International Journal of Electrical and Computer Engineering (IJECE)
Vol. 13, No. 1, February 2023, pp. 669~679
ISSN: 2088-8708, DOI: 10.11591/ijece.v13i1.pp669-679

Quadratic exponential random early detection: a new enhanced random early detection-oriented congestion control algorithm for routers

Samuel Oluwatosin Hassan[1], Adewole Usman Rufai[2], Vivian Ogochukwu Nwaocha[3], Samson Ojo Ogunlere[4], Aderonke Adelola Adegbenjo[5], Michael Olugbenga Agbaje[5], Theophilus Aniemeka Enem[6]
[1]Department of Mathematical Sciences, Olabisi Onabanjo University, Ago-Iwoye, Nigeria
[2]Department of Computer Sciences, University of Lagos, Lagos, Nigeria
[3]Department of Computer Science, National Open University, Abuja, Nigeria
[4]Information Technology Department, Babcock University, Ilishan-Remo, Nigeria
[5]Computer Science Department, Babcock University, Ilishan-Remo, Nigeria
[6]Cyber Security Department, Air Force Institute of Technology, Kaduna, Nigeria

Article Info

Article history:

Received Jan 16, 2022
Revised Jul 13, 2022
Accepted Jul 27, 2022

Keywords:

Active queue management algorithm
Congestion control
Delay
Network routers
Quadratic exponential random early detection

ABSTRACT

Network congestion is still a problem on the internet. The random early detection (RED) algorithm being the most notable and widely implemented congestion algorithm in routers faces the problems of queue instability and large delay arising from the presence of an ineffectual singular linear packet dropping function. This research article presents a refinement to RED, named quadratic exponential random early detection (QERED) algorithm, which exploits the advantages of two drop functions, namely quadratic and exponential in order to enhance the performance of RED algorithm. ns-3 simulation studies using various traffic load conditions to assess and benchmark the effectiveness of QERED with two improved variants of RED affirmed that QERED offers a better performance in terms of average queue size and delay metrics at various network scenarios. Fortunately, to replace/upgrade the implementation for RED algorithm with QERED's in routers will require minimal effort due to the fact that nothing more besides the packet dropping probability profile got to be adjusted.

Corresponding Author:

Samuel Oluwatosin Hassan
Department of Mathematical Sciences, Olabisi Onabanjo University
Ago-Iwoye, Nigeria
Email: samuel.hassan@oouagoiwoye.edu.ng

1. INTRODUCTION

In today's world, the internet has been described as an advanced technology which drives speedy development of the society as it supports contemporary communication, information sharing and services [1], [2]. Congestion control still continues as an active research area in the current internet due to huge magnitude of traffic triggered by users accessing internet services [3]–[7]. Network congestion refers to a condition in which the amount of generated traffic grows and exceeds network's resources processing capacity [3], [8]. It is one of the crucial problems that affect quality of network service [9], [10]. Therefore, to make certain of a better network performance, it is imperative to develop algorithms that avoid network traffic congestion [9], [11]–[13].

Aside the well-known responsibility of packet forwarding, the router plays an important role of implementing algorithms needed for congestion control in the network. Traditionally, drop-tail queue management algorithm, as its name implies, operates the first-in, first-out (FIFO) procedure which accepts all incoming packets into the queue of router's buffer until full capacity is reached [14]. The weaknesses of drop-Tail algorithm include full queues, high packet loss rate, and large delay in data delivery, network deadlock and global synchronization [14]–[16].

Zeng *et al.* [17] described the active queue management (AQM) algorithm employed by routers as a more efficient way to relieve congestion. AQM algorithm detects congestion at an early stage and delivers a feedback signal (through dropping of packets) to traffic sources to slow down their transmission rate before the buffer is fully occupied [18]–[20]. AQM algorithms aimed at reducing packet loss rate, keeping average queue size small, increasing throughput and reducing delay [15]. The internet engineering task force (IETF) has recommended the widespread deployment of AQM algorithms in internet devices (in particular, the router) as a solution for controlling congestion [14].

One notable representative AQM is the random early detection (RED) [21], as it is called. RED detects incipient congestion by determining the average queue size (*ave*) using a low pass filter exponential weighted moving average (EWMA) method (according to (1)). The result is compared with two pre-determined queue thresholds for average queue size, namely minimum threshold (*minTh*) and maximum threshold (*maxTh*).

$$ave = \left(1 - w_q\right) \times ave' + (wq \ \times q_c) \tag{1}$$

where q_c refers to the current queue size, ave' refers to the previously calculated ave, and w_q is a pre-configured weight parameter to calculate $ave$and ranges between 0 and 1.

Further, when the computed ave is less than $minTh$ threshold, all incoming packets are enqueued. When ave is between *minTh* and *maxTh* thresholds, incoming packets are dropped with a probability that increases linearly from *0* to P_{max} (that is, the maximum packet dropping probability). However, when ave is greater than *maxTh*, all arriving packets are dropped with a probability equal to 1. Therefore, the initial dropping probability (P_b) strategy of RED (dependent on ave) is given as (2).

$$P_b = \begin{cases} 0 & when \ \ ave \in [0, minTh), \\ P_{max}\left(\frac{ave-minTh}{maxTh-minTh}\right) & when \ \ ave \in \ [minTh, maxTh), \\ 1 & when \ \ ave \ \in [maxTh, +\infty), \end{cases} \tag{2}$$

Finally, the packet dropping probability (P_a) is computed as (3):

$$P_a = P_b \times \frac{1}{1-(count \times P_b)} \tag{3}$$

where *count* refers to the number of admitted packets since the last packet dropped.

To tackle the delay problem of RED, Abu-Shareha [22] proposed a delay-controller RED (DcRED) algorithm. The algorithm extends RED by computing a delay parameter (which relies on the queue size, departure rate, and arrival rate), at every packet arrival into the router's buffer to determine the dropping probability. Gentle RED (GRED) algorithm [23] aimed at increasing the throughput performance of RED. GRED utilizes three queue thresholds, namely *minTh*, *maxTh*, and two times *maxTh*. A linear packet dropping function is applied when *ave* varies between *minT*h and *maxTh*. Similarly, a linear packet dropping function is applied when *ave* is between *maxTh* and two times *maxTh* thresholds. QRTRED algorithm [24] relies on the computation of a metric named QRT which is used to determine the state of the network traffic and dynamically adjust the *minTh* and *maxTh* thresholds. The algorithm obtained improved link utilization.

Nonlinear RED (NLRED) algorithm [7] operates similar to RED except that a quadratic packet drop function is utilized when *ave* varies between *minTh* and *maxTh* queue thresholds and increased P_{max} by a factor of 1.5. This was done with the intention of addressing the aggressive problem of RED at heavy congestion. NLRED achieved an improved throughput performance. Double slope RED (DSRED) algorithm [25] is an enhanced version of RED such that it utilizes three queue thresholds which are *minTh*, *maxTh*, and a mid-point threshold (set as (*minTh* + *maxTh*)/2). When *ave* is between *minTh* and mid-point thresholds, a linear dropping function is used. When *ave* is between the mid-point and *maxTh* thresholds, another linear dropping function is used. The slopes of these two linear functions are complementary and adjustable by a mode selector. The algorithm obtained an improved throughput performance. Feng *et al.* [26] improved RED by employing three dropping functions, which are nonlinear, linear, and nonlinear between *minTh* and *maxTh* thresholds. TRED exhibits a trade-off between two performance metrics: delay and throughput.

MRED algorithm [27], an improved version of GRED applied a quadratic drop function when *ave* varies between *minTh* and *maxTh* threshold values (instead of a linear function used in GRED). MRED achieved an increased throughput performance. Baklizi *et al.* [28] proposed a dynamic GRED (DGRED) which is a modified version of GRED. The algorithm aimed at stabilizing *ave* at a computed specified value between *minTh* and *maxTh* while dynamically adjusting the queue thresholds: *maxTh* and two times *maxTh* in order to provide an improved performance in terms of packet loss rate. Baklizi *et al.* [29] suggested the stabilized DGRED (SDGRED) algorithm, a modified version of DGRED whereby the *maxTh* and two times *maxTh* queue thresholds were dynamically adjusted and aimed at stabilizing *ave* at a computed specified value between *minTh* and two times *maxTh* queue thresholds in order to obtain a reduced packet loss rate and queuing delay. Weight queue dynamic AQM (WQDAQM) algorithm [30] is an improvement over SDGRED algorithm. Based on the traffic load, WQDAQM dynamically adjust the queue weight and the two queue thresholds, namely *maxTh* and two times *maxTh* in order to perform packet dropping. The algorithm aimed at stabilizing the queue weight between *minTh* and *maxTh*. Therefore, the algorithm obtained an improved performance in terms of average queue size, packet loss rate and delay.

VRED algorithm [31] relies on the computation of the queue size growth velocity which is utilized as an indicator for congestion instead of average queue size (as in RED). VRED achieved an improved utilization. RED exponential (RED_E) algorithm [4] is an improved RED in the sense that the linear packet dropping function was replaced with an exponential (nonlinear) packet drop function which does not require P_{max} parameter. RED_E was reported to obtain an improved performance in terms of delay especially at heavy congestion scenario. Adamu [3] developed the self-adaptive RED (SARED) algorithm which relies on the computation of average queue size and current network traffic condition as indicators for congestion. Therefore, at low and moderate network traffic scenarios, a nonlinear packet dropping function is utilized in order to improve the throughput performance. However, at heavy network traffic scenario, a linear packet dropping function is utilized in order to improve the link utilization.

Mohammed *et al.* [10] developed the dynamic queue RED (DQRED) algorithm which operates by classifying incoming packets into the router's buffer into three different virtual queues: one for user data protocol (UDP)-based video applications, another for UDP-based audio applications, and the last third queue for transmission control protocol (TCP) based applications. Packets from these queues are dynamically scheduled with regards to the number of enqueued packets in each. The RED algorithm is implemented in each queue. Attiya and El-Khobby [32] proposed an algorithm which employs a classifier in order to split incoming packets into the router's buffer using three virtual queues, namely high priority UDP-based video traffic, medium priority UDP-based audio traffic, and low priority TCP-based traffic. Packets from these queues are scheduled according to ratio 3:2:1. All the three queues have RED algorithm implementation. The modified algorithm obtained an improved performance in terms of delay and packet loss which are essential for real-time applications. Su *et al.* [33] employed the Q-learning algorithm to determine and adjust P_{max} parameter for deployment in RED. The modified version was named q-learning RED (QRED). The algorithm obtained an improved throughput and packet loss rate.

Flexible RED (FXRED) algorithm [12] operates a self-adaptive mechanism similar to SARED except that a nonlinear packet dropping function is utilized for a low and moderate network traffic conditions in order to obtain an improved throughput and link utilization while a linear packet dropping function for a high network traffic condition in order to obtain an improved delay. The former conditions refer to a situation where *ave* is between *minTh* and a mid-point threshold set as (*minTh* + *maxTh*)/2 while the latter condition refers to a situation where *ave* is between the mid-point and *maxTh* thresholds. Smart RED (SmRED) algorithm [34] improved RED by using three queue thresholds, namely *minTh*, *Target* (set as *minTh* + (*maxTh*-*minTh*)/2) and *maxTh*. SmRED utilizes a nonlinear (quadratic) packet drop function when *ave* is between *minTh* and *Target* in order to obtain an improved link utilization; utilizes a (symmetric) square root packet drop function when *ave* is between Target and *maxTh* in order to obtain a reduced delay.

MultiRED (MRED) [35] utilizes a classifier to split incoming packets into the router queue using two virtual queues: one for TCP-type traffic applications, and the other for UDP-type traffic applications. Each of these queues has RED algorithm implementation. MRED achieved a reduced packet loss rate especially for sensitive traffic flows. Kumhar *et al.* [36] proposed a quadratic RED (QRED) algorithm. To overcome instability problem of RED, QRED replaced the linear drop function of RED with a quadratic drop function (expressed in (4) and (5)):

$$P_b = (\frac{ave-minTh}{maxQ-minTh})^2 \tag{4}$$

or

$$P_b = 1-(\frac{maxQ-ave}{maxQ-minTh})^2 \tag{5}$$

where *maxQ* indicates the buffer size.

To tackle the instability issue of RED algorithm, Gimenez *et al.* [37] developed the Beta RED algorithm which utilizes a (nonlinear) Beta distribution packet drop function in lieu of the linear packet drop function utilized by RED. Suwannapong and Khunboa [38] deploys two dropping functions which are linear (between *minTh* and *maxTh*) and exponential (between *maxTh* and *K:* in which *K* is the size of buffer). While RED directly applies the linear dropping function between *minTh* and *maxTh*, the model suggested in [39] does not. It instead exploits the *count* variable and drop packets based on (6):

$$P_b = 1 - \left(\frac{p_1(-\log(p_1))}{count+1}\right) \tag{6}$$

in which p_1 refers to RED's linear dropping function as (7).

$$p_1 = P_{max}\left(\frac{ave - minTh}{maxTh - minTh}\right) \tag{7}$$

Enhanced congestion control-RED (CoCo-RED) [40] is a refined version of CoCo-RED algorithm in which *minTh* and *maxTh* are adjusted depending on the congestion level. Kar *et al.* [20] described the dropping probability as the central part of an AQM algorithm. Also, Koo *et al.* [41] developing an AQM algorithm with a good drop probability was described as the hallmark of congestion control. Therefore, the main contribution of this paper is to propose an improved, RED-based algorithm named quadratic exponential random early detection (QERED), which utilizes a quadratic and an exponential packet dropping functions to distinguish between light-and moderate-network traffic load scenarios and a heavy network traffic load scenario with the aim of addressing the instability of queue size and large delay shortcomings of RED.

2. QUADRATIC EXPONENTIAL RANDOM EARLY DETECTION

In this section, we present a detailed description of a refined algorithm, referred to as QERED. Like RED algorithm, QERED works by measuring congestion level through the computation of the average queue size (earlier given in (1)). However, QERED as its name implies, replaces the RED's linear relationship between the packet dropping probability and average queue size with an interplay of a quadratic and an exponential drop function. The packet drop probability function for QERED algorithm is depicted in Figure 1. The proposed QERED algorithm distinguishes between two traffic scenarios: one is light and moderate loads, the other is heavy load. The distinction between these two network traffic load situations was accomplished through the introduction of a *Target* threshold parameter given in (8).

$$Target = minTh + \left(\frac{minTh + maxTh}{3}\right) \tag{8}$$

Specifically, the light-and moderate-traffic load scenarios account for situations when *ave* is close to the minimum threshold and when *ave* is near *Target*, respectively. Therefore, a quadratic drop function (expressed in (9)) is applied to drop packets because network congestion is not yet serious. Thus, a smaller packet dropping probability is achieved.

$$P_b = 9P_{max}\left(\frac{ave - minTh}{maxTh + minTh}\right)^2 \tag{9}$$

Also, heavy traffic load scenario refers to conditions when *ave* is between Target and the maximum threshold. Therefore, an exponential drop function (expressed in (10)) is applied to drop packets because network congestion is serious. Thus, there is a need to ease congestion in the network through the use of a higher packet dropping probability.

$$P_b = e^{\log(P_{max}) \times \left(\frac{3(maxTh - ave)}{2(maxTh - 2minTh)}\right)} \tag{10}$$

In other words, the initial dropping probability strategy of QERED is expressed as (11):

$$P_b = \begin{cases} 0 & when \quad ave \in [0, minTh), \\ 9P_{max}\left(\frac{ave - minTh}{maxTh + minTh}\right)^2 & when \quad ave \in [minTh, Target), \\ e^{\log(P_{max}) \times \left(\frac{3(maxTh - ave)}{2(maxTh - 2minTh)}\right)} & when \quad ave \in [Target, maxTh), \\ 1 & when \quad ave \in [maxTh, +\infty), \end{cases} \tag{11}$$

Algorithm 1 presents the pseudo-code for QERED algorithm.

Algorithm 1. Pseudo-code for QERED algorithm
```
1:  Preset input parameters: $minTh$, $maxTh$, $P_{max}$, $w_q$
2:  Set $ave = 0$
3:  Set $Target = minTh + (\frac{minTh+maxTh}{3})$
4:  For every incoming packet
5:  Calculate the average queue size ave according to (1)
6:  if $ave < minTh$ then
7:  The packet will be admitted into queue of router's buffer
8:  else if $minTh \le ave < Target$ then
9:  A quadratic function (given in (9)) is employed to calculate the packet dropping
    probability
10: The incoming packet is dropped based on the calculated probability
11: else if $Target \le ave < maxTh$ then
12: An exponential function (given in (10)) is employed to calculate the packet dropping
    probability
13: The incoming packet is dropped based on the calculated probability
14: else if $maxTh \le ave$ then
15: The incoming packet is rejected
16: end if
```

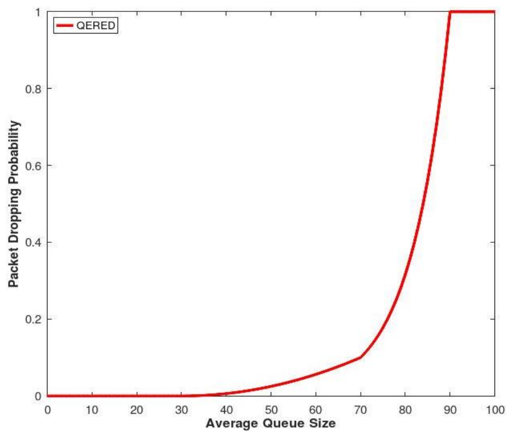


Figure 1. QERED drop function curve

3. RESULTS AND DISCUSSION

3.1. Simulation configuration

We implement the proposed QERED algorithm in network simulator version 3 (ns-3) [42]. The network topology considered in our simulation experiment is depicted in Figure 2. The topology has two intermediate devices-routers (which are R_1 and R_2), primarily for the purpose of directing data packet flows. Traffics are generated from N TCP/New-Reno sources. The bottleneck link is between router R_1 and router R_2, it has a bandwidth of 10 Mbps and a propagation delay of 100 ms. Every other link (either connecting the sources to router R_1 or connecting router R_2 to destination device, D) has a capacity of 100 Mbps and 3 ms propagation delay. The size of packets by default is 1000 bytes. AQM algorithms are implemented and deployed in the queue of router R_1 with a specified buffer size of 250 packets. For SmRED, we set $minTh$=30 packets, $Target$=60 packets, $maxTh$=90 packets, P_{max}=0.1, and w_q=0.002. For RED_E, we set $minTh$=30, $maxTh$=90 packets, and w_q=0.002. For QERED, we set $minTh$=30 packets, $Target$=70 packets, $maxTh$=90 packets, P_{max}=0.1, and w_q=0.002. Simulation time is specified as 100 sec.

Based on Figure 2, three experiments involving light, moderate and heavy traffic loads were conducted in order to evaluate the performance of RED_E, SmRED, and QERED algorithms. Comparison was done using average queue size, throughput, and delay as performance measures. The light, moderate, and heavy traffic loads involve N=5, N=20, and N=50 TCP flows respectively.

3.2. Scenario 1: light traffic

From Figure 3(a), we see that the proposed QERED algorithm regulates the oscillatory behavior of average queue size better than RED_E and SmRED algorithms. Statistical results presented in Table 1 shows that QERED obtained the lowest value in terms of average queue size. Therefore, in terms maintaining the average queue size at a stable and small value, QERED clearly outperformed RED_E and SmRED.

It can be observed from Figure 3(b) that the oscillation in delay is better controlled with QERED. Analysis presented in Table 1 further confirms that QERED achieved the least value. Thus, QERED has greater benefit in terms of delay. Figure 3(c) presents the throughput performance. From the figure, it can be seen that SmRED obtained the best performance in terms of throughput among all the three algorithms. However, QERED obtained a better result than RED_E. Table 1 confirms this result.

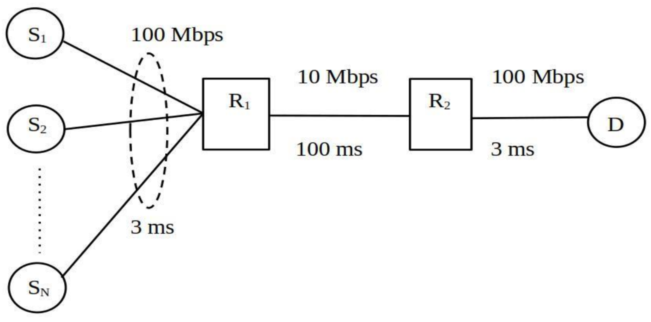


Figure 2. Simulation topology

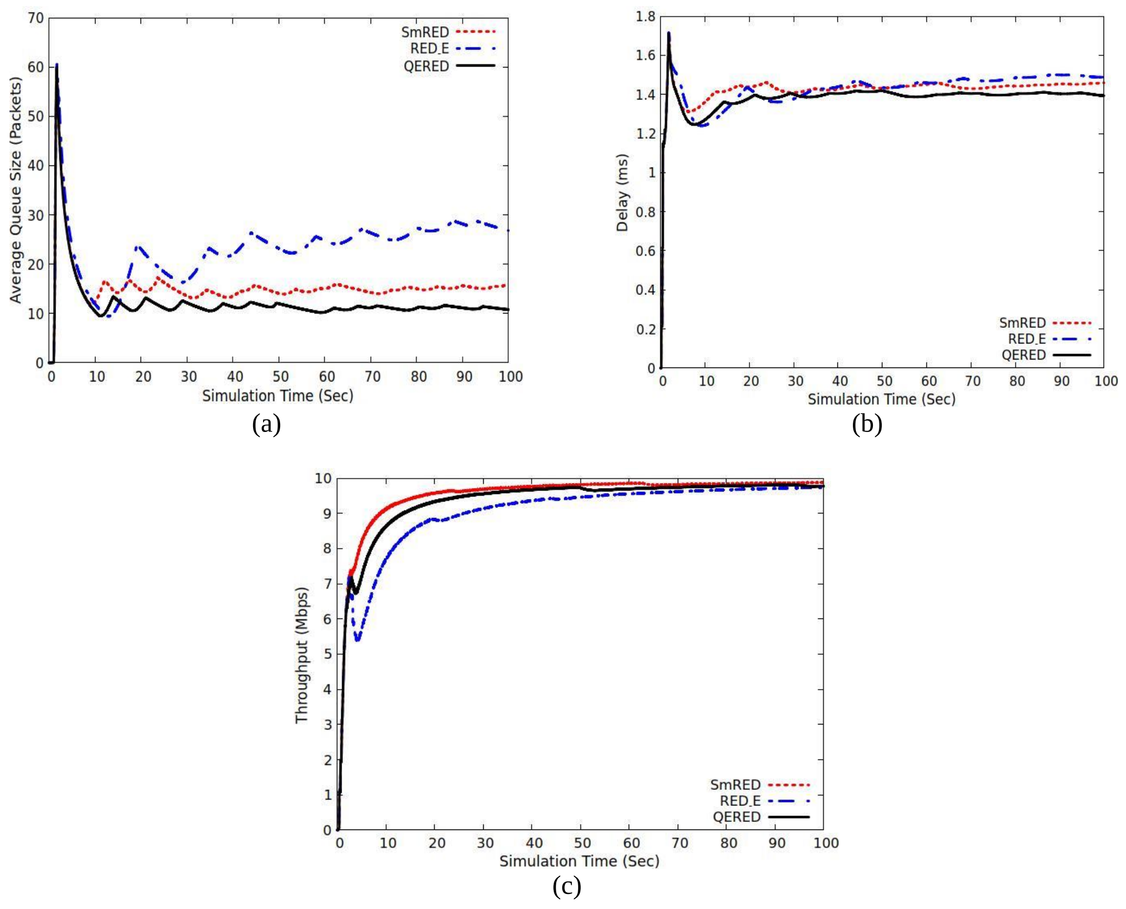


Figure 3. Light traffic load: (a) average queue size, (b) delay, and (c) throughput

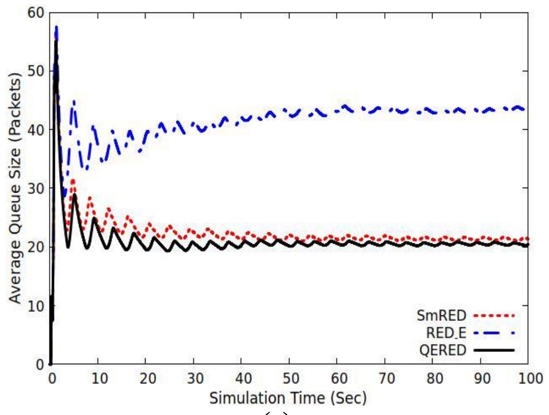
Table 1. Performance comparison under light traffic load

AQM algorithm	Average queue size (Packets)	Delay (ms)	Std-Deviation (ms)	Throughput (Mbps)
SmRED	15.4349	1.4234	0.1002	9.4980
RED_E	23.0978	1.4254	0.1158	9.0002
QERED	12.1596	1.3807	0.0995	9.3276

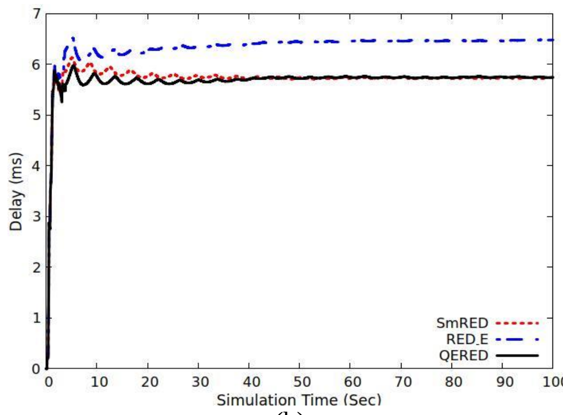
3.3. Scenario 2: moderate traffic

Figure 4(a) presents the average queue size performance of SmRED, RED_E and QERED algorithms. It is clearly seen that QERED better regulates and keeps the average queue size smaller than SmRED and RED_E algorithms. Analysis presented in Table 2 also confirms that the proposed QERED algorithm obtained the best performance in regard to average queue size metric.

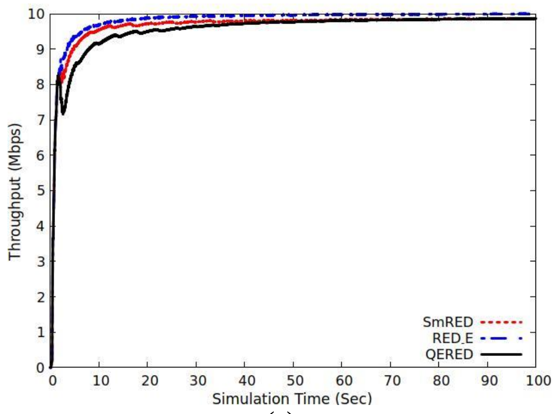
From the delay graph plotted in Figure 4(b), it can be observed that the proposed QERED algorithm obtained a better performance than SmRED and RED_E algorithms. In fact, it can be confirmed from the statistical results in Table 2 that QERED has a superior delay performance. The throughput results presented in Figure 4(c) shows that RED_E algorithm has the best performance. Table 2 shows the analysis. This result is a pointer to the fact that under this scenario, QERED obtained a reduced delay at the expense of throughput.



(a)



(b)



(c)

Figure 4. Moderate traffic load: (a) average queue size, (b) delay, and (c) throughput

Table 2. Performance comparison under moderate traffic load

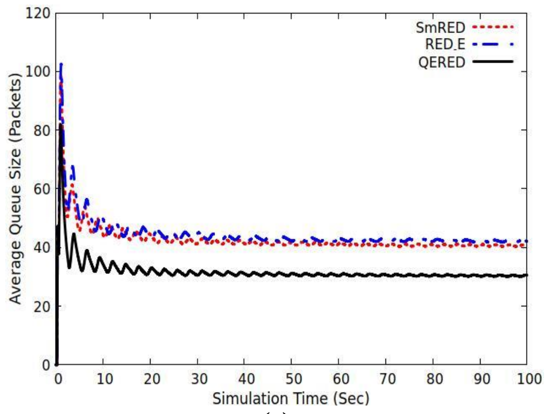
AQM algorithm	Average queue size (Packets)	Delay (ms)	Std-Deviation (ms)	Throughput (Mbps)
SmRED	22.4637	5.7066	0.4529	9.6450
RED_E	41.1539	6.3240	0.5280	9.7973
QERED	21.0029	5.6657	0.4470	9.5314

3.4. Scenario 3: heavy traffic

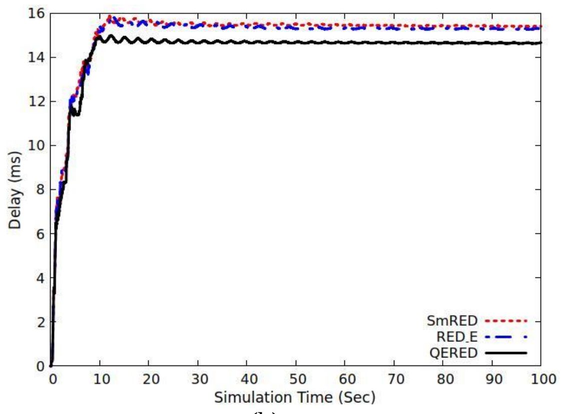
The graph shown in Figure 5(a) compares the average queue size for RED_E, SmRED, and QERED algorithms. As illustrated, at heavy traffic load when congestion is aggravated, QERED algorithm significantly reduces and stabilizes the average queue size. As presented in Table 3, QERED obtained the lowest value when compared with the other two algorithms. The reason is because, the exponential drop function utilized by QERED rises faster. As a result, more packets are dropped with QERED when heavy congestion occurs.

The plot depicted in Figure 5(b) compares the performance of the three algorithms in terms of delay. As demonstrated, QERED algorithm outperformed both RED_E and SmRED. Analysis provided in Table 3 confirms that QERED reduces the delay better than RED_E and SmRED. Also, the variance in delay is lowest with QERED.

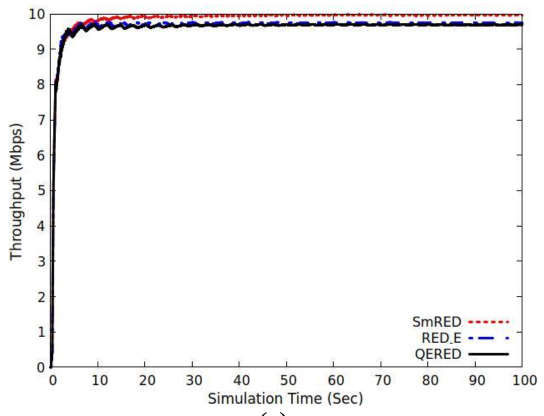
Figure 5(c) compares the throughput performance of RED_E, SmRED and QERED under heavy traffic load condition. It is evident that SmRED obtained the best performance. Analysis provided in Table 3 confirms that QERED obtained the least value among the three algorithms. Again, this throughput result is a pointer to the fact that under this scenario, QERED obtained a reduced delay at the expense of throughput.



(a)



(b)



(c)

Figure 5. Heavy traffic load (a) average queue size, (b) delay, and (c) throughput

Table 3. Performance comparison under heavy traffic load

AQM Algorithm	Average Queue Size (Packets)	Delay (ms)	Std-Deviation (ms)	Throughput (Mbps)
SmRED	43.6304	15.0049	1.8609	9.8273
RED_E	44.3700	14.8843	1.8481	9.6355
QERED	31.7554	14.2488	1.7728	9.5809

4. CONCLUSION

In this paper, the RED algorithm is re-investigated, and an improved, RED-oriented algorithm named QERED has been proposed. QERED utilizes the mixture of quadratic and exponential packet dropping strategy in lieu of an ineffectual linear packet dropping strategy utilized in RED. Demonstrations in ns-3 simulator revealed that QERED algorithm evidently outperformed two representative AQM algorithms which are RED_E and SmRED in terms of average queue size stability and delay performance measures under three traffic load conditions (which are light, moderate, and heavy). In future work, we plan to validate QERED’s algorithm implementation in Linux kernel and eventually assess its performance alongside some selected AQM algorithms on a real network.

REFERENCES

[1] J. Shi, Y. B. Leau, K. Li, and J. H. Obit, "A comprehensive review on hybrid network traffic prediction model," *International Journal of Electrical and Computer Engineering (IJECE),* vol. 11, no. 2, pp. 1450–1459, Apr. 2021, doi: 10.11591/ijece.v11i2.pp1450-1459.

[2] R. Alsabah, M. Aljshamee, A. M. Abduljabbar, and A. Al-Sabbagh, "An insight into internet sector in Iraq," *International Journal of Electrical and Computer Engineering (IJECE),* vol. 11, no. 6, pp. 5137–5143, Dec. 2021, doi: 10.11591/ijece.v11i6.pp5137-5143.

[3] A. Adamu, "SARED: A self-adaptive active queue management scheme for improving quality of service in network systems," *Computer Science*, vol. 22, no. 2, pp. 253–267, Apr. 2021, doi: 10.7494/csci.2021.22.2.4020.

[4] H. Abdel-Jaber, "An exponential active queue management method based on random early detection," *Journal of Computer Networks and Communications*, vol. 2020, pp. 1–11, May 2020, doi: 10.1155/2020/8090468.

[5] X. Yang, C. Song, J. Sun, and X. Wang, "Simulation and implementation of adaptive fuzzy PID," *Journal of Networks*, vol. 9, no. 10, pp. 2574–2581, Oct. 2014, doi: 10.4304/jnw.9.10.2574-2581.

[6] N. Hamadneh, M. Obiedat, A. Qawasmeh, and M. Bsoul, "HRED, an active queue management algorithm for TCP congestion control," *Recent Patents on Computer Science*, vol. 12, no. 3, pp. 212–217, 2019, doi: 10.2174/2213275912666181205155828.

[7] K. Zhou, K. L. Yeung, and V. O. K. Li, "Nonlinear RED: A simple yet efficient active queue management scheme," *Computer Networks*, vol. 50, no. 18, pp. 3784–3794, Dec. 2006, doi: 10.1016/j.comnet.2006.04.007.

[8] L. Pei, F. Wu, and S. Wang, "Periodic, quasi-periodic and chaotic oscillations in two heterogeneous AIMD/RED network congestion models with state-dependent round-trip delays," *International Journal of Bifurcation and Chaos*, vol. 31, no. 06, pp. 2150121–2150124, May 2021, doi: 10.1142/S0218127421501248.

[9] T. A. Assegie and H. D. Bizuneh, "Improving network performance with an integrated priority queue and weighted fair queue scheduling," *Indonesian Journal of Electrical Engineering and Computer Science (IJEECS)*, vol. 19, no. 1, pp. 241–247, Jul. 2020, doi: 10.11591/ijeecs.v19.i1.pp241-247.

[10] H. Mohammed, G. Attiya, and S. El-Dolil, "Active queue management for congestion control: Performance evaluation, new approach, and comparative study," *International Journal of Computing and Network Technology*, vol. 05, no. 02, pp. 37–49, May 2017, doi: 10.12785/ijcnt/050201.

[11] M. M. Hamdi, H. F. Mahdi, M. S. Abood, R. Q. Mohammed, A. D. Abbas, and A. H. Mohammed, "A review on queue management algorithms in large networks," *IOP Conference Series: Materials Science and Engineering*, vol. 1076, no. 1, Feb. 2021, doi: 10.1088/1757-899X/1076/1/012034.

[12] A. Adamu, V. Shorgin, S. Melnikov, and Y. Gaidamaka, "Flexible random early detection algorithm for queue management in routers," in *Distributed Computer and Communication Networks: 23rd International Conference, DCCN 2020*, 2020, pp. 196–208, doi: 10.1007/978-3-030-66471-8_16.

[13] S. Sunassee, A. Mungur, S. Armoogum, and S. Pudaruth, "A comprehensive review on congestion control techniques in networking," in *2021 5th International Conference on Computing Methodologies and Communication (ICCMC)*, Apr. 2021, pp. 305–312, doi: 10.1109/ICCMC51019.2021.9418329.

[14] F. Baker and G. Fairhurst, "IETF recommendations regarding active queue management," RFC 2309, Jul. 2015, doi: 10.17487/RFC7567.

[15] B. Braden *et al.*, "Recommendations on queue management and congestion avoidance in the internet," Internet Performance Recommendations, RFC 2309, Apr. 1998, doi: 10.17487/rfc2309.

[16] C. Brandauer, G. Iannaccone, C. Diot, T. Ziegler, S. Fdida, and M. May, "Comparison of tail drop and active queue management performance for bulk-data and Web-like Internet traffic," in *Proceedings. Sixth IEEE Symposium on Computers and Communications*, 2001, pp. 122–129, doi: 10.1109/ISCC.2001.935364.

[17] L. Zeng, H. Ni, and R. Han, "The yellow active queue management algorithm in ICN routers based on the monitoring of bandwidth competition," *Electronics*, vol. 10, no. 7, Mar. 2021, doi: 10.3390/electronics10070806.

[18] M. Q. Sulttan, M. H. Jaber, and S. W. Shneen, "Proportional-integral genetic algorithm controller for stability of TCP network," *International Journal of Electrical and Computer Engineering (IJECE)*, vol. 10, no. 6, pp. 6225–6232, Dec. 2020, doi: 10.11591/ijece.v10i6.pp6225-6232.

[19] M. Barczyk and A. Chydzinski, "AQM based on the queue length: A real-network study," *PLOS ONE*, vol. 17, no. 2, Feb. 2022, doi: 10.1371/journal.pone.0263407.

[20] S. Kar, B. Alt, H. Koeppl, and A. Rizk, "PAQMAN: A principled approach to active queue management," *arXiv:2202.10352v1*, Feb. 2022.

[21] S. Floyd and V. Jacobson, "Random early detection gateways for congestion avoidance," *IEEE/ACM Transactions on Networking*, vol. 1, no. 4, pp. 397–413, 1993, doi: 10.1109/90.251892.

[22] A. A. Abu-Shareha, "Controlling delay at the router buffer using modified random early detection," *International Journal of Computer Networks & Communications*, vol. 11, no. 6, pp. 63–75, Nov. 2019, doi: 10.5121/ijcnc.2019.11604.

[23] S. Floyd, "Recommendation on using the 'gentle' variant of RED," *www.icir.org/floyd/red/gentle.html*, 2000, [Online]. Available: https://cir.nii.ac.jp/crid/1571135650186783744 (accessed June 15, 2022).

[24] S. Jamali, N. Alipasandi, and B. Alipasandi, "An improvement over random early detection algorithm: A self-tuning approach," *An Improvement over Random Early Detection Algorithm: A Self-tuning Approach,* vol. 2, no. 2, pp. 57–61, 2014.

[25] Bing Zheng and M. Atiquzzaman, "DSRED: an active queue management scheme for next generation networks," in *Proceedings 25th Annual IEEE Conference on Local Computer Networks. LCN 2000*, 2000, pp. 242–251, doi: 10.1109/LCN.2000.891036.

[26] C.-W. Feng, L.-F. Huang, C. Xu, and Y.-C. Chang, "Congestion control scheme performance analysis based on nonlinear RED," *IEEE Systems Journal*, vol. 11, no. 4, pp. 2247–2254, Dec. 2017, doi: 10.1109/JSYST.2014.2375314.

[27] Y. Zhang, J. Ma, Y. Wang, and C. Xu, "MRED: An improved nonlinear RED algorithm," in *2011 International Conference on Computer and Automation Engineering (ICCAE 2011)*, 2012, vol. 44, doi: 10.7763/IPCSIT.2012.V44.2.

[28] M. Baklizi, H. Abdel-Jaber, M. Abualhaj, N. Abdullah, S. Ramadass, and D. A. Almomani, "Dynamic stochastic early discovery: A new congestion control technique to improve networks performance," *Dynamic stochastic early discovery: A new congestion control technique to improve networks performance*, vol. 9, no. 3, pp. 1113–1126, 2013.

[29] M. Baklizi, "Stabilizing average queue length in active queue management method," *International Journal of Advanced Computer Science and Applications*, vol. 10, no. 3, 2019, doi: 10.14569/IJACSA.2019.0100310.

[30] M. Baklizi, "Weight queue dynamic active queue management algorithm," *Symmetry*, vol. 12, no. 12, Dec. 2020, doi: 10.3390/sym12122077.

[31] S. Jamali, B. Alipasandi, and N. Alipasandi, "VRED: An improvement over RED algorithm by using queue length growth velocity," *Journal of Advances in Computer Research*, vol. 4, no. 1, pp. 31–38, 2013.

[32] G. Attiya and H. El-Khobby, "Improving internet quality of service through active queue management in routers," *International Journal of Computer Science Issues*, vol. 9, no. 1, pp. 279–286, 2012.

[33] Y. Su, L. Huang, and C. Feng, "QRED: A Q-learning-based active queue management scheme," *Journal of Internet Technology*, vol. 19, no. 4, pp. 1169–1178, 2018, doi: 10.3966/160792642018081904019.

[34] A. K. Paul, H. Kawakami, A. Tachibana, and T. Hasegawa, "An AQM based congestion control for eNB RLC in 4G/LTE network," in *2016 IEEE Canadian Conference on Electrical and Computer Engineering (CCECE)*, May 2016, pp. 1–5, doi: 10.1109/CCECE.2016.7726792.

[35] E.-B. Fgee, A. Smeda, and K. AbouElgaseem, "MRED: An algorithm to insure high QoS in IP networks," *Journal of Communications*, vol. 12, no. 4, pp. 200–206, 2017, doi: 10.12720/jcm.12.4.200-206.

[36] D. Kumhar, A. Kumar, and A. Kewat, "QRED: an enhancement approach for congestion control in network communications," *International Journal of Information Technology*, vol. 13, no. 1, pp. 221–227, Feb. 2021, doi: 10.1007/s41870-020-00538-1.

[37] A. Giménez, M. A. Murcia, J. M. Amigó, O. Martínez-Bonastre, and J. Valero, "New RED-type TCP-AQM algorithms based on beta distribution drop functions," *arXiv:2201.01105v1*, Jan. 2022.

[38] Suwannapong and Khunboa, "Congestion control in CoAP observe group communication," *Sensors*, vol. 19, no. 15, Aug. 2019, doi: 10.3390/s19153433.

[39] S. Patel and Karmeshu, "A new modified dropping function for congested AQM networks," *Wireless Personal Communications*, vol. 104, no. 1, pp. 37–55, Jan. 2019, doi: 10.1007/s11277-018-6007-8.

[40] C. Suwannapong and C. Khunboa, "EnCoCo-RED: Enhanced congestion control mechanism for CoAP observe group communication," *Ad Hoc Networks*, vol. 112, pp. 1–10, Mar. 2021, doi: 10.1016/j.adhoc.2020.102377.

[41] Jahon Koo, Byunghun Song, Kwangsue Chung, Hyukjoon Lee, and Hyunkook Kahng, "MRED: a new approach to random early detection," in *Proceedings 15th International Conference on Information Networking*, 2001, pp. 347–352, doi: 10.1109/ICOIN.2001.905450.

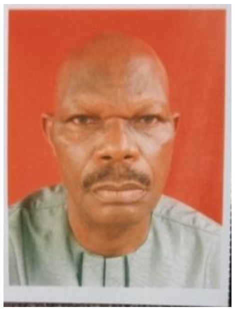
[42] ns-3, "The network simulator ns-3" *ns-3*. [Online]. Available: https://www.nsnam.org (accessed February 26, 2022).

BIOGRAPHIES OF AUTHORS

Samuel Oluwatosin Hassan received his M.Sc. and Ph.D. degrees in Computer Science from Obafemi Awolowo University, Ile-Ife, Nigeria. Currently, he is a Lecturer in the Department of Mathematical Sciences (Computer Science Unit), Olabisi Onabanjo University, Ago-Iwoye, Nigeria. He is a Certified Information Technology Practitioner (*C.itp*). He is a Member of Computer Professionals (Registration Council) of Nigeria (CPN), International Association of Engineers (IAENG). He is also a member of the following societies: Nigeria Computer Society (NCS), the IAENG Society of Computer Science, the IAENG Society of Internet Computing and Web Services, the IAENG Society of Scientific Computing, the IAENG Society of Software Engineering, and the IAENG Society of Wireless Networks. He is also a Member of MathTech Thinking Foundation, (MTTF) and a certified MTTF-Advisor. His research interests span computational mathematics, computer networks and communications, mathematical modeling and simulation, and Internet congestion control. He can be contacted at samuel.hassan@oouagoiwoye.edu.ng.

Adewole Usman Rufai is a Senior Lecturer in the Department of Computer Sciences, University of Lagos, Nigeria. His tertiary education includes the following: Bachelor of Science degree in Mathematics obtained from the University of Ilorin, PGD, M.Sc., Ph.D. (Computer Science), and MBA (Finance) all from the University of Lagos. He is a Fellow of the Nigeria Computer Society, a Certified Information Technology Practitioner, and a Member of the Computer Professionals Registration Council of Nigeria and a Member of Association of Computer Machinery. His areas of research include Software Engineering, Cognitive Computing, and Cyber Security. He can be contacted at arufai@unilag.edu.ng.

Vivian Ogochukwu Nwaocha is an Associate Professor of Computer Science and the Deputy Director, Africa Centre of Excellence on Technology Enhanced Learning at the National Open University of Nigeria. She holds a PhD in Computer Science, MSc. in Computer Science, PGD in Computer Engineering and a B.Eng. in Metallurgical and Materials Engineering. She has over thirteen years of experience in inclusive open and distance learning and has served as the Head of Department of Computer Science at the National Open University of Nigeria. She is a member of Computer Professionals Registration Council of Nigeria, Nigeria Computer Society, Association for Computing Machinery, Polearm Academy, and several international professional and scientific associations. She can be contacted at onwaocha@noun.edu.ng.



Samson Ojo Ogunlere is an Associate Professor at Babcock University, Information Technology (IT) Department in IT and Computer related courses. He is a registered member of Nigeria Society of Engineers (MNSE), Council for Regulation of Engineering in Nigeria (COREN), Chartered Information Technology Practitioner (C.itp), Member Computer Professionals Registration Council of Nigeria (MCPN) and Nigeria Computer Society (NCS) with many years of working experiences in IT and Computer Industries. He is also a member of Computer Science and IT Department Research group in Babcock University, Ogun State, Nigeria. He can be contacted at ogunleres@babcock.edu.ng.

Aderonke Adelola Adegbenjo is a Senior Lecturer in the Department of Computer Science, Babcock University, Nigeria. A graduate of Babcock University, Ilishan Remo, Ogun State, Nigeria. Master's degree in Computer Science Department at the Univeristy of Ibadan and MPhil degree at Babcock University and Ph.D. in the same institution, Babcock University. She is a member of Computer Professionals (Registration Council of Nigeria). Her research interest is in Networking. She can be contacted at adegbenjoa@babcock.edu.ng.

Michael Olugbenga Agbaje is an Associate Professor of Computer Science and Information security in the Department of Computer Science, Babcock University, Nigeria. Holds a Doctorate Degree from the Department of Computer Science, Babcock University, Nigeria. He can be contacted at agbajem@babcock.edu.ng.

Theophilus Aniemeka Enem holds a PhD in Computer Science from Babcock University, Ilishan-Remo, Ogun State, Nigeria. He has several years' experience of teaching computer science and cyber security courses at the university level. Currently, the HOD of Cyber Security Department, Air Force Institute of Technology, Kaduna, Nigeria. Enem is a full member of the Nigeria Computer Society (NCS) and the Computer Professional Registration Council of Nigeria (CPN). His areas of interest are Networking, Telecommunications, and Forensic Science. He has published works in several journals of international repute. He can be contacted at ta.enem@afit.edu.ng.