A Packet Drop Guesser Module for Congestion Control Protocols for High speed Networks

Ghani-ur-rehman[1], Muhammad Asif[2], Israrullah[3]

[1](Department of Computer Science, Khushal Khan Khattak University Karak, Pakistan)
Ghani84kk@yahoo.com
[2](Department of Computer Science, International Islamic University, Islamabad, Pakistan)
Asif253026@yahoo.com
[3](Department of Computer Science, National University of Computer & Emerging Sciences, Pakistan).
Israrullahkk@yahoo.com

ABSTRACT

Different high speed Transport layer protocols have been designed and proposed in the literature to improve the performance of standard TCP on high BDP links. They are mainly different in their increase and decrease formulas of their respective congestion control algorithm. Most of these high speed protocols consider every packet drop in the network as an indication of congestion and they immediately reduce their congestion window size. Such an approach will usually result in under utilization of available bandwidth in case of noisy channel conditions. We take CUBIC as a test case and have compared its performance in case of normal and noisy channel conditions. The throughput of CUBIC was drastically degraded from 50Mbps to 0.5Mbps when we introduced a random packet drops with 0.001 probability. When the probability of the packet drops increases then the throughput gets decreases. Indeed, we need to complement existing congestion control algorithms with some intelligent mechanisms that can differentiate whether a certain packet drop is because of congestion or channel error thus avoid unnecessary window reduction. In order to distinguish between packets drops, we have developed a k-NN based module to guess whether the packet drops are due to the congestion or any other reasons. After integrating this module with CUBIC algorithm, we have observed significant performance improvement.

KEYWORDS

Congestion, Traditional TCP, Throughput, High speed Protocols, CUBIC TCP and K-NN Technique.

1. INTRODUCTION

High speed networks [1] refer to the networks that usually have higher rate of data transmission such as high speed LAN and Ethernet. The rate may be varying from few Mega bit per seconds (Mbps) to Giga bit per seconds (Gbps). The common applications of high speed networks are telemedicine, videoconferencing and weather simulations etc. High speed network may perform better in the situation where the end system may regulate their flow of data for using the available network resources efficiently without more loads on the system. When there are more loads on the systems then it leads to congestion and throughput collapse. Simply high speed networks are introduced to sends a large amount of data quickly. Network congestion refers to the situation in which the capacity of a network is exceeded by the number of packets sent to it. It may mean load on the network. When this load is keep below the total capacity of the network then it is called congestion control. Congestion is possible in any system that involves waiting. In network

congestion occurs because routers and switches has queue which stores the packets. We can easily understand congestion control by taking an example of congestion control in TCP. TCP [2] is an acknowledged based protocol in which a receiver must sends an acknowledgment to the sender after receiving a packet. Sender can only sent a new packet after an ACK has been received from receiver. One of the important duties of the TCP is congestion control.TCP handle congestion in three steps: slow start, congestion avoidance and congestion detection. This slow start has an exponentially increase. This increase is dictated by the size of the congestion window (maximum number of packets that can be transmitted at a particular time) which starts with one maximum segment size (maximum amount of data that a segment can hold). During the connection creation the maximum segment size is determined by using the option of the same name. When an acknowledgment is received for the send segment the size of the cwnd is increased to 1 MSS. As the name implies the window start slowly and increases exponentially. For example the sender start with congestion window is equal to one maximum segment size means that sender can sends only one segment. For example we have seven segments, when an ACK is received for segment 1, then the size of the window is incremented by 1, means the value of cwnd is now 2 and 2 more segments can be sent. When an ACK is received for those two segments, the size of the window is incremented by one MSS for each segment. When all seven segments are ACK the congestion window is equal to 8. When there is delayed acknowledgement then the size of the window is incremented less than power of 2. Slow start cannot continue infinitely and stop when it reach threshold. The size of the window in slow start phase is increased exponentially. To avoid congestion, TCP defines congestion avoidance algorithm, which uses additive increase? When cwnd size reaches threshold, slow start stop and additive increase starts. In Additive increase algorithm the size of cwnd is incremented by 1 additively until congestion is detected. If congestion occurs, the size of the cwnd must be decreased. When sender knows that congestion has occurred, it retransmits a segment. The segment can be retransmitted for two reasons. The first one is when timer times out and the second is by receiving three duplicate ACKs. The threshold's size is halved due to these reasons. For these two reasons the size of the threshold is reduced to one half (multiplicative decrease). Thus, there are two choices for the TCP:

1) If a time-out happens, the possibility of congestion is very high. Here TCP has a strong reaction.
- Set the value of threshold to cwnd/2.
- Set cwnd to the size of one segment.
- Start with slow-start step again.

2) If three duplicate ACKs are received, the chances that congestion occurs are less. A network segment may have been dropped but not all the segments, because three duplicate ACKs are received for the segments after this one show that they are received safely. This is called fast transmission and fast recovery. For fast retransmission and fast recovery the size of the congestion window must be four packets or more. In this case TCP has not a stronger reaction:
- Set the value of threshold to cwnd/2.
- Set cwnd to the value of threshold.
- Start the congestion avoidance phase.

Congestion detection may occur in one of the followings two ways.

(1) If detection is due to first case, a new slow start phase begins.

(2) If detection is due to second case, a new congestion avoidance phase is begins. The increase and decrease formula for TCP is given below.

1. Increase when all acknowledge has been received for the window means fully acknowledged:

$$cwnd = cwnd + 1.$$

2. Decrease when acknowledgment for message is not received means message is lost:

$$cwnd = cwnd - cwnd/2.$$

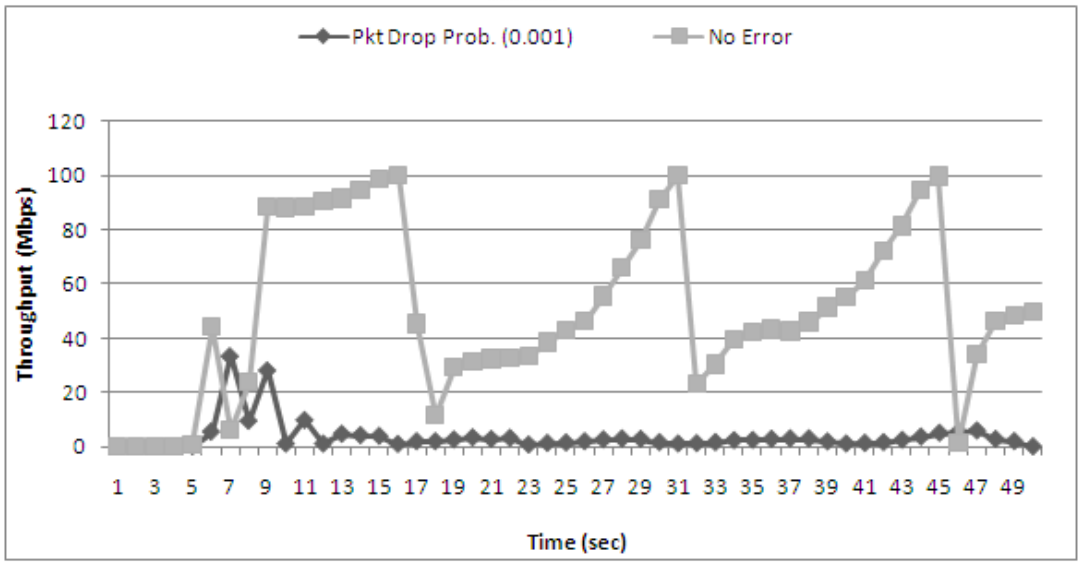

Figure 1: This shows how throughput of CUBIC is affected with random packet drop probability of 0.001

As TCP is not best suitable for high-speed networks, for this purpose different high speed network protocols are come into existence also called the variants of the TCP. These high speed protocol are BIC, CUBIC, FAST, Xcp, TCP VEGAS, SCALABLE and HSTCP etc. But the main problem with these High speed Protocols is that they consider every packet drop as an indication of network congestion. The performance of CUBIC protocol in term of throughput, congestion window and link utilization under normal network condition is fine. When packet drops randomly, then CUBIC Protocol performance in this case is degraded and does not achieve the desire bandwidth. With error rate of 0.001, the performance of CUBIC was drastically decreased as shown in Figure 4. So we incorporate k-NN technique in CUBIC protocol and the performance is much improved.

The rest of the paper is organized as follows: the related work is presented in Section II. Section III contains the pro- posed work. Results are discussed in Section IV. We conclude the paper in Section V.

2. RELATED WORKS

As we know that Conventional TCP [1] is not a good candidate on high bandwidth delay product link. TCP's congestion control algorithm takes long time to take benefit of large bandwidth and to recover from loss.TCP has the followings main drawbacks. The first main problem with TCP is that it considers the entire packet drop due to the network congestion. When a packet is lost in the network it cut the size of the window into half. If there is further packet drop in the network it again cut it into half. All packets drop may not be due network congestion only but it may be due some type of noisy errors. The second drawback with TCP is that when the RTT from source to

the destination and back increases it operates slower and slower. A number [1], [2] of high speed protocols are proposed to improve the traditional TCP performance on high speed network. The high speed protocols can be further classified into two types, loss-based and delay based. Protocols that depend on packet drops due congestion detection are referred to as Loss-based and delay based is those protocols that depend on queuing delay to collect information's about congestion detection. FAST is the only delay based protocol. Loss-based are further classified into two types: first, those that take the previous congestion window in consideration for calculating the next congestion widow, secondly, those that are time-based, since last packet dropped for calculating the next congestion window. The driving force behind developing BIC TCP was to improve the TCP's effectiveness on links with high bandwidth delay product. BIC TCP uses two schemes of window size control. Additive increase and binary search increase are two scheme used by BIC. In Binary search increase scheme the congestion control is consider to be a searching problem. In this, it provides 'Yes' or 'No' feedback for the packet loss. There are two starting points for this search. The first one is the current minimum window size Wmin (size of the window just after the reduction). At this point the cwnd is free of any losses. The second one is maximum window size Wmax (the size of window just before the reduction) where the available link bandwidth is utilized by the cwnd. Binary search increase repeatedly calculates the midpoint between Wmax and Wmin called target window size. This midpoint is the set as the current size. It also keeps watch on feedback as packet loss. The feedback being 'Yes' means that the packet is lost. Then the midpoint is set to the new Wmax. But when feedback is 'No' then it means that there is no packet loss. In this case the midpoint is set to the new Wmin. This process continues until the difference between the size of the window just after the reduction and the size of the window just before the reduction is below the preset threshold value called the minimum increment (Smin). This process is called binary search increase. Binary search increase is more aggressive at the start, at a time when the difference between the current window size and target window size is very large. Similarly, for this difference being small, the binary search increase is less aggressive. One of the important properties of the protocol is that its increase function is logarithmic. It can also reduce the chances of the packet loss. The main advantage of the binary search increase is that it provides a concave response function [3], [4]. Mixed with additive increase, the binary search increase gets faster convergence and RTT-fairness. With a large distance from the current minimum to the midpoint, the window size may be increased to that midpoint, resulting in more stress that the network observes. When the current window size's distance to a target in the binary search increase is larger than the Smax, the window's size is increase by Smax, Smax being maximum increment. This is done until the distance becomes less than Smax at time the window increases to the target after a large amount of window reduction. In the beginning this policy increases the window linearly and then logarithmically. The combination of binary search increase and additive increase is called binary increase. When the window is large and multiplicative decrease policy is mixed with the binary increase becomes an additive increase. When the size of the window is larger then there is a large reduction in the multiplicative decrease and hence long additive increase period. But for the size of the window being small then it is approximately equal to the binary search increase and hence shorter additive increase period. In Multiplicative decrease when packet lost is happen then there will be a reduction in the size of the cwnd by a multiplicative factor of β, which is commonly 0.875 used by others protocols. Of the many advantages of the BIC TCP, the main advantage [6] is that due to the use of the binary increase policy it uses the bandwidth in the most optimum way. CUBIC can be regarded as the extension of BIC protocol as it the BIC variant with many new features that add to the usefulness of BIC. The window control of the BIC TCP can further simplified by the CUBIC TCP. Although BIC guarantees good stability, fairness and scalability during current high speed TCP variants, but its growth function is not that good for TCP under smaller RTT and low speed networks. The window control's different phases make it too complex to understand the protocol easily. To discuss CUBIC protocol [5] in detailed the followings terms is used below. Where cwnd: value of the congestion window, C: scaling constant, T: time to denote the

last congestion event, Wmax: it denotes the size of the cwnd, β: multiplicative decrease factor, which is normally 0.8.

$$K = (Wmax\,\beta/C)^{1}/3$$

When an acknowledgment is arrived:

$$Cwnd = C(T - K)3 + Wmax$$

When packet is lost means no acknowledgement is received:

$$cwnd = \beta Wmax$$

The window growth function of CUBIC protocol is cubic function and its shape is very much the same as the growth function of the BIC. The CUBIC growth function grows very fast at Wmax with the origin. It may grow slow when it nears to the Wmax. Around the Wmax the window increment becomes 0. Above Wmax, CUBIC starts checking more bandwidth available in which the window gradually increases at the start and makes its growth faster when it moves away from Wmax. This slow growth ensures modification in the stability of the protocol while the fast growth ensures the scalability of the protocol. It also ensures the intra protocol fairness among different flow of the same protocol due to the cubic function of CUBIC protocol. As the rate of growth is dominated by the elapsed time t, thus cubic function also provides good RTT fairness property [3], [5]. This ensures linear RTT fairness of different competing flows. Limit is imposed on window increment so that it is less than or equal to Smax per second to add something more to the fairness and stability property. Due to this feature the window will grow linearly when it is not near to the Wmax. Under small RTT, the CUBIC is fairer than other high speed networks protocols. Of the many other advantages of CUBIC [6], [7], [8] is that it modifies the fairness characteristic of BIC while keeping its scalability and stability intact. The main disadvantage of the CUBIC is that it is so slower in increasing its congestion window to completely occupy the link. FAST continuously monitors the flow's RTT so that it is able to check the congestion level in the network. The congestion window is updated by FAST and every other RTT based on the RTT observed and which contains the number of packets it tries to maintain in the router queues. FAST is not good a performer for the size of the router queue being less than? When the observed minimum value is not changed it aggressively increases the congestion window. In FAST the congestion control of TCP is composed of four important components. These four components are not dependent upon each other. All these components can be designed separately. The work of data control component is to examine which packet to transmit, the window control examines how many packets to transmit, and the burstiness control examines when to transmit these packets. The estimation component provides some meaningful information. The other three components can make decisions upon this provided information. The estimation component also gives two types of information for the packet being sent. The data control component chooses the next packet to send from three available options: new packet, negatively acknowledgement packet and packet for which an acknowledgement is not yet been received. Window control sends packets in order at RTT timescale. In FAST, the network consists of a set of resources with limited capacity. These resources are transmission link, processing unit and memory etc. Different unicast flow shares the network. These unicast flow is identified by their sources. The main problem faced in this protocol is the unfairness and the instability in small buffer or in long delay networks [9]. HS-TCP was proposed by Sally Floyd [10] especially for large congestion window. It also solves the standard TCP's shortcoming of getting a large congestion window in a situation where the packet loss rate in not too much high. HS-TCP also proposed a little change in the increment and decrement parameters of the standard TCP. It also makes an attempt to improve the lost recovery time of the traditional TCP by bringing some modifications in its AIMD algorithm. This enhanced algorithm affects the higher congestion window. For the

congestion window being lower than the value of the threshold, called low window, then the standard AIMD algorithm is applied. Otherwise it applies high speed AIMD algorithm. The important and demanding task of HS-TCP is to makes its flows more aggressively having no response from receiver or link router. HS-TCP is specifically designed for low loss rate in a high bandwidth environment and tries its best to be more aggressive than standard TCP. Scalable TCP shortened as STCP was proposed by Tom Kelly [11], [12]. It is the enhanced version of High-Speed TCP or HS-TCP. The scalable TCP's basic goal is the improvement of loss recovery time, not catered for in the standard TCP. As mentioned earlier the scalable TCP being the enhanced version of HS-TCP, takes its main idea from HS-TCP. To compare and contrast the scalable TCP and standard TCP and HS-TCP: In standard TCP and HS-TCP connections, the time for packet loss recovery depends directly on the RTT and the size of the connection window. On the other hand, in scalable TCP, the packet loss recovery time depends only upon the RTT and not on the size of the connection window. With all the other changes and modifications, the standard TCP's slow start phase is not modified for the scalable TCP [8]. The scalable TCP increases the size of its congestion more speedily than the standard TCP while decreases it less speedily than the standard TCP. As in HS-TCP, the STCP has set a threshold for the window size and whenever the size of the cwnd is more than this threshold window size, STCP is used otherwise the standard TCP is used. The threshold window's is set to 16 segments, as the default value. STCP is deployed incrementally. Its behaviour is exactly the same as the parent SCPT for congestion window size lower than the threshold. It has the ability to double its sending rate for any other rate in 70 RTTs and thus the name scalable TCP. TCP Vegas was proposed by Habibullah Jamal and Kiran Sultan that uses packet delay instead of packet loss to determine the size of congestion window [13], [14]. Unlike other congestion control algorithms which take proper measures only after a packet drop has occurred, the TCP Vegas is sensitive to increase in the RTT. The TCP Vegas extends its retransmission mechanism in the following ways: with the transmission of each segment, system clock is read and then recorded; at the arrival of ACK, clock is again read. RTT is calculated using this time. The timestamp is recorded for the relevant segment. This is a more accurate RTT, using which the retransmission can be calculated (a). When a duplicate acknowledgement is received, the algorithm confirms whether the new RTT is greater than RTO or not, where RTT is current time minus timestamp recorded. If greater, the Vegas retransmit the segment without waiting for the third duplicate acknowledgement (b). On the reception of a non-duplicate acknowledgement, whether the first or second one after a retransmission, Vegas checks again to see if RTT is greater than RTO; if it is, then the segment is retransmitted. Thus, there are some ACKs, that help Vegas determine whether the timeout should happen or not. Explicit Control Protocol is the feedback-based congestion control scheme. For congestion being there in the network, it applies direct and precise router feedback to avoid it. It is especially developed for the purpose of scalability and generality. Explicit Congestion Notification router is used in this protocol to immediately inform sender about the congestion in the network. It shows better performance in high delay bandwidth product network. XCP uses router-assistance to exactly notify the sender about the congestion. The resource allocation function is divided between a fairness controller and congestion controller by XPC. The duty of the congestion controller is to make sure that flows make use of all the accessible capacity on other hand the duty of the fairness controller is to fairly allocate the capacity to all the flows. The majority of congestion control schemes are not able to perform this division. In XCP, the implementation and explanation of these two resource allocation functions is made possible due to this division.

3. PROPOSED SOLUTION

Proposed solution is based on k-NN and we have developed a packet drop guesser module considering only two parameters i.e. time and congestion window. Of course, we can have several other parameters but that's not taken into account for this work as the two chosen parameters are considered sufficient and the most relevant to the problem in hand. We get information about

network status in the form of two feedback events: a) when we receive an Ack packet, it shows non- congestion as the data is going through. b) When timer expires or Not-Ack packet is received, it shows that network may be congested. So we keep record of these two types of events in our module (i.e. Ack and drop events) and congestion window along with time instant of the event is recorded in the history which will help us in matching a pattern to distinguish a random packet drop caused by noisy channels from genuine packet drops due to network congestion. Obviously, packet drops caused by reasons other than congestion, will be randomly distributed and we will certainly have many ACK events in the neighborhood of such drop events. While packet drops due congestion will have consistent pattern and congestion window will almost be same for all of them. Thus using simple k-NN technique, we can distinguish packets drops caused by congestion from random packet drops. k-NN is a technique which maintains a history of previous instances on the basis of which it estimates whether the packet drop was due to congestion or not. This estimation is done the basis of majority of instances. The k-NN algorithm is presented below.

3.1 Packet Drop Guesser Module

Our Packet drop guesser module work as follows: Input is event and output is true or false. In first step when the type of the event is Ack then it is simply recorded and exit. But when the type of the event is other than Ack, means Drop then it is recoded and if one-third of the maximum size of the history is greater than the current size then it means there is no congestion and return false otherwise it return true means congestion.

Packet Drop Guesser Module

Input: Event
Output: Return TRUE/FALSE
Procedure:

1. *If Event_Type= ACK*
    a. *Record Event and Exit.*
2. *Else*
    a. *Record Event*
    b. *If MAX_SIZE/3 > CUR_SIZE*
        i. *Return FALSE // No congestion*
    c. *Else*
        i. *Result= KNN_Prediction_Algorithm()*
        ii. *Return Result*

3.2 K-NN Prediction Algorithm

When it return true means that there is congestion so k-NN prediction algorithm is invoked in this case which is given below?

KNN_Prediction_Algorithm

Input: Drop_Event, Event_Record, Range
Output: Return TRUE/FALSE
Procedure:

1. *TRUE_Count=0*
2. *FALSE_Count=0*
3. *For each Event in Event_Record*
   If Event is within Range of Drop_Event
   i. *If Event.Flag = TRUE*
   TRUE_Count++
   ii. *Else*
   FALSE_Count++
4. *End For*
5. *If TRUE_Count < FALSE_Count*
   Return FALSE
6. *Else*
   Return TRUE

4. EMPRICAL EVALUATION

4.1 Simulation Parameters

This section compares the performance of the proposed scheme with traditional CUBIC protocol. All the simulations will be performed using NS-2. Table I shows details of varying parameters used in simulation. All the simulations are performed in NS-2 using TCP/Linux code available at [15]. We have incorporated our module into CUBIC protocol as a test case. CUBIC was selected because its performance drastically degraded when random packet drops are introduced [6]. Secondly we used only two nodes called source node X and destination node Y. The link used between these two nodes is 100 Mbps. The simulation parameters are shown in the table below.

TABLE 1. Simulation Parameters

S.No	Parameters	Value
01	No of flows	01
02	Link Delay	64 msec
03	Packet Size	1448 bytes
04	Buffer size	220 Packets
05	Simulation Time	50 sec
06	Minimum Bandwidth	100 Mbps

4.2 Performance Metrics

The followings performance metrics are used in the comparison of the protocols.

• Throughput: Throughput is the main performance parameter of our undertaking. It tells how close to success our results are. It is the rate at which a device sends successfully and can be expressed in term of Mbps.
• Link utilization: It is the percentage of the bandwidth of the link that is currently being used.

• Congestion Window: Shortened as cwnd, the congestion window determines the maximum number of packets that can be transmitted at a particular time. The congestion window is a value read at particular time from CUBIC protocol. The size of the congestion window is however a variable that is gets reduced when the network in noisy.

But in this paper, only throughput is takes into account as a performance metrics for different error rates.

5. RESULTS AND EXPERIMENTS

The performance of CUBIC protocol in normal network condition, CUBIC without k-NN and CUBIC with k-NN technique is compared in this section. For throughput the results are from three different environments with packet drops rate of 0.1, 0.01, and 0.001 randomly. The figure (2) shows the throughput when packet drops rate is 0.1 randomly. An error rate of 0.1 means that there is 1 packet dropped out of every ten packets (needless to say this is an extremely high error rate). Whenever there is no error (normal CUBIC) in the network, then throughput is better as shown. Shown in the below graph are results of the CUBIC without k-NN which as expected and as it should be are almost zero at every point. Next is the results of CUBIC with k-NN technique, these are obviously better than the CUBIC without k-NN results and almost reaching an average throughput of 20 Mbps. These values are very sharp and to present a smoother graph moving average of 5 previous values is also shown. The figure (3) shows the throughput when error rate is 0.01. It means that there is 1 packet dropped out of every hundred packets, which is comparably stable environment than the one with an error rate of 0.1. Whenever there is no error (normal CUBIC) in the network, then throughput is better as shown. But when the error occurs at rate of 0.01, then the throughput is affected almost comes down to 0 Mbps.

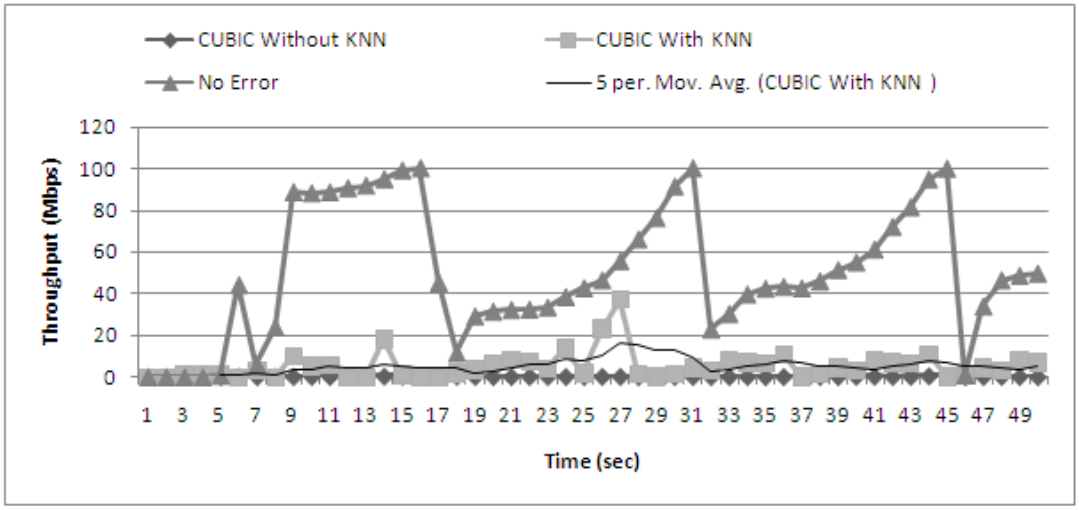

Figure 2. This shows how throughput of CUBIC is affected with random packet drop probability of 0.1

So we incorporate k-NN module to achieve some acceptable level of throughput which is an average of 30 Mbps. Similarly, the figure (4) below shows the throughput when error rate is 0.001. It means that there is 1 packet dropped out of every one thousand packets. Whenever there is no error (normal CUBIC) in the network, then throughput is very fine as shown. But when the error occurs at rate of 0.001, then the throughput is deteriorated badly and is slightly greater than 5 Mbps. So we incorporate k-NN module to achieve some acceptable level of throughput which is

almost 40 Mbps. So the throughput gets better and better when the random packet drop rate is decreased.

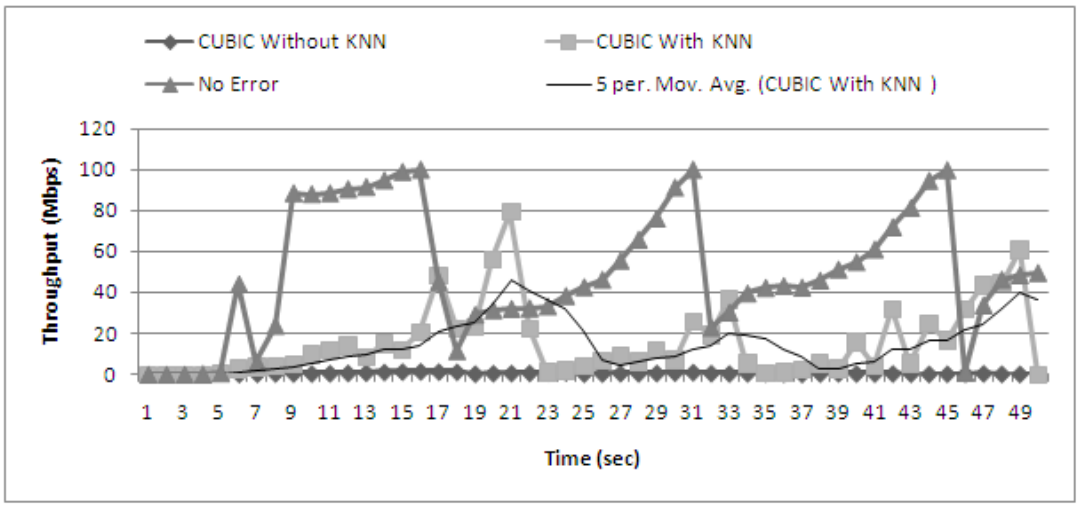

Figure 3. This shows how throughput of CUBIC is affected with random packet drop probability of 0.01

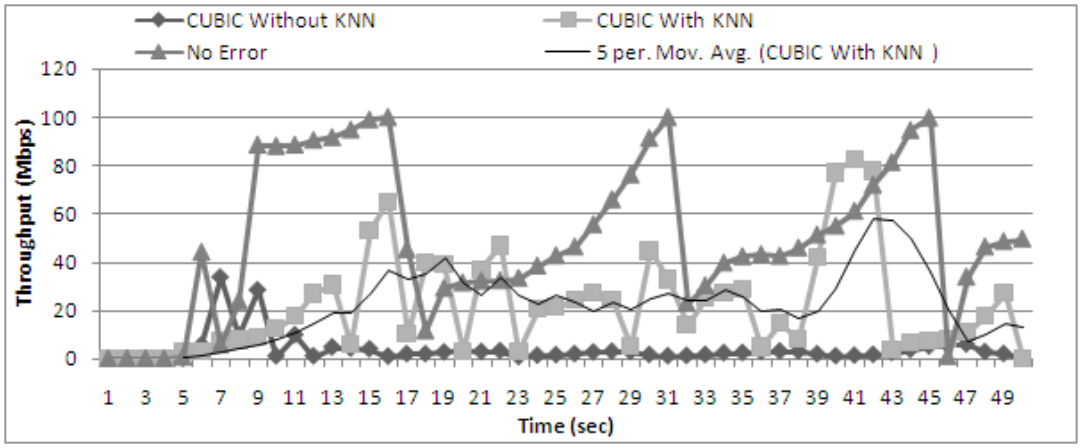

Figure 4. This shows how throughput of CUBIC is affected with random packet drop probability of 0.001

6. CONCLUSIONS

In this paper the performance of the CUBIC protocol was greatly improved by incorporating k-NN module. The performance of CUBIC algorithm in an environment with no errors and the results are seen for its throughput in different error rates. It is found that the performance of CUBIC in this environment was very well and comparable to any other high-speed network algorithm. The same procedure was done in a very noisy environment where the error rate was 0.1, 0.01 and 0.001 and its performance is degraded. So CUBIC's performance was gradually becoming better and better as the environment became less and less noisy and the error rate went on decreasing. Still more modifications and extensions can be made so that we get a better and better protocol.

ACKNOWLEDGEMENTS

All members of International Islamic University have contributed towards various developments reported in this paper. In particular, we would like to thanks to Dr.Sher and Dr.Arif, their contributions in improving the paper.

REFRENCES

[1] Junsoo Lee, Stephan Bohacek, Joao P. Hespanha Networks, University of Southern California, white paper 2007., Katia Obraczka, A Study of TCP Fairness in High-Speed
[2] S. Molnr, B. Sonkoly, and T. A. Trinh, "A comprehensive TCP fairness analysis in high speed networks," Computer Communications, vol. 32, no. 13-14, pp. 1460-1484, Aug. 2009.
[3] L. XU, K.Harfoush, and I.Rhee,"Binary Increases Congestion Control for fast long distance Networks", In Proceeding of the IEEE INFOCOM, Hong Kong, March, 2004.
[4] M.Jehan, G.Radhamani and T.Kalakumari"Experimental Evaluation of TCP BIC and Vegas in MANETs" International Journal of Computer Applications (0975-8887) Volume 16-No.1, 2010.
[5] Pankaj Sharma, "Performance Analysis of High-Speed Transport Control Protocols", Master of Science (Computer Science) Thesis, Clemson University, August 2006.
[6] K.Satyanarayan reddy and Lokanatha C.Reddy " A Survey on Congestion Control Protocols For high Speed Networks" IJCSNS International Journal of Computer Science and Network Security, Vol.8 No.7 July ,2008.
[7] S. Ha, I. Rhee, and L. Xu. Cubic: a new tcp-friendly high-speed tcp variant. SIGOPS Oper. Syst. Rev., 42(5):64-74, 2008.
[8] Injong Rhee, and Lisong Xu"CUBIC: A New TCP-Friendly High-Speed TCP Variant" Department of Computer Science, North Carolina State University, Raleigh, NC 27695-7534 USA, 2006.
[9] C. Jin, D. X. Wei, and S. H. Low, Hegde, S. FAST TCP: Motivation, Architecture, Algorithms, and Performance". IEEE/ACM Transactions on Networking, Volume 14, Issue 6, pp. :1246 - 1259, Dec. 2006.
[10] S. Floyd. "High-speed TCP for large congestion window", Internet Draft draft-floyd-tcp-highspeed-01.txt, February 2003.
[11] Tom Kelly "Scalable TCP: Improving performance in High-speed Wide Area Networks, 2004.
[12] Tom Kelly" Scalable TCP: Improving performance in High speed Wide Area Networks" 2005.
[13] Habibullah Jamal, Kiran Sultan" Performance Analysis of TCP Congestion Control Algorithms". INTERNATIONAL JOURNAL OF COMPUTERS AND COMMUNICATIONS, Issue 1, Volume 2, 2008.
[14] D. M. Lopez-Pacheco and C. Pham" Robust Transport Protocol for Dynamic High-Speed Networks: enhancing the XCP approach" in Proc. of IEEE MICCICON, 2005.
[15] http://netlab.caltech.edu/projects/ns2tcplinux/ns2linux/.

AUTHORS PROFILE

Mr.Ghani-ur-rehman is working as a Lecturer in the Department of Computer Science at Khushal Khan Khattak University Karak. He has more than 2 years of teaching high-speed Networks. He received his BS (CS) and MS (CS) from Kohat University of Science & Technology, Kohat and International Islamic University, Islamabad in year 2006, 2012 respectively. He is also a research scholar in Khushal Khan Khattak University Karak.

Mr.Asif is working as a Lecturer in Government degree College Ahmad Abad Karak. He has more than 2 years of teaching experience of teaching networking subjects. He received his MS (CS) degree from IIU, Islamabad. He is also a research scholar in Government degree College Ahmad Abad Karak.

Mr.Israrullah is working as a Lecturer in Virtual University of Pakistan. He has more than 4 years of teaching networking subjects. He received his MS (CS) degree from National University of Computer & Emerging Sciences, Islamabad. He is also a research scholar in National University of Computer & Emerging Sciences.