IOSR Journal of Computer Engineering (IOSR-JCE)
e-ISSN: 2278-0661,p-ISSN: 2278-8727, Volume 17, Issue 1, Ver. III (Jan – Feb. 2015), PP 70-72
www.iosrjournals.org

Congestion Control: A Dynamic Approach

[1]Keshav Sharma, [2]Aman Kedia, [3]Shivam Shrivastava
School of Computing Science and Engineering, VIT University, Vellore

Abstract: *Congestion control is a very important concept used in practical networks. Congestion of a network occurs when a network carries such high traffic of data that the quality of service deteriorates. In modern day TCP protocol (Tahoe and Reno), the network determines congestion on the basis of the number of data packets lost. We believe that this is not the most optimized way of determining congestion because of the bottleneck problem that is described later in the text.Hence, in this paper, we propose a new algorithm to deduce whether the network is congested or not and then apply proper measures to rectify it.*

Keywords: *Congestion control, TCP/IP, Duplicate acknowledgements, Packet Loss, Bottleneck problem of TCP/IP*

I. Introduction

In the current version of TCP, Tahoe and Reno, congestion control takes place on the basis of packet loss. The congestion window decreases because of three duplicate acknowledgements by the client or because of a timeout at the server side. Both the present versions alter the congestion window in different ways. While TCP Reno cuts the window into half in the case of three duplicate acknowledgements and shelves it to 1 MSS in case of a timeout, TCP Tahoe sets the window to 1 MSS in both the scenarios. However, they both increase the congestion window in the same manner, that is, both follow slow start (exponential growth) till threshold and then go into the collision avoidance phase (linear growth).

The drawback in the above scenario is mainly the bottleneck problem. As the congestion window grows exponentially, the rate of sending packets is too high to make the bottleneck link buffer be overflow. The issue that the multiple packets in the same sending window are discarded cause TCP performance drastically dropped. In this paper we propose a dynamic approach to change the congestion window, not on the basis of packet loss but on the basis on Round Trip Time. If implemented, this method is expected to provide satisfying results without unnecessary overheads.

TCP (Transmission Control Protocol) is a connection oriented protocol that provides reliable data transfer between two end systems by various means of services. Some of the important services TCP provides are, three way handshake between the sender and the receiver, flow control, congestion control etc. It ensures that the data sent by one system reaches the other system without any guarantee on the time limit. One of the main services provided by TCP is the congestion control.

A TCP connection probes the network for a possible congestion by increasing the number of packets sent into the network until a packet loss occurs. A packet loss gives an indication that the network is congested and the TCP then reduces its transmission rate.

In this paper, we calculate the size of the congestion window not on the basis of packet loss but on the basis of the Round Trip Time (RTT). Since a TCP connection already stores the Sample RTT (SRTT) for calculation of the Estimated RTT (ERTT), we believe that this approach will not result in any unnecessary overhead.

$$ERTT = (1-\alpha)*ERTT + \alpha*SRTT$$

Where $\alpha = 0.125$

Round Trip Time is the time taken by the packet to reach the destination plus the time taken by the acknowledgment of the same packet to reach the sender.

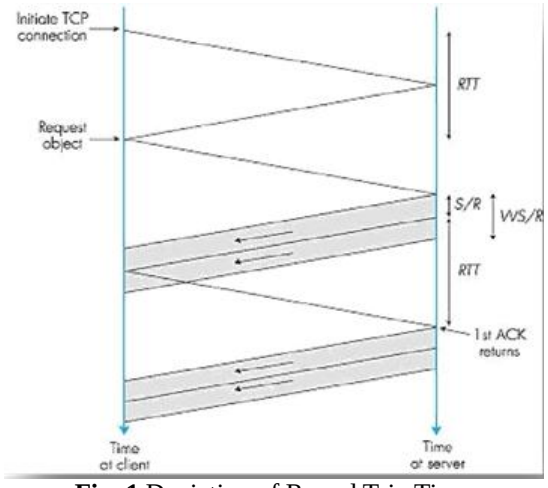

Fig. 1 Depiction of Round Trip Time

RTT can be a good measure of the congestion in the network, the more the RTT, the more congested is the network and vice versa. And thus, using RTT as a parameter we can change the size of the congestion window.

II. Working

Since TCP calculates and stores the Round Trip Time for every packet sent, we will use it to calculate the Estimated Round Trip Time using the above specified formula and then set this ERTT as a threshold for further part of the connection.

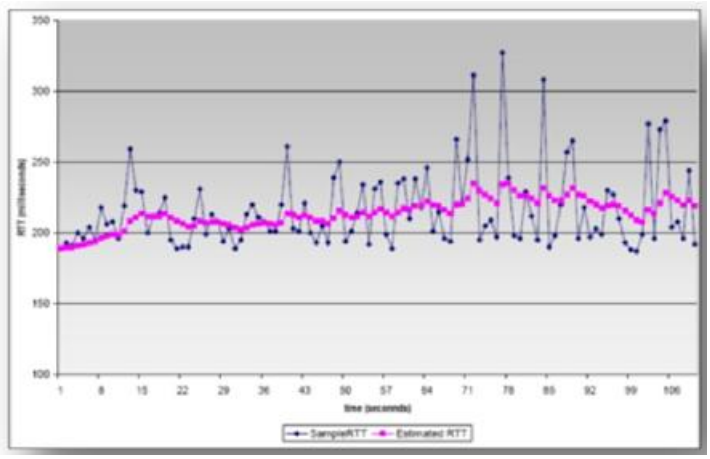

Fig. 2 Graph between Sample RTT and Estimated RTT

We use the Estimated Round Trip Time as the threshold value because it is the average of the Sample Round Trip Time and more stable as shown in the figure.

Now, we initialize the congestion window at 1 MSS and follow linear growth that is

CongWin = CongWin + MSS * (MSS/CongWin)

The congestion window follows a linear growth until the Sample RTT is less than the Estimated RTT. We make it a linear growth instead of an exponential one so as to overcome the bottleneck problem of the TCP Reno and Tahoe.

If the Sample RTT crosses the Estimated RTT, TCP detects that there is congestion in the network and cuts the congestion window into half. That is once Sample RTT is greater than the Estimated RTT, reduce CongWin = CongWin/2, which helps in reducing the congestion.

In case of a highly congested network, when Sample RTT is greater than (ERTT + 4*DRTT) where DRTT is the Deviation Round Trip Time, we reset the Congestion Window to 1 MSS. The Deviation RTT is calculated from the formula below

$DRTT = (1-\beta)*DRTT + (\beta*|SRTT - ERTT|)$

Where β is typically 0.25.

Thus, this method includes three phases, linear growth, multiplicative decrease (congestion window will not fall below 1 MSS) and resetting the window to 1 MSS in case of highly congested network (analogous to the timeout event).

III. Advantages

It overcomes the bottleneck problem of Tahoe and Reno as it does not include the slow start phase and hence the output buffer will remain in control and not overflow.

It is based on a more reliable and dynamic parameter, the Sample RTT, that gives a clear indication of the network traffic rather than depending duplicate acknowledgements and timeout.

It prevents the network from getting congested rather than doing the repair work after the damage is done.

IV. Drawbacks

It is a little slow in the beginning due to the absence of the slow start phase. Since it involves only the linear growth it takes some time to utilize the full bandwidth but once the level is attained, it is more or less stable.

The major drawback of this method is the ambiguous value of the Sample RTT as explained by the diagram below

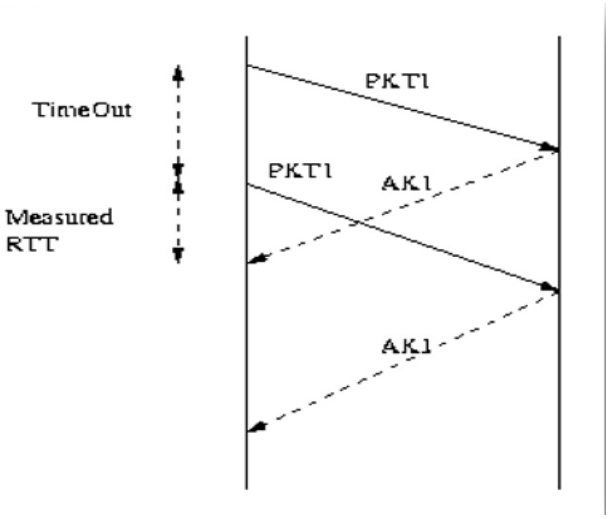

Fig.3 Diagram depicting drawback of the proposed algorithm

In the above figure, the true Sample RTT is the Measured RTT plus the Time out period but TCP takes only the measured RTT into account which increases the Congestion Window rather than decreasing it.

V. Conclusions

Like most modern algorithm, this algorithm has its pros and cons. In this paper we showed a novel way to change the Congestion Window on the basis of a differentparameters, the Round Trip Time and if simulated in real time environment, we expect it to provide satisfactory results.

References

[1]. Qian Wang, Dongfeng Yuan, "An Improved TCP Congestion Control Mechanism with Adaptive Congestion Window"
[2]. Dorgham Sisalem, Henning Schulzrinne, "Congestion Control in TCP: Performance of Binary Congestion Notification Enhanced TCP Compared to Reno and Tahoe TCP"
[3]. Ming Yan, Chengjun Yue, "Robust Sliding Mode Congestion Control Algorithm for TCP Networks"
[4]. Saverio Mascolo, " Smith's Predictor for Congestion Control in TCP Internet Protocol"
[5]. James F. Kurose, Keith W. Ross, "Computer Networking, A Top-Down approach"
[6]. Behrouz A. Forouzan, "Data Communications and Networking".