Revisiting the *Gentle* Parameter of the Random Early Detection (RED) for TCP Congestion Control

Nabhan Hamadneh[1], Mouhammd Al-Kasassbeh[2], Ibrahim Obiedat[1], Mustafa BaniKhalaf [3]
[1] Hashemite University, Zarqa 13133, Jordan
[2] Princess Sumaya University for Technology, Amman 11941, Jordan
[3] Yarmouk University, Irbid 21163, Jordan
Email: {nabhan, imsobeidat}@hu.edu.jo; m.alkasassbeh@psut.edu.jo; mbanikhalaf@yu.edu.jo

Abstract —The Random Early Detection (RED) is used as an Active Queue Management (AQM) Technique for TCP congestion handling. A modification of RED called the *Gentle* RED (GRED) has been proposed by adding the *Gentle* parameter to the original implementation of RED. This parameter has been turned on by default in the NS2 simulator versions *2.1b* and later; claiming that it helps in smoothing out traffic in routers and increases network performance. In this article we revisit this parameter and show, through simulation, that this parameter should be turned off in current simulations of RED using the NS2 simulator and it should be replaced by any adaptation parameter such as the *Adaptive* parameter in ARED.

Index Terms—RED, TCP, Gentle-RED, AQM, NS2

I. Introduction

Active Queue Management (AQM) [1] is a network approach of congestion prevention that works in conjunction with TCP source algorithms [2, 3, 4]. The Random Early Detection (RED) has been proposed as an AQM strategy for congestion handling [5]. In the literature of RED there have been many enhancements applied over its original implementation to increase the performance of TCP networks [6]. The majority of these strategies present an optimal parameter configuration for RED or add extra parameters that enhance the operation of it.

One of the most critical parameters that has been added to the algorithm of RED is the *Gentle* parameter which has been activated in the NS2 simulator [7] release *2.1b* and later releases. In this article we revisit this parameter and argue that it should be turned off in current simulations of RED using the NS2 simulator. We show through simulations that this parameter could harm the overall performance of RED in some situations.

This paper is organized as follows: Section 2 presents a literature review about RED and its implantation. Section 3, presents an analysis of RED's adaptation techniques. In this section, we also present three reasons about not to deploy the *Gentle* parameter of RED in the NS2 simulator. Section 4, presents the simulation topology used in this article. The results are presented in section 5 and section 6 concludes our paper.

Manuscript received May 12, 2018; revised February 23, 2019.
Corresponding author email: nabhan@hu.edu.jo
doi:10.12720/jcm.14.3.229-235

II. Literature Review

RED is a network strategy implemented by intermediate routers. It manages queues in network nodes to prevent congestion using a set of parameters. Table 1 illustrates these parameters. Traditional routers drop packets when the queue becomes full causing immediate reduction in sending rates by TCP sources. This technique is called the Tail Drop (TD) strategy [8]. Although it gives the queue an opportunity to drain out the overwhelming packets, it incurs two main problems. The first problem is the full queue problem and the second is the global synchronization [8, 9].

TABLE I: THE MAIN PARAMETERS OF RED

Parameter	Description
avg	Average queue size.
w_q	A weight parameter, $0 \leq w_q \leq 1$.
q	The current queue size.
p_b	Immediately marking probability.
max_p	Maximum value of p_b.
min_{th}	Minimum threshold.
max_{th}	Maximum threshold.
p_a	Accumulative drop probability.
$count$	Number of arrived packets since the last dropped one.

RED overcomes these problems by maintaining an average queue size parameter (avg) in (1) and drops packets depending on it instead of the actual queue size (q). When the average queue size is less than the minimum threshold no action needs to be taken. If the average is in between the minimum and maximum thresholds, packets are dropped with the probability parameter p_a in (3) which has an initial value of p_b in (2). All incoming packets are dropped when the average queue size exceeds the maximum threshold. In other words, the drop probability is set to one.

$$avg = (1 - w_q) * avg + w_q * q \quad (1)$$

$$P_b = max_p\left(\frac{avg - min_{th}}{max_{th} - min_{th}}\right), \quad (2)$$

$$P_a = P_b(\frac{1}{1-count*P_b}). \quad (3)$$

RED solved the problems of the TD strategy, but researches show that it suffers from some problems like parameter settings, the insensitivity to the input traffic load variations, and the mismatch between macroscopic [10] and microscopic behaviors of queue length dynamics [11]. Therefore, many RED variants have been proposed to solve these problems [12].

There is an approach to enhance the operation of RED by presenting an optimal parameter configuration, because the performance of RED is very sensitive to these configurations. Another approach of tuning the performance of RED is the adaptation of the dropping process. In this approach, the drop rate should be increased or decreased depending on the persistence of the congestion. This goal is achieved by altering some of the parameters of RED, dynamically, while the simulation is running to cope with aggressive traffic. Examples of these adjusting parameters are: the current drop probability, maximum drop probability, the average queue size and the weight parameter.

The *Gentle* parameter in GRED [13] is used to adapt the drop rate of RED dynamically using the NS2 simulator. The next sections argue through simulations that this parameter should be turned off for better configuration of RED and it has to be replaced by any RED adaptation technique.

III. Analysis of RED's Adaptation Technique

RED divides the queue into three segments to deal with the incoming packets in three different scenarios. It uses the average queue size as an indicator of the aggressiveness of congestion and drops packets depending on it. Fig. 1 shows these segments. The strategy will not drop any packet while the average queue size in segment one. All incoming packets will be dropped if the average queue size is in segment three. The drop rate is increased gradually while the average queue size travels through segment two toward segment three. This helps in absorbing congestion gradually before the queue is full. On the other hand, choosing a packet to be dropped randomly helps in preventing the global synchronization problem.

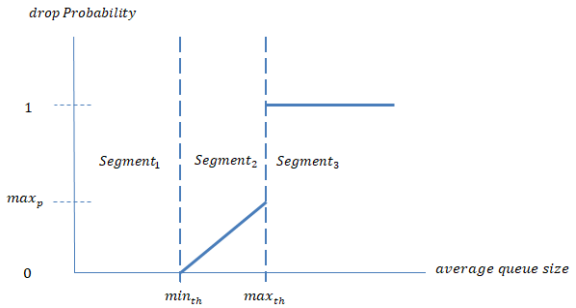

Fig. 1. RED's active queue management

Regardless of the higher network performance produced by applying RED over TD in routers there was a suggestion to add the *Gentle* parameter to RED's implementation [5]. This suggestion has been adopted by the NS2 simulator since version *2.1b* and it is still being used by the current version *2.35*. The idea of the *Gentle* parameter is to differ the dropping with probability one until the average queue size is twice the maximum threshold. Rather than the sudden jump of the drop probability from max_p to *1* in RED, the *Gentle* RED (GRED) strategy increases the drop rate gradually from max_p to *1*. In other words, the queue will be divided into four segments in GRED instead of three in RED. This paper is questioning: does the *Gentle* parameter add real benefits to the performance of RED? Before testing the goodness of the *Gentle* parameter, we present the following notations about the GRED strategy:

First, since every arriving packet is dropped when the average queue size is twice the maximum threshold, then this makes a third drop level for GERD which works exactly the same as the maximum threshold of RED.

Second, the task of the minimum threshold in both strategies RED and GRED is kept the same, in which no packets will be dropped when the average queue size is less than this value.

Third, the drop probability varies from max_p to *1* while the average queue size varies from the minimum threshold to twice the maximum threshold which supposed to make GRED work in adaptation approach like ARED [14] and Blue-RED [15]. In most cases this is not true as we will see in the next sections; because this makes GRED work only as a special case of RED with wider area for segment two. In the next section, we study how this affects the overall performance of GRED.

IV. Simulation Topology

This section tests GRED using two simulation scenarios. In the first scenario we examine the implementation of GRED under light weight traffic. In the second scenario, we examine GRED under aggressive traffic. Fig. 2 shows the network topology that is used to monitor the queue dynamics for different strategies.

In this figure there are three *ftp* applications, each of which is attached to a TCP agent. These agents start propagation from nodes *n0* and *n1* at time *0.02* seconds and stop propagation at time *10* seconds. The nodes *n0* and *n1* are connected through duplex links to *n2* forming feeding connections. The node *n2* is connected to *n3* through duplex link forming bottleneck connection. Finally, the node *n3* is connected to *n4* and *n5* through duplex links forming a TCP sinks.

Fig. 3 shows the parameter configuration for the feeding links which are *5Mbps* bandwidth with *2ms* delay. The maximum queue size is *20* packets with TD queue management algorithm. Fig. 4 shows the parameter configuration for the *ftp* application.

Scenario two uses the same topology but with heavier traffic. In order to make the traffic aggressive; we increase the capacity of the feeding links from *5Mbps* in

scenario one to *100Mbps* in scenario two. The delay of the feeding links is also increased from *2ms* to *10ms*. The bottleneck link capacity is increased from *2Mbps* in scenario one to *10Mbps* with delay *10ms* instead of *2ms*. The *ftp* application starts propagation at time *1.0* second until *10* seconds with *14* packets maximum queue size. Table II shows the configuration of RED's parameters used in the simulation and Table III illustrates the configuration of the topologies in scenarios one and two.

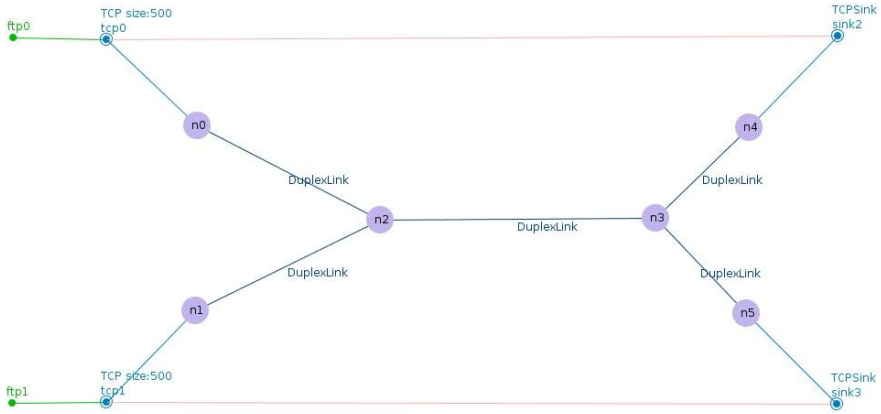

Fig. 2. Simulation network topology

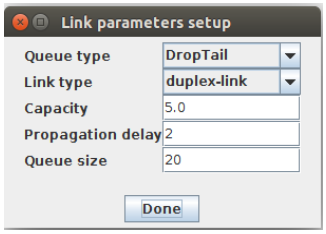

Fig. 3. Link parameter configurations

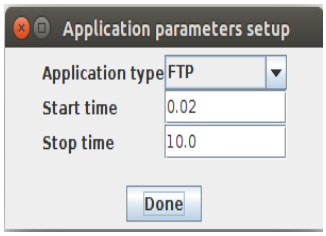

Fig. 4. *ftp* application timing

TABLE II: THE MAIN PARAMETERS OF RED

Scenario	min_{th}	max_{th}	Weight	Packet size
1	5	15	0.002	500
2	3	6	0.002	1500

RED queue is used in the bottleneck link with the configuration illustrated in Table II. In this table, the weight parameter is *0.002* which is the same in the two scenarios. For scenario one, the minimum threshold is *5* packets and the maximum threshold is *15* packets. The packet size is *500* bytes. For scenario 2, the values *3* and *6* are assigned to the minimum and maximum thresholds respectively with a packet size of *1500* bytes.

V. SIMULATION AND ANALYSIS

Since RED is sensitive to the parameter settings, its performance could be enhanced by providing an optimal parameter configuration. Another approach of enhancing RED's performance is through adapting the drop rate in segment two. When GRED was proposed, it was intended to follow the adaptation approach. However, we show in this section that the algorithm of GRED will end up as a new parameter configuration for RED instead of a new adaptation technique. We compare the behavior of GRED with RED and ARED. Here, ARED is considered as an adaptation strategy. Also, we include the Traditional TD strategy in this comparison.

TABLE III: TOPOLOGIES CONFIGURATION

Scenario	Feeding link Speed (Mbps)	Feeding link Delay (ms)	Bottleneck Link Speed (Mbps)	Bottleneck Link Delay (ms)	FTP start time	FTP end time	Max_q length in (packets)
1	5	2	2	2	0.02	10.0	35
2	100	10	10	10	1.0	10.0	14

We argue in this study that the adaptation approaches of RED comparing to the optimal parameter configuration approaches will always provide better congestion solution for the following reasons:

First, the objective of RED is to stabilize the average queue size in between the minimum and maximum thresholds. Therefore, segment two will be the best place to study the behaviors of traffics and the impact of

amendments over the drop probability which is the approach of all adaptation techniques.

Second, the third drop level of GRED which equals to twice the maximum threshold is a high value that is hard to be reached by the average queue size. Therefore, it is very likely for the packets to be dropped due to buffer overflow before GRED reacts to severe congestion scenario. It is much easier for the average queue size in adaptation techniques to hit the maximum threshold and respond quickly to congestion.

Third, even with a proper work for GRED "unpublished" [16], the average queue size will always be high due to the high drop level assigned to it. This makes it harder for the average queue size to go back to normal values after long congestion period. In this case, the strategy will continue dropping innocent packets due to wrong congestion indication. Hence, RED adaptation techniques are better than other techniques in normalizing the drop rate after severe congestion cases.

The performance of any RED strategy can be judged from the view of its queue dynamics. Queues of many oscillations reflect instability in network performance in which the queue goes from the full queue to the idle state and backward very frequent. Queues reflect these kinds of carves in severe congestions like scenario two of this paper. In case of light weight congestions, the dynamics of the average queue size need to be studied whether it is increasing or decreasing in segment two, also if it is stabilized near to the maximum threshold or the minimum threshold. Each case has an indication of network performance that will be discussed in this article.

A. Scenario One

Fig. 5 to Fig. 9 illustrate the queue behaviors of the strategies ARED, TD, GRED and RED respectively. Fig. 6 shows the behavior of the TD queue. In the beginning of the simulation the actual queue size had grown rapidly until the queue was full. A packet was dropped due to the full queue and the sending nodes reduced their sending rates simultaneously causing the global synchronization problem. The queue had been drained quickly and started to increase gradually due to the slower sending rates by the source algorithms. The full queue and global synchronization problems stimulate the proposal of the RED strategy.

Fig. 5. Scenario one, ARE's queue dynamics.

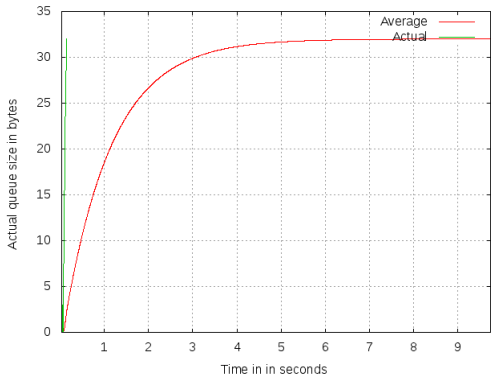

Fig. 6. Scenario one, TD's queue dynamics.

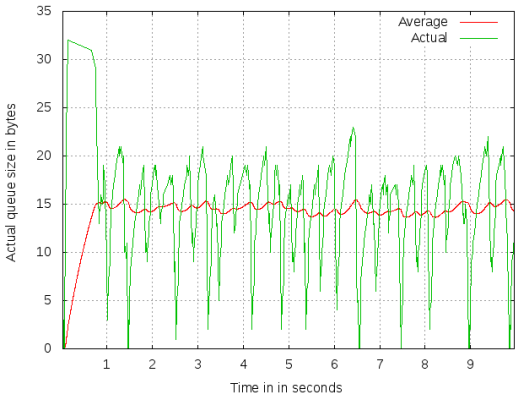

Fig. 7. Scenario one, GRED's queue dynamics.

The average queue size in TD has no impact in the operation of the strategy. Also, the average and actual queue sizes are identical in this scenario due to the light weight traffic. Hence, the behavior of the average queue size is to be ignored in the TD strategy.

The performance of TCP networks is measured by four performance parameters. The best congestion control strategy is the one that could achieve the highest throughput and link utilization with the lowest drop rate and average delay among a set of strategies [17]. All these parameters have tradeoffs in that a strategy could increase the link utilization associated with high loss rate and average processing time of packets. However, queue dynamics still give an overall view of the goodness of any RED based strategy [18].

The strategy that could reduce the oscillations of the actual queue size is considered to have benefits over strategies with much oscillated queues [19]. These kinds of vibrations between the idle and full queue states are incurred by the aggressive congestion reactions in congestion control strategies. These strategies increase the dropping rate in very short time, resulting in a big reduction of the sending rates by source algorithms which degrades the link utilization.

In these types of strategies, the total number of dropped packets for the whole network operation time is not as important as the distribution of these drops. For example, dropping ten packets in the rate of one packet every five seconds for *50* seconds simulation time is better than dropping five successive packets in the middle

of simulation, because this will result in excessive reduction in the sending rates depending on the TCP source algorithm. As a result, the system will take longer time to recover and gain higher sending rates. This behavior could be concluded from the dynamics of the average queue size.

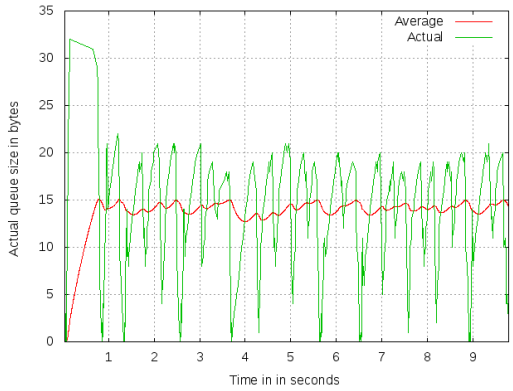

Fig. 8. Scenario one, RED's queue dynamics.

The strategy that could stabilize the average queue size in between the minimum threshold and maximum threshold is considered better in managing packet drops [20]. If the average queue size oscillates around the maximum threshold, then the strategy is very aggressive and the drop rate is high. When the average queue size oscillates around the minimum threshold then the traffic is light with low drop rate.

RED's queue in Fig. 8 comparing to the TD's queue in Fig. 6 stabilized the average queue size at the maximum threshold; preventing the full queue and global synchronization problems. It is clear in Fig. 7 that GRED has not outperformed the original RED strategy because the average queue size has been oscillated around the maximum threshold and never reached twice the maximum threshold. In reality, the *Gentle* parameter has not worked and the only impact of GRED is the graded increase in the drop probability after the average queue size exceeded the maximum threshold. The behaviors of RED and GRED were nearly the same in Figures 8 and 7 respectively. Since they stabilized the average queue sizes around the maximum threshold, they will deliver nearly the same drop rate. However, Table IV shows that RED outperformed GRED in throughput, link utilization, average delay and average processing time. The link utilization here is estimated by the total number of enqueued packets.

ARED comparing to RED and GRED does the adaptation in the right position in segment two. Fig. 5 reflects the behaviors of the average and actual queue sizes of ARED. The average queue size had been increased from the beginning of the simulation until it hit the maximum threshold. Here ARED reacted same as RED by dropping every arriving packet with probability one in order to overcome the severe congestion case. The drop rate had been continually degraded for ARED until the end of simulation time. ARED outperformed RED and GRED in throughput, link utilization, average delay and average processing time as illustrated in Table IV.

B. *Scenario Two*

In this scenario, we increase the severity of the congestion problem by increasing the bandwidth of the feeding links. Figures 9 to 12 show the behaviors of ARED, TD, GRED and RED respectively. Table 3 shows the performance of each strategy. We need such an aggressive network traffic to let the average queue size exceed segment four of GRED to test its implantation at this point [21]. Even with this aggressive scenario the average queue size had not crossed the fourth segment which means that having a drop level with a value twice the maximum threshold is not a practical solution.

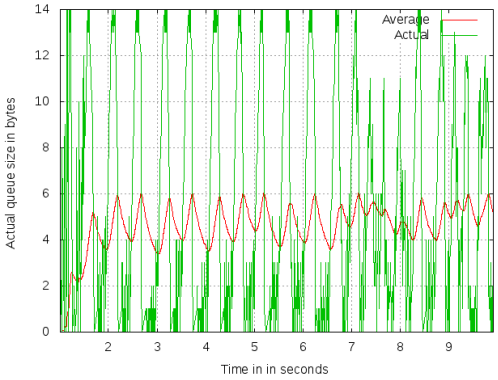

Fig. 9. Scenario two, ARED's queue dynamics.

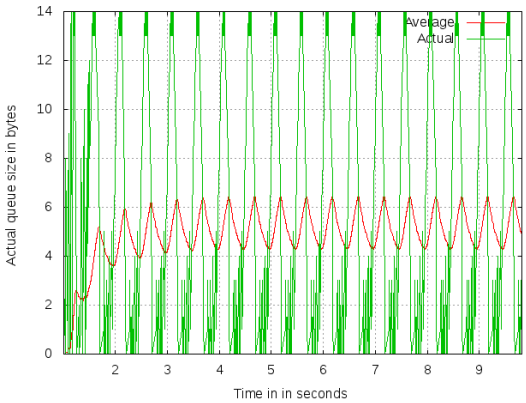

Fig. 10. Scenario two, TD's queue dynamics

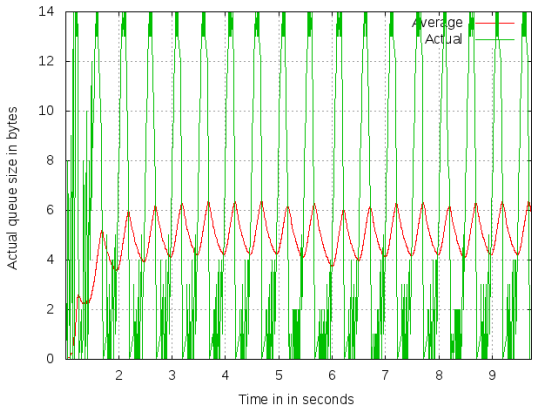

Fig. 11. Scenario two, GRED's queue dynamics.

From Figures 9 through 12, it is clear that all queues were heavily congested. The queues had been alternated between the full queue and idle cases very frequently and the majority of the drops were caused by the full queue problem. In this case, RED and GRED outperformed the TD strategy in the average processing time because the packets had started to be dropped from the tail of the queue when it entered the full case. This increased the delay of earlier arrived packets, as illustrated in Table V, which also increased the average delay time.

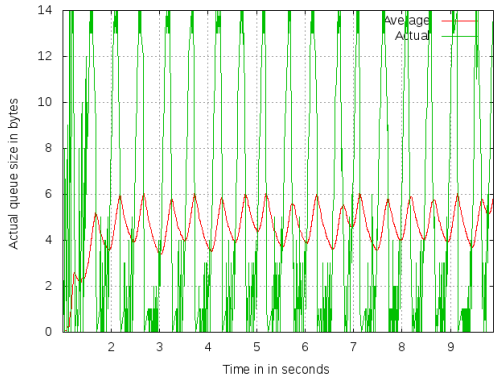

Fig. 12. Scenario two, RED's queue dynamics.

TABLE IV: NETWORK PERFORMANCE FOR SCENARIO 1

Strategy	Throughput	Utilization	Average End 2 End Delay	Average processing time
ARED	12367	12545	0.02455706591	0.003859093313
DT	11918	12042	0.02496078607	0.004267548079
RED	11960	12145	0.02468390241	0.003984954515
GRED	11911	12045	0.02485067799	0.004155023927

TABLE V: NETWORK PERFORMANCE FOR SCENARIO 2

Strategy	Throughput	Utilization	Average End 2 End Delay	Average processing time
ARED	9173	9237	0.02096642513	0.01288433969
DT	9219	9252	0.04286234704	0.03478230611
RED	9143	9265	0.02489266659	0.01680992234
GRED	9191	9235	0.02527038642	0.01718336111

ARED in Fig. 9 outperformed RED and GRED in Figures 12 and 11 respectively. In this scenario of congestion, ARED had successfully broken the full queue phenomenon two times through simulation; one between times *7* to *8* seconds and the second between times *9* to *10* seconds. These two cases are depicted by the reduction of the actual and average queue sizes in Fig. 9. It is clear that there had been no big oscillations in the actual and average queue sizes. In other words, the queue could recover from the full queue problem without going to the idle state. This means that ARED is unlike RED and GRED in heavy congestion situations in that it would absorb heavy traffics and tune out the drop rate. In Table V, ARED shows the lowest processing time and average delay with throughput and link utilization higher than RED and GRED.

VI. CONCLUSION

This paper shows that the *Gentle* parameter of RED using the NS2 simulator is impractical and has to be turned off and replaced by any adaptation parameter such as the *adaptive* parameter of ARED. This suggestion is shown in this article using two simulation scenarios of light weight and heavy weight traffics. The area between the minimum threshold and maximum threshold is the most dynamic area of the queue because the average queue size oscillates in this area for almost all the simulation time. Therefore, the best adaptation space of RED is segment two which is illustrated by the results of this paper. GRED is trying to adapt the drop rate at segment four which is impractical solution. Hence, GRED has to be considered as a special parameter configuration of RED rather than being considered as an adaptation strategy.

REFERENCES

[1] K. Nichols, V. Jacobson, A. M. Ed, and J. I. Ed, "Controlled delay active queue management," *IETF*, 2018.

[2] S. K. Bisoy, B. Pati, C. R. Panigraph, and P. K. Pattnaik, "Analysis of TCP variant protocol using active queue management techniques in wired-cum-wireless networks," in *Computational Intelligence in Data Mining, Part of: Advances in Intelligent Systems and Computing*, Singapore, 2017.

[3] C. A. Grazia, N. Patriciello, and M. Klapez, "A cross-comparison between TCP and AQM algorithms: Which is the best couple for congestion control?" in *Proc. 14th International Conference on Telecommunications (ConTEL)*, Zagreb, Croatia, 2017.

[4] C. A. Grazia, N. Patriciello, and M. Klapez, "Which AQM fits IoT better?" in *Proc. IEEE 3rd International Forum on Research and Technologies for Society and Industry (RTSI)*, Modena, Italy, 2017.

[5] R. Adams, "Active queue management: A survey," *IEEE Communications Surveys & Tutorials*, vol. 15, no. 3, pp. 1425-1476, 2013.

[6] R. J. Briscoe and C. Cairano-Gilfedder, "Signalling congestion," USA Patent US9634916B2, 25 4 2017.

[7] S. Keshav. (2018). [Online]. Available: http://www.cs.cornell.edu/skeshav/real/overview.html

[8] N. Sharma, S. S. Rajput, A. K. Dwivedi, and M. Shrimali , "P-RED: Probability based random early detection algorithm for queue management in MANET," in *Proc. Advances in Computer and Computational Sciences. Advances in Intelligent Systems and Computing*, Singapore, 2018.

[9] W. Lautenschlaeger and A. Francini, "Global synchronization protection for bandwidth sharing TCP flows in high-speed links," in *Proc. 16th International Conference on High Performance Switching and Routing*, Budapest, Hungary, 2015.

[10] T. Burbridge, A. Smith, and P. L. Eardley, "Processing data items in a communications network," USA Patent US20180091431A1, 2018.

[11] D. A. Alwahab and S. Laki, "A simulation based survey of active queue management algorithms," in *ICCBN*, Singapore, 2018.

[12] V. Kushwaha and R. Gupta, "Congestion control for high-speed wired network: A systematic literature review," *Journal of Network and Computer Applications*, vol. 45, pp. 62–78, 2014.

[13] S. Floyd. (2000). Recommendation on using the gentle_ variant of RED. [Online]. Available: http://www.icir.org/floyd/red/gentle.html

[14] L. Wei-yan, Z. Xue-lan, L. Tao, and L. Bin, "A dynamic and self-adaptive TCP friendly congestion control mechanism in next generation networks," in *Proc. Intelligent Systems and Applications ISA*, Wuhan, 2009.

[15] W. Feng, D. D. Kandlur, D. Saha, and K. G. Shin, "BLUE: A new class of active queue management algorithms," *IEEE/ACM Transactions on Networking (TON)*, vol. 10, no. 4, pp. 513-528, 2002.

[16] N. Hamadneh, I. Obeidat, A. Qawasmeh, M. Bsoul, and M. Kharabsheh, "HRED, an active queue management algorithm for TCP congestion control," *Recent Patents on Computer Science*, 2018.

[17] H. Jamal and K. Sultan, "Performance analysis of TCP congestion control algorithms," *International Journal of Computers and Communications*, vol. 2, no. 1, 2008.

[18] S. K. Bisoy and P. K. Pattnaik, "Design of feedback controller for TCP/AQM networks," *International Journal o Engineering Science and Technology*, vol. 20, no. 1, pp. 116-132, February 2017.

[19] P. Wang, D. Zhu, and X. Lu, "Active queue management algorithm based on data-driven predictive control," *Telecommunication Systems*, vol. 64, no. 1, p. 103–111, January 2017.

[20] P. K. Dash, S. K. Bisoy, N. K. Kamila, and M. Panda, "Parameter setting and stability of PI controller for AQM router," in *Handbook of Research on Wireless Sensor Network Trends, Technologies, and Applications*, 2017, pp. 371-393.

[21] S. K. Patel and S. Bhatnagar, "Adaptive mean queue size and its rate of change: Queue management with random dropping," *Telecommunication Systems*, vol. 65, no. 2, p. 281–295, 2017.

