Combination of GREEN and SHRed AQM For Short-Lived Traffic

Ahmad Qadafi Shaii, Roslan Ismail, Jamilin Jais

UNITEN, College of Information Technology, Universiti Tenaga Nasional, Km 7, Jalan Kajang-Puchong, 43009 Kajang, Selangor, Malaysia.

ABSTRACT

The majority of traffic flows dominate Internet traffic is Web interactions where they are short-lived HTTP connections handled by TCP. Short-lived traffic is more sensitive to delay and has small congestion windows cwnds. This paper introduces a new active queue management (AQM) algorithms based on combination of GREEN algorithm and SHRED, to tackle issues on Short-lived flows. Active Queue Management (AQM) refers to a method to enhance congestion control, and to achieve tradeoff between link utilization and delay. Several example of AQM model is Random Early Detection (RED), Blue and GREEN (Generalized Random Early Evasion Network). RED has the potential to overcome some of the problems such as synchronization of TCP flows. To evaluate the performance of new algorithm, network simulation has been done using NS-2 simulation. This study provides a series of NS-2 experiments to investigate the behavior of new algorithm. The results show improvement on short-lived traffic.

Keywords

Active Queue Management

1.0 INTRODUCTION

Network congestion control is a critical issue and high priority, especially the growing size, demand, and speed (bandwidth) of the increasingly integrated services networks. Designing effective congestion control strategies for these networks is a challenge because of the complexity of the structure of the networks, nature of the services supported, and the variety of the dynamic parameters involved.

There is several definition of network congestion. According to [5], network congestion is defined by an increase in packet delays or packet loss from buffer overflow. International Telecommunication Union (ITU) defines congestion as a state of network elements (e.g. switches, router and transmission links) where network is not able to meet the negotiated network performance objectives for the established connections and/or for the new connection requests. Congestion is not a resource shortage problem but a dynamic allocation problem. For example, if a network buffers in a router is expanded to alleviate congestion problem, it might not eliminate congestion. If more buffers are in place, it will lead to a delay in processing a packet because larger buffer and adding buffers to a congested network might even increase the amount of congestion since retransmission due to protocol time-outs consume more resource than consumed by retransmission caused by queue overflow. Several studies have been embarked on congestion control technique since the "congestion collapsed" [1] in the eighties. Congestion collapse had been predicted by Nagel [7] in 1984.

Active queue management is one of the network congestion solutions. Queue management is defined as the algorithms that manage the length of packet queues by dropping packets when necessary or appropriate. Active queue management is expected to eliminate global synchronization and improve quality of service (QoS) of networks. The expected advantages of active queue management are increase in throughput, reduced delay, and avoiding lock-out.

The main goals of the paper are 1 Design of a new AQM which combined of existing GREEN and SHRED active queue management (AQM). 2 Simulation studies of new algorithm and it comparison to others AQM. 3 Discovery through experimental and analysis.

The paper is organized as follows. In section II will highlight into the techniques that are available that could alleviate the congestion problem faced in internet network. The active queue management system (AQM) namely SHRED and BLUE are explored in section III. Section IV will explain the new proposed algorithm is developed. The simulation carried out showed that the proposed algorithm able to address short-lived traffic is shown at chapter section V. Lastly section VI will discuss on conclusions.

2.0 ACTIVE QUEUE MANAGEMENT AQM

Active Queue Management (AQM) algorithms have been designed to control the queue utilisation in routers supporting TCP traffic. This would therefore be able to prevent congestion and resulting packet loss as much as possible. Basically AQM will impose dropping mechanism to arriving packet for purpose to maintain it queue occupancy. AQM consider as a proactive method in preventing a full queue. Many different algorithms have already been presented in literature. In the current Internet, the TCP protocol detects congestion only after a packet has been dropped at the router. The TCP source uses the receipt of the three duplicate acknowledgements or the expiration of a retransmit timer as indication of congestion. Active queue management mechanisms detect congestion before the queue overflows and provide an indication of this congestion to the end nodes. With this approach TCP does not have to rely only on buffer overflow as the indication of congestion since notification happens before serious congestion occurs.

2.1 Active Queue Management AQM Algorithm

Section 2 has discussed on AQM and this section will discuss more GREEN and SHRED algorithm.

2.1.1 GREEN

The GREEN algorithm [2,4] applies mathematical models of the steady state behavior of TCP connections as indicator for dropping packets. GREEN attempts to prevent congestion from occurring and to ensure a higher of fairness. Mathis et al. [3] show that a connection's throughputs at steady state calculate using the following steady state equation.

BW is the bandwidth/throughput of the connection, MSS is the maximum segment size, RTT is its round trip time, p is the packet loss probability and c is a constant depending on the acknowledgement strategy being used, as well as on whether packets are assumed to be lost periodically or randomly.

When this value is used as the dropping probability for congestion notification, GREEN forces flows to send at their fair-share rate. Since p value depends on the number of flows and the RTT of each flow, congestion notification is more aggressive for large N and small RTT. By including the RTT as an inverse parameter, GREEN also eliminates the bias to connections with smaller RTT with respect to throughput, flows with smaller RTT can increase their window size faster due to this smaller RTT and are therefore more aggressive. These flows grab more than their fair share of bandwidth, which leads to this bias. GREEN does not require any information about the congestion window size compare SHRED which will be discussed later. The implementation of GREEN relies on the knowledge of flow RTTs and the total number of active flows N.

2.1.2 SHRED

SHRED[9] (SHort-lived flows friendly RED) address short-lived flows using value of congestion windows (cwnds) because the short-lived is indicated by small *cwnds* and average is around 10kb . The SHRED algorithm operates similar as RED but SHRED algorithm used the tcp congestion windows *cwnd* to compute drop probability value within minimum and maximum threshold. Drop probability is based on ratio of a *cwnd* to the weighted average *cwnd*, called short-lived flow adjustments (SA).

Short lived traffic is indicated by slow start *tcp* and also by low value of *cwnds* because the request is small and average is around 10kb. In this algorithm, probabilistic value is governed by *cwnds*. The way SHRED algorithm operates is

similar to RED but SHRED algorithm is based on tcp congestion windows *cwnd* to compute drop probability value within minimum and maximum threshold. Drop probability is based on ratio of a *cwnd* to the weighted average *cwnd*, called short-lived flow adjustments (SA).

$min_{th\text{-}mod} = min_{th} + ((max_{th} - min_{th\text{-}mod}) \times (1 - (cwnd_{sample} / cwnd_{avg}))$

$max_{p\text{-}mod} = max_{p} \times (max_{th} - min_{th\text{-}mod}) / (max_{th} - min_{th})$

$p_b = max_{p\text{-}mod} \times (max_{th} - min_{th\text{-}mod}) / (max_{th} - min_{th\text{-}mod})$

The average *cwnd* used in this algorithm is a weighted average that weighs *cwnds* of arriving packets. Value *cwndavg* and *cwndsample* important because it dictates drop probabilistic. The ratio of *cwndavg* and *cwndsample* control value *minth-mod*. Based on below formula value *minth-mod* is equal *minth* if *cwnd* ratio value is equal to 1. If *cwnd* ratio value is more than 1, value *minth-mod* is lower *minth* and value *minth-mod* is remain between *minth* and *maxth* if cwnd value is less than 1.

3.0 THE NEW PROPOSED ALGORITHM

The new proposed algorithm combine the methods found in GREEN and SHRED. This new algorithm is based on a mathematical model on steady state flow and cwnd. The algorithm calculates the probabilistic drop value for steady state using knowledge of the steady state behavior of TCP connections. Short-lived traffic is identified by value rtt less than 1ms, this because short-lived has small *cwnd* 10kb and it lead to less *rtt* value. Below is the new algorithm.

```
if(estimate_rtt >= 1ms)
    {
     Calculate p value using steady state formula
    }
else
   {
     Calculate p value using SHRED method
    }
```

Each of traffic will go through a filter process for classification purposes. The algorithm will then check the *rtt* value and if the *rtt* value is more than 1 ms it will categorize the traffic as steady flow. If the *rtt* value is less than 1 ms, it would then mark it as a short lived traffic. The new algorithm will then calculate the probabilistic value once the classified process is done.

4.0 SIMULATION RESULT

This section show and discuss simulation results that have been carried.. The graphs show the performance of the simulation results so that the performance metrics are compared and analyzed. Section 5.1 will discuss utilization of queue size based on the new algorithm. Section 5.2 will explains the results of throughput. This chapter will be summarized at the end of this section.

4.1 Queue Size

Figure 1.1 shows that Drop Tail utilise most of the queue but the new algorithm has a lower queue length than Drop Tail. Drop Tail uses almost all the queue spaces because Drop Tail has no control on the queue and would start to drop packet when queue is full. Once a packet is drop tcp will slow down the packet transmission. If we look at the new algorithm, it starts to receive packets at the beginning of the simulation and starting to drop packets at the 3rd second. The new algorithm does not allow packets use all the queue length as Drop Tail. The new algorithm would adjust the packet enter to queue to 1000 packets compared to Drop Tail where only 3000 packets queueed on 4th second. The simulation start to release short lived on the 12th to 24th sec and show the new algorithm started to drop packets but still, if we compare it to Drop Tail as it totally drop packet as expected because Drop Tail is not friendly to short lived packet.

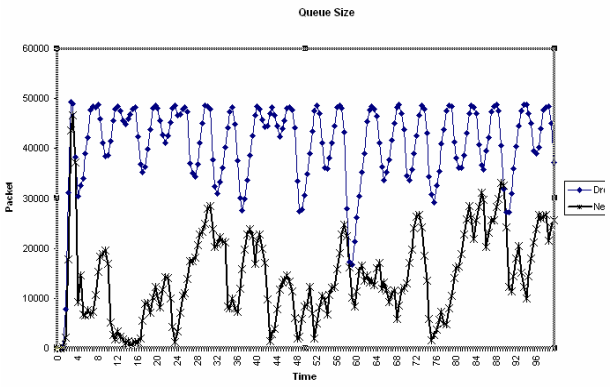

Figure 1.1

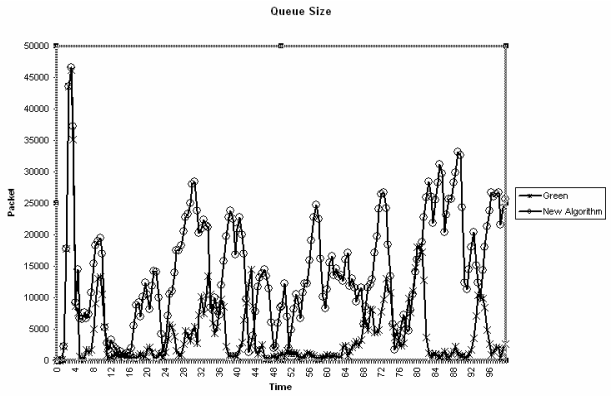

Figure 1.2

Figure 1.2 shows that the Green algorithm used lower queue length compared to Drop Tail and the new proposed algorithm. Green utilised almost all queue length at beginning of simulation starting from time 1st to 2nd second, it allow maximum packet enter in queue. Then on the 2nd to 4th second it started to drop the incoming packets. It shows that Green trigger the dropping mechanism once the queue was almost full at the beginning of simulation. Starting from the 4th second, the Green algorithm allows several packet to enter the queue and from 12th second it started to drop all packet because short lived packet starts to coming in and the Green algorithm will drop all busty or short lived traffic. If we look at the following chart, it shows it drop all traffic at 12th to 30th time and 45th to 64th time. From figure 5.1 and 5.2 it show new algorithm optimize almost haft of queue occupant compare Drop Tail which use almost the maximum queue and Green algorithm which Dorothy not tolerate short lived traffic.

On 44th to 64th second and on 84th to 92nd second, the same patent showed the same behavior of the algorithms as explained before. At the end of simulation, Green still fluctuate and Drop Tail still consumed the majority of the queue. On the other hand, the new algorithm utilize only half of the queue. These simulations showed to us that the new algorithm does much less queue compared to the Drop Tail and the Green algorithm.

In Figure 1.3, all the three algorithms usage of queue is shown on the same chart. This makes it easier to show comparison in term of queue usage by all the three algorithms. Figure 1.3 show us that the new algorithm has better utilization over queue length compared to Drop Tail and green algorithm. The new algorithm will not drop packets and it optimized the majority of incoming traffic. It has a balanced combination with process packets and at same time will drop some percentage to control the incoming traffics. Towards the end of the simulation, Drop Tail still optimises almost the maximum queue. Green used less queue than Drop Tail; but its queue utilization is still much higher when compared with the new algorithm

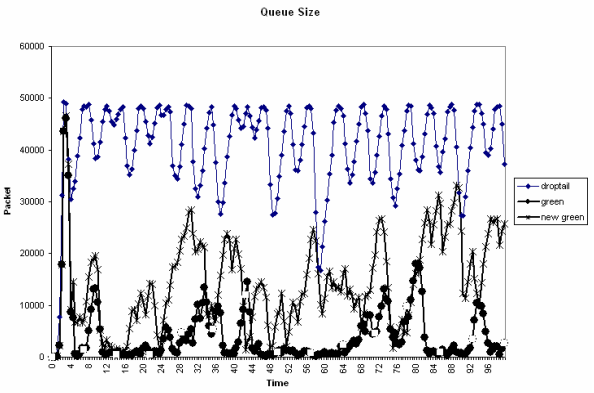

Figure 1.3

4.2 Throughput

Comparison of throughput between Drop Tail and the new algorithm was simulated in figure 1.4. Throughout the process, it can be clearly seen that Drop Tail has lower throughput when compared with the new algorithm. This is due to

the fact that Drop Tail hold more packet in queue and therefore it take more time for a packet to be processed in Drop Tail. Even though Drop Tail do not drop more packets but the effect is on throughput. New algorithm has a better throughput compared to Drop Tail, but it still showed a drop in throughput between 12^{th} to 17^{th} seconds (figure 1.4). This is because simulation started to release short lived traffic during that period.

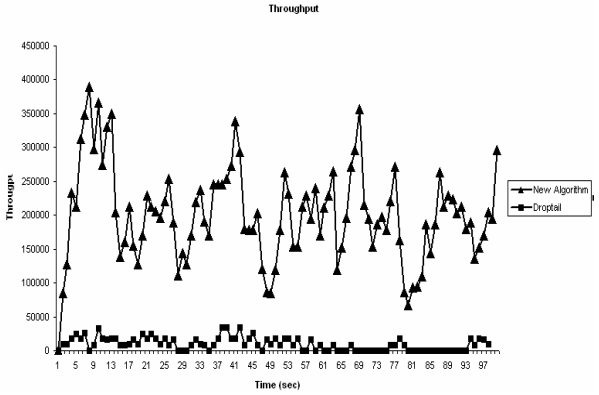

Figure 1.4

In figure 1.4, in term of throughput, the new algorithm shows consistent higher throughput until the end of simulation without aggressively dropping packets. The new algorithm maintains high throughput compared to Drop Tail and at the same time the new algorithm optimises queue occupancy. The difference in throughput is huge as it can be clearly seen that Drop Tail throughput never exceeded 50,000 throughout the simulation while the new algorithm's throughput is always within 100,000 to 300,000.

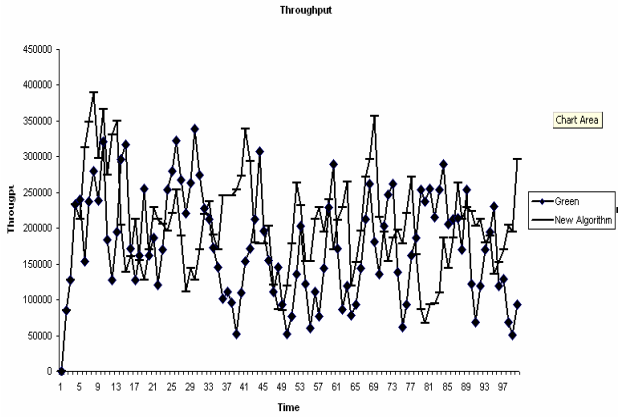

Figure 1.5

When referring to Figure 1.5, it can be shown that generally Green has lower throughput compared to the new algorithm In term of the queue utilisation, Green drop aggressively because it does not entertain short lived traffic. Green also shows a declining trend on throughput at end of simulation.

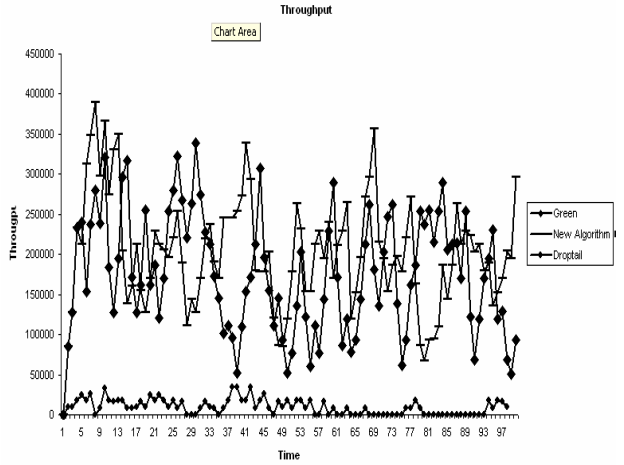

Figure 1.6

Figure 1.6 shows throughput result and comparison between GREEN, Droptail and tne new algorithm. It clearly shows that the new algorithm has outperformed all algorithm and it maintained a higher throughput until the end of the simulation. Thus, we can conclude that the new algorithm has a better throughput when compared with Drop Tail and Green.

5.0 CONCLUSION

The results of this paper show that the new algorithm performs better than GREEN and Drop Tail. New algorithm optimized moderate queue occupancy compared to the Drop Tail that used almost 100% queue occupancy. This lead to high throughput for the new algorithm and packets are queued for shorter time compared to Drop Tail before it gets to be processed. GREEN optimized lower queue occupancy and it lead end point need slow packet transmission because GREEN allows fewer numbers of packet to enter the queue. Throughput result for new algorithm show it has better output or result compared to Drop Tail and GREEN. This shows that new algorithm has out performed all AQM by maintaining a higher throughput throughout the entire simulation.

6.0 FUTURE RESEARCH

The new algorithm has been demonstrated in the simulation topology with FTP flows and HTTP flows. The following is a list of possible directions for future research.

- To further refine new algorithm by employing ECN. The expectation is that ECN will only improve new algorithm even more than this study has showed and thus would give us a better internet connection quality.

- To equip new algorithm with delay-sensitive traffic such as streaming real-time audio and video. Since many of these are UDP-based, this benefit provides incentive for UDP and similar transport layer protocols to adopt TCP friendliness in congestion-control mechanisms.

REFERENCES

[1]. B.Suter, T. V. Lakshman, D. Stiliadis, A. K. Choudhury, "Buffer Management Schemes for Supporting TCP in Gigabit Routers with Per-flow Queuing," IEEE Journal on Selected Areas in Communications, Vol.17, No.6, June, 1999, pp1159-1169.

[2]. Mikkel Christiansen, Kevin Jeffay, David Ott, F. Donelson Smith, "Tuning RED for Web Traffic," ACM SIGCOMM2000, Stockholum, Sweden, August, 2000

[3]. V. Firoiu, M. Borden, "A Study of Active Queue Management for Congestion Control," IEEE INFOCOM2000, Tel-Aviv, Israel, March 26-30,2000

[4]. S. Shenker, L. Zhang, D. Clark, “Some Observations on the Dynamics of a Congestion Control Algorithm, “ACM Computer Communication Review, 20(4), October 1990.

[5] Jim Martin and Arne Nilsson, “The Evolution of Congestion Control in TCP/IP: from Reactive Windows to Preventive Rate Control”, North Carolina State University

[6] Sally Floyd, "TCP and Explicit Congestion Notification," ACM Computer Communication Review, vol 24, pp. 8-23, Oct. 1995.

[7] A. Mauldin, "Global Internet Backbone Growth Slows Dramatically", http://www.telegeography.com/press/releases/2002/16-oct-2002.html: TeleGeography, Inc., 2002

[8] B. Wydrowski and M. Zukerman, "GREEN: An Active Queue Management Algorithm for a Self Managed Internet" Proceedings of ICC 2002, New York, 2002. pp. 2631-2635.

[9] Mark Claypool, Robert Kinicki, Matthew Hartling, Y. Jin, D. Raz, Y. Shavitt, and L. Zhang, “Active Queue Management for Web Traffic” Dept. of Comput. Sci., Worcester Polytech. Inst., MA, USA.