Paper

RED-LE: A Revised Algorithm for Active Queue Management

Samuel O. Hassan

Department of Mathematical Sciences, Olabisi Onabanjo University, Ago-Iwoye, Nigeria

https://doi.org/10.26636/jtit.2022.160022

Abstract—The random early detection (RED) algorithm was developed in 1993. Nearly three decades later, several improved variants have been proposed by scientists. The use of a (pure) linear function for computing packet drop probability has turned out to be a disadvantage, leading to the problem of large delays. Such a problem may be addressed by using linear and non-linear (i.e. as exponential) packet drop probability functions. This paper proposes a revised RED active queue management algorithm named RED-linear exponential (RED-LE). This variant involves an interplay of linear and exponential drop functions, in order to improve the performance of the original RED algorithm. More importantly, at low and moderate network traffic loads, the RED-LE algorithm employs the linear drop action. However, for high traffic loads, RED-LE employs the exponential function for computing the packet drop probability rate. Experimental results have shown that RED-LE effectively controls congestion and offers an improved network performance under different traffic loads.

Keywords—active queue management, network congestion, routers, RED-LE, simulation.

1. Introduction

Network congestion may be described as a condition in which the amount of incoming data packets (generated traffic) is greater than the amount that the network's available resources are capable of accommodating [1]–[5]. The problem of network congestion affects the quality of service (QoS), as it leads to high packet delays, loss rates, and low throughput [2], [4], [5]–[9].

A router plays an important role in the process of controlling network congestion, as it allows to achieve improved network performance rates [9]. Router-based congestion control algorithms, such as active queue management (AQM), effectively circumvent network congestion by dropping packets at an early stage, before the buffer becomes full and sends a feed- back signal to sources in order to reduce their transmission rates [10]–[12].

The most prominent type of an AQM algorithm is random early detection (RED), developed by Floyd and Jacobson in 1993 [11]. RED continues to serve as a basis for many new AQM algorithms [12]. Upon the arrival of each packet at the router, RED updates the average queue size (denoted *avg*) which is used as an indicator for congestion detection. To perform this computation, the current status of the queue is examined.

If the router's queue is non-empty, *avg* value is determined using the exponential weighted moving average (EWMA) mechanism in the following manner:

$$avg = (1 - W_q)avg' + (W_q \times q_{cur}) \ , \tag{1}$$

where $W_q \in [0,1]$ represents a preset weighting factor, avg' represents the previously computed average queue size, and q_{cur} represents the current queue size.

However, if the router's queue is empty, *avg* is determined as:

$$avg = (1 - W_q)^n \times avg' \ , \tag{2}$$

with

$$n = f(q\_current\_time - q\_idle\_time) \ , \tag{3}$$

where $q\_current\_time$ denotes the current time, q_idle_time denotes the beginning of queue idle time, and $f(t)$ denotes a linear function of time t.

The probability of dropping a packet in RED depends on *avg* in the following manner:

$$P_b = \begin{cases} 0, & \text{if } avg \in [0, minTH) \\ maxP(\frac{avg - minTH}{maxTH - minTH}), & \text{if } avg \in [minTH, maxTH) \\ 1, & \text{if } avg \geq maxTH \end{cases} , \tag{4}$$

where $minTH$ is the router's minimum queue threshold, $maxTH$ represents the router's maximum queue threshold, $maxP$ represents the maximum packet drop probability, and P_b stands for the initial packet dropping probability.

In RED, if $avg \in [0, minTH)$ then no packet will be dropped and if $avg \in [minTH, maxTH)$, then the packets are randomly dropped with the probability of:

$$P_b = maxP(\frac{avg - minTH}{maxTH - minTH}) \ . \tag{5}$$

Finally, if $avg \geq maxTH$, then the packet is forced to be dropped, with a probability of one. The final packet drop probability P_a therefore given by:

$$P_a = \frac{P_b}{1 - count \times P_b} \ , \tag{6}$$

where *count* represents the number of packets that arrived since the last dropped packet.

There are several models in literature that modify the linear probability function of RED algorithm in an attempt to overcome its weaknesses. In this paper, another improvement-aiming modification is suggested, known as the random early detection-linear exponential (RED-LE) algorithm. The RED-LE algorithm uses both linear and exponential packet drop functions instead of a (pure) linear packet drop probability function of RED, while retaining RED's other characteristics.

The rest of the paper is organized as follows. A review of related works on the RED algorithm is provided in Section 2. A description of RED-LE is given in Section 3. The simulation configuration is presented and the results are discussed in Section 4. Finally, a brief conclusion is given in Section 5.

2. Related Works

To increase the throughput of RED, Floyd developed, in [13], the gentle RED (GRED) variant in which a linear function is employed to compute the packet drop probability when *avg* lies within the $minTH$ and $maxTH$ queue threshold range – Eq. (5) – while another linear function is employed to compute the packet drop probability when *avg* is within the $maxTH$ and $2 \times maxTH$ threshold range – Eq. (7):

$$P_b = maxP + (1 - maxP)\frac{avg - minTH}{maxTH} \ . \tag{7}$$

To ensure higher stability, Giménez *et al*. developed a new RED variety called BetaRED, which involves a beta distribution function to compute the packet drop probability instead of a linear function when *avg* value is within the $minTH$ and $maxTH$ threshold range [12].

In [14], an attempt to reduce the number of input parameters for RED was made by Abdel-Jaber, known as Exponential RED (RED_E) in which a (pure) exponential drop function given in Eq. (8) is employed to compute the packet drop probability when *avg* value is between the $minTH$ and $maxTH$ queue thresholds.

$$P_b = \begin{cases} 0, & \text{if } avg \in [0, minTH) \\ \dfrac{e^{avg} - e^{minTH}}{e^{maxTH} - e^{minTH}}, & \text{if } avg \in [minTH, maxTH) \\ 1, & \text{if } avg \geq maxTH \end{cases} \ . \tag{8}$$

To increase RED's throughput, Zhang *et al*. proposed, in [15], the MRED variety in which a quadratic function is employed to compute the packet drop probability when *avg* is within the $minTH$ and $maxTH$ queue threshold range given by Eq. (9), while a linear function is employed to compute the packet drop probability when *avg* is within the $maxTH$ and $2 \times maxTH$ queue threshold range as stated in Eq. (10).

$$P_b = maxP\frac{avg^2 - minTH^2}{maxTH^2 - minTH^2} \ , \tag{9}$$

$$P_b = maxP + (1 - maxP)\frac{avg - minTH}{maxTH} \ . \tag{10}$$

To achieve a trade-off between delay and throughput performance metrics, Paul *et al*. suggested, in [16], the Smart RED (SmRED) scheme, given in Eq. (11), in which a quadratic function is employed to compute the packet drop probability when *avg* lies within the $minTH$ and $Target$ queue threshold range, while a linear function is employed to compute the packet drop probability when *avg* lies within the $Target$ and $maxTH$ queue threshold range:

$$P_b = \begin{cases} 0, & \text{if } avg \in [0, minTH) \\ maxP\left(\dfrac{avg - min_{TH}}{max_{TH} - min_{TH}}\right)^2, & \text{if } avg \in [minTH, Target) \\ maxP\sqrt{\dfrac{avg - min_{TH}}{max_{TH} - min_{TH}}}, & \text{if } avg \in [Target, maxTH) \\ 1, & \text{if } avg \geq maxTH \end{cases} \tag{11}$$

in which

$$Target = minTH + \frac{maxTH - minTH}{2} \ . \tag{12}$$

In order to obtain improved throughput, Suwannapong and Khunboa developed, in [17], yet another variety of RED, named Congestion Control RED (CoCo-RED) which involves both linear and an exponential drop functions. The linear function is employed when *avg* value is within the $minTH$ and $maxTH$ queue threshold range, while the exponential function is employed when *avg* value is within the $maxTH$ and K queue threshold range:

$$P_b = \begin{cases} 0, & \text{if } avg \in [0, minTH) \\ maxP\dfrac{avg - minTH}{maxTH - minTH}, & \text{if } avg \in [minTH, maxTH) \\ ab^{avg}, & \text{if } avg \in [maxTH, K) \end{cases} \tag{13}$$

in which

$$a = \frac{1}{\left(e^{\frac{\ln(1/maxP)}{K - maxTH}}\right)^{maxTH}} \times maxP \tag{14}$$

and

$$b = e^{\frac{\ln(1/maxP)}{K - maxTH}} \ . \tag{15}$$

Feng *et al*. [18] proposed a three-section RED (TRED) which employs the usage of a non-linear drop action, a linear drop action, and a non-linear drop function for low, moderate, and high buffer occupancy rates, respectively. TRED results in high throughput at high traffic loads and achieves a reduced delay at high traffic loads.

In order to increase throughput, Zhou *et al*. proposed, in [19], another variant named non-linear RED (NLRED) in which a quadratic function is employed to compute the packet drop probability when *avg* value is between the $minTH$ and $maxTH$ queue thresholds.

To reduce the packet loss rate, Kumhar *et al*. developed, in [20], quadratic RED (QRED) in which a quadratic function is deployed to compute the packet drop probability

when avg value lies within the $minTH - maxTH$ queue threshold range:

$$P_b = \left(\frac{avg - minTH}{K - minTH} \right)^2 \tag{16}$$

or

$$P_b = 1 - \left(\frac{K - avg}{K - minTH} \right)^2 , \tag{17}$$

in which K represents the buffer size.

To achieve increased throughput, Adamu *et al*. developed, in [21], the flexible RED (FXRED) algorithm. At low and moderate traffic loads, i.e. when avg value lies within the $minTH$ and Δ queue threshold range, FXRED uses a non-linear function to drop packets. However, at high traffic loads, i.e. when avg value lies within the Δ and $maxTH$ queue threshold range, FXRED switches to a linear pattern for aggressive drop action. In FXRED, Δ was chosen in the following manner:

$$\Delta = \frac{minTH + maxTH}{2} . \tag{18}$$

Furthermore, to improve throughput performance, the self-adaptive RED (SARED) algorithm developed by Adamu *et al*. in [4] is quite similar to RED, except that either a quadratic or linear drop function is employed when avg lies within the $minTH$ and $maxTH$ threshold range. The quadratic drop function is employed for lower and moderate buffer occupancies, while a linear drop function is employed for higher buffer occupancies, respectively.

3. Random Early Detection-Linear Exponential (RED-LE)

The proposed algorithm is named random early detection-linear exponential (simply denoted by RED-LE). This revised RED algorithm involves an interplay of linear and exponential drop functions in order to increase the performance of the original RED algorithm. The improved packet drop probability is:

$$P_b = \begin{cases} 0, & \text{if } avg \in [0, minTH) \\ 2maxP \dfrac{avg - minTH}{maxTH - minTH}, & \text{if } avg \in [minTH, Target) \\ e^{\log(maxP)\frac{2(maxTH - avg)}{maxTH - minTH}}, & \text{if } avg \in [Target, maxTH) \\ 1, & \text{if } avg \geq maxTH \end{cases} \tag{19}$$

in which

$$Target = \frac{maxTH + minTH}{2} . \tag{20}$$

Considering the curve given in Fig. 1, RED-LE breaks the section between $minTH$ and $maxTH$ queue thresholds into two parts which include both a linear drop function and an exponential drop function, such that:

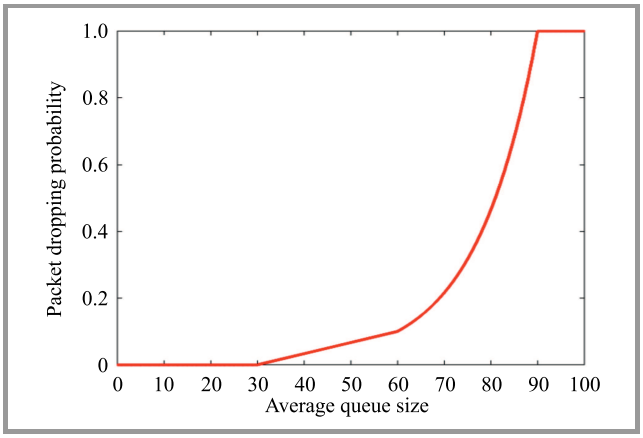

Fig. 1. RED-LE's drop probability function curve.

- At low and moderate traffic loads which account for cases with $minTH \leq avg < Target$, the packet drop function is expressed as:

$$P_b = 2maxP \frac{avg - minTH}{maxTH - minTH} . \tag{21}$$

  $Target$ is a mid-point threshold defined according to Eq. (20).

- At high traffic load which account for cases where $Target \leq avg < maxTH$, the packet drop function is expressed as:

$$P_b = e^{\log(maxP)\frac{2(maxTH - avg)}{maxTH - minTH}} . \tag{22}$$

  Using Eq. (22), a more aggressive drop action will be achieved at high load.

It is worth to mention that $Target$ serves the purpose of distinguishing between two traffic scenarios: lower and moderate buffer occupancies, and higher buffer occupancies. A detailed pseudo-code for RED-LE is presented in Algorithm 1.

4. Simulations

In this section, the proposed RED-LE AQM algorithm is implemented using the ns-3 simulator [22]. The effectiveness of RED-LE is evaluated and compared against two algorithms, namely TRED and RED_E, under three different network traffic loads: low, moderate, and high.
The simulation double dumbbell topology (shown in Fig. 2) consists of N TCP connecting sources transmitting to one sink (denoted by D) via two routers R_1 and R_2. These two routers R_1 (with the algorithm implemented) and R_2 are connected together via a bottleneck link with a capacity of 10 Mbps and a propagation delay of 100 ms. Other hosts are connected to the routers via 100 Mbps links characterized by a propagation delay of 5 ms. The N number of

Algorithm 1. Detailed RED-LE's algorithm

Initialization:
$avg = 0$
$count = -1$
Upon every packet arrival do
Calculate the average queue size avg
if router's queue is non-empty **then**
  $avg = (1 - W_q)avg' + (W_q \times q_{cur})$
else
  Compute n in which
  $n = f(q\_current\_time - q\_idle\_time)$
  $avg = (1 - W_q)^n \times avg'$
end if
if $avg < minTH$ **then**
  Accept the packet
  Set $count = count - 1$
else if $minTH \leq avg < Target$ **then**
  Set $count = count + 1$
  Based on the linear drop function compute the final drop probability P_a:
  $P_b = 2maxP(\frac{avg - minTH}{maxTH - minTH})$
  $P_a = P_b/(1 - count \times P_b)$
  Drop arriving packet according to P_a
  Set $count = 0$
else if $Target \leq avg < maxTH$ **then**
  Set $count = count + 1$
  Based on the exponential drop function compute the final drop probability P_a:
  $P_b = e^{\log(maxP)\frac{2(maxTH - avg)}{maxTH - minTH}}$
  $P_a = P_b/(1 - count \times P_b)$
  Drop arriving packet according to P_a
  Set $count = 0$
else if $maxTH \leq avg$ **then**
  Drop arriving packet
  Set $count = 0$
end if
if count= -1 **then**
  When the router's queue becomes empty
  Set $q\_idle\_time = q\_current\_time$
end if

Saved variables:
avg: average queue size
q_idle_time: beginning of queue idle time
$count$: packets since last dropped packet

Preset input parameters:
$minTH$: router's queue minimum threshold
$maxTH$: router's queue maximum threshold
$maxP$: maximum packet drop probability
W_q: weighting factor

Other:
P_b: current packet marking probability
q_{cur}: current queue size
$q\_current\_time$: current time
$f(t)$: a linear function of time t

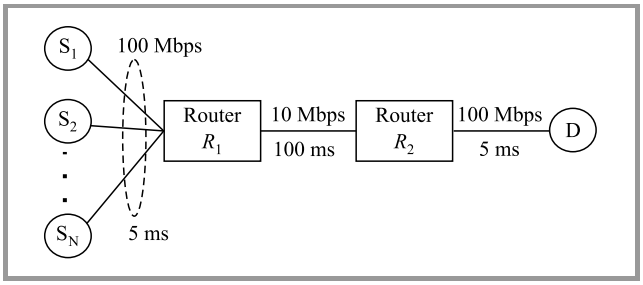

Fig. 2. Network topology.

flows was varied to indicate various levels of traffic loads in the network. The TCP implementation used is New Reno. The buffer size was set to 250 packets, while simulation time was set to 100 s. Other configurations are shown in Table 1.

Table 1
Simulation setup

Input parameter	Algorithm	Value
$minTH$	TRED, RED_E & RED-LE	30 packets
$Target$	RED-LE	60 packets
$maxTH$	TRED, RED_E & RED-LE	90 packets
$maxP$	TRED & RED-LE	0.1
W_q	TRED, RED_E & RED-LE	0.002

4.1. Scenario 1 – Low Load

In this scenario, the number of connecting sources is set to 5. As shown in Fig. 3a, the RED-LE algorithm reduces the average queue size better than both TRED and RED_E algorithms. As shown in Table 2, RED-LE reduced the queue size by 1.9437% and 14.1522% compared with TRED and RED_E, respectively.
Delay performance is shown in Fig. 3b, RED-LE outperformed both TRED and RED_E. As shown in Table 3, delays in RED-LE were by 0.0226% and 0.1140% shorter when compared with TRED and RED_E, respectively.
Figure 3c shows throughput performance. RED_E clearly outperformed both TRED and RED-LE, although the results of RED-LE were better than those of TRED. A detailed analysis is presented in Table 4.

4.2. Scenario 2 – Moderate Load

In this scenario, the number of connecting sources is set to 20. As shown in Fig. 4a, the proposed RED-LE algorithm is clearly more efficient at reducing the average queue size than both TRED and RED_E algorithms. As shown in Table 2, RED-LE reduced it by 5.2565% and 29.7475% percentage decrement when compared with TRED and RED_E, respectively.
Delay performance is shown in Fig. 4b, RED-LE satisfactorily outperformed both TRED and RED_E. As shown

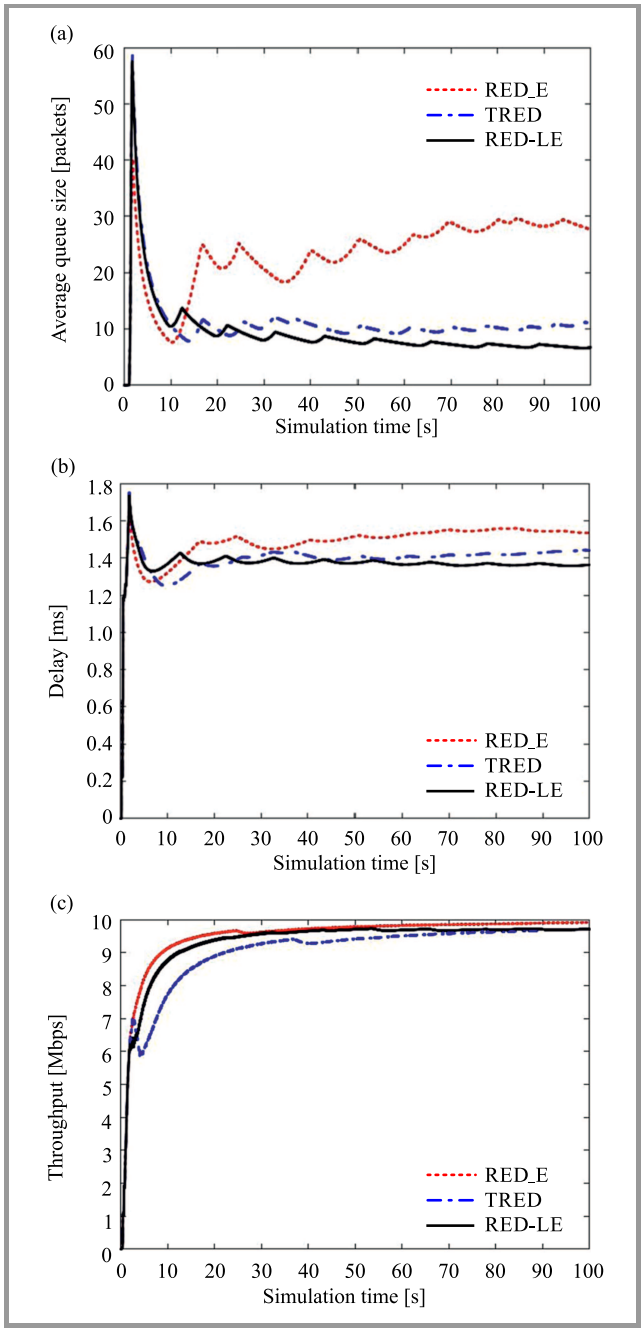

Fig. 3. Low load condition graphs: (a) average queue size, (b) delay, (c) throughput.

Table 2
Performance in terms of average queue size [packets]

Traffic load	AQM algorithm		
	TRED	RED_E	RED-LE
Low	11.1701	23.3786	9.2264
Moderate	16.3599	40.8509	11.1034
High	40.7549	62.0096	27.9556

in Table 3, RED-LE reduced the delay by 0.1777% and 0.8593% compared with TRED and RED_E, respectively.

Table 3
Performance in terms of delay [ms]

Traffic load	AQM algorithm		
	TRED	RED_E	RED-LE
Low	1.3918	1.4832	1.3692
Moderate	5.8040	6.4856	5.6263
High	15.2848	16.8606	14.3817

Figure 4c shows throughput performance. RED_E clearly obtained the highest value when compared with TRED and RED-LE. An analysis is presented in Table 4.

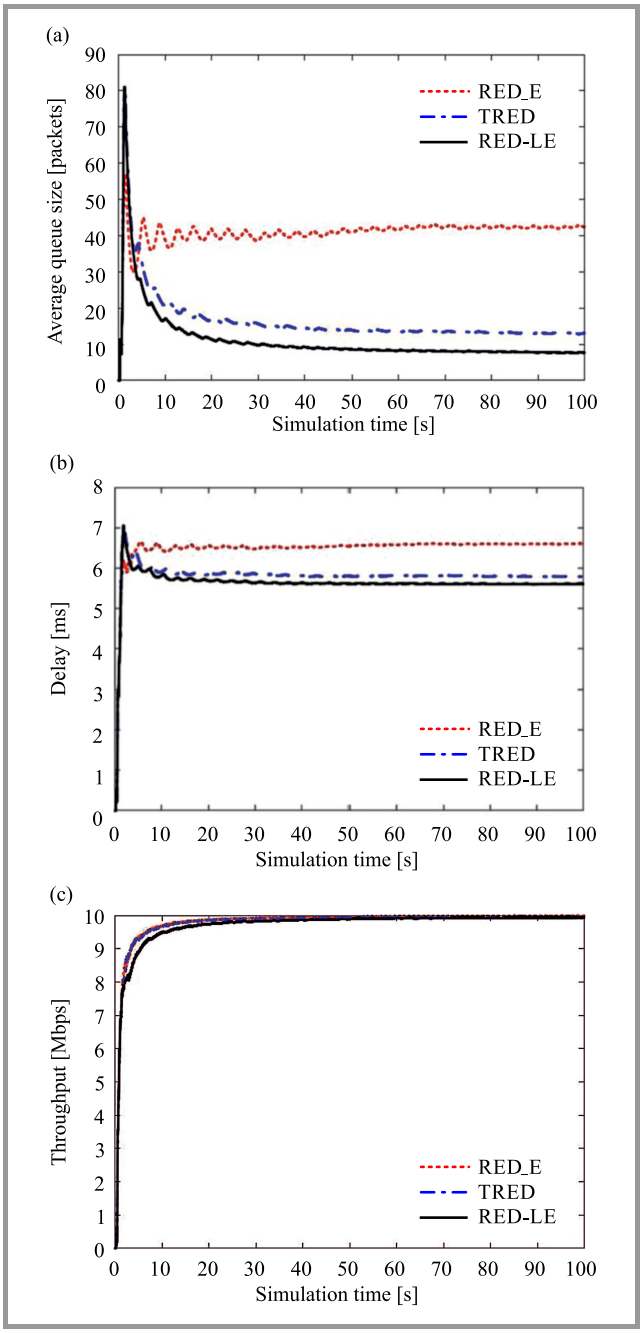

Fig. 4. Moderate load condition graphs: (a) average queue size, (b) delay, (c) throughput.

Table 4
Performance in terms of throughput [Mbps]

Traffic load	AQM algorithm		
	TRED	RED_E	RED-LE
Low	9.0105	9.4899	9.3054
Moderate	9.7709	9.7915	9.6858
High	9.7917	9.8696	9.5417

4.3. Scenario 3 – High Load

In this scenario, the number of connecting sources is set to 50. Again, as shown in Fig. 5a, the proposed RED-LE algorithm clearly outperformed both TRED and RED_E algorithms in terms of maintaining a small average queue size. As presented in Table 2, RED-LE reduced the queue size by 12.7993% and 34.0540% compared with TRED and RED_E, respectively.

The delay plot for the three algorithms is shown in Fig. 5b. RED-LE clearly obtained shorter delays than both TRED and RED_E. As presented in Table 3, RED-LE reduced them by 0.9031% and 2.4789% compared with TRED and RED_E, respectively.

Figure 5c shows throughput performance. RED_E obtained the highest value when compared with both TRED and RED-LE. A more detailed analysis is presented in Table 4.

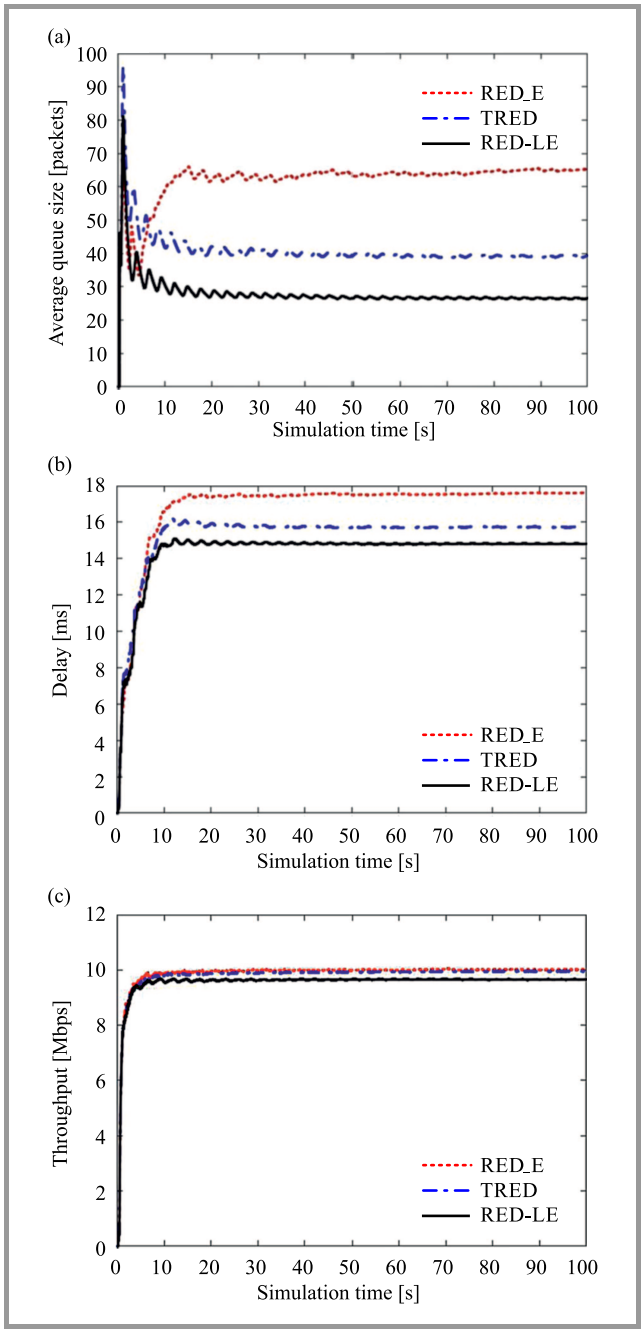

Fig. 5. High load condition graphs: (a) average queue size, (b) delay, (c) throughput.

5. Conclusion

In this study, a modest modification is introduced to the packet drop probability function of the RED algorithm. More specifically, a RED-linear exponential (RED-LE) algorithm was suggested for implementation in routers. A comparison was made between the RED-LE algorithm and two other AQM algorithms, namely TRED and RED_E, under various traffic load scenarios in a widely-used network simulator. From experimental results, it can be concluded that RED-LE ensures better and more efficient congestion control by obtaining reduced average queue size and delay values.

References

[1] A. A. Abu-Shareha, "Controlling delay at the router buffer using modified random early detection", *Int. J. of Comp. Netw. and Commun.* (*IJCNC*), vol. 11, no. 6, pp. 63–75, 2019 (DOI: 10.5121/ijcnc.2019.11604).

[2] S. B. Danladi and F. U. Ambursa, "DyRED: An enhanced random early detection based on a new adaptive congestion control", in *Proc. of the 15th Int. Conf. on Electron., Comp. and Comput. ICECCO 2019*, Abuja, Nigeria, 2019 (DOI: 10.1109/ICECCO 48375.2019.9043276).

[3] M. Baklizi, H. Abdel-Jaber, S. Ramadass, N. Abdullah, and M. Anbar, "Performance assessment of AGRED, RED and GRED congestion control algorithms", *Inform. Technol. J.*, vol. 11, no. 2, pp. 255–261, 2012 (DOI: 10.3923/itj.2012.255.261).

[4] A. Adamu, Y. Surajo, and M. T. Jafar, "SARED: Self-adaptive active queue management scheme for improving quality of service in network systems", *J. of Comp. Sci.*, vol. 22, no. 12, pp. 253–267, 2021 (DOI: 10.7494/csci.2021.22.2.4020).

[5] N. Kaur and R. Singhai, "Congestion control scheme using network coding with local route assistance in mobile adhoc network", *Int. J. of Comp. Appl. in Technol.*, vol. 60, no. 3, pp. 242–253, 2019 (DOI: 10.1504/ijcat.2019.100298).

[6] 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", *Int. J. of Bifurcation and Chaos*, vol. 31, no. 6, 2150124, 2021 (DOI: 10.1142/S0218127421501248).

[7] H. Mohammed, G. Attiya, and S. El-Dolil, "Active queue management for congestion control: Performance evaluation, new approach, and comparative study", *Int. J. of Comput. and Netw. Technol.*, vol. 5, no. 2, pp. 37–49, 2017 (DOI: 10.12785/IJCNT/050201).

[8] A. Ahmed and N. Nasrelden, "New congestion control algorithm to improve computer networks performance", in *Proc. of the 28th Int. Conf. on Innovat. Trends in Comp. Engin. ITCE 2018*, Aswan, Egypt, 2018, pp. 87–93 (DOI: 10.1109/ITCE.2018.8316605).

[9] H. Abdel-Jaber, A. Shehab, M. Barakat, and M. Rashad, "IGRED: An improved gentle random early detection method for management of congested networks", *J. of Intercon. Netw.*, vol. 19, no. 2, 1950004, 2019 (DOI: 10.1142/S021926591950004X).

[10] J. Aweya, M. Ouellette, and D. Y. Montuno, "A control theoretic approach to active queue management", *Comp. Netw.*, vol. 36, no. 2, pp. 203–235, 2001 (DOI: 10.1016/S1389-1286(00)00206-1).

[11] S. Floyd and V. Jacobson, "Random early gateway for congestion avoidance", *IEEE/ACM Trans. on Network.*, vol. 1, no. 4, pp. 397–413, 1993 (DOI: 10.1109/90.251892).

[12] A. Giménez, M. A. Murcia, J. M. Amigó, O. Martínez-Bonastre, and J. Valero, "New RE-type TCP-AQM algorithms based on beta distribution drop functions" [Online]. Available: https://arxiv.org/pdf/2201.01105.pdf

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

[14] H. Abdel-Jaber, "An exponential active queue management method based on random early detection", *J. of Comp. Netw. and Commun.*, vol. 2020, article ID 80904682020, 2020 (DOI: 10.1155/2020/8090468).

[15] Y. Zhang, J. Mab, Y. Wang, and C. Xu, "MRED: An improved nonlinear RED algorithm", in *Int. Proc. of Comp. Science and Inform. Technol.*, vol. 44, no. 2, pp. 6–11, 2012 (DOI: 10.7763/IPCSIT.2012.V44.2).

[16] A. K. Paul, H. Kawakami, A. Tachibana, and T. Hasegawa, "An AQM based congestion control for ENB RLC in 4G/LTE network", in *Proc. of the IEEE Canadian Conf. on Elec. and Comp. Engin. CCECE 2016*, Vancouver, BC, Canada, 2016 (DOI: 10.21109/CCECE.2016.7726792).

[17] C. Suwannapong and C. Khunboa, "Congestion control in CoAP observe group communication", *Sensors*, vol. 19, no. 3433, pp. 1–14, 2019 (DOI: 10.3390/s19153433).

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

[19] K. Zhou, K. L. Yeung, and V. O. K. Li, "Nonlinear RED: A simple yet efficient active queue management scheme", *Comp. Netw.*, vol. 50, pp. 3784–3794, 2006 (DOI: 10.1016/j.comnet.2006.04.007).

[20] D. Kumhar, A. Kumar, and A. Kewat, "QRED: An enhancement approach for congestion control in network communications", *Int. J. of Inform. Technol.*, vol. 13, pp. 221–227, 2021 (DOI: 10.1007/s41870-020-00538-1).

[21] 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, Moscow, Russia, September 14-18, 2020, Revised Selected Papers*, V. M. Vishnevskiy, K. E. Samouylov, and D. V. Kozyrev, Eds. *LNCS*, vol. 12563, pp. 196–208. Springer, 2020 (DOI: 10.1007/978-3-030-66471-8_16).

[22] "The Network Simulator ns-3" [Online]. Available: http://www.nsnam.org

Samuel O. Hassan received his M.Sc. and Ph.D. degrees in Computer Science from Obafemi Awolowo University, Ile-Ife, Nigeria. Currently, he is a Lecturer at the Department of Mathematical Sciences (Computer Science Unit), Olabisi Onabanjo University, Ago-Iwoye, Nigeria. He is a Certified Information Technology Practitioner (*C.itp*). His research interests spans computational mathematics, computer networks and communications, mathematical modeling and simulation, and Internet congestion control.

E-mail: samuel.hassan@oouagoiwoye.edu.ng
Department of Mathematical Sciences
Olabisi Onabanjo University
Ago-Iwoye, Nigeria