International Journal of Science and Engineering Applications (IJSEA)
Volume 2 Issue 3, 2013, ISSN - 2319-7560 (online)

Performance Enhancement of Network Transmission by Change of Delay Parameter to Increase Throughput

Manish Giri
MIT Academy of Engineering,
Pune, India

Manik Hendre
MIT Academy of Engineering,
Pune, India

Abhijeet Kolte
MIT Academy Of Engineering,
Pune, India

Kaustubh Jawalekar
MIT Academy Of Engineering,
Pune, India

Aakash Chaudhari
MIT Academy Of Engineering,
Pune, India

Abstract: Congestion is a major problem which really degrades network performance. Its effect on the delay is quite high which decrease the throughput in considerable amount. In this paper we have proposed a new congestion scheme which will enhance performance of network transmission by considering delay and throughput as main components. For this we are using Adaptive Explicit Congestion Notification method which allows us to control the network transmission from source to a receiver in a hierarchical manner by alterations in the receiving sending capacity. To achieve predictable average delays with adaptive explicit congestion notification would require constant tuning of the parameters to adjust to current traffic conditions. The sending and receiving ends are tuned according to its capacity and current network traffic condition once ECN (Explicit congestion notification) notification packet comes into action. The aim of this paper is to solve the parameter tuning problem of the AECN by dynamically setting up the network parameters to overcome delay in network transmission and hence to increase throughput .We will be comparing the performance of the ECN enabled system with the AECN enabled system. For this we are going to use JAVA , JNETPCAP and WINPCAP API's by which we can create TCP packets, alter the fields of headers, send and receive packets etc.

Keywords: congestion window, Explicit, Packet, Marking, Notification

1. INTRODUCTION

The accumulation of the packets in the queue results in the saturation at the end points in the communication links. The TCP/IP protocol as a primitive measure drops some packets to overcome the overflow as in the active queue management. Now if the end points are made ECN capable, the receiver sends a notification Packet of its status .The sender then has to tune up considering the severity of the receiver ends to what extent the rate can be decreased. Also making use of ECN we can increase transmission rate if senders sends data with a capacity less than of its receiver.

1.1Explicit Congestion Notification

ECN is an extension to the Internet Protocol and to the Transmission Control Protocol and is defined in RFC 3168 (2001). ECN allows end-to-end notification of network congestion without dropping packets. ECN is an optional feature that is only used when both endpoints support it and are willing to use it. It is only effective when supported by the underlying network on which the transmission is active When ECN is successfully negotiated, an ECN-aware router may set a mark in the IP header instead of dropping a packet in order to signal impending congestion. The receiver of the packet echoes the congestion indication to the sender, which reduces its transmission rate as though it detected a dropped packet.

1.2. Multilevel Approach of ECN

Marking of Bits:-

AECN uses two bits that is being specified for the use of ECN, in the IP header bit 6 and 7 in the TOS octet in Ipv4, or the Traffic class octet in Ipv6 to indicate four different levels of congestion, instead of the binary feedback provided by ECN.

Table 1. ECT and CE marking

CE(Congestion Experienced) Bit	ECT(ECN Capable Transport) Bit	Congestion state
0	0	ECN
0	1	No
1	0	Incipient
1	1	Moderate
Packet Drop		Severe

00 is used for identifying non-ECN capable packets and other combinations are used for indicating different levels of congestion which are then used to take proper action at TCP source depending on level of congestion as given in Table 1.

Sender and receiver side:-

Bit marking in IP header is reflected by receiver, to the TCP ACK. We use 3 combinations of 2 bits 8, 9 in TCP header and other combination used by source has to indicate that congestion window reduced.

Table 2. Receiver side CWR and ECE marking

CWR Bit	ECE Bit	Congestion	CWND change
0	1	No congestion	Increase cwnd additively
1	0	Incipient congestion	Decrease multiplicatively
1	1	Moderate Congestion	Decrease multiplicatively
Packet Drop		Severe congestion	Decrease multiplicatively

In TCP header it has the ECN-Echo (ECE) flag and Bit 8 is designated as the Congestion Window Reduced (CWR) flag. These two bits are used both for the initializing phase in which the sender and the receiver negotiate the capability and the desire to use ECN, as well as for the subsequent actions to be taken in case there is congestion experienced in the network during the established state.

When a router has decided from its active queue management mechanism, to drop or mark a packet, it checks the IP-ECT bit in the packet header. It sets the CE bit in the IP header if the IP-ECT bit is set. When such a packet reaches the receiver, the receiver responds by setting the ECN-Echo flag (in the TCP header) in the next outgoing ACK for the flow. The receiver will continue to do this in subsequent ACKs until it receives from the sender an indication that it (the sender) has responded to the congestion notification. Upon receipt of this ACK, the sender triggers its congestion Avoidance algorithm by halving its congestion window, cwnd, an updating its congestion window threshold value

1.3. Evaluation of the proposed module

In Adaptive MECN, the objective is to maintain the queue near the *tar*get queue. If the average queue doesn't vary and remains constant at *tar*get queue, then the probability of packet drop/mark will remain fixed. Let this probability be *ptarget*.

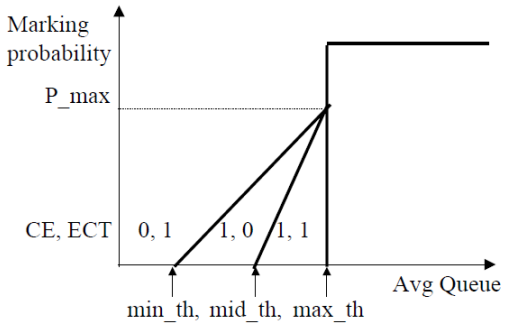

Figure 1. Probabilities of marking packets

We set the *tar*get queue to be in between *minth* and *midth*. Hence only the first probability curve will be active, in this region. Hence the probability *ptar*get is given by,

$$P_{target} = \frac{P_{max}}{max_{th}\text{-}min_{th}} * Averagequeue - min_{th} \quad (1)$$

Since in the above equation, *ptarget, minth, maxth* are all constant, we can say that,

$$Average\ queue \propto \frac{1}{P_{max}} \quad (2)$$

In any network, we do not have the control over the traffic and the average queue increases or decreases with the load. But the aim is to have the *avgqueue*, always equal to the *targetqueue*. Hence if the *avgqueue*, is greater than *targetqueue*, at any instant, we need to increase *pmax* which would decrease the *avgqueue* so that it becomes equal to *targetqueue* and if the *avgqueue*, is less than, at any instant, we need to decrease *pmax*, to allow the queue, to grow, which would give a better throughput. Thus to keep a constant queue we need to adapt the ***pmax***.

Also we need to get the other parameters like ***wq, maxth, midth*** and ***minth*** automatically. Adapt ***pmax*** in response to measured queue lengths and set ***wq, maxth, midth*** and ***minth*** automatically, based on the link speed and target queue.

2. DESIGN AND IMPLEMENTATION:

Algorithm:-

For every (Time Interval) do

If (avgqueue > ptarget and pmax<=Time_Interval) do

Increase ***pmax*** by adding α *to it*

EndIf

ElseIf(avgqueue<ptarget and pmax>=0.01)

Decrease ***pmax*** by multiplying β to itEnd ElseIf

End For.

The overall Adaptive ECN, which is implemented, has the following features:

Pmax is adapted to keep the average queue size with a target range half way between ***minth*** and ***maxth***

Pmax is adapted slowly, over time scales greater than a typical round-trip time and in small steps. The time scale is generally 5-10 times the typical RTT of the network.

Pmax is constrained to remain with range of [0.01,Time_Interval]

Instead of multiplicatively increasing and decreasing ***pmax*** here the policy used is additive-increase multiplicative-decrease (AIMD) policy.

The robustness of this algorithm comes from its slow and infrequent adjustment of ***pmax.*** The price of these slow modifications is that after a sharp change in the level of congestion, it could take some time, before ***pmax*** adapts to its value. But also adapting ***α*** *and* ***β*** makes this process faster and decrease the response time of the system.

2.1 Setting the Parameters:

The range for ***pmax:*** The upper bound of 0.5 on ***pmax*** can be justified because, when operating under the gentle mode, this would mean that the packet drop rate varies from 0 to ***pmax***, when average queue varies from ***minth to maxth*** and varies from ***pmax*** to 1.0, if queue changes from ***maxth*** to 2**maxth.*** For scenarios with very small drop rates, MECN will perform fairly robustly with ***pmax*** set to the lower bound 0.01, and no one is likely to object to an average queue size less than the target range.

Parameters ***α*** *and* ***β****:*

The **α** is an increase factor which can be given as,

$$\propto = const * \frac{avg - target}{target} * P_{max}$$

And **β** is a decrease factor which can be given as,

$$\beta = 1 - X * \frac{target - avg}{target}$$

where,

$$X = const * \frac{target}{target - min}$$

Here the *const* varies from 0 to 1. According to networks speed or condition we are going to set its value. It takes 0.49/***α*** intervals for ***pmax*** to increase from 0.01 to 0.5; this is 24.5 seconds, if ***α*** is set as 0.01. Similarly, it takes at least log 0.02/ ***β*** intervals for ***pmax*** to decrease from 0.5 to 0.01; with the default values, which is 20.1 seconds. Therefore if there is a sharp change in the router load, then it may take as long as 24.5 seconds for the average queue to reach the target range. This time is really a long time in network. Hence we believe that ***α*** and ***β*** should also be adapted, according to the position of the average queue, with respect to the target queue. So the value of ***α*** and ***β*** are also recalculated every 0.5 seconds when the ***pmax*** calculation is done. We scale the value of ***β*** from 0.83 to 1.0 when average queue, varies from 0 to target queue.

Thus use the formula given below to adapt ***β.***

$$\beta = 1 - (0.17 * \frac{(target - avg)}{(target - min)}) \qquad (3)$$

Setting **midth, maxth** *and* **wq:** To reduce the need for other parameter-tuning, we also give some guidelines for setting the ***midth, maxth*** and ***wq.*** The ***maxth*** is set to three times the ***minth***. In this case the target average queue size is centered around 2 * ***minth***. We believe that, the target queue should be kept in the low congestion region (i.e. between ***minth*** and ***midth***), to maximize the throughput, but at the same time the ***midth*** should not be too far from the ***targetqueue***, so that when the average queue rises above target, a quick response to congestion is achieved, when the second probability curve,

comes into action. This belief, led us to setting the ***midth*** slightly above the ***targetqueue***.

Thus ***midth*** was set at ***2.25*** * ***midth (targetqueue=2*minth).***

If the queue size changes from one value to another it takes -1/ln (1-wq) packet arrivals for the average queue to reach 63% of the way to the new value. Thus we refer to -1/ln (1-wq) as the time constant of the estimator for the average queue size. We set wq as a function of the link bandwidth. For MECN in automatic mode, we set wq to give a time constant for the average queue size estimator of one second.

Thus we set,

$$wq = 1 - exp\left(\frac{-1}{C}\right) \qquad (4)$$

where C is the link capacity in packets/second, computed for packets of the specified default size.

3. CONCLUSION

In today's TCP networks, explicit congestion notification (ECN) is the only explicit mechanism which delivers congestion signals to the source. We present a traffic management scheme based on an enhanced ECN mechanism. In particular, we used adaptive ECN, which conveys more accurate feedback information about the network congestion status than the current ECN scheme. We have designed a TCP source reaction that takes advantage of the extra feedback information that have received from receiving end in the form of notification packet and tunes better, its response to the congestion than the current schemes. So to enhance the networks transmission by using the Adaptive ECN technique we attain feasible solutions to avoid the network congestion. The further work is to implement this technique in real time system and to observe the behavior of the systems.

4. REFERENCES

[1]Random Early Detection Gateways for Congestion Avoidance Sally Floyd and Van Jacobson, Lawrence Berkeley Laboratory, University of California

[2] Network Working Group K. Ramakrishna, Request for Comments: 3168 TeraOptic Network.

[3] ECN protocols and the TCP paradigm, Teunis J. Ott ,June 30 1999[5] Bayesian Packet Loss Detection for TCP mahur Fonseca and Mark Crovella

[4] An ECN-based end-to-end congestion-control framework: experiments and evaluation Koenraad Laevens Peter Key Derek McAuley

[5] The Power of Explicit Congestion Notification Aleksandar Kuzmanovi Department of Compute Science Northwestern University

[6] S. Floyd and V. Jacobson, "Random early detection gateways for congestion avoidance," IEEE/ACM Transactions on Networking, vol. 1 no. 4, pp. 397–413, August 1993

[7] Promoting the Use of End-to-End Congestion Control in the Internet Sally Floyd and Kevin Fall : IEEE/ACM Transactions on Networking, vol. 7, no. 4, pp. 458–472, 1999. [Online]. Available: citeseer.nj.nec.com/article/floyd99promoting.html

[8] Adaptive Multi-level Explicit Congestion Notification, Mukundan Sridharan, Arjan Durresi, Raj Jain Dept. of Computer and Information Science, The Ohio State University, Columbus, OH

[9] S. Floyd, "Red: Discussions of setting parameters,"http://www.aciri.org/floyd/RED parameters.txt, November 1997.

[10]V. Jacobson, K. Nichols, and K. Poduri, "Red in a different light,"1999. [Online]. Available:citeseer.nj.nec.com/jacobson99red .html

[11] T. Ziegler, S. Fdida, and C. Brandauer, "Stability criteria for red with bulk-data tcp traffic," Technical report, August 2001. [Online]. Available: http://www-rp.lip6.fr/ sf/WebSF/PapersWeb/red.net2000.pdf

[12] K. Ramakrishnan and S. Floyd. The addition of explicit congestion notification to IP, Sept. 2001. Internet RFC 3168

[13] A. Kuzmanovic. The power of explicit congestion notification (extended version). Northwestern University Technical Report, May 2005.

[14] L. Le, J. Aikat, K. Jeffay, and F. Smith. The effects of active queue management on web Performance. In Proceedings of ACM SIGCOMM '03, Karlsruhe, Germany, Aug. 2003.

[15] L. Le, J. Aikat, K. Jeffay, and F. Smith. Differential congestion notification: Taming the elephants. In Proceedings of IEEE ICNP '04, Berlin, Germany, Oct. 2004.

[16] R. Mahajan, S. Floyd, and D. Wetherill. Controlling high-bandwidth flows at the congested router. Proceedings of IEEE ICNP '01, Riverside, CA, Nov. 2001.

[17] V. Paxson and M. Allman, Computing TCP's retransmission timer, Nov. 2000. Internet RFC 2988.

[18] Jain, R., "A Delay-Based Approach for Congestion Avoidance in Interconnected Heterogeneous Computer Networks", Computer Communication Review, V.19 N.5, October 1989, pp. 56-71.

[19] Jain, R., "Congestion Control in Computer Networks: Issues and Trends", IEEE Network, May 1990,pp. 24-30.