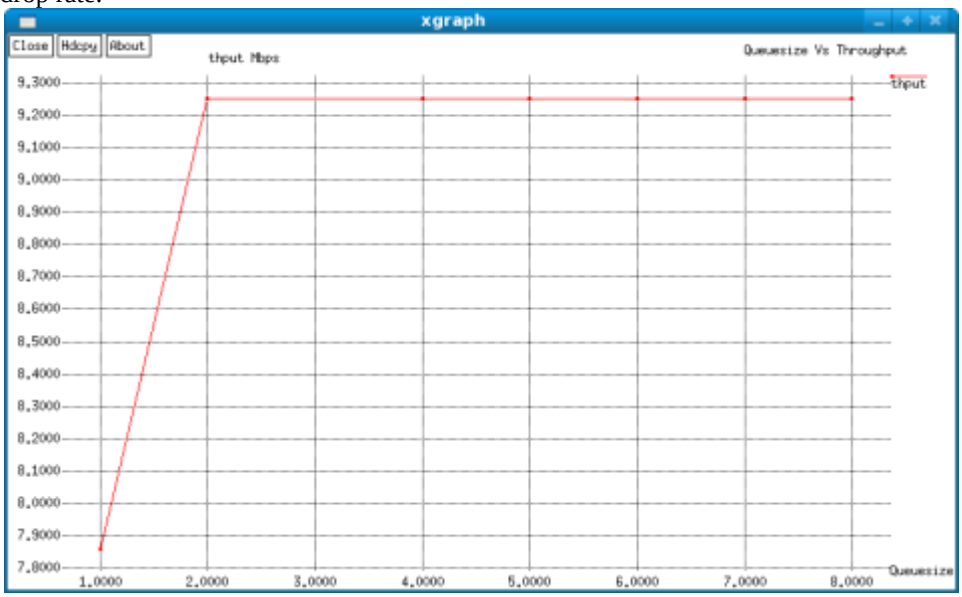

Fig. 6.1.5: MPTCP Throughput

The above figure 6.1.5 shows the throughput of the system (Mbps) for Queuesize 1, 2, 4,5,6,7 and 8.
As seen from the results, the simulation parameters remain constant when the buffer is approximately 15% full and there will be no congestion further and the packet drop will decrease resulting in enhancing the throughput of the system.

VII. EXPERIMENT RESULTS

The results obtained from the experiment are given below.

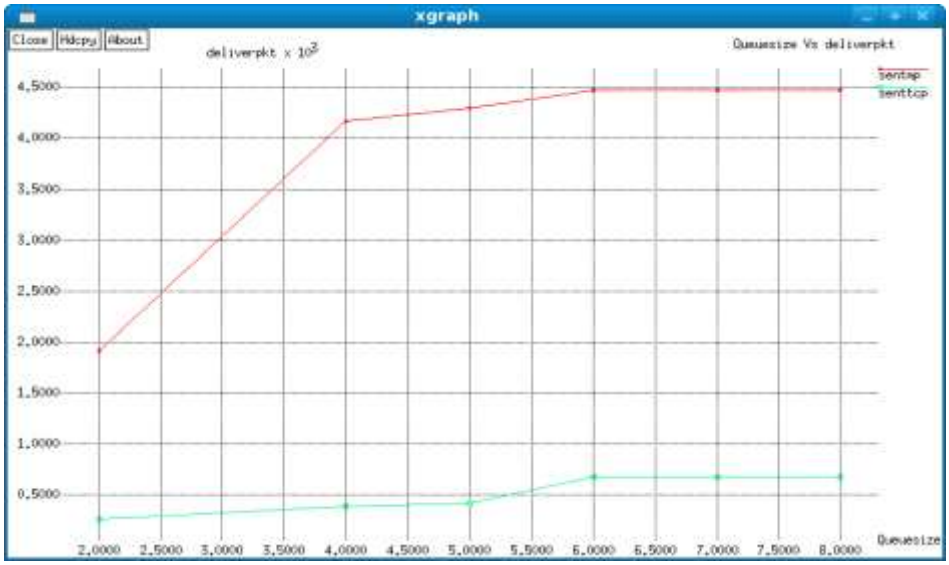

Fig. 7.1: MPTCP & TCP Sent Packets

Here in the above figure 7.1 it is shown the comparison of sent data packets of MPTCP and TCP.

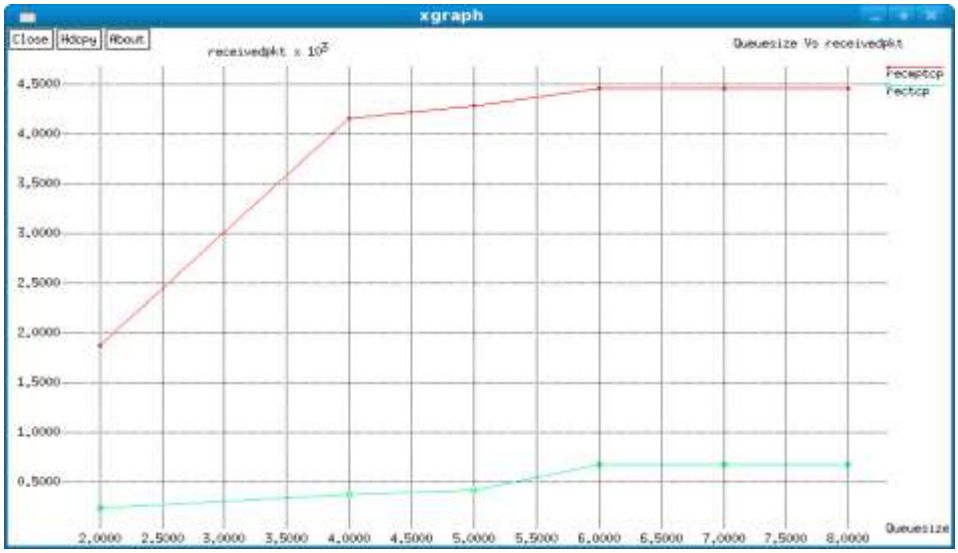

Fig. 7.2: MPTCP and TCP Received Packets

The above figure 7.2 shows the comparison of received packets between MPTCP and TCP.

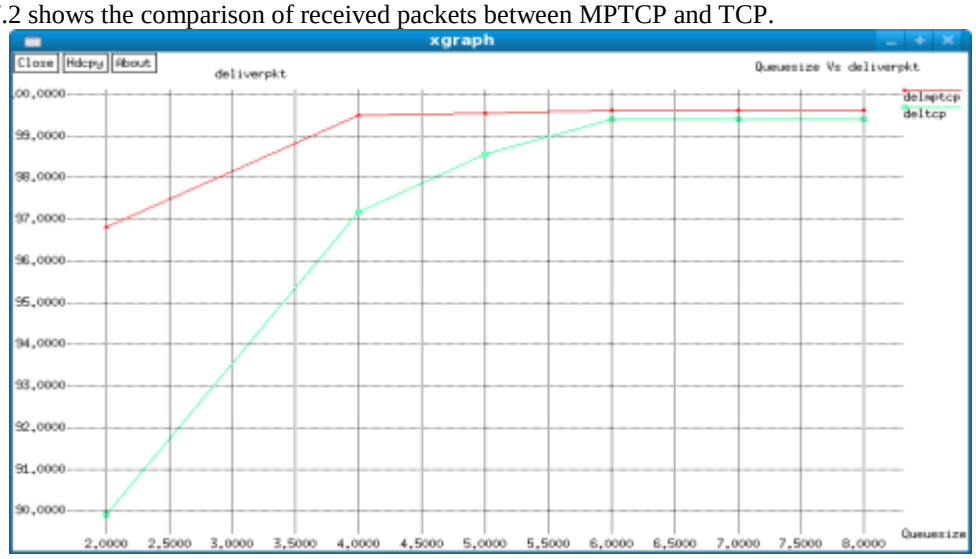

Fig. 7.3: MPTCP & TCP Delivered Packets

Here in the above figure 7.3 it is shown the how the delivered packet percentage of MPTCP is good than the fundamental TCP.

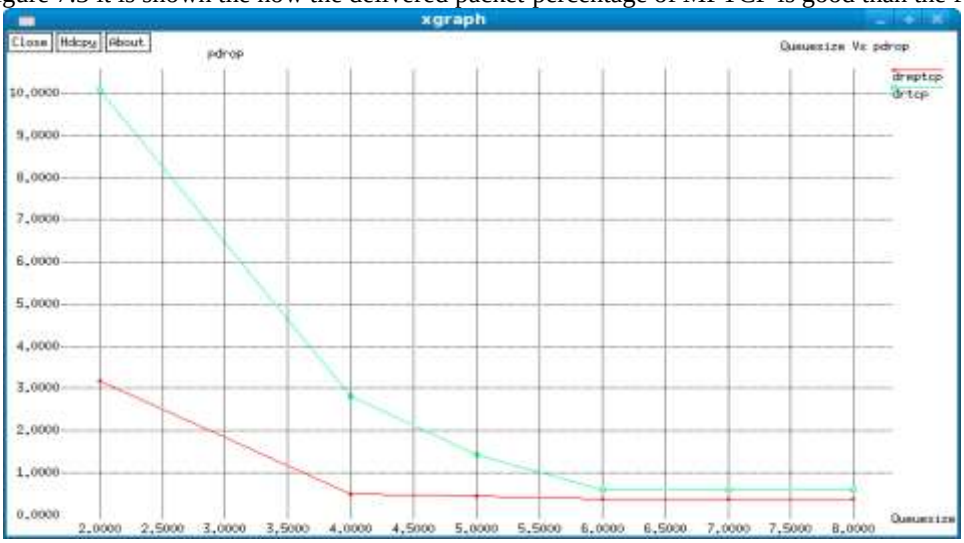

Fig. 7.4: MPTCP & TCP Dropped Packets

In above figure 7.4, it is shown that the dropped packet percentage is less in MPTCP than that of the fundamental TCP.
Thus, concluding with the figure 7.5 given below, showing the throughput comparison between MPTCP and TCP. It is clearly seen that the throughput of MPTCP in congestion is better than that of the throughput of TCP when in congestion.

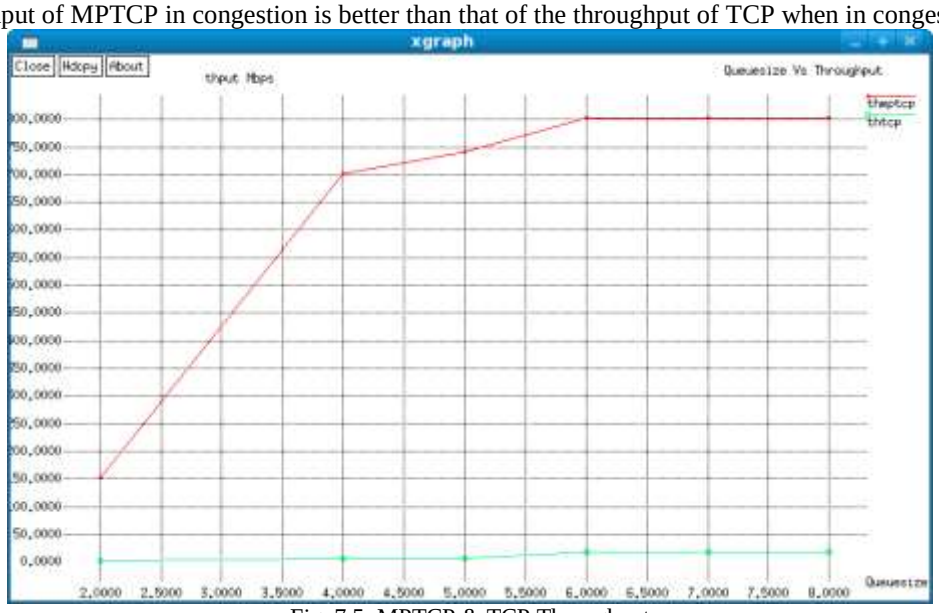

Fig. 7.5: MPTCP & TCP Throughput

VIII. CONCLUSION & FUTURE WORK

In this experiment, it has been likewise described the ongoing structure of the implementation for MultiPath TCP, by naming the objectives characterized for this protocol, describing its structure and how it works, since the start until the end of a session. In this experiment NS2.38 simulator the end user performance of wired network consisting 10 nodes has been used. It is been found out that network simulator (NS2), is used as a tool to design the result of the simulation and transfer information securely between the communicating nodes using multiple paths.

An extensive variety of test situations were created in order to evaluate the utilization of MPTCP protocol on real network environments. It is demonstrated that MPTCP can possibly enhance network flexibility in different situations where multihoming and multi path connections are available. Besides, an extensive evaluation of the MPTCP protocol is performed, and is confirmed that it can in certainty balance network congestion, offering higher throughput and being more resilient to failures over the network. In a few situations, MPTCP showed overall performance better when contrasted with the conventional TCP. Summarily in this experiment, it demonstrates that an implementation of this protocol with varying queue size the congestion can be reduced over web network resulting in the increase in its overall performance. Hence from this experiment it is shown that MPTCP is more efficient than TCP when in congested network.

As it has been shown that for a wired topology, the throughput for MPTCP is good and has less drop packet percentage when compared to TCP for FTP communication in a congested network. In future, it may be implemented using UDP for LAN, WAN, WLAN, SAN and other communication networks.

REFERENCES

[1] Geoff Huston and Telstra "TCP Performance" - The Internet Protocol Journal - Volume 3, No. 2- June 2000.
[2] Wischik D., Raiciu C., Greenhalgh A., and M. Handley, "Design, Implementation and Evaluation of Congestion Control for Multipath TCP" Usenix NSDI March-2011, <http://www.cs.ucl.ac.uk/staff/c.raiciu/files/mptcp-nsdi.pdf>.
[3] Bagnulo, M., "Threat Analysis for TCP Extensions for Multipath Operation with Multiple Addresses", RFC 6181- March 2011.
[4] S. Barre, C. Paasch, and O. Bonaventure, "Multipath TCP: From theory to practice," in networking 2011, ser. Lecture Notes in Computer Science, J. Domingo-Pascual, P. Manzoni, S. Palazzo, A. Pont, and C. Scoglio, Eds. Springer Berlin Heidelberg, 2011, vol. 6640, pp. 444–457.
[5] J. Iyengar - Franklin and Marshall College; S. Barre - University Catholique de Louvain; A. Ford - Roke Manor Research; C. Raiciu and M. Handley - University College London;"Architectural Guidelines for Multipath TCP Development." Internet Engineering Task Force, ISSN: 2070-1721- March 2011.
[6] M.Scharf and A.Ford "Multipath TCP (MPTCP) Application Interface Considerations "Internet Engineering Task Force (IETF) Request for Comments: 6897 Alcatel-Lucent Bell Labs Category: Informational ISSN: 2070-1721 Cisco March 2013.
[7] A. Ford Pexip, C. Raiciu Univ. Politechnica of Bucharest, M. Handley Univ. College London, O. Bonaventure Univ. Catholique de Louvain, C. Paasch Apple, Inc "TCP Extensions for Multipath Operation with Multiple Addresses." draft-ietf-mptcp-rfc6824bis-11, May 15, 2018.