IOSR Journal of Computer Engineering (IOSR-JCE)
e-ISSN: 2278-0661, p- ISSN: 2278-8727Volume 12, Issue 2 (May. - Jun. 2013), PP 25-30
www.iosrjournals.org

Improvement of Congestion window and Link utilization of High Speed Protocols Through k-NN Module

Ghani-ur-rehman[1], Muhammad Asif[2], Israrullah[3]
[1](*Department of Computer Science, Khushal Khan Khattak University Karak, Pakistan)*
[2](*Department of Computer Science, International Islamic University, Islamabad, Pakistan)*
[3](*Department of Computer Science, National University of Computer & Emerging Sciences, Pakistan).*

Abstract: *Many high speed protocols of transport layer have been comes into existence and proposed in the literature to improve the performance of Traditional TCP on high bandwidth delay links. All high speed protocols are distinguished on the basis of their incremented and decremented formulas of their respective congestion control algorithm. Many of these high speed protocols consider every packet lost as denotion of congestion in the network and cut their congestion window size. Such an approach will usually result in under utilization of available bandwidth in case of noisy channel conditions. For this reason we take a CUBIC protocol as a test case and compare its performance in noisy channel conditions and normal network conditions. The congestion window and Link utilization of the CUBIC protocol was suddenly decremented, when we incorporate a random packet drops. When the probability of packet lost incremented then both congestion window and link utilization decreases. Thus we need an intelligent technique that distinguish whether a certain packet lost is due to the congestion or any noisy conditions to recover from unnecessary cutting in the window. We have proposed a k-NN based module differentiates whether the packet drop is congestion or any other noisy error. After incorporating this technique in CUBIC protocol, we found better performance improvement.*

Keywords: *Congestion, Congestion window, High-speed Networks, k-NN, Link utilization and TCP.*

I. Introduction

High speed networks [1] are network which has higher rate of data transmission. The most common applications of high speed networks are telemedicine, weather simulations and videoconferencing. Simply high speed networks are especially comes into existence to transmit a large amount of data more quickly. One of the important problems that normally found in high speed protocols is network congestion. Network congestion is the situation in which the capacity of a network is exceeded by the number of packets sent to it. It is not wrong to say that simply load on the network. When this load is reduced to some extent then it is called congestion control. Congestion may occur in any system that involves waiting. In network, congestion occurs because some network devices such as routers and switches has queue which stores the packets. Most high speed TCP variants treat every packet drop as an indication of network congestion and use the decrease formula in reaction. This results in their degraded performance, especially in noisy channel conditions. The objective of this work is mainly to develop some intelligent mechanism that can enable these protocols, CUBIC in this case, to differentiate between packet drops due to congestion and noise/error in the channel. The idea is to make use of history and identify patterns to correctly guess whether the packet loss is due to error in the channel, transmission impairment or congestion in the network. One possible solution is to estimate whether the dropped packet was due to congestion or any other reason so as to decide whether or not to reduce the congestion window, or *cwnd*. We use the k-nearest neighbour algorithm (k-NN) for such packet drop estimation. The k-Nearest Neighbour Algorithm, commonly shortened as k-NN, is a classification method that intuitively classifies unlabeled and non-classified objects based on their closest examples from within a feature space of already classified objects. It is a type of instance-based learning, (other examples of instance-based learning being: weighted regression and case-based reasoning). k-NN is one of the simplest algorithms of machine learning. In k-NN, all instances correspond to points in an n-dimensional Euclidean space. Classification is delayed until the arrival of new instances. Whenever a new instance arrives, it is classified by comparing it with the feature vectors of the different points which are its neighbours, and thus the name k-nearest neighbour. The target function may be discrete or real-valued. This is sensitive to the local structure of the data. The basic and simple version of k-NN is noted for the ease of implementation. It is done by calculating the distance from the test sample to all the neighbouring instances. But this version has vigorous computations, especially for the larger size of k. Over the years, many flavours of k-NN have been presented. To use the most suitable flavour, makes k-NN computationally tractable even for large values of k.

The rest of the paper is organized as follows: Section II contains the related works. Section III contains the proposed work. Empirical evaluation is discussed in Section IV. Section V contains the results and experiments. The paper is finally concluded in Section V.

II. Related Works

TCP [2] is one of the acknowledgement based protocols. When a packet is successfully reached to the destination from source then:

$$\text{cwnd} = \text{cwnd} + 1$$

On other hand when packet or message is lost then decrease in window will occurs.

$$\text{cwnd} = \text{cwnd} - \text{cwnd}/2$$

But the performance of TCP [3] is not satisfactory on high bandwidth delay products link. The congestion control algorithm of TCP takes much more time to get an advantage of large bandwidth and to recover from loss.TCP has the followings problems. The first major drawback with TCP is that it considers every drop of packet is due to network congestion. When the packet does not reach to the desired destination then the size of the congestion window is cut down into half. When the packet drops continuously, then the size of the congestion window will be reduced continuously. For example when the size of the congestion window is 100 packets and a packet drop occur then the now the size of the congestion window is cut down into half means 50 packets. It is not necessary that all the packet drops is due to network congestion but it may be due some noisy errors. When the RTT from source to the destination and back increases then it may be operates slower and slower is also a drawback of TCP.

A number [4] of high speed protocols are designed to improve the standard TCP performance on high speed network. The following are the different types of high speed network protocols such as BIC [5], CUBIC [6], XCP [11], Layered TCP [8], Compound TCP [9] and Scalable TCP [10]. BIC TCP [5] controls the size of the window by using two types of schemes. The first one is an Additive increase and the second one is binary search increase. The congestion control is considered to be a searching problem in binary search scheme. This scheme provide two values Yes and No for the packet loss. This search has two starting points. The one is the current minimum window size Wmin (size of the window just after the reduction). At this point the cwnd do not have any losses. The second one is maximum window size Wmax (the size of window just before the reduction) where the cwnd should utilize the available bandwidth of the link. This search repeatedly calculates the midpoint between Wmax and Wmin. The current size is set to the midpoint. It also keeps watch on feedback as packet loss. Whenever the feedback is 'Yes' means lost of packet. 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. But for the size of the window being small then it is equal to the binary search increase and hence shorter additive increase period. The CUBIC [6] protocol is another High speed variant of traditional TCP. The low utilization problem of the TCP Protocol can be solved through CUBIC protocol, because it uses a cubic function instead of linear increment in window for congestion control mechanism to improve the scalability and stability in fast and long distance networks. As the growth function [7] of CUBIC protocol is always defined in real time so growth function will be independent of RRT.

While retaining its scalability and stability, CUBIC also improves the fairness characteristics of Binary Increase Control Protocol. We also try to find balance between wmin (window size just after reduction) and wmax (window size just before reduction). Despite this, the real time increment in the window extremely improves the friendliness of standard TCP protocol. TCP window growth is much faster than CUBIC protocol. In situation to make the growth rate as similar to that of TCP, it uses a new TCP mode. In this mode it uses the similar congestion control mechanism as traditional TCP protocol while the RTT is not too much long. The congestion control algorithm or CUBIC congestion function is follows:

$$\text{Wcubic} = \text{C}\,(\text{T} - \text{K})\,3 + \text{Wmax}$$

where C is a scaling constant, T is the elapsed time from the last window reduction, Wmax is the size of window before the last window reduction, and $K = 3\sqrt{\ }\,p\ Wmax\beta/C$, where β is a multiplication decrease constant whose value is 0.875. Explicit Control Protocol [8] has congestion control scheme which uses feedback. Whenever there is congestion in the network, then it applies a direct and exact router feedback to avoid it. It is developed especially for scalability and generality functions. It uses router namely explicit congestion notification routers which immediately notifies or tell the sender about the congestion in the network. The performance of this protocol is much better in high bandwidth-delay product link. It also uses router-assistance to precisely inform the sender about the congestion. Layered TCP shortened as LTCP [9] is one of the sender-side adjustments to the response function of the standard TCP and tries its best to makes it more scalable in high speed network. The response function of the cwnd of the Layered TCP is clearly defined in two extents, at macroscopic level and at microscopic level. At macroscopic level, Layered TCP employs an

idea of the layering to speedily and efficiently explore for the available bandwidth. At the microscopic level, Layered TCP extends the present AIMD algorithm of TCP to find out the per acknowledgment performance. The primary purpose of the LTCP is to make the size of the congestion response function in high speed network under the followings conditions: (a) All LTCP flows which have same RTT should be fair to one another. (b) For the window size not being more than threshold then all Layered TCP flows should be fair to the standard TCP flows. This threshold defines some rule under which Layered TCP is found to be so friendly to the implementation of the standard TCP. In high speed network the selection of window scale makes able the receiver to announce the size of window too much large. In order to make certain that Layered TCP is fair to standard TCP, all the new connection of the Layered TCP that starts with only single layer and performs the same as standard TCP. The response function of the congestion window is changed only when this window is increase ahead of the threshold. When congestion window of Layered TCP flow grows ahead of the LTCP window threshold, the increase performance is changed to perform two dimensionally. (a) The layers are increased in the case when the congestion is not practical for more time. (b) The per-ACK congestion window performance of TCP is comprehensive so that a flow can increase its congestion window when it working at higher layer faster than compared to working at lower layer. The Layered TCP also called an ACK-clocked and with incoming acknowledgment LTCP flow changed its congestion window. Though LTCP flow increasing the congestion window forcefully compared to the implementation of common TCP based on the layer at which it work. The layers are added if no congestion is observed over an extended period of time. To do this, a simple method of layering is used. When the current congestion window gets higher than the window corresponding to the last addition of a layer, a new layer is added. The main advantage is that "At macroscopic level, Layered TCP employs an idea of the layering to speedily and efficiently explore for the available bandwidth". At the microscopic level, Layered TCP extends the present AIMD algorithm of TCP to find out the per acknowledgment performance. Scalable TCP shortened as STCP is the modified version of High-Speed TCP or HS-TCP. The basic idea behind Scalable TCP is to improve the loss recovery time. As Scalable TCP is an enhanced version of HS-TCP, takes its main idea from HS-TCP. In standard TCP and HS-TCP connections, the packet loss recovery time depends directly on the Round Trip Time and the connection window size. But in scalable TCP, the time for packet loss recovery depends upon the RTT only. The slow start phase of standard TCP is not modified for the scalable TCP besides all other changes. The congestion window is increased fastly than that standard TCP but decreased slowly as compared to it. Like HS-TCP, the STCP has set a threshold for the window size and whenever the threshold size is less than size of the congestion window , STCP is used otherwise the standard TCP is used. The default value for threshold windows is set to 16 segments. STCP is incrementally deployed. When the congestion window size is lower than the threshold then its behaviour is exactly the similar to the parent STCP. Compound TCP was basically design to satisfy simultaneously the efficiency and friendliness requirement. The main idea is that when the link is under-utilization then the high speed protocol will be aggressive and increase the transmitting rate quickly. As the Compound TCP is too much aggressive, so main problem with this protocol is like TCP unfairness. The core idea of our CTCP is to incorporate a scalable delay-based component into the standard TCP congestion avoidance algorithm. CTCP can be very quick to obtain network bandwidth which is free, by adopting a rapid increase rule in the delay-based component such as multiplicative increase. By employing the delay- based component, CTCP can reduce the transmitting rate when the link is fully utilized. In this way, a CTCP flow will not cause more self-induced packet losses than a standard TCP flow, and therefore maintains fairness to other competing regular TCP flows. So it also improves the RTT fairness compared to standard TCP.

III. Proposed Algorithm

The proposed solution depends on k-Nearest Neighbour (K-NN) technique and we have developed a packet drop guesser module that considering only two parameters i.e. time and congestion window. Of course, we can take other parameters, but that's not considered for this work as the two chosen parameters are consider enough and the most appropriate to the problem in hand. Two feedback events give us information's about network status as follows: a) When an ACK is received; it shows no congestion as the data is going through. b) When timer expires, 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 neighbourhood 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

In our event structure, we keep record of three different parameters Time, Congestion window and Event type. Our Packet drop guesser module work as follows: Input is usually event and output will be in the form true or false. Whenever the type of the event is Acknowledgement (ACK) then it will be simply recorded and exit. But when the type of the event is not Acknowledgement (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 for 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 drop event then k-NN prediction algorithm is invoked 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

IV. Emperical Evualation

4.1 Simulation Parameters

We have compared the performance of the proposed scheme with traditional CUBIC protocol. All the simulations will be performed using NS-2. Table I shows details of various parameters used in simulation. All the simulations are performed in NS-2 using TCP/Linux code available at [12]. 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 [13]. Secondly we used only two nodes called source node A and destination node B. 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

Now the performance Metrics which are used in the comparison of the protocols are namely Link utilization and Congestion window.

• Link utilization: It is the percentage of the bandwidth of the link that is currently being used.
• Congestion Window: The congestion window is the maximum number of packets that can be sent 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.

V. Results & Experiment

We compared two types of results, the performance of CUBIC protocol without k-NN and CUBIC with k-NN technique for two parameters i.e. link utilization and congestion window. The first figure (1) shows the results of congestion window when there is a random packet drop rate of 0.1, 0.01 and 0.001. Here 0.1 means there is 1 random packet drop of every 10 packets sent, 0.01 means 1 random packet drop of every 100 packets sent and 0.001 means 1 random packet drop of every 1000 packets sent. Thus for error rate 0.1, the average congestion window of CUBIC without K-NN is approximately 02 packets shown in black colour. But average congestion window of CUBIC protocol with K-NN is 25 packets shown in gray colour. Similarly for error rate 0.01 and 0.001, the average congestion window of CUBIC protocol without k-NN is 08 packets and 15 packets respectively. But when k-NN module is incorporated in CUBIC protocol the average congestion window will now be improved to 50 packets and 80 packets respectively. It is clear that when the error rate is decreased then the average congestion window gets increased.

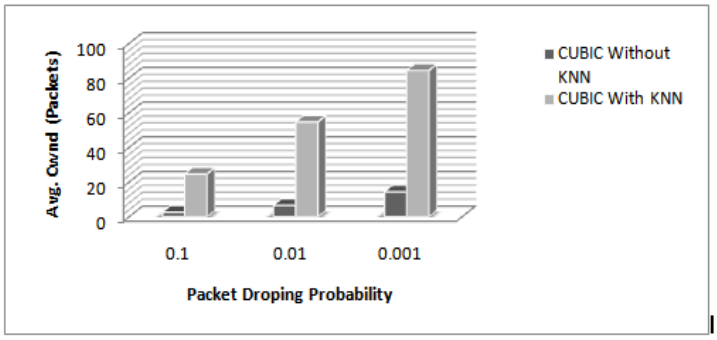

Figure 1. Effect on Avg. congestion window under different packet dropping probability rate

Similarly, the figure (2) shows the link utilization of CUBIC protocol without k-NN module and with k-NN module. For error rate of 0.1 means one random packet drop of every 10 packet sent, 0.01 means one random packet drop of every 100 packet sent and 0.001 means one random packet drop of every 1000 packet sent, so the link utilization of CUBIC protocol without k-NN is 2% for error rate of 0.1, 3% for error rate of 0.01 and 5% for error rate of 0.001 respectively. But when k-NN module is incorporated in CUBIC protocol then the link utilization is improved up to 8% despite error rate of 0.1, 20% for error rate of 0.01 and 40% for error rate of respectively. It means that when the probability of random packet rate is decreased then the percentage link utilization of the CUBIC protocol is increased. It is clear that by incorporating k-NN module in CUBIC protocol the performance is much more increased in the form of average link utilization and average congestion window.

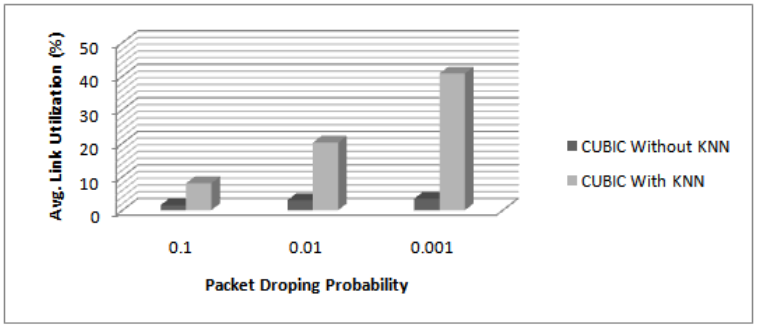

Figure 2. Effect on Link utilization under different packet dropping probability rate

VI. Conclusions

In this paper by incorporating k-NN module in CUBIC protocol, the performance was much improved. The performance of CUBIC algorithm in an environment there is no errors and the results are seen for its congestion window and link utilization. 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 used 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 medications and extensions can be made so that we get a better and better protocol. The main drawback of the CUBIC protocol is that it can't notify about the congesting immediately.

References

[1]. W. R. Stevens, TCP/IP Illustrated, Volume 1, Addison-Wesley, Reading, MA, November 1994.
[2]. Tomoya Hatano, Hiroshi Shigeno and Kenichi Okada, "TCP friendly congestion control for high-speed network", IEEE, 2007.
[3]. M. Allman and V. Paxson, "TCP Congestion Control", Internet Engineering Task Force, RFC 2581, April 1999.
[4]. Y.-T. Li, "Evaluation of tcp congestion control algorithms on the windows vista platform," Tech. Report SLAC-TN-06-005, Stanford University, Tech. Rep., June 2006.
[5]. L. Xu, K. Harfoush, and I. Rhee, "Binary increase congestion control for fast long distance networks," in INFOCOM, 2004.
[6]. I. Rhee, L. Xu. CUBIC: A New TCP-Friendly High-Speed TCP Variant. In Proc. PFLDnet, 2005.
[7]. Tomoya Hatano, Hiroshi Shigeno, Kenichi Okada, "TCP-friendly Congestion Control for High Speed Network", International Symposium on Applications and the Internet- SAINT'07, pp.10, 2007.
[8]. Habibullah Jamal, Kiran Sultan" Performance Analysis of TCP Congestion Control Algorithms". INTERNATIONAL JOURNAL OF COM- PUTERS AND COMMUNICATIONS, Issue 1, Volume 2, 2008.
[9]. K. Tan, J. Song, Q. Zhang, and M. Sridharan, "A compound tcp approach for high-speed and long distance networks," in INFOCOM, 2006.
[10]. T. Kelly, "Scalable TCP: Improving Performance in High-Speed Wide Area Networks," ACM SIGCOMM Computer Communication Review, Volume 33, Issue 2, pp. 83-91, April 2003.
[11]. 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.
[12]. 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.
[13]. http://netlab.caltech.edu/projects/ns2tcplinux/ns2linux/.