Hindawi
Journal of Computer Networks and Communications
Volume 2020, Article ID 8090468, 11 pages
https://doi.org/10.1155/2020/8090468

Research Article

An Exponential Active Queue Management Method Based on Random Early Detection

Hussein Abdel-Jaber

Faculty of Computer Studies, Department of Information Technology and Computing, Arab Open University, Saudi Arabia

Correspondence should be addressed to Hussein Abdel-Jaber; habdeljaber@arabou.edu.sa

Received 15 November 2019; Revised 15 February 2020; Accepted 5 May 2020; Published 22 May 2020

Academic Editor: Roberto Nardone

Congestion is a key topic in computer networks that has been studied extensively by scholars due to its direct impact on a network's performance. One of the extensively investigated congestion control techniques is random early detection (RED). To sustain RED's performance to obtain the desired results, scholars usually tune the input parameters, especially the maximum packet dropping probability, into specific value(s). Unfortunately, setting up this parameter into these values leads to good, yet biased, performance results. In this paper, the RED-Exponential Technique (RED_E) is proposed to deal with this issue by dropping arriving packets in an exponential manner without utilizing the maximum packet dropping probability. Simulation tests aiming to contrast E_RED with other Active Queue Management (AQM) methods were conducted using different evaluation performance metrics including mean queue length (mql), throughput (T), average queuing delay (D), overflow packet loss probability (P_L), and packet dropping probability (D_P). The reported results showed that E_RED offered a marginally higher satisfactory performance with reference to mql and D than that found in common AQM methods in cases of heavy congestion. Moreover, RED_E compares well with the considered AQM methods with reference to the above evaluation performance measures using minimum threshold position (min threshold) at a router buffer.

1. Introduction

With the fast development in computer hardware and data communications, Quality of Service (QoS) in computer networks has become an important concern for end users [1, 2]. QoS can be defined as the network performance for the different services sought by the user while data is traversing via this network [1]. QoS has different levels provided to the users based on the requirements of applications used, providers of network services, or Service Level Agreement (SLA) among users [2]. Best effort is one of the services used in the Internet to deliver packets and not differentiate between packets generated by different classes of service [3]. QoS can be evaluated by different metrics such as bandwidth, packet loss, jitter, and delay.

In modern communication and computer networks, different network applications have been developed such as voice over IP (VoIP), video conferencing, live video, e-mail, and file transfer. These applications require different QoS. For instance, VoIP, video conferencing, and live video necessitate high bandwidth and low delay and jitter. Moreover, network applications have low sensitivity to packet loss. On the other hand, the applications of e-mail and file transfer have high sensitivity to the requirements of packet loss and low sensitivity to bandwidth, delay, and jitter. Enhancing the performance of a network can be performed based on obtaining the QoS for that network. Many researchers have proposed congestion control techniques [4–9, 10--12] that are implemented via simulation to enhance network performance with different QoS requirements [1, 13, 14]. Other congestion control techniques have been developed as analytical models to deal with QoS issues [1, 2]. For example, in [15–18], the authors proposed discrete-time queue analytical models aimed at early control of congestion. Researchers [1, 19, 20] also developed analytical models to solve problems related to the delay in the computer network.

RED is one of the key AQM techniques that were proposed to enhance the classic drop-tail method [21] performance. Nevertheless, and despite superiority of RED over the drop-tail method, the advances in traffic service diversity have revealed the following performance issues associated with RED's performance [2, 22].

(1) Usually the key congestion measure (average queue length (aql)) used by RED varies according to the level of congestion. For instance, when aql is near to the min threshold, light congestion occurs, whereas if the aql is near the max threshold, heavy congestion occurs. These may cause the router buffer to overflow and, therefore, the dropping of arriving packets.

(2) Another issue is tuning certain parameters of RED into specific values, that is, the maximum value of packet dropping ($D_{\max}$), to ensure a satisfactory performance. This produces biased results.

(3) Often aql depends on the number of TCP connections. When the numbers of TCP connections are increased, the aql increases as well and may exceed the max threshold position; hence, every arriving packet is dropped.

This paper deals with the second problem described above which is tuning $D_{\max}$ to a certain value to guarantee high yet biased QoS. We would like to minimize this problem by proposing a new method that relaxes the dependency on the $D_{\max}$ and instead employs a new exponential measure called D_{init} (see equation (5)) that is based on average queue length (aql). In particular, when aql is between the minimum and the maximum thresholds, the proposed congestion control method drops arriving packets exponentially. This will result in more realistic performance results rather than those that are biased. We call the proposed algorithm RED-Exponential (RED_E).

The proposed method is implemented using a simulation and a discrete-time queues approach [23]. The discrete-time queues approach is used to model the arrival and departure of packets in every single time unit called a slot. Further details on using the discrete-time queues approach with the proposed method is given in Subsection 4.1. The benefits of the proposed RED_E algorithm are as follows:

(i) Calculating D_{init} not involving using parameter $D_{\max}$, relaxing users' subjectivity when setting up RED's parameters;

(ii) Enhanced performance measures of mql and D when compared to RED and one of the RED-based AQM methods such as NLRED especially in cases where the value of packet arrival probability is very high.

This paper is organized as follows. Related work, including RED and NLRED methods, is introduced in Section 2. Section 3 presents the proposed RED_E. Simulation details are introduced in Section 4 as well as performance measures results. Finally, conclusions and future work are provided in Section 5.

2. Related Work

2.1. RED and Nonlinear RED. RED is one of the known AQM techniques that was proposed to detect and control congestion [7]. RED relies on certain parameters to calculate its D_p value. These parameters are min threshold, max threshold, $D_{\max}$, and qw. For every arriving packet at the router buffer, RED computes the aql value. When the aql value is less than the min threshold, no congestion occurs and thus no packets can be dropped. However, if the aql value is less than max threshold and equal to or larger than the min threshold, this gives an indication there is congestion, and the router buffer drops arriving packets probabilistically. Lastly, when the value of aql is equal to or greater than the max threshold, heavy congestion is presented, and every arriving packet will be dropped in order to manage this congestion. This is achieved either by dropping the arriving packets at the router buffer or by marking them using Explicit Congestion Notification (ECN) [18]. The pseudocode of the RED technique is shown in Algorithm 1.

Algorithm 1 shows that RED utilizes several factors to compute aql and D_p. The factors are explained below:

current_time: current time.

idle_time: starting idle time at the RED router buffer.

n: number of packets sent to the RED router buffer during an idle interval time.

C: number of packets arrived at the RED router buffer not dropped since the last packet was dropped.

D_p: instantaneous packet dropping probability.

D_{init}: initial packet dropping probability.

q_instantaneous: instantaneous queue length.

qw: queue weight.

$D_{\max}$: maximum value of D_p.

q(time): linear function for the time.

One of the main issues of RED is that setting the parameters cannot guarantee stable performance under varying traffic loads [24]. This can be attributed to the linear packet dropping probability function that is often aggressive when the traffic load is light and not aggressive when the traffic is low, which may cause the value of average queue length to reach the value of the maximum threshold. To overcome this problem, [24] proposed Nonlinear RED (NLRED) that utilizes a nonlinear packet dropping function similar to those proposed in [4, 25]. The pseudocode of the NLRED method is given in Algorithm 2.

Algorithm 2 shows the NLRED router buffer is working as RED's router buffer when the value of aql is less than the min threshold or greater than or equal to the max threshold. In cases when the aql is between the min threshold and max threshold, then NLRED uses the quadratic function to compute the packet dropping probability, and this function is given in

$$D_{\text{init}} = D'_{\max} \times \left(\frac{(\text{aql} - \text{min threshold})}{(\text{max threshold} - \text{min threshold})}\right)^2, \quad (1)$$

(1) Initialisation stage
$C = -1$
aql = 0.0
(2) For every arriving packet at a RED router buffer
Calculate the aql for this packet at the RED router buffer
Examine the queue status at a router buffer, is it empty or not
if the queue at a RED router buffer is empty then
Compute n, where $n = q$(current_time − idle_time)
aql = aql × $(1 - qw)^n$
Else
aql = aql × $(1 - qw) + qw \times q$_instantaneous
3. Determine a congestion status at the RED router buffer
if aql value is less than min threshold value then
Calculate D_p value for the arriving packet as follows:
$D_p = 0.0$
$C = -1$
else if aql value is greater than or equal to min *threshold* value and less than max threshold value then
$C = C + 1$
$D_{\text{init}} = (D_{\max} \times (\text{aql} - \text{min threshold}))/(\text{max threshold} - \text{min threshold})$
$D_p = D_{\text{init}}/(1 - C \times D_{\text{init}})$
Mark/Drop the arriving packet probabilistically regarding to D_p value due to occurrence of congestion
$C = 0$
else
Mark/Drop every arriving packet $D_p = 1.0$ due to occurrence of heavy congestion
$C = 0$
4. When the RED router buffer becomes empty
idle_time = current_time

Algorithm 1: RED technique detailed pseudocode.

For each arriving packet
Compute the average queue length (aql)
if aql ≤ M in threshold
No packet will be dropped
else if M in threshold ≤ aql ≤ Max threshold
Compute the packet dropping probability using quadratic function
Drop the arriving packet with the computed probability
else
Drop the arriving packet

Algorithm 2: The pseudocode of the NLRED method [24].

where $D_{\max}{}'$ is the maximum value of packet dropping probability. If the same value is used for $D_{\max}$ in RED and $D_{\max}{}'$ in NLRED, then NLRED will be gentler than RED for all traffic loads since the packet dropping probability value of NLRED is less than that of RED. If the total of packet dropping probabilities of RED and NLRED are similar, then the value of $D_{\max}{}'$ will be set as follows:

$$D_{\max}{}' = 1.5 \times D_{\max}. \quad (2)$$

The packet dropping probability functions of RED and NLRED methods are shown in Figure 1.

2.2. Other AQM Techniques. To solve the first shortcoming of RED, aql varies according to the level of congestion, so the Adaptive RED (ARED) technique was proposed in [8]. ARED aimed to stabilize the aql value at a specific level between the min threshold and max threshold positions. This may prevent the dropping of large numbers of packets.

Gentle RED (GRED) technique [9] was proposed to reduce packets being dropped. This was accomplished by preventing the dropping of every arriving packet when the aql value equals the max threshold position as in RED [7]; instead, GRED drops arriving packets probabilistically based on values in the range of $D_{\max}$ to 1.0.

Dynamic Random Early Drop (DRED) technique [5] was proposed to deal with aql dependency on the number of TCP connections; BLUE technique [6] was developed to offer better network requirements of packet loss rates and queue size than those of RED [7].

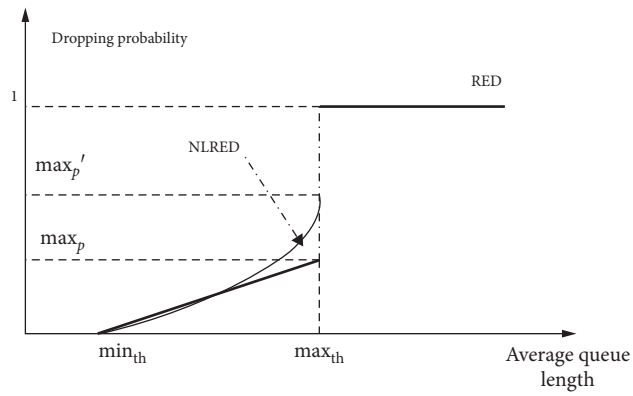

Figure 1: Packet dropping probability functions of RED and NLRED methods [26].

Effective RED (ERED) technique was proposed to reduce packet loss rates in a simple and scalable manner. The authors made a few changes to the packet dropping function of RED and the rest of RED's parameters remain unchanged. They refined and controlled the packet dropping function using average queue size and instantaneous queue size parameters. Simulation results demonstrated that ERED achieved higher throughput and lower packet dropping than RED.

The authors of [27] proposed an AQM algorithm based on RED called NPD-RED. This algorithm is dependent on self-tuning feedback control for proportional and differential. NPD-RED relies on the current queue length and the instantaneous differential error signal to length of the buffer. The authors presented an analysis for stability of the system and also provided procedures for choosing the gains of the feedback of TCP/RED in order to maintain the current queue length. NPD-RED was compared with RED, using simulation NS2; the simulation results showed that NPD-RED outperformed RED with reference to average queue length, average throughput, and stability.

An AQM algorithm based on RED was proposed by [28]. This Improved Adaptive RED algorithm uses nonlinear smooth function for packet loss rate through exploiting the ascend demi-cauchy of fuzzy distribution. In the ARED algorithm, the increasing velocity of packet loss rate near the maximum threshold position is quick, whereas it is slow near the minimum threshold position. Also, the maximum value of packet dropping probability ($D_{\max}$) of ARED modified the size of the mean queue and the target for adapting the altering of network conditions.

Two discrete-time queue analytical models based on RED, called RED-Exponential and RED-Linear, were aimed at improving RED's issue in cases of heavy traffic were presented by [29]. These two analytical models dealt with this issue by using instantaneous queue length instead of average queue length as a congestion measure. The two models outperformed the classic RED regarding the performance results of mean queue length and average queuing delay when congestion occurs.

A learning-automata-like method for avoiding congestion incidence in wired networks called LALRED was introduced by [30]. The aim of LALRED was to enhance the value of average queue size used for congestion control and as such decrease the total packet loss at the queue.

An analytical framework model for RED queues using blended types of traffic such as TCP and User Datagram Protocol (UDP) was introduced by [31]. Steady-state goodput expressions for every flow and average queuing delay at every queue were obtained. The analytical framework was extended to include a class of RED queues, which offers different services for flows with multiple classes [31]. The analytical framework was validated with several network configuration simulations of RED. The results of these simulations showed that the analytical framework matched with several of the network configuration simulations of RED with a 5% value for mean error.

An adaptive queue management with random dropping method that uses information related to the average queue length and its rate of change was proposed [32]. This method was named AQMRD, and it uses the rate of change in the queue length as an extra parameter to detect the congestion. This method avoids the average queue length frequently exceeding the maximum threshold value by quickly responding to congestion thus avoiding buffer overflows.

An AQM method named ModRED that employs three dynamic packet dropping probabilities depending on the incoming traffic was presented by [33]. This method can use different traffic shapes. ModRED utilizes an Additive-Increase Multiplicative-Decrease (AIMD) algorithm with the purpose of using different packet dropping probability for dissimilar traffic loads [33]. Depending on the kind of traffic load, the packet dropping probability is calculated. ModRED handles congestion at the receiver side and as such it is used for TCP congestion control. ModRED provided better performance than RED with respect to throughput, goodput, packet delivery ratio, and delay simulation results.

A new RED method based on the Hemi-Rise Cloud model (CRED) was introduced by [34]. CRED uses a nonlinear packet loss approach, and this method enhances the uncertainty and sensitivity of the parameters. Controlling of network congestion and efficient use of network resources were accomplished [34]. CRED's stability was investigated, and this showed that CRED might enhance the stability and it gives a more acceptable performance than either RED or ARED.

RED is sensitive to its parameters and the traffic, so when traffic load is low, then bandwidth is underutilized. However, when traffic load is high, this will give a large delay [26]. Three-section random early detection (TRED) method based on nonlinear RED was proposed in [26]. In TRED, packet dropping probability function is split into three parts with the aim of discriminating between different loads: light, moderate, and high. This distinguishing between loads aims to obtain a tradeoff in the throughput and delay among low and high loads.

3. The Proposed RED-Exponential Technique

As mentioned previously in Section 1, RED has drawbacks that contribute in deterioration of its performance. This

(1) Initialisation stage
  $C = -1$
  aql = 0.0
(2) For every arriving packet at a RED-Exponential router buffer
  Calculate the aql for this packet at the RED-Exponential router buffer
  Examine the queue status at a router buffer, is it empty or not
    if the queue at a RED-Exponential router buffer is empty then
      Compute n, where $n = q$(current_time − idle_time)
      aql = aql × $(1 - qw)^n$
    Else
      aql = aql × $(1 - qw) + qw \times q$_instantaneous
(3) Determine a congestion status at the RED-Exponential router buffer
  if aql value is less than min threshold value then
    Calculate D_p value for the arriving packet as follows:
    $D_p = 0.0$
    $C = -1$
  else if aql value is greater than or equal to min threshold value and less than max threshold value then
    $C = C + 1$
    $D_{\text{init}} = (e^{\text{aql}} - e^{\text{min threshold}})/(e^{\text{max threshold}} - e^{\text{min threshold}})$
    $D_p = D_{\text{init}}/(1 - C \times D_{\text{init}})$
    Mark/Drop the arriving packet probabilistically regarding to D_p value due to occurrence of congestion
    $C = 0$
  else
    Mark/Drop every arriving packet $D_p = 1.0$ due to occurrence of heavy congestion
    $C = 0$
(4) When the RED-Exponential router buffer becomes empty
    idle_time = current_time

Algorithm 3: RED_E technique detailed pseudocode.

paper deals with the second problem (presetting RED's parameters), to ensure achieving a realistic performance without the need to tune parameters to a certain value, in particular, the $D_{\max}$. Therefore, the RED_E method was developed, which employs the aql as the congestion measure. Yet it differs on the way packets are dropped whenever a packet arrives at a router buffer, particularly when the aql value is equal to or larger than the min threshold and less than the max threshold. In this scenario, classic RED drops packets probabilistically using the following equations.

$$D_{\text{init}} = D_{\max} \times \frac{(\text{aql} - \text{min threshold})}{(\text{max threshold} - \text{min threshold})}, \quad (3)$$

$$D_p = \frac{D_{\text{init}}}{(1 - C \times D_{\text{init}})}. \quad (4)$$

In equations (3) and (4), D_{init} is the initial packet dropping probability and, in equation (4), C represents the number of packets which arrived at the router buffer and have not dropped since the last packet was dropped [7]. On the other hand, RED_E as shown in Algorithm 3 does not utilize the $D_{\max}$ parameter in computing D_{init} as employed in RED and many of its successors. Instead of calculating D_{init} as equation (3), RED–E employs its own computed D_{init} as shown in equation (5).

$$D_{\text{init}} = \frac{(e^{\text{aql}} - e^{\text{min threshold}})}{(e^{\text{max threshold}} - e^{\text{min threshold}})}. \quad (5)$$

It can be seen in equation (5) that RED_E no longer uses the $D_{\max}$ parameter, which is typically set in the preliminary stage as in RED to guarantee satisfactory performance. RED_E drops arriving packets exponentially using equations (4) and (5). Figures 2 and 3 display the D_{init} mechanism versus the aql mechanism for RED and RED_E, respectively. The proposed technique increases the D_{init} value exponentially from 0.0 to 1.0 as the aql value increases from the min threshold value to the max threshold value. The pseudocode of RED_E, given in Algorithm 3, has other goals besides removing the biased results by presetting the parameters, such as offering a more satisfactory performance in heavy congestion scenarios.

4. Simulation Results

4.1. Simulation Setup. RED, NLRED, and RED_E methods are simulated using a single queue node system that is shown in Figure 4. One packet can arrive and/or depart at a time unit called a slot that is used in a discrete-time queue [23]. The implementations of RED, NLRED, and proposed methods are conducted based on discrete-time queues in the Java environment. Packets interarrival and service times are geometrically distributed with means $1/\alpha$ and $1/\beta$, respectively, where α is the probability of packet arrival in a slot and β is the probability of packet departure in a slot. The arrival process used in both methods is the Bernoulli process [23]. In Figure 4, the finite capacity of a single queue node for RED, NLRED, or RED_E is K packets. First come, first

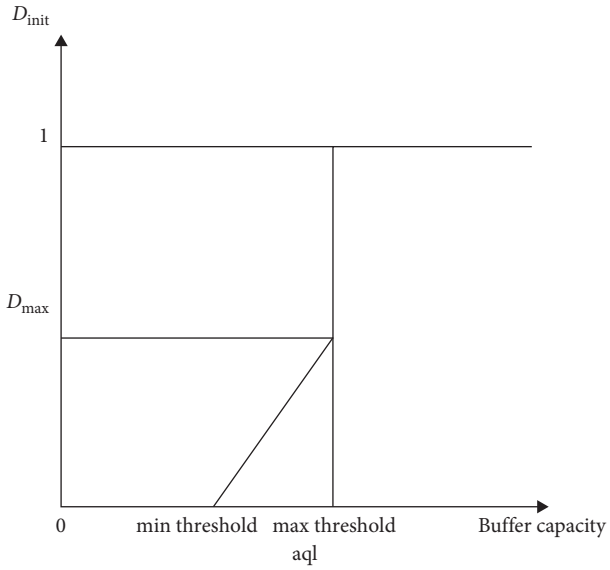

Figure 2: D_{init} versus aql of RED.

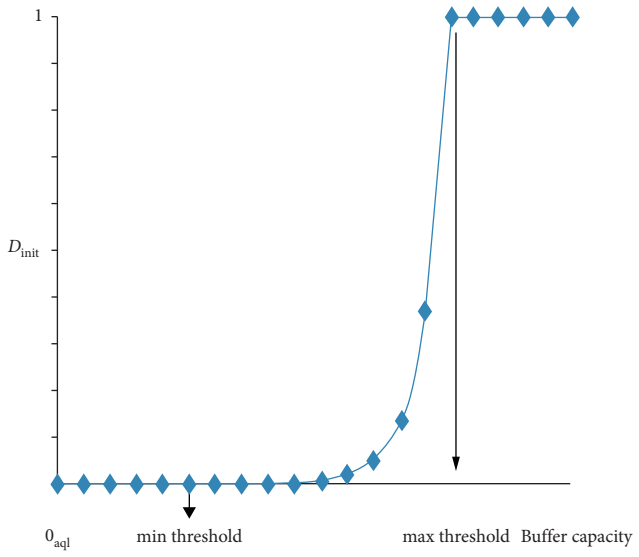

Figure 3: D_{init} versus aql of RED_E.

served (FCFS) is the queuing discipline which is used in RED, NLRED, or RED_E.

4.2. Parameter Settings. The parameters of RED, NLRED, and RED_E are tuned in Table 1.

In Table 1, the α values vary between 0.18 and 0.93, with half of these values (i.e., 0.18, 0.33, and 0.48) being less than β and the rest (i.e., 0.63, 0.78, and 0.93) being greater than β. These α and β values were set in order to test the performance measure results when 1) $\alpha < \beta$ and 2) $\alpha > \beta$ (congestion scenarios). K is set to 20 packets to test congestion with small buffer sizes. max threshold is triple that of min threshold as in [8]. The qw and D_{max} in Table 1 are set to the values as in [7]. The D_{max}' is set to a value calculated using equation (2). The set value of number of slots is given in order to guarantee it reaches a steady state.

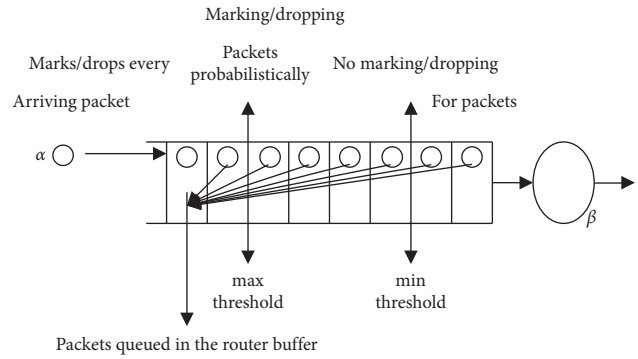

Figure 4: The single router buffer for RED, NLRED, or the proposed RED_E.

4.3. Simulation Environment. Special things are used in this simulation such as packet arrival probability and packet departure probability in discrete-time queues. Such special things were not employed by existing simulators. Moreover, these special things can be applied in simulation environments such as Java, and for this reason Java has been chosen as the simulation environment of RED, NLRED, and RED_E methods in this paper.

4.4. Results Analysis. A warming-up period is conducted before generating the performance measure results. When the system reaches a steady state, the performance measure results are then obtained. Each performance measure result represents the arithmetic mean of ten time runs for each α value. For every run, a seed value is changed utilizing a random number generator to remove any biased results. A decision on which method offers better satisfactory results is reached only using the setting values of α.

RED, NLRED, and RED_E algorithms are compared with reference to the following performance measures: mql, T, D, P_L, and D_P. The abbreviation mql denotes mean queue length; T is the throughput that represents the number of packets that have been successfully passed through a queue node for every time unit; D is the average queuing delay for packets; P_L is the probability of packet loss due to buffer overflow; and D_P is the packet dropping probability before a router buffer becomes full. This comparison aims to evaluate RED_E performance in different congestion situations and without utilizing the D_{max} parameter.

Figures 5–9 show the performance measure results (mql, T, D, P_L, and D_p) versus α for RED, NLRED, and the RED_E. The column chart type is used rather than scatter chart type due to the performance measure results of the compared methods being slightly different based on α, and these differences are shown more clearly by using column chart type than scatter chart type. The performance measures are functions of α.

4.4.1. Performance Evaluation Results Based on Varying α Values. After analyzing Figures 5 and 7, RED, NLRED, and the proposed RED_E provide similar D and D results when α is less than or equal to 0.33. In other words, RED and

Table 1: Parameter settings of RED, NLRED, and RED_E.

Parameter	The set value
α	0.18–0.93
β	0.5
K	20
min threshold	3
max threshold	9
qw	0.002
$D_{\max}$	0.1
Number of slots	2,000,000

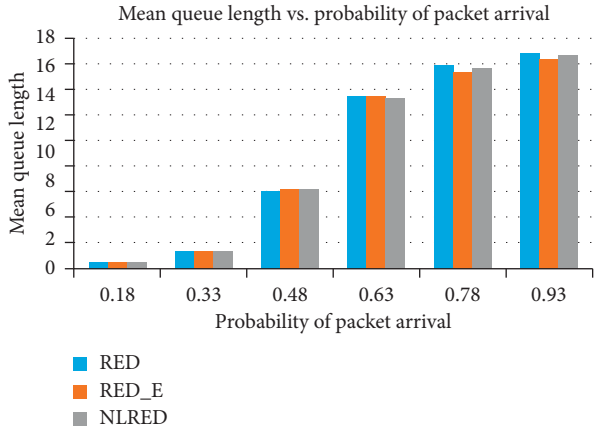

Figure 5: mql versus α.

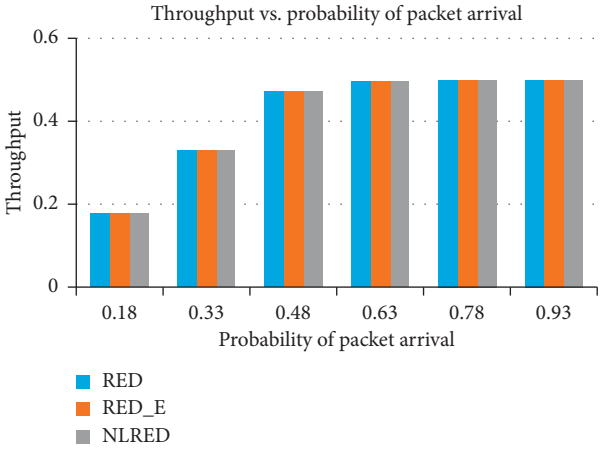

Figure 6: T versus α.

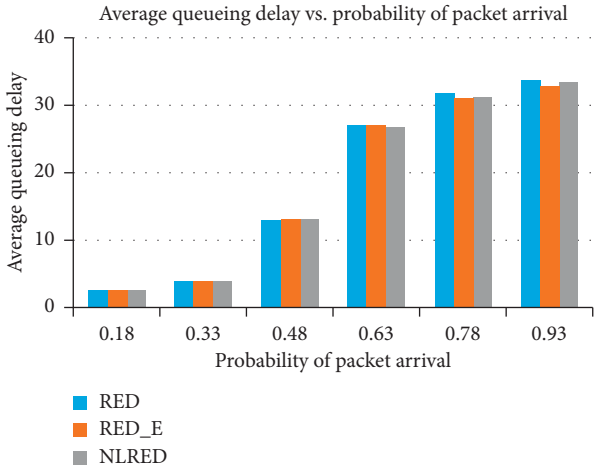

Figure 7: P_L versus α.

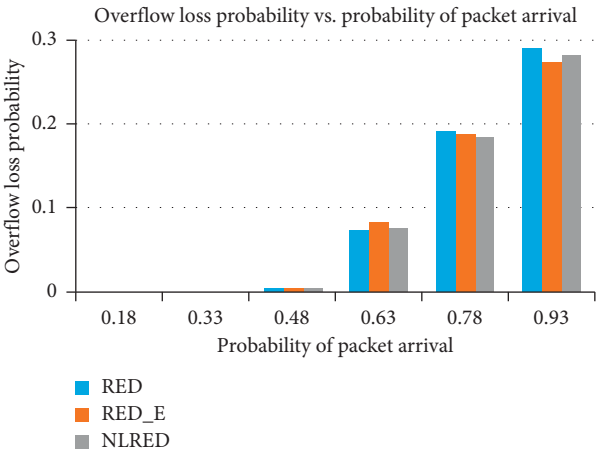

Figure 8: D versus α.

RED_E give similar mql and D results in noncongestion situations at a router buffer of a queue node. This is due to both RED and E_RED dropping the same numbers of packets before the router buffer is full (see Figure 9), that is, zero, and loses the same number of packets due to overflow (see Figure 8), that is, zero.

When α is greater than 0.33 and less than or equal to 0.48, RED provides a slightly lower mql and D results than NLRED and RED_E since RED drops slightly more packets than NLRED and RED_E before the buffer is full. RED_E and NLRED gave similar results concerning mql and D because RED_E and NLRED drop similar packets before the buffer is full. Moreover, RED, NLRED, and RED_E lose comparable number of packets due to overflow (P_L).

When α is larger than 0.48 and less than 0.63, NLRED provides slightly smaller *mql* and D results among the compared methods because of the previously mentioned reason. RED and RED_E have analogous mql and D results. RED and NLRED lose fewer packets than RED_E, and this is because the router buffers of RED and NLRED are

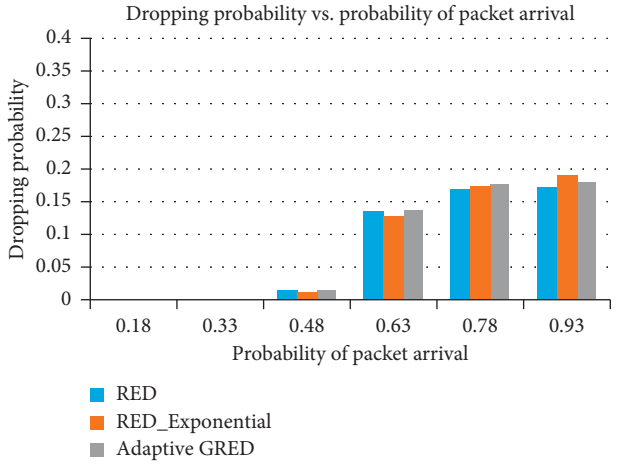

Figure 9: D_p versus α.

overflowed on number of occasions, less than that of RED_E. Furthermore, RED_E has fewer dropped packets than RED and NLRED, and both RED and NLRED drop similar packets before their router buffers are full.

When α is greater than 0.63 such as $\alpha = 0.78$, RED_E achieves the most satisfactory measured performance among the compared methods in terms of *mql* and D results since RED_E maintains less mean queue length. Additionally, NLRED to some extent gave better results for mql and D than RED because of the previous mentioned reason. RED provides higher P_L result than NLRED and RED_E since RED's buffer overflows more times than NLRED and RED_E. NLRED has a reduced P_L result than RED_E. In addition, RED drops fewer packets than the other two methods, and RED_E has slightly better D_p result than that of NLRED. In the presence of heavy congestion such as $\alpha > 0.78$ such as $\alpha = 0.93$, RED_E accomplishes lower mql, D and P_L results than RED or NLRED due to RED_E dropping a higher number of packets (D_p) than either RED or NLRED. Furthermore, NLRED produces better mql, D, and P_L results than RED since NLRED drops further packets than RED.

It is noted in Figure 6 that RED, NLRED, and RED_E offer similar T results in the existence of congestion or without congestion.

4.4.2. Performance Evaluation Results Based on Varying α Values. Further simulation tests between RED, NLRED, and RED_E based on different values for min threshold to evaluate their effect on performance results were conducted. α was set to 0.78 since this value can produce heavy congestion and we needed to evaluate the effectiveness of the min threshold parameter on the compared techniques in the presence of heavy congestion. The min threshold was set to different values, ranging from 3 to max threshold − 1 to observe the effectiveness of each min threshold value on the performance measure results. The performance measure results of RED, NLRED, and RED_E versus min threshold values are given in Figures 10–14. The chart type used in this subsection is scatter because the results of the compared methods can be shown clearly.

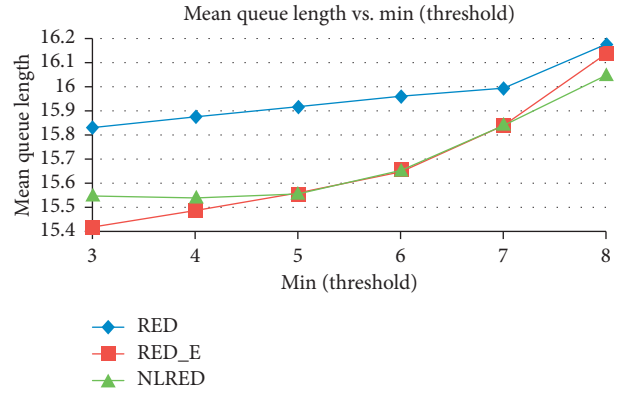

Figure 10: mql versus min threshold.

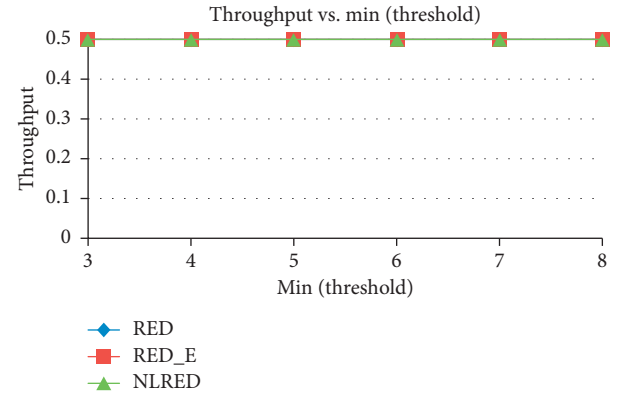

Figure 11: T versus min threshold.

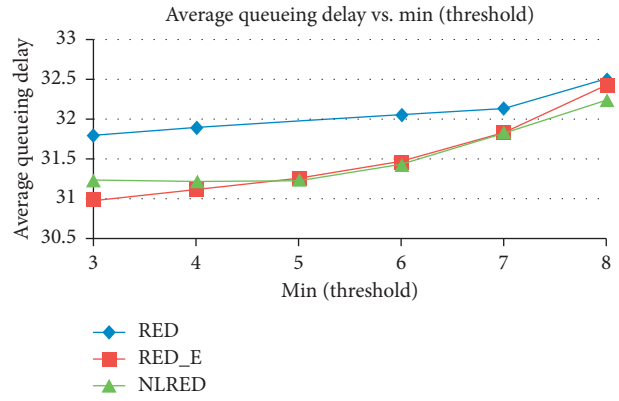

Figure 12: D versus min threshold.

It is noted from Figures 10, 12, and 13 that RED offers higher mql, D, and P_L results than NLRED and RED_E for all min threshold values. This is because the RED router buffer drops fewer packets than NLRED and RED_E (see Figure 14). In addition, RED_E provides smaller mql and D results than NLRED when the min threshold value is 3 or 4, and these values represent the farthest given values from the max threshold. In case that the value of min threshold is set

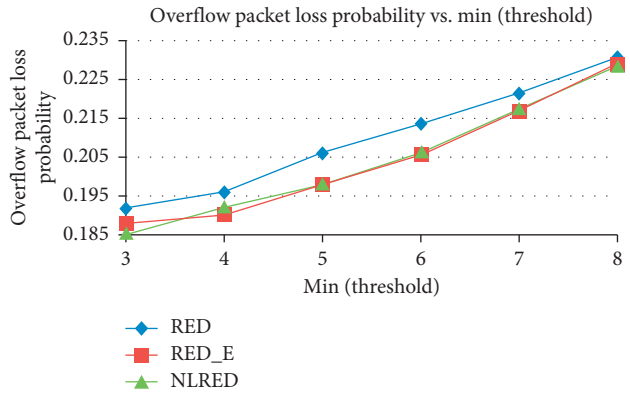

Figure 13: P_L versus min threshold.

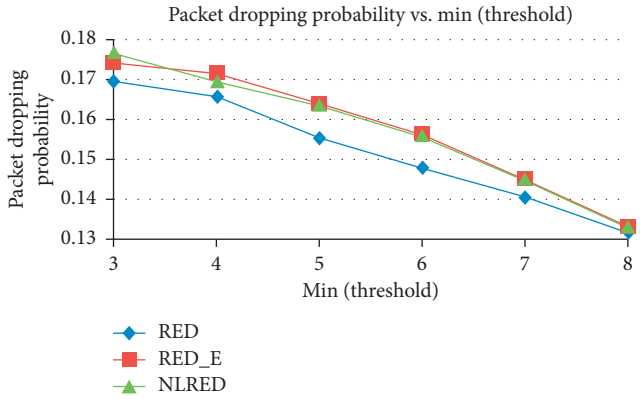

Figure 14: D_p versus min threshold.

to 8 (the nearest value to the max threshold), then NLRED generates lower mql and D results than that of RED_E; this is due to the length of mean queue of NLRED being smaller than that of RED_E. The performance results of NLRED and RED_E regarding mql and D are comparably similar when the min threshold value is set to 5, 6, and 7 (halfway between the min threshold and the min threshold). NLRED generates a smaller P_L than that of RED_E when the min threshold is set to the farthest value from the max threshold, whereas P_L derived smaller P_L than NLRED when the min threshold is given as 4.

It is clear from Figure 14 that RED drops fewer packets than NLRED or RED_E since both NLRED and RED_E offer smaller mql results than RED. RED_E obtains smaller D_P than NLRED when the min threshold is given to a value that is the farthest value from the max threshold. However, NLRED achieves smaller D_P than RED_E when the min threshold value is set to 4.

For the other min threshold values (5, 6, 7, and 8), both NLRED and RED_E generate similar P_L and D_P results since they lose and drop a similar number of packets. In addition, T performance results for RED, NLRED, and RED_E are similar regardless of the min threshold value and stabilized on β value and are not affected by the min threshold parameter. Lastly, D_P results for RED, NLRED, and RED_E are decreased if the value of the min threshold is increased. It can be inferred that the performance measures' results of RED, NLRED, and the RED_E are all affected by the min threshold parameter except for the T results (see Figure 11).

5. Conclusions and Future Work

This research paper addressed one of the AQM problems in congestion, namely, the configuration of the $D_{\max}$ parameter to generate a good yet biased performance. We proposed a new exponential AQM method called RED-Exponential (RED_E) to minimize reliance of these methods on the $D_{\max}$ parameter. The proposed technique differs from RED and its successors by dropping arriving packets in an exponential manner rather than tuning the $D_{\max}$ parameter. This exponential packet dropping process will minimize the dependency on pretuned parameters, particularly the $D_{\max}$ parameter, when calculating the performance measures of the AQM methods.

Simulation results have been generated to measure the advantages and disadvantages of RED_E. The focus of the simulation results was on utilizing different α and min threshold values to measure the performance of RED, NLRED, and RED_E in different situations (no congestion, light congestion, and heavy congestion). The key metrics used to measure the performance of the considered AQM techniques are T, mql, D, D_p, and P_L. Hereunder is the summary of the results.

RED_E enhanced the performance measures results with reference to mql and D when heavy congestion occurred and offered better mql and D results than RED or NLRED.

When $\alpha = 0.78$, RED_E outperformed RED and NLRED with reference to mql and D results. Also, NLRED provided the best P_L result among the compared methods, whereas RED produced the most satisfactory D_p result. In the case $\alpha = 0.93$, RED_E obtained the most acceptable mql, D, and P_L results compared with RED and NLRED, while RED presented better D_p result than NLRED and RED_E.

When light congestion exists ($\alpha = 0.48$), RED slightly outperformed RED_E with regard to mql and D results, whereas RED_E and NLRED have similar mql and D results. However, when $\alpha = 0.63$, RED outperformed NLRED and RED_E with reference to mql and D results. Moreover, both RED and NLRED outperformed RED_E regarding to P_L results, whereas RED_E outperformed RED and NLRED in D_p results.

The considered techniques offered similar T results when heavy congestion occurred. In addition, RED, NLRED, and RED_E offered similar mql, T, D, P_L, and D_p results when there was no congestion in a router buffer.

In addition, NLRED and RED_E provided better mql, D, and P_L performance results than RED using different values of the min threshold parameter. For instance, RED_E offered marginally higher P_L results and marginally lower D_p than those of NLRED when the min threshold was set to the farthest value from the max threshold, and it generated slightly lower P_L results and slightly higher D_p than those of NLRED when the min threshold was set to the second

farthest value from the max threshold. For the other given values of the max threshold (5, 6, 7, and 8), both RED_E and NLRED produced similar P_L and D_p results.

The results of mql, D, and P_L of RED, NLRED, and RED_E techniques increased as long as the value of the min threshold is increased, whereas the D_P results decreased when the min threshold is increased. Hence the results of RED, NLRED, and RED_E have been impacted by the min threshold parameter except in the case of the T metric.

In the near future, we intend to develop the RED_E in order to further improve performance results based on dynamic values of the min threshold, max threshold, and qw.

Data Availability

There are no data availabile.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

References

[1] J. Wang, L. Guan, L. B. Lim et al., "QoS enhancements and performance analysis for delay sensitive applications," *Journal of Computer and System Sciences*, vol. 77, no. 4, pp. 665–676, 2011.

[2] M. Welzl, *Network Congestion Control: Managing Internet Traffic*, p. 282, Wiley, Hoboken, NJ, USA, 2005.

[3] D. D. Clark and W. Wenjia Fang, "Explicit allocation of best-effort packet delivery service," *IEEE/ACM Transactions on Networking*, vol. 6, no. 4, pp. 362–373, 1998.

[4] S. Athuraliya, S. H. Low, V. H. Li, and Q. Qinghe Yin, "REM: active queue management," *IEEE Network*, vol. 15, no. 3, pp. 48–53, 2001.

[5] J. Aweya, M. Ouellette, and D. Y. Montuno, "A control TheoRED_Eic approach to active queue management," *Computer Network*, vol. 36, no. 2-3, pp. 203–235, 2001.

[6] W. Feng, D. Kandlur, D. Saha, and K. G. Shin, "Blue: a new class of active queue management algorithms," Technical Report, UM CSE-TR-387-99, University of Michigan, Ann Arbor, MI, USA, 1999.

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

[8] S. Floyd, G. Ramakrishnan, and S. Shenker, "Adaptive RED: an algorithm for increasing the robustness of RED's active queue management," Technical Report, ICSI, New Delhi, India, 2001.

[9] S. Floyd, "Recommendations on using the gentle variant of RED," 2000, http://www.aciri.org/floyd/red/gentle.html.

[10] B. Abbasov and S. Korukoglu, "Effective RED: an algorithm to improve RED's performance by reducing packet loss rate," *Journal of Network and Computer Applications*, vol. 32, no. 3, pp. 703–709, 2009.

[11] W. Chen, Y. Li, and S. H. Yang, "An average queue weight parameterization in a network supporting TCP flows with RED," in *Proceedings of the 2007 IEEE International Conference on TuesM02 Networking, Sensing and Control*, pp. 590–595, London, UK, April 2007.

[12] W. Chen and S. H. Yang, "The mechanism of adapting RED parameters to TCP traffic," *Proc. of Computer Communications*, Elsevier, vol. 32, no. 13-14, pp. 1525–1530, 2009.

[13] J. Hong, C. Joo, and S. Bahk, "Active queue management algorithm considering queue and load states," in *Proceedings of the 13th International Conference on Computer Communications and Networks*, pp. 140–145, Chicago, IL, USA, October 2004.

[14] K. Okokpujie, C. Emmanuel, O. Shobayo, E. Noma-Osaghae, and I. Okokpujie, "Comparative analysis of the performance of various active queue management techniques to varying wireless network conditions," *International Journal of Electrical and Computer Engineering (IJECE)*, vol. 9, no. 1, p. 359~368, 2019.

[15] H. Abdel-Jaber, M. Woodward, F. Thabtah, and A. Abu-Ali, "Performance evaluation for DRED discrete-time queueing network analytical model," *Journal of Network and Computer Applications*, vol. 31, no. 4, pp. 750–770, 2008.

[16] H. Abdeljaber, M. Woodward, F. Thabtah, and M. Al-diabat, "Modelling Blue active queue management using DiscRED_ Ee-time queue," in *Proceedings of the 2007 International Conference of Information Security and Internet Engineering (ICISIE'07)*, pp. 568–573, London, UK, July 2007.

[17] M. Al-Diabat, H. Abdeljaber, F. Thabtah, O. Abou-rabia, and M. Kishta, "Analytical Models based DiscRED_Ee-time queuing for the congested network," *International Journal of Modeling, Simulation, and Scientific Computing (IJMSSC)*, vol. 3, no. 1, p. 22, 2012.

[18] K. Ramakrishnan and S. Floyd, "The addition of explicit congestion notification (ECN) to IP," *RFC*, vol. 3168, 2001.

[19] A. G. Fayoumi, "Delay performance evaluation of shared-buffer based all-optical multihop networks," in *Proceedings of the 15th IEEE International Conference on Networks*, pp. 166–169, Adelaide, SA, Australia, November 2007.

[20] L. B. Lim, L. Guan, A. Grigg et al., "Controlling mean queuing delay under multi-class bursty and correlated traffic," *Journal of Computer and System Sciences*, vol. 77, no. 5, pp. 898–916, 2011.

[21] 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 of the Sixth IEEE Symposium on Computers and Communications*, pp. 122–129, IEEE, Hammamet, Tunisia, July 2001.

[22] R. Braden, D. Clark, J. Crowcroft et al., "Recommendations on queue management and congestion avoidance in the internet," *RFC*, vol. 2309, 1998.

[23] M. E. Woodward, *Communication and Computer Networks: Modelling with discRED_Ee-Time Queues*, Pentech Press, London, UK, 1993.

[24] 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, 2006.

[25] B. Zheng and M. Atiquzzaman, "DSRED: an active queue management scheme for next generation networks," in *Proceedings of 25th Annual IEEE Conference on Local Computer Networks (LCN'00)*, pp. 242–251, Tampa, Fl, USA, November 2000.

[26] F. Chen-Wei, 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, 2015.

[27] N. Xiong, A. V. Vasilakos, L. T. Yang et al., "A novel self-tuning feedback controller for active queue management supporting TCP flows," *Information Sciences*, vol. 180, no. 11, pp. 2249–2263, 2010.

[28] J. Zhang, W. Xu, and L. Wang, "An improved adaptive active queue management algorithm based on nonlinear smoothing," *Procedia Engineering*, vol. 15, pp. 2369–2373, 2011.

[29] H. Abdel-Jaber, F. Thabtah, and M. Woodward, "Modeling discrete-time analytical models based on random early detection: exponential and linear," *International Journal of Modeling, Simulation, and Scientific Computingfic Computing*, vol. 6, no. 3, Article ID 1550028, 22 pages, 2015.

[30] S. Misra, B. J. Oommen, S. Yanamandra, and M. S. Obaidat, "Random early detection for congestion avoidance in wired networks: a discretized pursuit learning-automata-like solution," *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, vol. 40, no. 1, pp. 66–76, 2010.

[31] A. A. Abouzeid and S. Roy, "Modeling random early detection in a differentiated services network," *Computer Networks*, vol. 40, no. 4, pp. 537–556, 2002.

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

[33] K. Kachhad and A. Lathigara, "ModRED: modified RED an efficient congestion control algorithm for wireless networks," *International Research Journal of Engineering and Technology (IRJET)*, vol. 5, no. 5, pp. 1879–1884, 2018.

[34] Z. Yuhong, M. Zhonggui, Z. Xuefeng, and T. Xuyan, "An improved algorithm of nonlinear RED Based on membership cloud theory," *Chinese Journal of Electronics*, vol. 26, no. 3, pp. 537–543, 2017.