Engineering and Technology Journal e-ISSN: 2456-3358
Volume 08 Issue 01 January-2023, Page No.-1789-1794
DOI: 10.47191/etj/v8i1.01, I.F. – 6.484

Using Random Early Detection for Enhancing Network Performance by Adjusting Lower Thresholds and Average Queue Sizes

Nghiem Van Tinh[1], Pham Quang Hieu[2], Dao Ngoc Ton[3]
[1,2,3] Thai Nguyen University of Technology, Thai Nguyen University Thai Nguyen, Vietnam

ABSTRACT: In internet media, the created information will move to network nodes called the routing information, from one node to another based on the request made. When data loads increase in communication links, routers must be designed to supply reasonable data flow from resource to its target, to all network nodes. In case accessed data is heavy, routers can be deadlocked and reduce router's bandwidth which leads to increase packet loss. In order to sustainably transmit packets through such routers, they need to be designed and provided with effective deadlock avoidance algorithms. Formerly, these algorithms were used as Active queue management-AQM, Drop Tail algorithm (DT), and Random Early Detection-RED. This study suggests an innovative RED named LtRED (Lower threshold of RED) to address RED's shortcomings. Research results evaluated on NS2 showed the innovation LtRED algorithm was better than RED in terms of packet loss rate, average queue delay, and average throughput.

KEY WORD: Deadlock, Active queue management, RED, Average queue size.

1. INTRODUCTION

Based on internet protocol, the internet was designed to provide transmit data services for users by utilizing Transmission Control Protocol-TCP or user Datagram Protocol-UDP protocols. Recently, the rapid development of the internet resulted in increasing internet deadlock. This led to a decrease in internet efficiency. The deadlock was found when buffer memory was full and out-of-date packets were discarded leading to decreasing in network throughput. Deadlock is the main problem that influences on quality of network service. The amount of packet loss, delay due to transmission and average throughput is the principal issue of network. Therefore, cutting down packet loss rate, delay, and average throughput are the keys in the management techniques to improve quality and network service [1]. TCP is a protocol of transport layer which is a reliable and accessible protocol that is used popularly on the Internet. It supplied an avoidance mechanic and control deadlock on the Internet. When being used, TCP applied several techniques to achieve network efficiency and prevent deadlock [2, 3]. In order to address the problem of network deadlock, many algorithms were suggested such as Drop Tain and the AQM. In there, Drop Tail resolved packets at the queue store following rule FIFO- when hang hoi is full, the next packets are lost. Active queue strategies were used to replace Drop Tail in the Router. AQM predicted the possible deadlock and deleted packets based on random probability instead of waiting for... As a result, the router informed controlling speed power button instead of going window size down. This resulted in decreasing packet loss and increasing average throughput in the network. Many queue management strategies were proposed such as RED, ARED,… Floyd and Jacobson [4] suggested the source RED. RED prevented congestion by applying avg parameter showing the average queue. Average parameter was calculated by relying on queue weighting, queue size, and the previous average value. Following that, using avg is to compare with lower thresh and upper thresh tomake decisions on addressing packets. The packet is not dropped if average is lower than the lower threshold. The packet will be dropped according to the probability determined based on the average, lower and higher thresh if avg is between them. The packet will be deleted if the average value is higher than the threshold. By detecting deadlock early, RED has shown its advantages over Drop Tail in reducing packet loss, reducing transmission delay, and increasing average throughput. However, when network data spiked, RED proved ineffective in improving network performance. In this study, the author will present a method to enhance the above-mentioned network performance.

2. RANDOM EARLY DETECTION

Active queue management mechanism based on queue size has been introduced by researchers in recent years, such as RED, ENRED, ARED [4, 6, 7].

RED algorithm, which was proposed by Floyd and Jacobson [4] was designed with the objectives to minimize packet loss and

queuing delay, avoid global synchronization of sources, maintain high link utilization and remove biases against bursty sources. RED is possible to be resistant to source nodes simultaneously reducing the window size, maintaining high throughput through the RED queue, as well as low latency, along with fair treatment between outgoing TCP connections through the queue. The implication of RED is that, for each packet arriving at the router, the average queue size is calculated using a low-pass filter in the cases where the queue is empty and the queue is not empty. The calculated avg is then compared with two thresholds (lower min_hold and upper max_hold) in the router's buffer to decide when to drop the packet. These thresh values are fixed, in the performance evaluation simulations of RED [4], [5], the authors take max_hold = 3. Min_hold.

RED algorithm implements its process in two stages: One is for computing the average queue size and the packet drop probability, which determines the degree of burstiness in the router buffer. The average queue size is calculated based on the current queue size q, the queue weight (ω_q) and the previous average queue size according to the following formulas:

$$aver = (1 - \omega_q).aver + \omega_q.q \quad (1)$$

$$p_d = \begin{cases} 0, & aver < min_{hold} \\ \frac{aver - min_{hold}}{max_{hold} - min_{hold}}.max_p, & min_{hold} \le aver \le max_{hold} \\ 1, & aver > max_{hold} \end{cases} \quad (2)$$

Where $aver$ is the average queue size, p_d is the packet drop probability, max_p is maximum value of p_d

RED the algorithm is given in Algorithm 1.

Algorithm 1: RED

Initialize: *aver* = 0;
For each arrival packet
Calculate *the new aver*
- ***if*** $min_{hold} \le aver < max_{hold}$
- *Calculate the packet drop probability*

With probability p_d
Drop and mark arrival packet
else if $max_{hold} \le aver$
Mark or discard arrival packet
else
Accept arrival packet

End for

This algorithm is used to calculate the decision to drop packets based on the current deadlock level. The goal is to have fairness in marking packets at regular intervals, to avoid mistakes, source nodes and window size reduction, and to RED has solved the problem of early detection of congestion, increased transmission efficiency, and avoided global synchronization. RED has variations that tend to control average queue latency, while maintaining high line utilization efficiency, reducing packet loss rates, reducing global synchronization, and bursting connections.

2.2 Enhanced Random Early Detection (ENRED)

Alshimaa and colleagues proposed ENRED [6]. ENRED uses another parameter besides the queue weight ω_q, called the target queue q_t. This parameter q_t is determined by the difference of the current queue size and the average of the lower and upper thresholds. The average queue size is calculated by the formula (3). ENRED reduces the average queue size of RED, which in turn reduces queue latency and reduces packet loss.

$$aver = q_t(1 - \omega_q) + q.(q_t - \omega_q) \quad (3)$$

Here, ω_q is the queue weight and q is the current queue. If the value of q is small enough, the $aver$ will tend to change little, and will be less affected for short bursts of packets.

3. THE SUGGESTED IMPROVING NETWORK PERFORMANCE APPROACH

The proposed method(LtRED) combines fine-tuning of the lower threshold and average queue size to control congestion in the router's cache in the early state before the cache is full which is an extension of RED [8] – [12]. It takes into account the period when the queue is empty (the idle period) by estimating the number m of small packets that could have been transmitted

by the router during the idle period and the number of packets resides in the buffer over a period of time. The purpose of this proposed method is to increase the average throughput, reduce the average queuing delay and reduce the packet loss rate in the cases of congestion: light congestion, and severe congestion. LtRED extends RED by combining a lower threshold fine-tuning based on the average queue size avg and recomputing avg when comparing the current queue size with the thresholds. It calculates the average queue size each time a packet arrives based on the current queue size and the previously calculated $aver$. We refine the lower thresh and average queue size according to the following expression:

$aver = (1 - \omega_q).aver/a + \omega_q.q$, with a > 1 (4)

$min_{hold} = b.aver$, with b > 1 (5)

Where, a and b are two values which reasonable selected to get the low queue-delay**.**

When having arirval packets, we calculated the average queue size depending on the empty queue or not. Then, compare aver with two thresholds in the router's cache to determine the level deadlock in the queue. If $aver$ is less than the lower thresh, we recalculate the avg and tweak the lower threshold and then execute RED. If $aver$ is greater than or equal to the upper threshold, then call RED implementation like the original algorithm, ie, if $aver$ is greater than the upper threshold, discard the packet with probability 1, and if avg is between the lower threshold and the upper threshold, the accuracy is calculated. drop rate and perform packet drop according to that calculated probability.The proposed method is improved from RED described in algorithm 2 as follows:

Algorithm 2: LtRED

Initialize: *aver: = 0; count =0;*

For each arrival packet

Calculate *the new aver as follows:*

- ***if*** *q==0 then aver:=* $(1-\omega_q)^{f(time-q\_time)}$**aver;*
- ***if*** *(q!=0) and(*$aver < min_{hold}$*) then aver:= (1-*ω_q*)* aver +* ω_q **q;*

Calculate D_p *and its related parameters, and implements packet dropping, as:*

If ($min_{hold} \leq aver < max_{hold}$)

count=count+1

$p_d = \frac{aver - min_{hold}}{max_{hold} - min_{hold}}.max_p$;

$D_p = \frac{p_d}{1 - count * p_d}$;

With probability D_p

Mark arrival packet; count:=0;

else if *(*$max_{hold} \leq aver$*)*

Drop and mark arrival packet; count=0;

else

count=-1; $min_{hold} == b.aver$;

When q==0 **;** *q_time= time;*

End for

Where: averg: average queue size; q_time: start of the queue idle time; count: packets since last marked packet; D_p: current packet-marking probability; Time: current time.

4. SIMULATE AND EVALUATE THE NETWORK EFFICIENCY

In this study, we perform simulation using NS-2 network tool for two methods RED and proposed method LtRED.

4.1. Simulation topology

Simulation topology and parameters are designed as shown in Figure 1. While the bandwidth and queue-delay of the links are 40Mbps and 4ms, except for the R-R5 link which has 3Mbps and 40ms bandwidth and queue-delay. The R-R5 link queue size is 40 packets. The duplex link between node R and R5 uses a queue type of RED or LtRED. Apart from the parameter of bandwidth, queue-delay, queue size, and others about network throughput time and network throughput stop time simulation time need to make sure to be the same in simulation comparing.

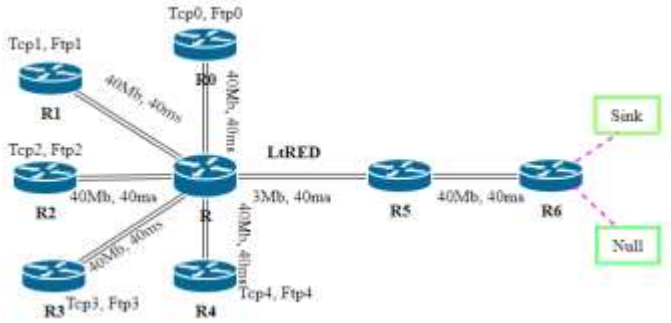

Fig. 1: Simulation topology

4.2. Number of dropped packets

The number of dropped packets is determined in two cases corresponding to two different levels of deadlock in the network:

Case 1: Mild deadlock

The values for the total number of incoming packets and the total number of dropped packets for both strategies in the case of mild deadlock are shown in Table 1. Table 1 shows that the number of dropped packets of LtRED is significantly reduced compared to RED.

Table 1: Compare dropped packets in case of mild deadlock

RED			**LtRED**		
The amount of arrival packet	The amount of drop packet	The percent rate of drop packet	The amount of arrival packet	The amount of drop packet	The percent rate of drop packet
287450	1636	0,568	290443	1016	0,350
287090	1637	0,606	292056	1015	0,347
280097	1692	0,574	294135	1012	0,344
287697	1694	0,578	290466	992	0,342
289250	1698	0,577	294088	1073	0,359

Case 2: Heavy deadlock

The values for the total number of incoming packets and the total number of dropped packets for both strategies in the case of heavy deadlock are shown in Table 2.

In this case, the number of dropped packets of LtRED is greatly reduced compared to RED. In the mild deadlock, the total packet transmission of LtRED is larger than that of RED, while the number of dropped packets is much smaller than that of RED. The reason is that LtRED reduces the value of avg every time it finds it below the lower threshold. This is the case where the packet will not be dropped.

Table 2: Compare dropped packets in case of heavy deadlock

RED			**LtRED**		
The amount of arrival packet	The amount of drop packet	The percent rate of drop packet	The amount of arrival packet	The amount of drop packet	The percent rate of drop packet
345448	4013	1,159	338527	2256	0,667
346065	4013	1,160	337132	2254	0,672
345045	3987	1,154	334330	2204	0,658
345582	4023	1,165	334305	2205	0,659
345341	4402	1,273	337745	2258	0,669

4.3. Average queue delay of packets

The average queue delay is determined by the total delay of the packets in the queue divided by the number of packets entering the queue. It is identified in two cases: Mild obstruction and severe obstruction. The results of the comparison are shown in Figure 2 in case of mild deadlock of and Figure 3 in case of heavy deadlock.

In both cases, the average queue latency of LtRED is smaller than that of RED: with a mild deadlock of 0.024, and with a heavy deadlock, it is 0.032. The more severe the deadlock, the greater the average queuing delay of packets.

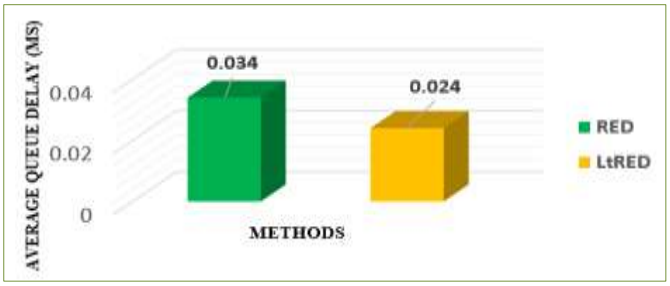

Fig. 2. Average queue delay of packets comparison in case of mild deadlock

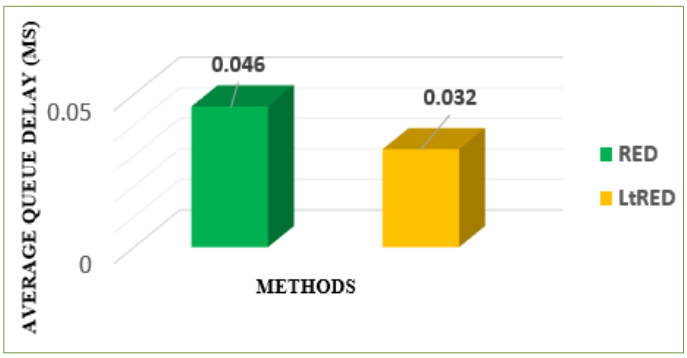

Fig. 3. Average queue delay of packets comparison in case of heavy deadlock

4.4 Average throughput

Average throughput is defined as the total size of packets received divided by the difference between the time to receive the last packet minus the time to receive the first packet.

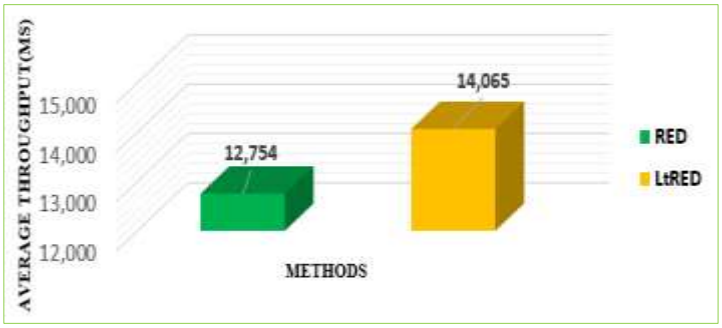

Fig. 4. Average throughput comparison between RED and LtRED

In summary, the aforementioned results demonstrated that the proposed method outperforms the RED method based on the total number of dropped packets, the average queue delay, and the average throughput.

5. CONCLUSION

This study presents an approach for modifying and replacing the existing RED's indicator with new indicators. Based on such a replacement, we obtained a new method, which provided us with a new network model. According to the comparison and evaluation results shown in Tables 1-2 and Figures 2-4, it showed that the proposed LtRED algorithm gave better results than the RED queue mechanism in the following cases: packet loss rate, average queue delay of packets, and average throughput. As a result, the reliability of LtRED's performance enhancement mechanism can be illustrated. In the future, we will study and improve the synchronous operation of queue management mechanisms by improving random early detection in a new-generation network environment.

ACKNOWLEDGMENT

This work was supported by Thai Nguyen University of Technology (TNUT).

REFERENCES

1. Abbasov, B., & Korukoglu, S. Effective RED: An algorithm to improve RED's performance by reducing packet loss rate. Journal of Network and Computer Applications, 32(3), 703-709, 2009.
2. R. J. La, P. Ranjan, and E. H.Abed, Analysis of Adaptive Random Early Detection (ARED), Networking , IEEE/ACM Transaction, Volume 12, Pages: 1079 – 1092, 2004.
3. D. Que, Z. Chen, and B. Chen, "An Improvement Algorithm Based on RED and Its Performance Analysis," 9th International Conference on Signal Processing, 2008.
4. S. Floyd and V. Jacobson, "Random Early Detection Gateways for Congestion Avoidance," *Institute of Electrical and Electronics Engineers(IEEE)*, pp. 1-22, August 1993.
5. M. Khatari and G. Samara, "Congestion Control Approach based on Effective Random Early Detection and Fuzzy Logic," MAGNT Research Report, Jordan, 2015.
6. A. H. Ismail, A. EL-Sayed, I. Z. Morsi, and Z. Elsaghir, "Enhanced Random Early Detection (ENRED)," *International Journal of Computer Applications*, vol. 92(9), pp. 20-24, April 2014.
7. Minseok Kwon and Sonia Fahmy, *A Comparison of Load-based and Queue-based Active Queue Managgement Algorithms*, Dept. Of computer Science, Purdue University, West Lafayette, IN 47906-1398, USA, 2010
8. A. M. Alakharasani, M. Othman, A. Abdullah, and K. Y. Lun, *An Improved Quality-of-Service Performance Using RED's Active Queue Management Flow Control in Classifying Networks*, 2017.
9. J. Song and Z. Zhixue, "Research on the Improvement of RED Algorithm in Network Congestion Control," *Applied Mechanics and Materials*, vol. 713, pp. 2471-2477, 2015.
10. Jianyong Chen, Cunying Hu, and Zhen Ji , Self-Tuning Random Early Detection Algorithm to Improve Performance of Network Transmission, Hindawi Publishing Corporation Mathematical Problems in Engineering Volume, Article ID 872347, 17 pages doi:10.1155/2011/87234, 2011

11. M. Khatari and G. Samara, "Congestion Control Approach based on Effective Random Early Detection and Fuzzy Logic," MAGNT Research Report, 2015.
12. M. M. Abualhaj *et al.*, "FLRED: an efficient fuzzy logic based network congestion control method," *Neural Computing and Applications*, vol. 30, no. 3, pp. 925-935, November 2016.