FUZZY ENHANCED TCP VEGAS FOR CONGESTION CONTROL

Prof. Deepa Jose
Lecturer, MCA Department
Y.M.T College of Management
Kharghar, NaviMumbai ,Maharashtra
deepa_jos@rediffmail.com

Abstract - In data networking and queuing theory, network congestion occurs when a link or node is carrying so much data that its quality of service deteriorates. Typical effects include queuing delay, packet loss or the blocking of new connections. A consequence of these latter two is that incremental increases in offered load lead either only to small increases in network throughput, or to an actual reduction in network throughput. TCP Vegas detects congestion at an incipient stage based on increasing Round-Trip Time (RTT) values of the packets in the connection unlike other flavors like Reno, NewReno, etc., which detect congestion only after it has actually happened via packet drops. The algorithm depends heavily on accurate calculation of the Base RTT value. In this paper an adaptive fuzzy enhanced version of TCP Vegas is proposed so that performance of traditional Vegas is optimized.

Keywords-TCPVegas,fuzzy,TCP/IP,bandwidth,congestion ,queue delay,window size.

I.INTRODUCTION

Congestion is said to occur in the network when the resource demands exceed the capacity and packets are lost due to too much queuing in the network. During congestion, the network throughput may drop to zero and the path delay may become very high. A congestion control scheme helps the network to recover from the congestion state. A congestion avoidance scheme allows a network to operate in the region of low delay and high throughput. Such schemes prevent a network from entering the congested state. Congestion avoidance is a prevention mechanism while congestion control is a recovery mechanism. Congestion control has been receiving increased attention lately due to an increasing speed mismatch caused by the variety of links that compose a computer network today. Congestion occurs mainly at routers (intermediate nodes, gateways) and links in the network where the rate of incoming traffic exceeds the bandwidth of the receiving node or link. [1]. Nowadays, communication networks are among the fastest-growing engineering areas and are driving extraordinary developments in communication industry. An increasing amount of research is devoted to the deployment of new communication networks that merge the capabilities of telephone networks and of computer networks in order to transmit multimedia over a fully integrated universal network[2]. The Internet's TCP guarantees the reliable, in-order delivery of a stream of bytes. It includes a flow-control mechanism for the byte streams that allows the receiver to limit how much data the sender can transmit at a given time. In addition, TCP implements a highly tuned congestion-control mechanism. The idea of this mechanism is to throttle how fast TCP sends data to keep the sender from overloading the network.The idea of TCP congestion control is for each source to determine how much capacity is available in the network, so that it knows how many packets it can safely have in transit. It maintains a state variable for each connection, called the congestion window, which is used by the source to limit how much data it is allowed to have in transit at a given time.TCP uses a mechanism, called additive increase/multiplicative decrease, that decreases the congestion window when the level of congestion goes up and increases the congestion window when the level of congestion goes down. TCP interprets timeouts as a sign of congestion. Each time a timeout occurs, the source sets the congestion window to half of its previous value. This halving corresponds to the multiplicative decrease part of the mechanism. The congestion window is not allowed to fall below the size of a single packet (the TCP maximum segment size, or MSS). Every time the source successfully sends a congestion window's worth of packets, it adds the equivalent of one packet to the congestion window; this is the additive increase part of the mechanism. TCP uses a mechanism called slow start to increase the congestion window "rapidly" from a cold start in TCP connections. It increases the congestion window exponentially, rather than linearly. Finally, TCP utilizes a mechanism called fast retransmit and fast recovery. Fast retransmit is a heuristic that sometimes triggers the retransmission of a dropped packet sooner than the regular timeout mechanism[3]. TCP Vegas is a TCP congestion avoidance algorithm that emphasizes packet delay, rather than packet loss, as a signal to help determine the rate at which to send packets. It was developed at the University of Arizona by Lawrence

Brakmo and Larry L. Peterson. TCP Vegas detects congestion at an incipient stage based on increasing Round-Trip Time (RTT) values of the packets in the connection unlike other flavors like Reno, NewReno, etc., which detect congestion only after it has actually happened via packet drops. The algorithm depends heavily on accurate calculation of the Base RTT value. If it is too small then throughput of the connection will be less than the bandwidth available while if the value is too large then it will overrun the connection. A lot of research is going on regarding the fairness provided by the linear increase/decrease mechanism for congestion control in Vegas. One interesting caveat is when Vegas is inter-operated with other versions like Reno. In this case, performance of Vegas degrades because Vegas reduces its sending rate before Reno as it detects congestion early and hence gives greater bandwidth to co-existing TCP Reno flows. TCP Vegas is one of several "flavors" of TCP congestion avoidance algorithms. It is one of a series of efforts at TCP tuning that adapt congestion control and system behaviors to new challenges faced by increases in available bandwidth in Internet components on networks like Internet2.TCP Vegas has been implemented in the Linux kernel, in FreeBSD and possibly in other operating systems as well[4].This research paper aims at enhancing TCP Vegas with fuzzy Logic when Vegas is in presence of other flows .The rest of the paper is organized as follows , 2.Background for the work,3.Performane of Vegas flow with a Reno flow,4.Ns2-The Network Simulator,5.Fuzzy Logic 6.Simulation Setup and Results,7.Fuzzy enhancement of TCP Vegas 8.Conclusion .

II.BACKGROUND

In this section, we briefly describe TCP NewReno and TCP Vegas. TCP NewReno TCP congestion control relies on the dynamic manipulation of the window size used by TCP's window-based flow control. In particular, a TCP source controls its average data sending rate by adjusting its congestion window size, which is an estimate of how much data can be safely transmitted into the network without experiencing packet loss. To determine an appropriate data sending rate, the TCP source operates in two different modes: slow start and congestion avoidance.

A .NewReno

1)Slow start: source begins transmitting a new data flow in slow start. The initial congestion window is typically one segment. Each time a NewReno source receives an acknowledgment (ACK) packet, it increases the congestion window cwnd by one segment. With this approach, the congestion window size expands multiplicatively, doubling every RTT. An additional TCP parameter, the slow start threshold, determines when this aggressive window expansion mode should end. Once the threshold is reached, a NewReno flowtransitions into congestion avoidance.

2)Congestion avoidance: In congestion avoidance, aNewReno source increases its congestion window linearly, rather than multiplicatively. Upon receiving each ACK, the congestion window is changed as follows:

$$cwnd = cwnd + 1/cwnd \quad (1)$$

Thus, each subsequent successful delivery of a data packet leads to a small increase of TCP congestion window, and after the exchange of a complete window's worth of data, cwnd increases by one segment. This linear increase continues until the flow is finished, or a packet loss occurs.

3)Loss recovery: There are two ways for a NewReno source to detect packet losses. One approach is a coarse grained timeout, which happens when a NewReno source has not received any ACK during a Retransmission Timeout (RTO) interval. When a timeout occurs, a NewReno source resets thecongestion window size to one packet, and resumes slow start. Another approach is called triple duplicate ACK, in which three repeated cumulative ACKs carry the same redundantinformation, indicating a lost packet within a window of packets.In this case, a NewReno source decreases the congestion window size by half.

B.TCP Vegas

TCP Vegas is one of the most prominent examples of delay based transport protocols. The key difference between loss based protocols like TCP NewReno and delay-based protocols like TCP Vegas is that the latter uses changes in end-to-end delay (rather than loss) as the means for detecting congestion.In particular, since TCP throughput is inversely related to RTT, Vegas measures the difference between the expected throughput and the actual throughput. The idea is that the actual throughput should match the expected throughput if there is no congestion along the network path. A lower actual throughput indicates increased delay, and hence congestion,on the network path.Similar to NewReno, Vegas has slow start and congestion avoidance modes.

1) Slow start: One difference between NewReno and Vegas is that the initial congestion window size is two packets instead of one. Another difference is that Vegas doubles its congestion window every other RTT, rather then every RTT. This approach improves measurement accuracy, since the congestion window is fixed when comparing Expected Throughput and Actual Throughput. When the value of diff in Equation (2) exceeds a threshold parameter , Vegas switches to congestion avoidance mode.

2) Congestion avoidance: The adjustments of the congestion window size are made based on the value of diff, which determines both the direction and the

magnitude of adjustment required. The value of diff is given by:

diff = (Expected Throughput –Actual Throughput) * Base RTT (2)

where Expected Throughput is the expected Throughput, Actual Throughput is the actual observed throughput, and Base RTT is the minimum RTT observed for the network path. The Expected Throughput and Actual Throughput are calculatedusing:

Expected Throughput =cwnd/BaseRTT (3)

Actual Throughput=cwnd/RTT (4)

wherecwnd is the current congestion window size (in segments),and RTT is the actual RTT observed. A Vegas source updates its congestion window size once per RTT, as follows:

cwnd =cwnd +1 if diff < α

=cwnd -1 if diff>β

= cwnd otherwise

where α and β are specified parameters. In other words, Vegas increases cwnd by one packet if the per-flow queue at a bottleneck link is smaller than α, decreases cwnd by one packet if the per-flow queue is larger than β , and keeps cwnd unchanged otherwise. In our work, we use the typical settingsof these parameters, namely

α = 1 and β = 3 .

3) Loss recovery: The loss signals for TCP Vegas are similar to those for TCP NewReno. One difference is that the congestion window after a timeout is 2 packets. Furthermore, if a loss is detected with triple duplicate ACKs, then the congestion window decreases by 25%, rather than 50% [6].

NS2 EXPERIMENTAL SETUP

NS2 is an open-source event-driven simulator designed specifically for research in computer communication networks. Since its inception in 1989, NS2 has continuously gained tremendous interest from industry, academia, and government.Having been under constant investigation and enhancement for years, NS2 now contains modules for numerous network components such as routing,transport layer protocol, application, etc. To investigate network performance, researchers can simply use an easy-to-use scripting language to configure a network,and observe results generated by NS2. Undoubtedly, NS2 has become the most widely used open source network simulator, and one of the most widely used network simulators[7].

In this paper simulations have been done on Ns2.The simulation setup consist of 4 nodes,Node0 Node1,Node2 and Node3,refer Fig 1

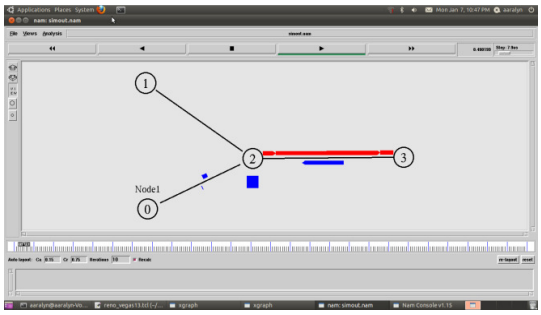

Fig 1

Node 0 and Node 1 are the source nodes and both have the sink node as Node 3.The traffic in the link Node2 to Node3 is studied in detail which is the bottleneck link. Four cases are considered by varying the bandwidth of the link Node2 to Node3 as .1Mb, 1Mb, 5 Mb and, 10 Mb. The other links are kept at a constant bandwidth of 4 Mb. Node 0 sends Reno traffic to the sink Node3 and Node 1 sends Vegas traffic to Node3.The simulation is run for 30 seconds in all cases.

Case1 :In Case 1 the bottleneck link is kept at .1Mb and simultaneously Vegas and Reno traffic are started.The throughput obtained in this case is as given in fig2. It can be observed that the throughput obtained by the Vegas flow is very less compared to that of Reno in the first case .

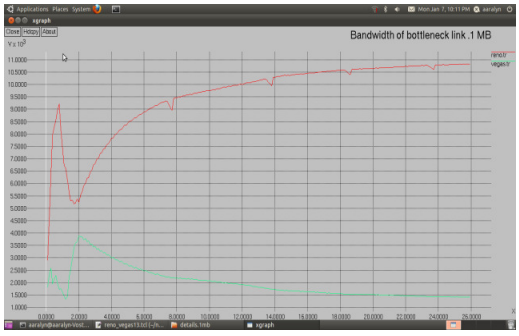

Fig2

Case2:In Case 2 the bottleneck link is kept at 1Mb and simultaneously Vegas and Reno traffic are send as in previous case .The throughput obtained in this case is as given in fig3.The observation is that the throughput obtained is not very high but still more consistent that in Case 1.

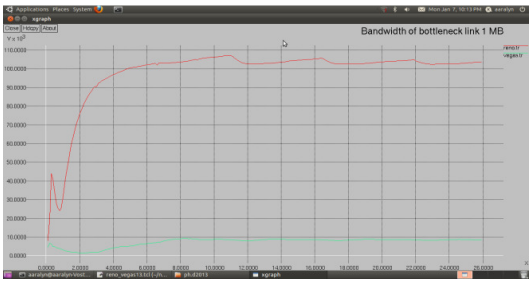

Fig3

Case3:In Case 3 the bottleneck link is kept at 5Mb and simultaneously Vegas and Reno traffic are evaluated. The throughput obtained in this case is as given in fig4.It can be easily observed that there is a drastic change in the throughput obtained by the Vegas flow.

Case4: In Case 4 the bottleneck link is kept at 10Mb and simultaneously Vegas and Reno traffic are allowed .The throughput obtained in this case is as given in fig5.When the bandwidth is increased considerably and there is no competition as in the previous cases the throughput of the Vegas flow is not only high but more than that of Reno flow.

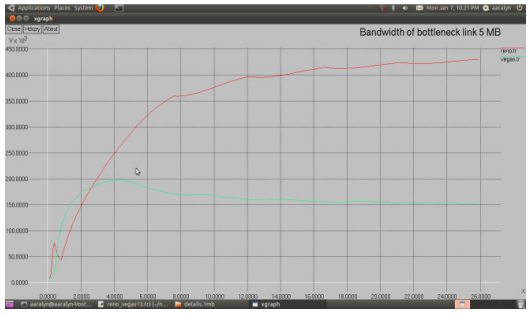

Fig4

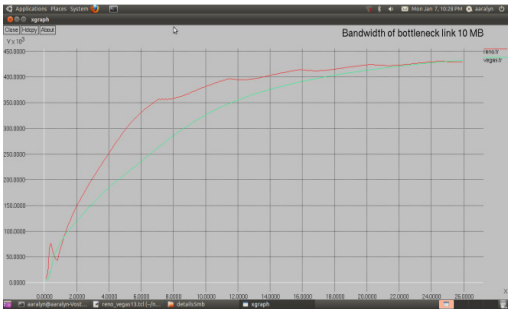

Fig5

FUZZY ENHANCED TCP VEGAS

A network system is a large distributed complex system, with difficult often highly non-linear, time varying and chaotic behavior. There is an inherent fuzziness in the definition of the controls (declared objectives and observed behavior).Dynamic or static modeling of such a system for (open or closed loop) control is extremely complex. Measurements on the

state of the network are incomplete, often relatively poor and time delayed. Its sheer numerical size and geographic spread are mind-boggling.

Therefore, in designing the network control system, a structured approach is necessary. The traditional techniques of traffic engineering, queuing ana1ysis, decision theory, etc. should be supplemented with a variety of novel control techniques, including (nonlinear) dynamic systems, computational intelligence and intelligent control (adaptive control, learning models, neural networks, fuzzy systems, evolutionary/genetic algorithms), and artificia1 intelligence[8].

In this paper , a fuzzy based approach to enhance the Vegas preserving all its benefits is considered. As input the bandwidth utilization and the packet loss parameters are taken. The output parameter is the newly adjusted sending window. Here the two input parameters ie,the bandwidth utilization and packet loss are chosen because bandwidth utilization gives idea about fairness in bandwidth sharing whereas the parameter packet loss gives idea about the presence of congestion in the network.So the input parameters give a clear picture as to whether fairness is present and if not till where we can extend the sending window. In this way user can increase the sending window if bandwidth utilization of vegas is low and there is no sign of congestion.

The fuzzy inference rules can be formed as follows:

b_util- the bandwidth utilization

p_loss- packet loss

w_sze- sending window size

If b_util is low and p_loss is low then w_sze increased.

If b_util is low and p_loss is medium then w_sze maintained.

If b_util is low and p_loss is high then w_sze

decreased.

If b_util is medium and p_loss is low then w_sze maintained.

If b_util is medium and p_loss is medium then w_sze maintained.

If b_util is medium and p_loss is high then w_sze decreased.

If b_util is high and p_loss is low then w_sze maintained.

If b_util is high and p_loss is medium then w_sze decreased.

If b_util is high and p_loss is high then w_sze decreased.

The rules are formulated in such a way that the fuzzy inference system fires a particular rule which matches the situation of the network.

CONCLUSION

In this paper a novel approach for enhancing TCP Vegas by keeping intact all its benefits has been done with the help of fuzzy logic.It has been already proved that Vegas achieves 31 to 70 % more throughput compared to other flows. Fuzzy logic derives much of its power from its ability to draw conclusions and generates responses based on vague, ambiguous, incomplete or imprecise information.

The work presented here is an attempt to make up for the reduction in bandwidth that Vegas experience in the presence other aggressive flows like Reno.Also since the parameter of packet loss is also considered it is made sure that the bandwidth of the Vegas flow is not increased to the point that it will create congestion.The idea presented here is expected to improve the Vegas by making it a suitably adapted algorithm for network.

REFERENCES

[1].Raj Jain, K. K. Ramakrishnan,Congestion Avoidance in Computer Networkswith a connectionless Network Layer Part1:Concepts ,Goals and Methodology, IEEE Press, pp. 1-12,1998.

[2].SaverioMascolo,Congestion control in high-speed communication networks using the Smith principle, S. Mascolo / Automatica 35 (1999) 1921}1935

[3].J.C. Majithia, et al., \Experiments in Congestion Control Techniques," Proc. Int. Symp. Flow Control Computer Networks, Versailles, France. February 1979.

[4].John Nagle, \Congestion Control in TCP/IP Internetworks," Computer Communication Review, Vol. 14, No. 4, October 1984, pp. 11-17.

[5].K. K. Ramakrishnan, \Analysis of a Dynamic Window Congestion Control Protocolin Heterogeneous Environments Including Satellite Links," Proceedings of ComputerNetworking Symposium, November 1986

[6].A.S. Tanenbaum, Network Protocols. Prentice-Hall: Englewood Cli_s, NJ, 1981.

[7].Dah-Ming Chiu and Raj Jain, \Congestion Avoidance in Computer Networks with a Connectionless Network Layer. Part III: Analysis of Increase/Decrease Algorithms,"Digital Equipment Corporation, Technical Report #TR-509, August 1987.

[8].M. Gerla and L. Kleinrock, \Flow Control: A Comparative Survey," IEEE Transactions on Communications, Vol. COM-28, No. 4, April 1980, pp. 553 - 574.

[9].Raj Jain, \Using Simulation to Design a Computer Network Congestion Control Protocol," Proc. Sixteenth Annual Modeling and Simulation Conference, Pittsburgh, PA,April 1985.

[10]. Raj Jain, \A Timeout-Based Congestion Control Scheme for Window Flow-Controlled Networks," IEEE Journal on Selected Areas in Communications, Vol. SAC-4, No. 7,October 1986, pp. 1162-1167.