Analysis of TCP Congestion Control Queuing Mechanism and Investigation for High Throughput and Low Queuing Delay

Iqtidar Ali, Tariq Hussain, Fatima Perviz and Altaf Hussain

April 22, 2020

ANALYSIS OF TCP CONGESTION CONTROL QUEUING MECHANISM AND INVESTIGATION FOR HIGH THROUGHPUT AND LOW QUEUING DELAY

Iqtidar Ali[1], Tariq Hussain[1*], Fatima Perviz[1], Altaf Hussain[1]

[1]University of Agricultural, Peshawar

Iqtidar @aup.edu.pk, uom.tariq@gmail.com*, faaatimakhaaan31@gmail.com, altafkfm74@gmail.com

Abstract: A protocol is a set of rules that governs data communication which decides when to communicate, how to communicate and where to communicate and also what to communicate. One of them is Transmission Control Protocol (TCP) that is the most popular and known protocol for controlling of the data transmission from source to destination or from one node to another node. It gives the best results in offline streaming of the data as compared to User Datagram Protocol (UDP). Congestion is the mechanism in networking that takes place in the time of communication when the data exceeds from its actual limit and it becomes overhead then congestion problem occurs. It usually occurs on the network like if there exist a router then the overhead occurs on router when there is limited time-baud buffer due to which the data may loss or overhead occurs. For this solution TCP is the best option to control and avoid from this problem. In this paper, an analysis has been made with the help of TCP for examining the congestion control queuing mechanism. Along with that, some parameters have been taken into account take are throughput and delay. These parameters have been tested under different settings and have been showed that by utilizing TCP congestion control queuing approach high throughput and low delay of queuing has encountered. From OPNET simulation results it has been concluded that TCP have shown remarkable and outstanding performance for controlling congestion issue with the help of other existing schemes that has shown poor performance.

Keywords: Transmission Control Protocol, Congestion Control Queuing, Throughput, Queuing Delay.

1. INTRODUCTION

The core of TCP congestion control is in the additive increase, multiplicative decrease and halving the window for congestion after receiving each window containing some packet loss. Another important component of the congestion control mechanism is high retransmit timer which includes the exponential back off when a retransmission packet itself discarded. Subsequently, the third fundamental element is the start mechanism for initial probing to know the available bandwidth rather than sending a high rate data in beginning which may not be supported by a network [1]. Furthermore, another congestion mechanism is a term called acknowledgement clocking (ACK) where the appearance of acknowledgement at the instigator is used to clock out the broadcast of new data [2]. When we merge all these mechanisms such as retransmit timers, ACK clocking, additive increase multiplicative decrease and slow start, there is a tremendous possibility of distinct behaviours [3]. For an instance, many things can be taken into consideration such as Return round trip, specific algorithm for retransmit timeout, the reply to rearrange or delay packets, the length of the congestion window in the initial time.

Henceforth, distinct TCP implementation differs in some extent to compete for available bandwidth however as they all adhere to some set mechanism, there is no starvation of bandwidth among completing TCP connections. And as a result, equal bandwidth sharing is not common between TCP connections and it is unlikely that one TCP implementation will prevent other TCP connection for an adequate bandwidth sharing of the available limit (Martin

[4]. Our literature review section discusses different themes based on TCP congestion algorithm. One of the authors discussed about TCP congestion control algorithms tend to increase robustness all across the

environment instead of fine tuning for a specific network requirement or traffic type at the sake of another TCP connection [5].

A secondary ideology is the independent changes are in development and evaluating one change needs considering the other account interaction with other changes in progress. On top of considering the influence of a specific impact of change in TCP, provided the recent environment it is also useful to recognize the potential influence of a proposed change some year down the road when other changes are taking place to the network. Now the last theme theory is based on an inevitable heterogeneity in the congestion control mechanism deployed TCP deployment. For instance, an uneven implementation allows more robust operation when several packets are discarded from a congestion widow of data which is now widely deployed [3].

There have been various changes to TCP congestion control mechanism which intend to prevent various unimportant retransmit timeouts for small transfers to advance performance in terms with delay time, corrupt packets and reordering. Rather than associating with fundamental changes to congestion control mechanisms these variations would bring TCP closer to purify congestion control behaviours elaborated briefly in further sections such as additive increase, increase/multiplicative decrease, fast transmit and fast recovery [1]. Retransmit timeouts are a need for last resort in TCP flow control used with TCP sender has no other mechanism to identify that a retransmission is required. In addition, back off algorithms for retransmit timers are a basic element for congestion window of a segment. However, when the congestion window is bigger than one segment, TCP uses the fundamental such as additive increase, increase/multiplicative decrease, fast transmit and fast recovery which are fundamental congestion control mechanisms and particularly in this case it would favourable to prevent unwanted retransmit timeout efficiently [6].

When we consider recent TCP implementations which have two possible mechanisms for figuring out the packet loss and fast retransmit. A TCP connection in generic terms recovers more swiftly from a packet loss with the help of fast retransmit and inferring the communication with fast retransmit with three duplicate ACKs packets. When fast retransmit is initiated then the TCP source node retransmit the segment inferred to be lost and reduced its congestion and continuing the data transfer. If the TCP data source nodes do not receive three supplicate ACKs after a packet loss, the source nodes go for a considerable delay for waiting for the transmit timer to expire. Many experimental studies discussed about the performance costs to small flows of unwontedly waiting for expiration of retransmit timer [2].

2. RELATED TERMS

Now after the above investigation of TCP congestion control mechanism, we would raise a question that why we would require any enhancement of the algorithms constituting several elements such as additive increase, increase/multiplicative decrease, fast transmit and fast recovery. Previous studies and our thorough study literature review section shows that there still possibility to fine tune TCP's retransmit timeout algorithm to attain efficient balance between retransmit timeouts and unwanted delay of sensing the delay. However, it is quite not possible to design timeout algorithms which would not result in unwanted retransmit timeout. Likewise, while it would be possible to tune TCP's fast retransmit algorithm to attain improved equilibrium between unwanted unnecessary delay and unwanted fast transmission in sensing loss [5]. Therefore, we would not expect to fine tune TCP algorithm to flawlessly function as it is not possible to devise an algorithm which correctly identify the reception of a duplicate ACK. Henceforth it is desirable for congestion control algorithm to perform well even with retransmit timeout and fast retransmits[7].

Our study suggests that there is a huge scope of improving the congestion control mechanisms by experimenting the existing elements and testing them together to find out the most efficient way based on different scenarios. We consider different scenarios as the congestion control gets affected by the changes in network as a packet travel through several networks [4]. One more factor influencing the behaviours of TCP congestion mechanism control is the scheduling mechanism used by various routing devices however the research will include only FIFO scheduling method [6].

We firmly believe that hypothesis carries clarity and focuses on main issues on a research. Previous studies have argued that any hypothesis should be based on some real time observation which we believe quite true [14]. Our research has influenced by some hypothesis which are quite generic for congestion control mechanisms such as: -

- Elements of TCP congestion control mechanism have an impact on the performance of a packet
- Element impacts the behavior of each other along with the complete congestion control mechanism

As research hypothesis is always based on examination of some tests that evoke suspicion on current ideas and concepts. We are not trying to approve or disapprove our research hypothesis however we will endeavor to delve down into TCP congestion control mechanism to find the scope of improvement [8]. It is also not indispensable to create a research hypothesis to investigate on issues however we have deduced the hypothesis on the basis of our literature review, and we will be testifying the above statements through our experiments in Sheffield Hallam Laboratories.

3. Methodology, Tools and Techniques

Experimentation method is known as empirical research which includes conclusion and can be verified with experiments and observation. Our research can be termed as explanatory research where majorly quantitative method has a great participation. The laboratory work will be performed in Sheffield Hallam University. As elaborated in the literature review sections, the hypothesis will be evaluated through independent and dependent on variables [9]. Our resultant will be produced by changing these variables in the laboratory and data will be evaluated and analyzed to see measure performance of different elements and TCP congestion control mechanisms in totality [10]. The research methodology will be deductive in nature where we instigate our research with a pre-defined idea and further evaluating with experimentations. In our research, it is crucial to find facts and views at the source nodes and to simulate required scenario.

a. Techniques

We will consider first two elements of TCP congestion control mechanism by implementing on a simulation tool which will transmit a packet. The secondary step will evaluate two elements behaviours and influenced caused by them to the TCP congestion control mechanism in totality. Once we record the captured behaviours of these elements on congestion control mechanism, we would consider the resultants as empirical data which can be analyzed [14].

b. Simulation Environment

We will implement a network in one of the laboratories of Sheffield Hallam University where we will use the OPNET IT Guru Academic Edition (OPNET, 2011). The OPNET tool will facilitate us to create a network set up which can testify various TCP congestion control mechanisms. The tool will also empower us to monitor network packets and simulate congestion control environment so that all possible elements such as fast recovery, additive increase and multiplicative decrease can be assessed. IT Guru Academic Edition application is meant for networking related experiments and designed to test various laboratory work (OPNET, 2011).

c. Experiment Design

Experimental designs are intrusive in nature and it is complicated to carry out in real world circumstance [11]. And as often experimentation is an incursion, we will formulate an artificial environment so that we will be able to evaluate the relationship between with high validity. In generic terms, our experiment will have two groups and it will be interesting to determine whether these groups produce different congestion window size. In the research work, we will be designed to reveal TCP congestion control algorithms which will consist of various simulation scenarios with different elements such as additive increase, increase/multiplicative decrease, fast transmit and fast recovery [8]. The element will have varied values which can be changed which can provided us different resultants. In the laboratory of Sheffield Hallam University, we will set up a network where TCP protocol will be used and an end to end transmission will be established between two devices. Our network set up will allows capturing the congestion window with different mechanisms.

4. RESULTS AND DISCUSSION

In this research we will discuss the different scenarios that were designed and implemented on that network and the graphs that are obtained as a result of performing the simulation using the OPNET IT Guru Academic version. The scenarios are designed to investigate the congestion control algorithms of TCP along the queuing mechanism. The algorithms are tested on the network design for different scenarios. The network design will use TCP as its end to end transmission protocol for FTP application and will analyses the size of congestion window and sent segment sequence number for basic flow of FTP. On the routers one by one, one of the queuing mechanisms among the FIFO, PQ and WFQ will be implemented between router and IP32_Cloud to examine the effect of congestion control algorithm along these queuing mechanisms.

The three scenarios No-Drop, Drop-fast and Drop-Nofast have been checked for FTP using three queuing mechanisms FIFO, WFQ and PQ. In the No-Drop scenario the fast retransmit was enable and assign Reno to fast recovery to Server Hallam also to the IP cloud packet discard ratio of 0.05% was assigned. In

Drop-Nofast to the IP cloud author assign the packet discard ratio same 0.05% and enable the fast retransmit and fast recovery mechanism of Server Hallam, while in Drop-fast scenario author have enable the fast retransmit and assign Tahoe to fast recovery mechanism of Server Hallam and have also keep the packet discarded ratio 0.05% same to the IP Cloud. The duration of the simulation is kept 8 minutes for all the scenarios. The resultant and comparative graphs obtained from these scenarios are discussed below.

4.1. Congestion Control Algorithm along FIFO

To compare the graphs obtained from all three scenario i.e., No-Drop, Drop-fast, Drop-Nofast, select the Compare Result from Result menu. Expand the Object Statistics and select the Congestion window size and Sent Segment Sequence Number. The graph can be displayed by first clicking on the Object Statistics under the Global statistics, after that click on your network, in this case click on Choose from Map, Sheffield Hallam University, Server Hallam, TCP Connection and then select your parameter congestion Window Size or Sent Segment Sequence Number To see clear view of the resultant graph, Click Show. The resulting graphs are showing the comparison of all the three scenarios below for Congestion window size Fig 1 and Sent Segment Sequence Number Fig 2.

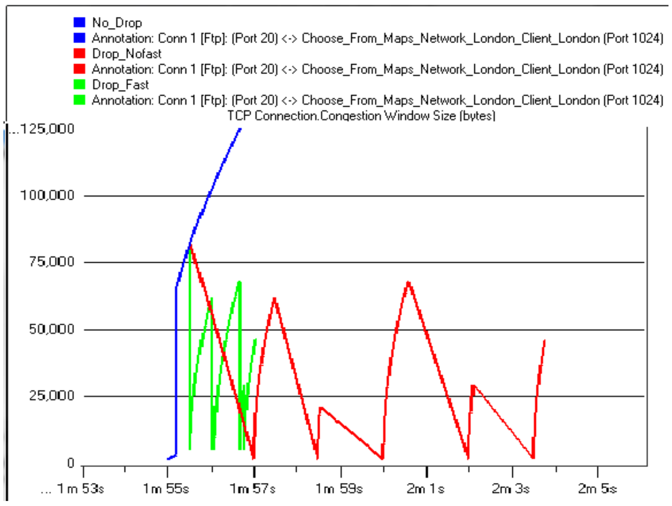

Fig 1: Window Size with FIFO

In fig 4.2 the blue colour is representing the No-Drop scenario, the green colour is representing the Drop-Fast scenario while the red colour is representing the Drop-Nofast scenario. The X-axis is showing the simulation time while the Y-axis is showing the size of the congestion window. After 1min and 56 second the change is shown by all the scenarios, the No-Drop blue line in graph is showing a constant increase in congestion it is represented by a straight line while in the other two scenario there is a fluctuation, it is showing irregular increase and decrease in the size of congestion window transmission. In Drop-Nofast the stability of the congestion window size transmission is low as compared to Drop-Fast. The packet discard ration of 0.05% is also assigned to the IP Cloud, and the fast recovery and fast retransmit algorithms are enabled, so it takes time to stable. The green colour is showing that the Drop-Fast recover soon because the recovery Tahoe is assigned, and retransmission is enabled in this scenario.

4.1.1. Sent Segment Sequence Number Along FIFO

In fig 2 the No Drop and the Drop Fast is showing approximately the same growth in Sent Segment Sequence Number with increase in traffic while segment sequence number has slowest growth in Drop_NoFast Scenario using the FIFO queuing Mechanism on the links connecting the routers to the IP Cloud. With every drop-in size of the congestion window one can see the change in graph in Sent Segment Sequence Number.

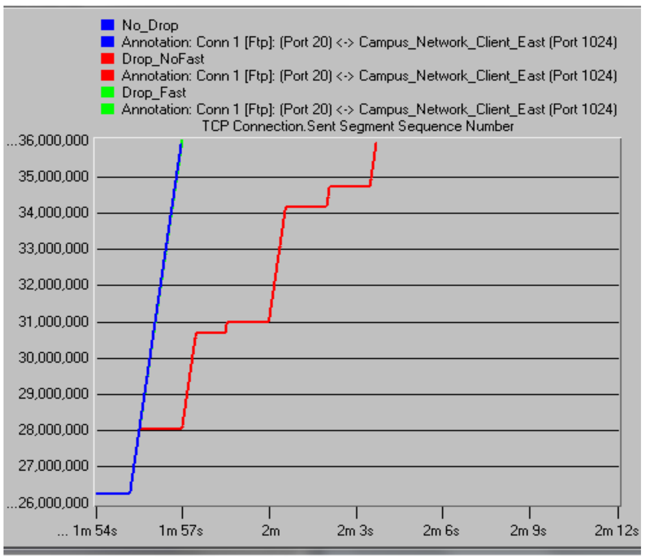

Fig 2: Sent Segment Sequence Number with FIFO

4.1.2 Graphs obtained from congestion control algorithms along PQ

The same three scenarios No_Drop, Drop_Fast and Drop_NoFast have been implemented with the PQ and the results were obtained on the basis of the same matrices (Congestion Window and Sent Segment Sequence Number) and were compared, the compared graphs are discussed and given below.

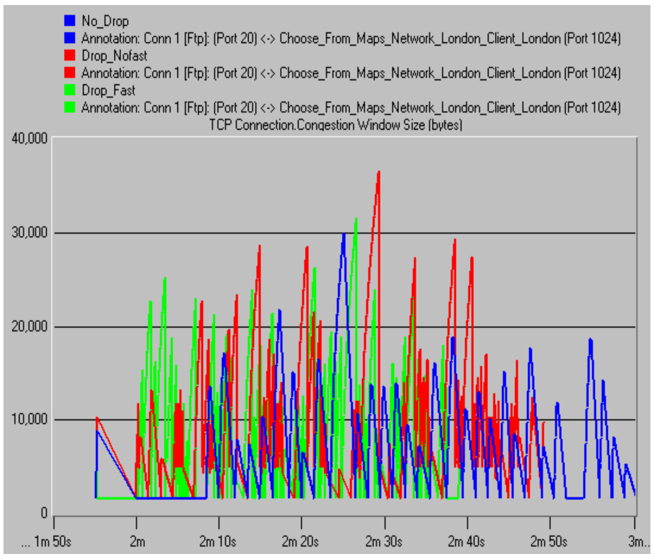

Fig 3: Congestion window size with PQ

Fig 3 shows that a change in the three scenarios can be noticed after 1 minute and 55 seconds has elapsed, the congestion window size graph show fluctuation for all the scenarios, the congestion window is increasing and then decreasing up to 1. The No-Drop scenario has taken more time to maintain a constant congestion window size for transmission, while in the rest of the two scenarios the Drop-Nofast scenario taken more time to maintain a constant congestion window size transmission. In the Drop-Fast scenario the congestion window drops and is then increased and stabilize early compared to other two scenarios.

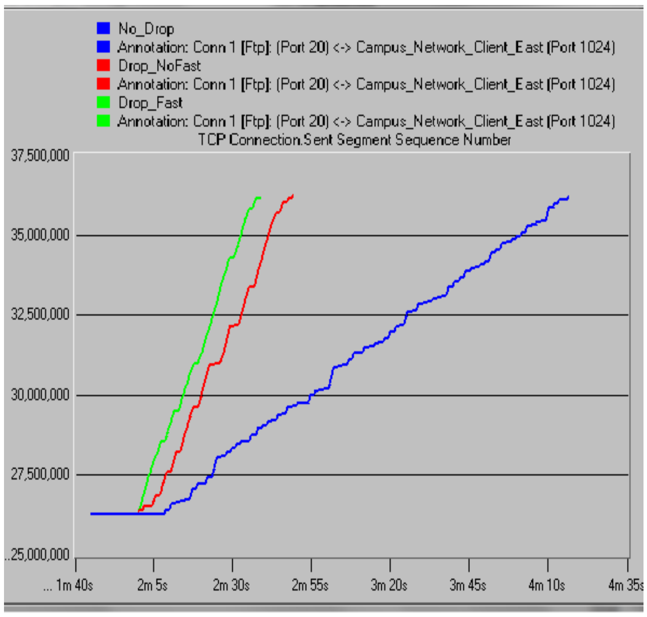

Fig 4: Sent Segment Sequence Number with PQ

In fig 4 the graph is showing that the Sent Segment Sequence numbers of the three graphs are same up to 1 minute and 54 seconds. After that a change in all three scenarios can be examined. With every drop in the congestion window size, the Sent segment sequence number is decreasing. After 1 min and 54 seconds it is observed that during transmission after the lost packet, the segment data is delayed until retransmission timer expires. The Drop_Fast has given the best result for sent segment sequence number as compared to other two scenarios by sending more in less time compared to other two scenarios. With the passage of time the number of Sent Segment Sequence Number of the No_Drop starts decreasing.

4.1.3 Graphs obtained from congestion control algorithms along WFQ

The same three scenarios No-Drop, Drop-Fast and Drop-No Fast have been implemented with WFQ and the results were obtained on the basis of the same matrices (Congestion Window Size and Sent segment Sequence number) results were compared, the compared graphs are discussed and given below.

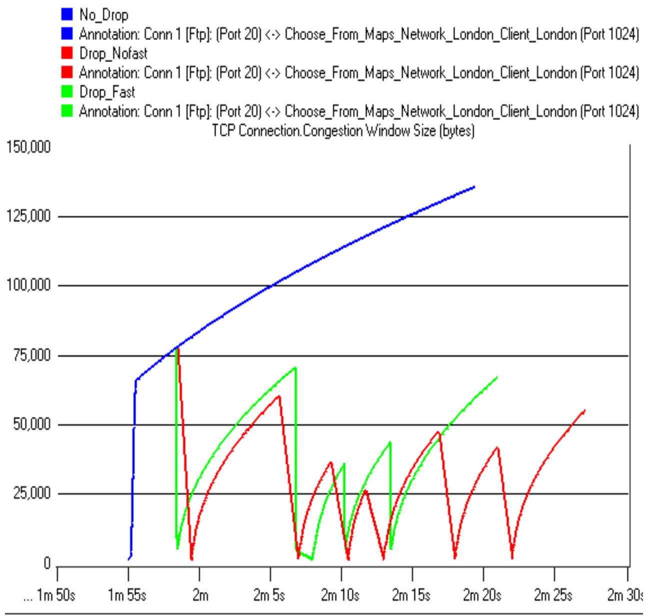

Fig 5: Congestion window size with WFQ

In figure 5 the peaks are showing that the transmission of the congestion window size is high and is then decreases dramatically, the No-Drop is showing a constant increase after 1 mint and 55 sec of the simulation and is taking more time to stabilize transmission compared to all other scenarios. The congestion window size of the Drop-fast become constant after 2 mint and 18 sec, while the Drop-No-fast was the slowest one in all which took more time for stabilizing the congestion window size in last. The fluctuation is showing that the congestion window is decreasing and then again grows up.

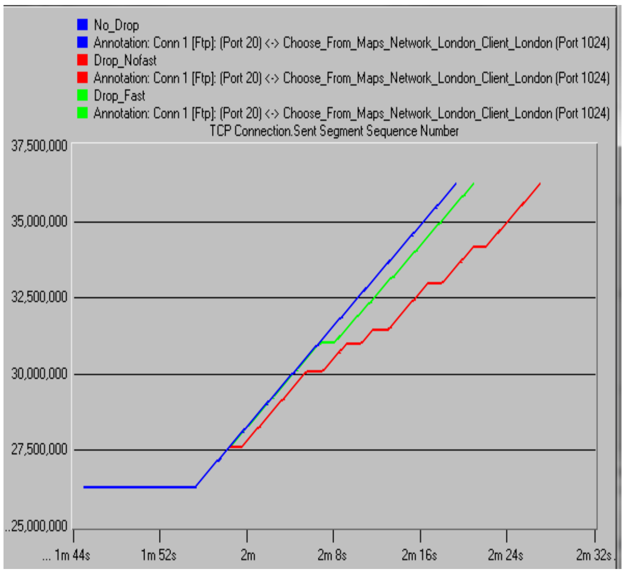

Fig 6: Sent Segment Sequence Number with WFQ

In fig 6 the change in the sequence number can be seen after passing of 1 minute and 58 seconds, one can see the change in segment sequence number of the three scenarios, with every drop in the congestion window has effect on the Sent Segment Sequence Number. The Drop-No fast is showing the slowest growth in number of Sent Segment Sequence Number.

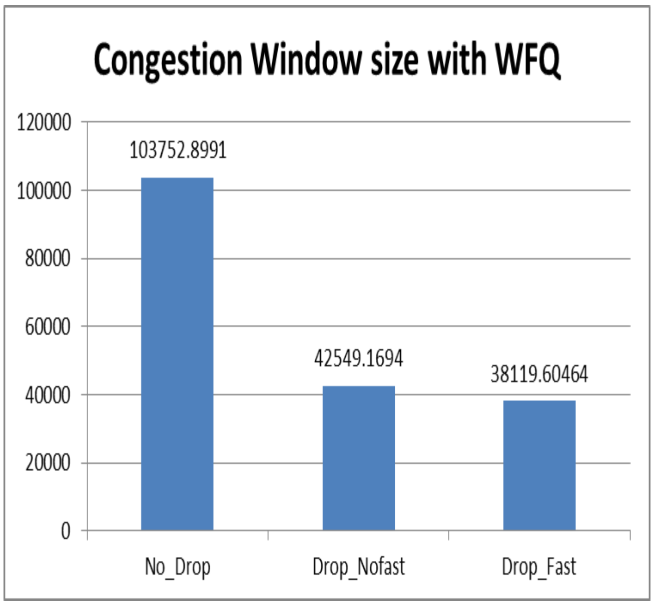

Fig 7: Congestion Window size with WFQ

The above fig 7 is clearly showing the average of the congestion window size growth of the three-scenario using WFQ as queuing mechanism, it is clear from fig 8 that the Drop-fast stable the size of congestion window size quickly so suffer less congestion. While the No-Drop show that it has taken more time to stabilize its congestion window size for transmission.

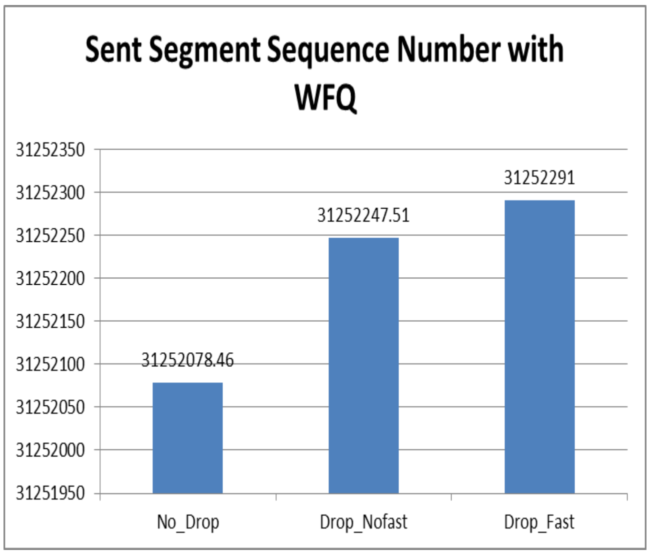

Fig 8: Sent Segment Sequence Number with WFQ

Observing the fig 8, the data is showing that the No-Drop is having the less increase in Sent Segment Sequence number as compared to other to scenarios Drop-No-fast and Drop-Fast. The data is clearly showing that the No-Drop is having the lowest increase in the Sent Segment Sequence number. That’s because with every drop-in size of the congestion window result drop in the sent segment sequence number while the Drop-Fast is giving the best result for Sent Segment Sequence Number.
Congestion Window with FIFO.

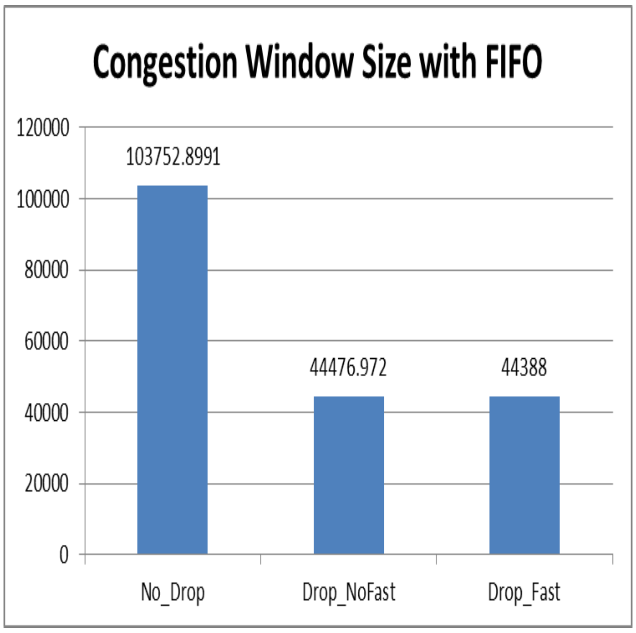

Fig 9: Congestion Window Size with FIFO

In fig 9 one can clearly see that Drop-fast takes less time to stable its congestion window for transmission, in Drop-fast the transmission is low, while the No-Drop has taken more time to stable its size of congestion window.

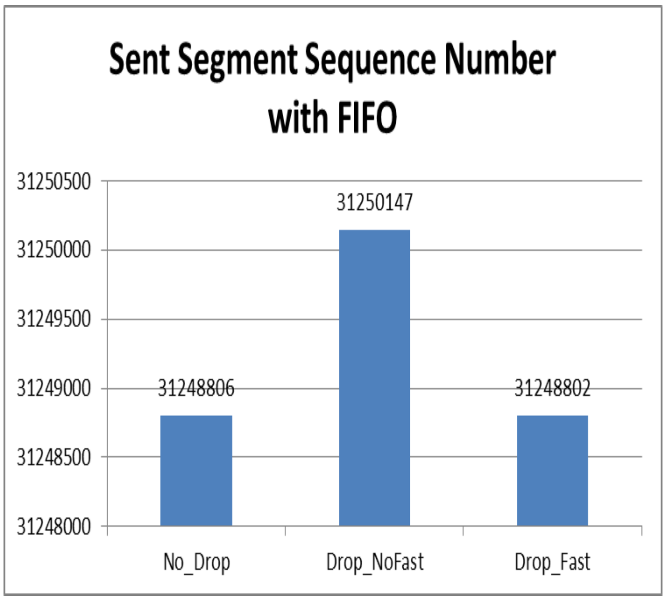

Fig 10: Sent Segment Sequence Number with FIFO

The excel graph in Fig 10 is showing that the No_Drop and Drop_Fast is having approximately the same increase in Sent Segment Sequence Number using FIFO as a queuing mechanism while the Drop_NoFast is having the highest increase in Sent Segment Sequence Number.

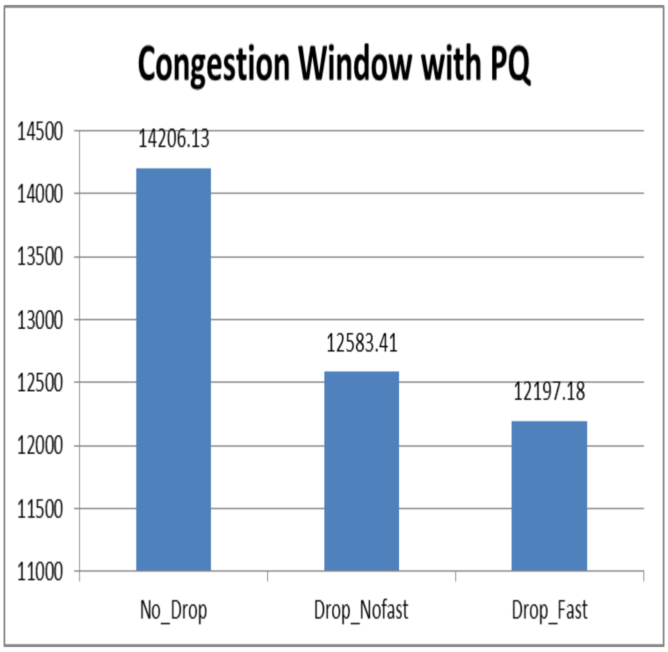

Fig 11: Congestion Window Size with PQ

Fig 11 is clearly showing that the congestion window size of Drop_Fast with PQ is giving good result. The Drop_Fast took less time to stables the size of the congestion window from fluctuation and recover fast compared to all the other scenarios. The No_Drop scenario is showing to be the worst one which took more time to stable its congestion window size. The value of the average of the congestion window size of 12197.18 makes it the best amongst all the mechanisms.

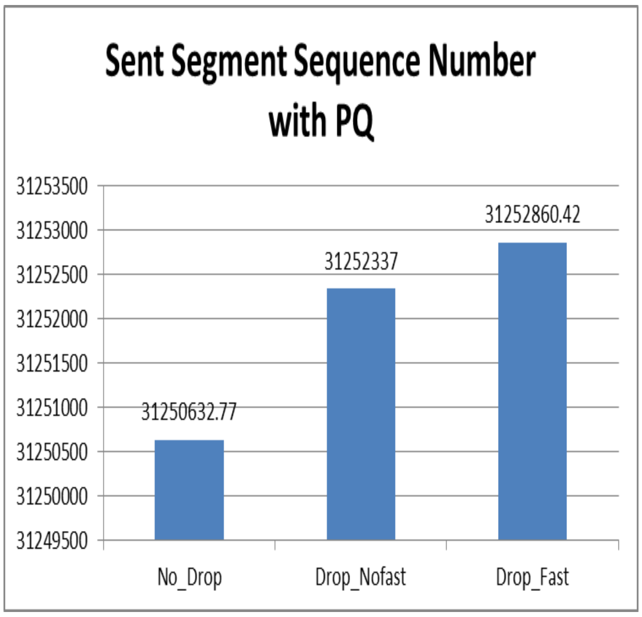

Fig 12: Sent Segment Sequence Number with PQ

The results of the three different scenario in fig 12 is clearly showing that the Drop_Fast Scenario in which the fast retransmit was enable and to fast recovery Tahoe is assigned showing the behaviour of fast recovery of congestion window has also increased the number of Sent Segment Sequence Number compared to all other scenario. Below the table is showing the summary of the graphs data.

Table 1: *Comparison* Table of all Scenarios and showing Average behaviour for all the Scenarios

Elements	No_Drop	Drop_Nofast	Drop_Fast
Avg CWnd with FIFO	103752	44476	44388
Avg CWnd with WFQ	103752	42549	38119
Avg CWnd with PQ	14206	12583	12197
Avg SSSeq no's with FIFO	31248806	31250147	31248802
Avg SSSeq no's with WFQ	31252078	31252247	31252291
Avg SSSeq no's with PQ	31250632	31252337	31252860

Table 1 is showing the average values for congestion window the time it takes to be stable for transmission and sent segment sequence number collected for each scenario.

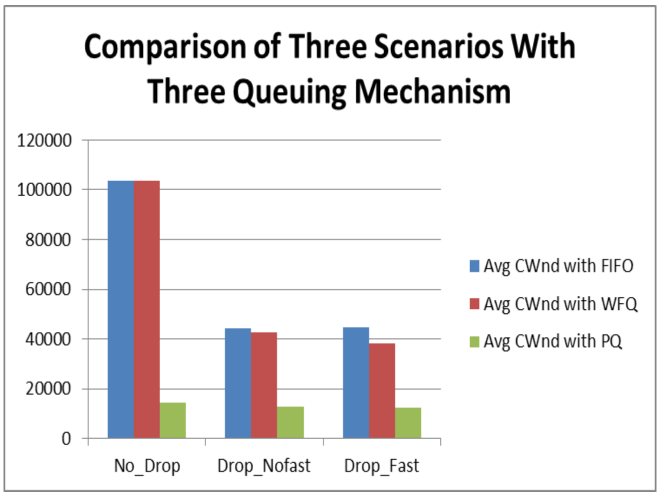

Fig.13: Graph based on average for Comparison of various algorithms using different Queuing Mechanisms

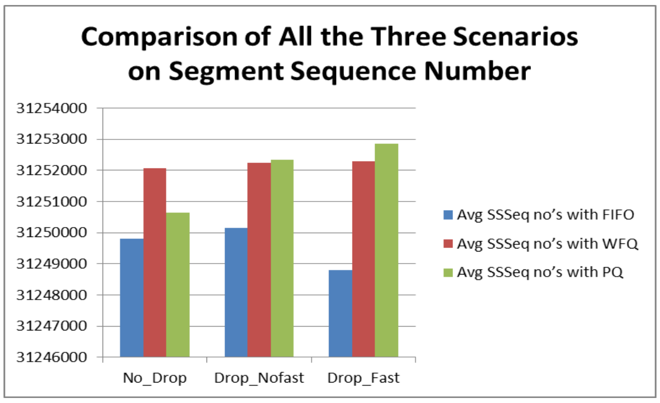

Fig 14: Graph based on average for Comparison of various algorithms using different Queuing Mechanisms

By comparing the graphs of Fig 13, Fig 14 and values in table 1 above Its clearly showing that the Drop-Fast scenario (Scenario in which fast retransmit enable and to the fast recovery assigned Tahoe also to the packet discard ratio 0.05% was assigned) using PQ deliver best result for congestion window size and sent segment sequence number for transmission 1MB file transmission. Using the Tahoe, when the packet is dropped, Algorithms in Tahoe TCP i.e. fast recovery can be used effectively when compared to other mechanisms to minimize the congestion. Tahoe takes very less time to recover from congestion by enabling fast retransmit mechanism and its fast recovery mechanism, thus congestion is reduced, and congestion window is increased and become stable in very less time. As a whole the simulation results are showing that the queuing mechanism PQ along Tahoe and enable retransmission appears to be the best combination for controlling congestion on TCP. The Overall summary of all these graphs is that the queuing mechanisms have an impact on the congestion control algorithms of TCP. The sent segment sequence number increase and decrease in different scenarios and the increase and decrease in congestion window size of these are clearly showing that there is still a chance of improvement

5. CONCLUSION AND FUTURE WORK

In this research, we have concluded the discussion on the basis of literature review and resultant graphs that are obtained from simulation. The most difficult part of the dissertation was calculating the performance of experiment and relating results to the relevant literature review. A lot of work was done on this hot topic and is gaining more attention because of the increase in the number of internet users, still limited research work is carried on congestion control algorithms along queuing mechanism.

The topic was broad, so authors had to divide the study into 4 parts; Study of TCP, related study of congestion control algorithms, queuing mechanisms and implementation of TCP congestion control algorithms in OPNET. The investigation of the congestion control algorithms along the queuing mechanism was very challenging but knowledgeable and interesting for authors because of the challenges that authors faced during learning process and later for getting the simulation performance results.
After performing the study of TCP, Congestion control algorithms, Queuing mechanism and OPNET. A simulation was performed, and it was finding out from the resultant graph, that the queuing mechanisms that are configured on the network devices and the congestion control algorithms of TCP that are working for controlling congestion on transport layer for reliable transport of data have an impact on reliability. It was also found that there is still a chance of improvement if appropriate queuing mechanism is used on these devices. I will conclude the discussion by saying that *"the results obtained were showing that the queuing mechanisms have an impact on performance of network"*.

REFERENCES

1. Yun, L., et al. *An improved TCP congestion control algorithm over mixed wired/wireless networks*. in *2009 2nd IEEE International Conference on Broadband Network & Multimedia Technology*. 2009. IEEE.
2. Afanasyev, A., et al., *Host-to-host congestion control for TCP*. IEEE Communications surveys & tutorials, 2010. **12**(3): p. 304-342.

3. Wang, H. and K.G. Shin, *Robust TCP congestion recovery*. Journal of High Speed Networks, 2004. **13**(2): p. 103-121.
4. Martin, J., A. Nilsson, and I. Rhee, *Delay-based congestion avoidance for TCP*. IEEE/ACM Transactions on networking, 2003. **11**(3): p. 356-369.
5. Lai, Y.-C., *TCP-friendly congestion control to guarantee smoothness by Slack Term*. Computer communications, 2007. **30**(2): p. 341-350.
6. Wei, D.X., et al., *FAST TCP: motivation, architecture, algorithms, performance*. IEEE/ACM transactions on Networking, 2006. **14**(6): p. 1246-1259.
7. Anker, T., et al. *TCP-friendly many-to-many end-to-end congestion control*. in *22nd International Symposium on Reliable Distributed Systems, 2003. Proceedings*. 2003. IEEE.
8. Ruddle, A., C. Allison, and P. Lindsay. *Visualising interactions between TCP's congestion and flow control algorithms*. in *Proceedings. Eleventh International Conference on Computer Communications and Networks*. 2002. IEEE.
9. Dawson, C., *A practical guide to research methods: A users friendly manual for mastering research techniques and projects*. 2007: How To Books.
10. Tsaoussidis, V. and C. Zhang, *The dynamics of responsiveness and smoothness in heterogeneous networks*. IEEE Journal on Selected Areas in Communications, 2005. **23**(6): p. 1178-1189.
11. Leedy, P. and J. Ormrod, *Practical Research Planning and Design . New Jersey: Pearson Merrill Prentice Hall*. 2005.

12 OPNET (2011) [Online] Jul [cited 2013 July 15]; Available from: http://www.opnet.com/

13 Plato Quotes (2011) [Online] last accessed date 04 July 2011 at: www.cyrket.com/p/palm/com.brighthouselabs.platoquotes/

14 Kothari C R, 2006, Research Methodology: Methods and Techniques, second editions, New Delhi, New Age International (P), Limited, Publishers