ISSN: 2278 – 1323
International Journal of Advanced Research in Computer Engineering & Technology (IJARCET)
Volume 2, Issue 4, April 2013

Balanced window size Allocation Mechanism for Congestion control of Transmission Control Protocol based on improved bandwidth Estimation.

Dusmant Kumar Sahu[1], S.LaKshmiNarasimman2, G.Michale[3]
[1]P.G Scholar, [2]P.G Scholar, [3]Professor &Assistant Professor
Bharath University, Chennai, India

***Abstract*- TCP is the widely used protocol for its reliable data communication over the network. Though it is used for enabling communication over the large network, it has some incapability in handling continues data transmission with reduced congestion over the internet. One of the most problems of TCP is fixing window size that turned to have a balanced allocation of buffer in accordance to handling congestion and traffic in a network. In this paper we present a framework for TCP congestion, called "Delay and Bandwidth based estimation", which defines an exact window size that to be assigned to have balanced configuration of bandwidth with respect to delay. It tries to find out the round-trip-time to decide to fix window size by increasing or decreasing the boundaries of a buffer. Firstly we have discussed overview of TCP and its AQM. Then steady state of the window size is investigated via normalizing bandwidth factor. Finally simulation is conducted to compare with traditional TCP-Vegas under different network environments.**

Keywords: **Transmission Control Protocol (TCP), Congestion, Delay and Bandwidth estimation, RTT.**

I. INTRODUCTION

The Internet plays a significant role nowadays in our lives. The Two main protocols are being implemented in the transport layer of the Internet, namely, UDP (User Datagram Protocol) and TCP (Transmission Control Protocol). They are distinguished by their connection type. UDP is a connectionless protocol that suits for multimedia transmissions; on the other hand, TCP is a connection-oriented one that is designed to provide a reliable transmission policy in an unreliable network, which may vary due to different users, routers, bandwidths, or other cases.TCP lies between the application layer and network layer and serves as the intermediary between the application programs and the network operations. TCP, like UDP is a process-to - process protocols.

TCP uses flow and error-control mechanisms at transport level. In recent studies [12], [5] pointed out that 80% of the Internet traffic is TCP-based, which is the main problem and causes congestion. To avoid this problem, controlling the sending rate of data packets is necessary and the issue has been discussed and developed over the past quarter of a century. A major breakthrough was achieved by Jacobsen [8], [9], who proposed a mechanism called Tahoe, which made the congestion window change dynamically. In Tahoe, the essence is to make the window size increase gradually until congestion is detected, and then it resets the window size to one and starts to increase it again. After that, plenty of studies were done to reach better performance based on this idea.

Two famous ones are Reno [9] and Vegas [1].Another approach proposed [3] an idea that uses the RTT to adjust the window size directly. Also in [11] and [12], the predefined thresholds were dynamically adjusted to make TCP-Vegas more adaptive. On the other hand, some have put emphasis on the router, which is in charge of delivering packets from input links to output links rather than the sender. The traditional, simplest, and most intuitive queuing method implemented at the router is so-called Drop-Tail [2], which drops the packet only when the buffer is full. Recently, AQM (Active Queue Management) has been the main topic in this area. As implied in the name, researchers intend to develop another mechanism to manage the queue more actively. RED (Random Early Detection) is one of the most popular ones, where one uses a marking probability to randomly drop the incoming packet before the occurrence of congestion [6].The past few years have brought analytical models into TCP/AQM systems. Given different analysis methods, researchers can investigate the performance of previous works and then rectify results afterwards. This idea motivated this paper to work on a probe into the TCP/AQM design [7].

A new TCP framework with appropriate AQM mechanisms is developed and analyzed for its feasibility and reliability. The details of the proposed TCP mechanisms will be explained in the following sections. In this paper, a defect of TCP-Vegas, which has been identified in recent years, is rectified. The

model called bandwidth-based TCP is developed to adjust window size according to the estimated bandwidth (transfer rate). Different operating systems and/or protocols of the connecting computers and other devices are available with the proposed scheme, thus one can conclude that the environment is heterogeneous. Some control theories are quoted here to find out the equilibrium points where the state variables will be in steady state first, and then, linearization is used to deal with such a nonlinear system, and finally, stability can be examined by inspecting the small neighborhood of these equilibrium points. With the afore-mentioned analysis, simulations are used to demonstrate that the proposed work has some advantages over TCP-Vegas.

II. TCP-VEGAS AND ITS ISSUES

TCP-Vegas [9] and TCP-Reno [1] are the two algorithms discussed briefly; here we have some detailed explanation of TCP-Vegas and its drawbacks [3]. These two algorithms are designed to control congestion which is caused during transmission of data by using reliable protocol called TCP. Although TCP gives a mechanism for handling congestion by using flow control through adjusting sender's windows size still it is not optimal to give a better throughout and minimized packet loss. TCP-Reno [9], [1] is been designed to detect the congestion after it is happened in a large network and it will not give an effective mechanism to manage it. When TCP-Vegas is compared to TCP-Reno, TCP-Vegas gives better performance well under heavy network load and it detect predicts congestion early it appears by estimating its RTT and delay in getting acknowledgment. The flow control mechanism in TCP is windows-based; here the destination node sends acknowledgment for packets that are received correctly based on sequence number it follows. Generally a sender keeps a variable called window size that calculates the maximum number of outstanding packets that have been transmitted but not yet been acknowledged. Based on this acknowledgment time it decides to increase or decrease windows size. When this buffer is got exhausted, the source has to wait until it gets the reply form the destination node before sending a new packet. So it's very important to keep track of window size when we are designing a new algorithm for handling congesting over TCP.

The figure Fig.1 depicts the so-called window size allocation of TCP over the internet. In this figure, initially the sender fixes to send four packets without any acknowledgment received back from the receiver until the fifth transmission; the current window size is determined to be four. It says number of outstanding packets in the network that have been transmitted but have not been acknowledged from receiver. Once it gets an acknowledgement form the receiver the algorithm increases window size until it recognizes congestion or relay in receiving corresponding acknowledgment. There are three algorithms called Taheo, Reno and Vegas based on congestion size and change the adjustment of their window size by self-generated congestion in order t achieve higher throughput. TCP-Vegas is an algorithm which concentrate on its round-trip-time (RTT) to vary the adjustment in window size.

TCP-Vegas uses three techniques to increase throughput but lower losses than TCP-Reno. These modifications are summarized as follows [1], [2].

1. New Retransmission Mechanism-Uses more accurate RTT estimation to decide to retransmit a dropped packet segment.
2. Congestion Avoidance Mechanism-Gives a method to calculate and control the amount of extra data this connection has in transit.

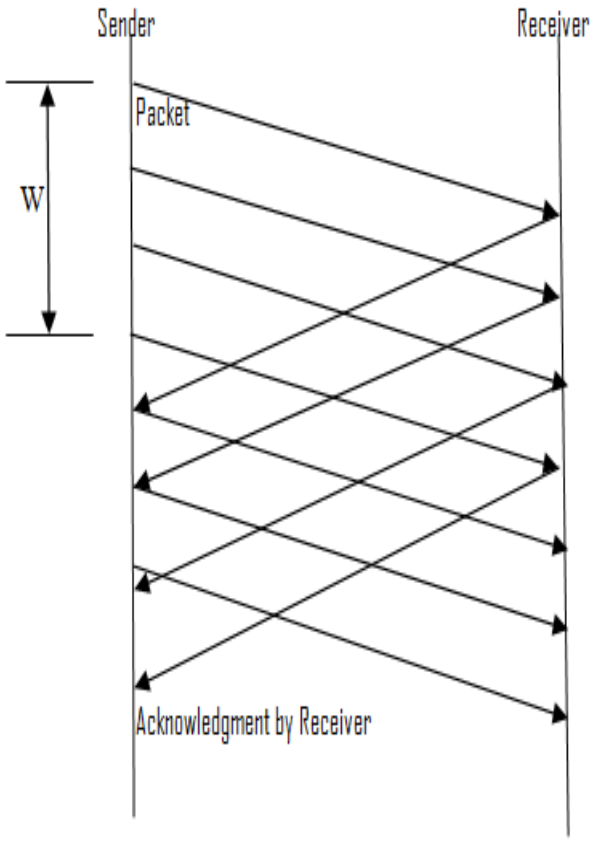

Fig.1Example Window Size Allocation.

3. Modified Slow Start Mechanism-Modifies TCP's slow-start to avoid packet losses while trying to find

the available bandwidth during the initial use of slow-start.

Let RTTmin be the minimum of all measured RTTs (commonly the RTT of the first packet)

If not overflowing the connection, then

ExpRate = Congestion Window / RTTmin

Source calculates current sending rate (ActRate) o nce per RTT

Source compares ActRate with ExpRate

- Diff = ExpRate - ActRate if Diff < a
- increase Congestion Window linearly else if Diff > b
- decrease Congestion Window linearly else
- -leave Congestion Window unchanged

The Vegas algorithm is explained in above procedure, as we can notice that TCP-Vegas uses a more identical method of window size adjustment to prevent packet losses. At first it determines RTT form the first acknowledgement got from the receiver. Then this algorithm uses it as a reference variable, called RTTmin. Henceforward, every RTT is compared with its initial data rate called ActRate. Each time it is compared with its ActRate, ExpRate is a predefined rate and when the Diff is less than a predefined index, called its threshold (ExpRate) then it implies the network is so smooth that the RTT is close to the RTTmin. So that the sender can increase window size by one, if it is grater than threshold value the sender decreases window size by one otherwise it remains same. Experimentally shown that TCP-Vegas gives higher throughput, it has some drawbacks to solve.

- The TCP Reno Congestion avoidance scheme is aggressive in the sense that it leaves little room in the buffer for other connections, while TCP Vegas is conservative and tries to occupy little buffer space.
- When a TCP Vegas connection shares a link with TCP Reno connection, the TCP Reno connection uses most of the buffer space and the TCP Vegas connection backs off, interpreting this as a sign of network congestion.

III. IMPROVED BANDWIDTH-BASED TCP DESIGN

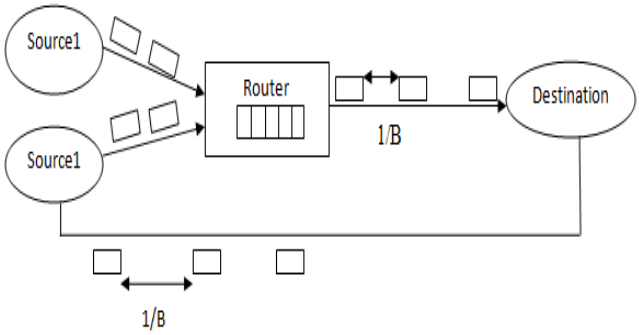

Fig.2 Improved Bandwidth estimation model

Improved bandwidth based TCP is proposed in this section for managing the problem discussed in the above section. By this technique, the sender can estimate the available bandwidth and subsequently predict the balanced congestion window size under the different criteria. Compared to TCP-Vegas, the slow-start phase is slightly modified while using a whole new adjustment manner for window size in the congestion avoidance phase. This approach includes the following three key features:

1. Enhanced Slow- Start,
2. Improved Bandwidth Estimation and
3. Improved Bandwidth-based adjustment of window size.

Fig.2 illustrates the bandwidth estimation method. If the bandwidth of a router is B (packets/sec), the router will send out packets to the destination every $1/B$ seconds. Suppose there is no loss during the transmission process, then the acknowledging rate must be B as well. Therefore, once an acknowledgment is received, the sender records the current time and calculates the ACK interval by subtracting the same value gotten in the previous period. Computing the reciprocal, the bottleneck bandwidth estimation is accomplished. If the router fairly processes the incoming packets from different senders, the ACK interval received by a sender should become N times longer, where N is the number of senders, and the estimated bandwidth $\hat{B}(n)$ is equivalent to B/N consequently. We rewrite this equation as

$$\hat{B}(n) = B/N \pm Traffic\,\mathrm{Re}laxation \qquad (1)$$

Here the traffic relaxation gives enhancement in processing data as it starts execute on each bytes of packet. Now we fix the Traffic Relaxation value as some x, which is based on previous history of packet acknowledgment time. Then we decide to add or subtract it from the B/N value. If is greater than fixed threshold value we fixed then it subtracts else it will

add, enhancement in processing packet is achieved. We can also change it dynamically as the number of users increase in a network. As per TCP-Vegas when the sender enters into congestion avoidance phase, it executes a steady-state prediction according to improved bandwidth estimation. We decompose the round-trip time as

$$r(n) = Tg + 1/B + q(n)/B \qquad (2)$$

Where *Tg* is the fixed propagation delay, *q*(*n*) is the queen length at slot *n*, *B* is the bandwidth of a router and 1/*B* is the processing delay. The improved bandwidth estimation finds adjustment of window size in three important steps.

A. Queuing Delay Estimation

Queuing delay can be calculated easily by subtracting RTTmin from the measured RTT. Here RTTmin is a minimum round-trip time taken to receive acknowledgement for a segment of data.

$$r(n) - RT\min = (Tg + 1/B + q(n)) - (Tg + 1/B) = q(n) \qquad (3)$$

B. Queue size Estimation

From this below equation we can get an exact queue size.

$$q(n) = B\hat{}(n)(r(n) - RT\min) \qquad (4)$$

C. Balanced Steady-state value of RTT and Window Size

Steady-state value of the window size is an important variable to analysis whether a network can process extra data or we make it reduce to control the network traffic by adjusting window size. This extra data is denoted by $d_k(x)$, it is depending on adjustment in size of the window. Let us consider α_1 and α_2 are two different variations in the window size, from this we derive $d_k(x) = \alpha_{1+}\ \alpha_2/2$. Based on the current RTT, dynamically adjust the window length. The current RTT is denoted as

$$r\hat{}c = 1/B + (\alpha 1 + \alpha 2)/2 \qquad (5)$$

We rewrite this equation in terms of measured RRT r(n),estimated queue q(n) and estimated bandwidth.

$$\begin{aligned} r\hat{}c &= Tg + 1/B + q(n)/B\hat{}(n) + ((\alpha 1 + \alpha 2)/2 - ((q\{n\}))/2 \\ &= r(n) + ((\alpha 1 + \alpha 2)/2 - ((q\{n\}))/2 \end{aligned} \qquad (6)$$

At last balanced point of window size is estimated ad

$$Wk = (((\alpha 1 + \alpha 2)/2) - q(n)) * (r\hat{}c/(r\hat{}c - RT\min) \qquad (7)$$

This prediction is used for adjusting window size in each RTT estimation. The algorithm is illustrated as below:

Improved TCP-Vegas Algorithm.
Slow Start
Ws_init=1;
Es_init=2;
For each ack
If (RTT==RTmin) Then
$B\hat{}(n)$= $B\hat{}(n)$+0;/Window size remain same.
Ws=W+ $B\hat{}(n)$;
Es=Es;
Else
Congestion avoidance state
End if
Congestion Avoidance
For each packet transmission
Estimate the Bandwidth B;
If (! Pkdrop_loss) Then
Es= $r\hat{}_{C+}$ $B\hat{}(n)$*(0.5-queuing size);*
Else
Es=Es+1;
*W= (Es+B*RTmin-Ws)/Ws;*
End if
Fast Retransmit
If (NACK) Then
Retransmit the lost packets
Ws= (α_{1+} α_2)/ 2;
Es=Es/2;
Enter congestion avoidance.

After a series of estimations, the sender lets the congestion window adjust linearly to the prediction value Es in an RTT and then repeats these three steps for every round-trip-time. The initial value of Es is set as two in the beginning. After an RTT, the sender calculates the queuing delay and then the growing factor is decided. Es is increased continuously. When the network is smooth, the sender should have

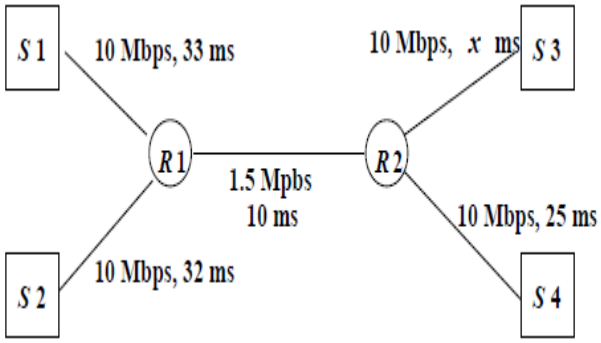

Fig.3 Multi-Connection Configuration

self-awareness to increase Es to lead to a larger window size, and decrease otherwise. I f a dropped packet is detected; Es becomes half of its original value. The growing rule is switched to decrease as soon as a packet loss is detected.

IV. SIMULATION RESULTS

Simulations have conducted by using Network Simulator version2 (NS2), we have used different kinds of networks such as homogeneous and heterogeneous configurations to implement the proposed algorithm. Firstly a simple one-connection configuration is considered and then it is expanded into a complex multi-connection configuration as shown in Fig.3 is used with setting both the sender-router and receiver-router. Source S1 uses 10Mbps with 33ms propagation delay, and the bottleneck router uses a 100paket buffer space with 1.5 Mbps and 10 ms delay. The values of Es, b(n), q(n) ,Wk are calculated to analysis the result. The graph in Fig.4 achieves improved performance than the TCP-Vegas. The equilibrium point of a window size is obtained to gain high performance which is compared with traditional TCP-Vegas algorithm. Even though every algorithm can reach a steady state in this simple case, the selection of (α1, α2)/2 deeply influences the equilibrium of window size in both the previous version and TCP-Vegas.

The multi-sender configuration is given to show the key behavior of the proposed algorithm. From this, the benefit of bandwidth-based TCP can be seen– adjusting the congestion window independent of the external threshold settings. Furthermore, the window size is larger than the other two.

V. CONCLUSION

Underlying TCP-Vegas algorithm is studied to obtain improved performance of the proposed system called improved bandwidth estimation. It has bandwidth

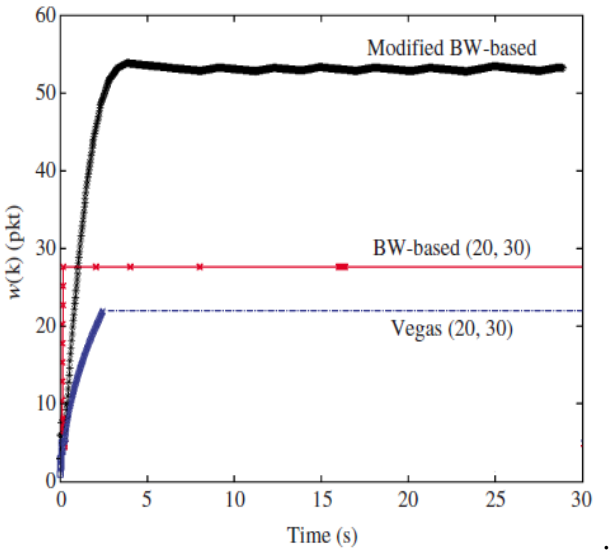

Fig.4 Window size comparison with TCP-Vegas

estimation, and improved bandwidth- based adjustment. In brief, it predicts an equilibrium point of window size then approximates it based on the measured bandwidth in every round-trip-time. Through the simulations, the proposed scheme is shown to have better performance than Vegas under a homogeneous and a heterogeneous environment.

REFERENCE

[1] Brakmo, L. S., and Peterson, L. L.,"TCP Vegas: End to End Congestion Avoidance on a Global Internet," *IEEE Journal on Selected Areas in Communications*, Vol. 13, No. 8, pp. 1465-1480, 1995.

[2] Richard J. La, Jean Walrand, and Venkat Anantharam, "Issues in TCP-Vegas," Steady report at Department of Electrical Engineering and Computer Sciences University of California at Berkeley, 2001.

[3] Chen, J. R., Chen, Y. C., and Lee, C. L., 2000, "TCP Vegas-A: Improving the Performance of TCP Vegas," *Computer Communications*, Vol. 23, No. 16, pp. 1537-1547.

[4] Habibullah Jamal, Kiran Sultan, "Performance Analysis of TCP Congestion Control Algorithms," International Journals on Computer Science and Communications, Issue 1,Volume 2, 2008.

[5] U. Hengartner1, J. Bolliger1 and Th. Gross1;2, "TCP-Vegas Revisited, " Study report at Carnegie Mellon University, 2010.

[6] Floyd, S., and Jacobson, V., "Random Early Detection Gateways for Congestion avoidance," *IEEE /ACM Transactions on Networking*, Vol. 1, No. 4, pp. 397-413, 1993.

[7] Hollot, C. V., Misra, V., Towsley, D., and Gong, W. B., "On Designing Improved Controllers for AQM Routers Supporting TCP Flows," *in*

Proceedings of IEEE INFOCOM, Vol. 3, pp. 1726-1734, Alaska, USA, 2001.

[8] Jacobson, V. "Congestion Avoidance and Control," *in Proceedings of ACM SIGCOMM*, pp. 314-329, Stanford, CA, USA, 1988.

[9] Jacobson, V., , "Berkeley TCP Evolution from 4. 3-Tahoe to 4.3-TCP-Reno," *Proceedings of the 18th Internet Engineering Task Force*, Vancouver, BC, Canada, 1990.

[10] Moar, A., and Mansour, Y., "AdaVegas: Adaptive Control for TCP Vegas," *Proceedings of the IEEE GLOBECOM'03*, Vol. 7, pp. 3647-3651, 2003.

[11] Srijith, K. N., Jacob, L., and Ananda, A. L., "An End-to-End Flow Control Approach Based on Round Trip Time," *Computer Communication*s, Vol. 28, No. 4, pp. 429-440, 2005.

[12] Low, S. H., Paganini, F., and Doyle, J. C., 2002, "Internet Congestion Control," *IEEE Control Systems Magazine*, Vol. 22, No. 1, pp. 28-43.