

## Random early detection-quadratic linear: an enhanced active queue management algorithm

Samuel Oluwatosin Hassan<sup>1</sup>, Vivian Ogochukwu Nwaocha<sup>2</sup>, Adewole Usman Rufai<sup>3</sup>, Tola John Odule<sup>1</sup>,  
Theophilus Aniemeka Enem<sup>4</sup>, Lukman Adebayo Ogundele<sup>1</sup>, Suleiman Abu Usman<sup>4</sup>

<sup>1</sup>Department of Mathematical Sciences, Olabisi Onabanjo University, Ago-Iwoye, Nigeria

<sup>2</sup>Department of Computer Science, National Open University, Abuja, Nigeria

<sup>3</sup>Department of Computer Sciences, University of Lagos, Akoka, Nigeria

<sup>4</sup>Department Cyber Security, Air Force Institute of Technology, Kaduna, Nigeria

### Article Info

#### Article history:

Received Apr 1, 2022

Revised May 16, 2022

Accepted June 16, 2022

#### Keywords:

Delay

Internet routers

RED

RED-QL

Simulation

### ABSTRACT

This paper identifies the lone linear drop function for computing the dropping probability between certain queue threshold values as a major weakness for the random early detection (RED) algorithm as it leads to large delay and queue instability. To address this concern, we propose an enhanced RED-based algorithm called random early detection-quadratic linear (referred to as "RED-QL") active queue management (AQM) which leveraged the benefit of a quadratic packet drop function for a light-to moderate traffic load conditions together with a linear packet drop function for a heavy traffic load condition respectively. Results from ns-3 network simulator using different experimental scenarios clearly reveals that the proposed RED-QL algorithm yields a substantial reduction in delay performance and indeed a reduced average queue size than other three representative AQM algorithms. RED-QL is robust, easy to implement and deploy in routers (both in software and hardware) as no more than the packet drop probability profile of the classic RED's algorithm implementation needs modest alteration.

This is an open access article under the [CC BY-SA](https://creativecommons.org/licenses/by-sa/4.0/) license.



### Corresponding Author:

Samuel Oluwatosin Hassan

Department of Mathematical Sciences, Olabisi Onabanjo University

Ago-Iwoye, Nigeria

Email: samuel.hassan@oouagoiwoye.edu.ng

## 1. INTRODUCTION

Network congestion is considered to mean a situation in which an increase in the volume of network data traffic flows arising from growth in the active number users deploying internet-based services and applications is greater than the inflowing bandwidth capacity of the device [1]–[4]. A buffer can simply be defined as a physical volume which stores a queue or even a set of queues [5]. Active queue management (AQM) algorithm, usually implemented in routers timely manages the buffer by dropping packets before it gets saturated in order to avoid overflow and packet loss unlike the traditional Drop-Tail algorithm which drop packets only when the buffer gets saturated thereby leading to lock-out phenomenon and global synchronization problems [6]–[9]. According to RFC 7928 [1], the aim of an AQM algorithm is two: i) to achieve a reduced average queue size, and ii) to accomplish a minimized (end-to-end) delay.

The most eminent AQM mechanism is the random early detection (RED) [10]. Essentially, at packet arrivals into the router, RED computes the average queue size (represented by *ave*) which is used as a measure of congestion based on whether the router's buffer is empty or not. If the packet arrives to an empty

queue, the idle time parameter (denoted  $m$ ) is computed, which is in turn used to calculate the average queue size as (1):

$$m = f(\text{time} - q\_idle\_time) \quad (1)$$

where,  $\text{time}$  is the current time;  $q\_idle\_time$  refers to the beginning of the queue idle time and  $f(t)$  is a linear function of time  $t$

$$ave = ave' \times (1 - w_q)^m \quad (2)$$

where,  $ave'$  refers to the computed previous value for average queue size and  $w_q \in [0, 1]$  is a pre-configured weight parameter to calculate  $ave$ . However, if the packet arrives to a non-empty queue, the average queue size is determined using a low-pass filter which is an exponential weighted moving average (EWMA):

$$ave = ave' \times (1 - w_q) + (w_q \times q) \quad (3)$$

where  $q$ , is the current queue size.

When  $ave$  is less than  $min_{Th}$  (queue minimum threshold), then no packet is dropped. When  $ave$  is higher than  $max_{Th}$  (queue maximum threshold), the packet is dropped with a probability of 1. However, if  $ave$  varies between  $min_{Th}$  and  $max_{Th}$ , then the packet is dropped with a linear drop function that grows from 0 to  $max_P$  (maximum drop probability at  $max_{Th}$ ). The actual dropping probability function ( $p_a$ ) for RED is given as:

$$p_b = \begin{cases} 0, & ave < min_{Th} \\ max_P \left( \frac{ave - min_{Th}}{max_{Th} - min_{Th}} \right), & min_{Th} \leq ave < max_{Th} \\ 1, & max_{Th} \leq ave \end{cases} \quad (4)$$

$$p_a = \frac{1}{1 - (count \times p_b)} \times p_b \quad (5)$$

in which:  $count$  indicates the number of arrived packets since the last dropped packet and  $p_b$  indicates the initial packet dropping probability function.

RED remains the most studied AQM [11]–[14], even till present [14]. A lot of AQM algorithms which leverages its simple model continues to evolve in literature [10]. Examples include: gentle RED (GRED) by [15], smart RED (SmRED) by [16], nonlinear RED (NLRED) by [11], SmRED-i by [17], Q-learning-based RED (QRED) by [18], modified RED (MRED) by [19], virtual queue RED (VQ-RED) by [20], hazard estimate red (HERED) by [21], RED-exponential (RED\_E) by [22], RED with reconfigurable maximum dropping probability (RRMDP) by [23], delay-controller RED (DcRED) by [24], three-section RED (TRED) by [25], multi RED by [26], adaptive drop-tail by [27], just to mention a few.

Zhou *et al.* in [11] proposed NLRED which improved throughput performance of RED through the introduction of a quadratic packet dropping function when  $ave$  varies from  $min_{Th}$  to  $max_{Th}$ . However, the  $max_P$  parameter was increased by a factor of 1.5. Floyd [15] developed gentle RED (GRED) algorithm which improved the throughput performance of RED algorithm through the introduction of an additional threshold ( $2max_{Th}$ ). When  $ave$  lies between  $max_{Th}$  and  $2max_{Th}$ , GRED linearly grows the packet dropping probability from  $max_P$  to 1 (as expressed by (6)).

$$p_b = max_P + (1 - max_P) \left( \frac{ave - min_{Th}}{max_{Th}} \right) \quad (6)$$

Zhang *et al.* [19] proposed MRED algorithm which replaced the linear packet dropping function utilized by GRED when  $ave$  varies from  $min_{Th}$  to  $max_{Th}$  with a quadratic drop function. MRED obtained an improved throughput performance. Abu-Shareha [24] developed the delay-controller RED (DcRED) algorithm which calculates a delay parameter for each packet as it arrives the router in order to estimate the dropping probability. In Exponential RED (RED\_E) algorithm proposed by Abdel-Jaber [22], the  $max_P$  parameter was eliminated and utilizes an exponential drop function when  $ave$  varies from  $min_{Th}$  to  $max_{Th}$ . The RED\_E's drop probability function is given by:

$$p_b = \begin{cases} 0, ave < min_{Th} \\ \left( \frac{e^{ave - min_{Th}}}{e^{max_{Th} - min_{Th}}} \right), min_{Th} \leq ave < max_{Th} \\ 1, max_{Th} \leq ave \end{cases} \quad (7)$$

Huang and feng [18] integrate the Q-learning algorithm into RED so as to adaptively determine  $max_P$  in order to achieve an improved link utilization. The improved algorithm was named QRED. Sun *et al.* [23] proposed an AQM named adaptive drop-tail which dynamically adjust the buffer size based on network traffic loads. This was done with the intention of reducing the rate of packet loss. Al-Allaf *et al.* [23] proposed the RRMDP (which stands for RED with reconfigurable max-inum dropping probability). The RRMDP algorithm achieved an improved delay through a reconfiguration parameter used to reconfigure  $max_P$  of RED. Abbasov and Korukoglu [21], the authors proposed a (nonlinear) hazard packet dropping function for RED. The algorithm was named HERED and aims at reducing the rate of packet loss. Lin *et al.* [20] developed virtual queue-RED (VQ-RED) which classifies flows into virtual queues which are in turn managed by VQ manager (VQM). The algorithm aimed at addressing the fairness problem of downlink or uplink in the infrastructure WLAN (wireless local area network).

Based on RED algorithm, Lu *et al.* [26] developed two algorithms for congestion control in core routers for a two-layer network. One algorithm implements fair link AQM while the other implements unequal link AQM. Feng *et al.* [25], the authors proposed three-section RED (TRED) algorithm which divides the segment between  $min_{Th}$  and  $max_{Th}$  of RED into three equal parts in order to obtain a trade-off between throughput and delay. The three sections utilizes a nonlinear drop function (expressed in (8)) when  $min_{Th} \leq ave < min_{Th} + \Delta$ ; a linear drop function (expressed in (9)) when  $min_{Th} + \Delta \leq ave < min_{Th} + 2\Delta$ , and a nonlinear drop function (expressed in (10)) when  $min_{Th} + 2\Delta \leq ave < max_{Th}$ .

$$p_b = 9max_P \left( \frac{ave - min_{Th}}{max_{Th} - min_{Th}} \right)^3 \quad (8)$$

$$p_b = max_P \left( \frac{ave - min_{Th}}{max_{Th} - min_{Th}} \right) \quad (9)$$

$$p_b = 9max_P \left( \frac{ave - max_{Th}}{max_{Th} - min_{Th}} \right)^3 + max_P \quad (10)$$

where,

$$\Delta = \left( \frac{max_{Th} - min_{Th}}{3} \right) \quad (11)$$

The smart RED (SmRED) algorithm was developed by scholars in [16]. SmRED utilizes a (nonlinear) quadratic and a linear packet dropping functions in order to obtain a trade-off between delay and throughput. The SmRED's drop probability function is given by:

$$p_b = \begin{cases} 0, ave < min_{Th} \\ max_P \left( \frac{ave - min_{Th}}{max_{Th} - min_{Th}} \right)^2, min_{Th} \leq ave < Target \\ max_P \sqrt{\frac{ave - min_{Th}}{max_{Th} - min_{Th}}}, Target \leq ave < max_{Th} \\ 1, max_{Th} \leq ave \end{cases} \quad (12)$$

where

$$Target = min_{Th} + \left( \frac{max_{Th} - min_{Th}}{2} \right) \quad (13)$$

Paul *et al.* [17] proposed an extension to SmRED and is therefore named SmRED-i where the packet dropping function can be tuned according to different values of parameter  $i$  with the aim of achieving a trade-off between delay and throughput. The SmRED-i's drop probability function is given by:

$$p_b = \begin{cases} 0, ave < min_{Th} \\ max_P \left( \frac{ave - min_{Th}}{max_{Th} - min_{Th}} \right)^i, min_{Th} \leq ave < Target \\ max_P \left( \frac{ave - min_{Th}}{max_{Th} - min_{Th}} \right)^{1/i}, Target \leq ave < max_{Th} \\ 1, max_{Th} \leq ave \end{cases} \tag{14}$$

where  $i = 2, 3, 4, 5, \dots$

The congestion control RED (CoCo-RED) algorithm suggested by Suwannapong and Khunboa in [28] utilizes a linear together with an exponential drop functions in order to enhance the performance of RED. The CoCo-RED's drop probability function is given by:

$$p_b = \begin{cases} 0, ave < min_{Th} \\ max_P \left( \frac{ave - min_{Th}}{max_{Th} - min_{Th}} \right), min_{Th} \leq ave < max_{Th} \\ ab^{ave}, max_{Th} \leq ave < K \end{cases} \tag{15}$$

where  $K$  is the buffer capacity;

$$a = \frac{max_P}{\frac{\ln(1/max_P)}{(e^{K-max_{Th}})^{max_{Th}}}} \tag{16}$$

and

$$b = e^{\frac{\ln(1/max_P)}{K-max_{Th}}} \tag{17}$$

Kumhar *et al.* in [29] developed a quadratic drop function (given in (18) and (19)) to replace the linear drop function utilized by RED in order to improve the queue stability of RED. The modified algorithm was called quadratic RED (QRED).

$$p_b = \left( \frac{ave - min_{Th}}{Q_{max} - min_{Th}} \right)^2 \tag{18}$$

Or

$$p_b = 1 - \left( \frac{Q_{max} - ave}{Q_{max} - min_{Th}} \right)^2 \tag{19}$$

where  $Q_{max}$  in these equations indicates the buffer size.

When  $min_{Th} \leq ave < max_{Th}$ , Patel and Karmeshu in [30] proposed a model which does not directly apply the linear dropping function of RED but goes further to compute a drop function as follows:

$$p_b = 1 - \frac{p_l(-\log(p_l))}{count+1} \tag{20}$$

where  $p_l$  refers to the linear drop probability function of RED. The modified algorithm was reported to outperformed both RED and TRED at heavy traffic state.

Using the approach of discrete-time queuing, the scholars in [31] proposed two analytical models i) RED-Exponential and ii) RED-linear. These two models are based on RED, however, the instantaneous queue length was employed for congestion measure. Recently, Giménez *et al.* [10] developed a beta distribution drop function to replace the linear drop function utilized by RED in order to provide a more stable average queue size. The modified algorithm was called BetaRED. It is interesting to note that AQM algorithms that utilizes nonlinear drop functions has been reported in literature to exhibit acceptable performance especially in terms of providing better stability of the queue size [14], [29].

Unlike many previous RED-based AQM algorithms, we propose a slight enhancement to RED while maintaining its original features. Specifically, the proposed random early detection-quadratic linear (RED-QL) algorithm leveraged the benefit of a nonlinear (that is, quadratic) together with a linear packet dropping functions to distinguish between light- to moderate and heavy network traffic loads respectively in lieu of a lone linear drop function utilized in RED. The quadratic packet dropping function ensures a reduced aggressive drop action which in turn provides an increased throughput performance at light- to moderate traffic load conditions.

The linear packet dropping function ensures an increased aggressive drop action, which in turn offers an improved delay performance at heavy traffic load.

## 2. THE PROPOSED RED-QL AQM ALGORITHM

The proposed AQM algorithm named RED-QL (de-picted in Figure 1) leverages on the simple packet dropping mechanism of RED, such that the router's queue interval between the  $min_{Th}$  and  $max_{Th}$  threshold values is subdivided into two regions: i) light- to moderate traffic load conditions; ii) heavy traffic load condition. The distinction between these two parts was achieved through the involvement of a new *Target* threshold, according to (21):

$$Target = 2 \left( \frac{max_{Th} + min_{Th}}{3} \right) - min_{Th} \quad (21)$$

Similar to RED algorithm operations, RED-QL first calculate *ave* by applying the mechanism of RED algorithm according to (1)-(3) needed for computing the drop probability. Accordingly, when *ave* is  $min_{Th}$  threshold value, no packet will be dropped. When *ave* lies between  $min_{Th}$  and *Target* threshold values (which implies that congestion in the network is not too serious for light- to moderate network traffic loads), the packet dropping function that slowly increase from 0 to  $max_p$  is described by a quadratic function:

$$p_b = 9max_p \left( \frac{ave - min_{Th}}{2(max_{Th} - 2min_{Th})} \right)^2 \quad (22)$$

When *ave* lies between *Target* and  $max_{Th}$  (which implies that the network is experiencing serious congestion - heavy network traffic load), the packet dropping function that grows rapidly from  $max_p$  to 1 is described by a linear function:

$$p_b = max_p + 3(1 - max_p) \left( \frac{ave - Target}{max_{Th} + min_{Th}} \right) \quad (23)$$

However, when *ave* is higher than  $max_{Th}$  threshold value, incoming packets are dropped with a probability of 1. RED-QL's  $p_b$  is given as follows. The RED-QL algorithm is provided in Algorithm 1.

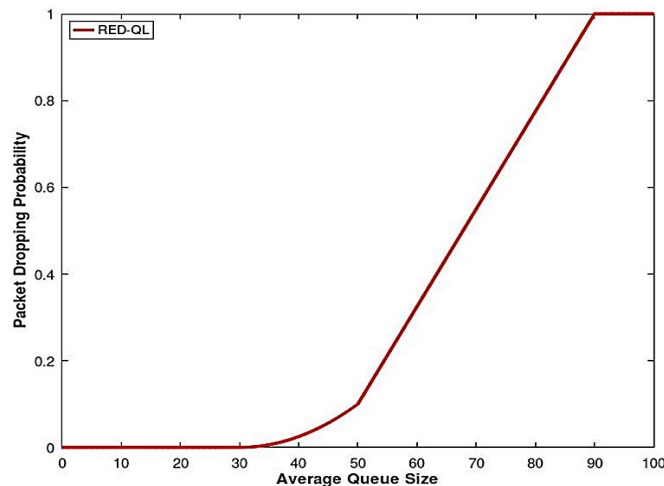


Figure 1. RED-QL's dropping function curve

$$p_b = \begin{cases} 0, ave < min_{Th} \\ 9max_p \left( \frac{ave - min_{Th}}{2(max_{Th} - 2min_{Th})} \right)^2, min_{Th} \leq ave < Target \\ max_p + 3(1 - max_p) \left( \frac{ave - Target}{max_{Th} + min_{Th}} \right), Target \leq ave < max_{Th} \\ 1, max_{Th} \leq ave \end{cases} \quad (24)$$

**Algorithm 1** RED-QL algorithm

1. Pre-configured parameters:  $min_{Th}$ ,  $max_{Th}$ ,  $max_P$ ,  $w_q$
2. Set  $ave = 0$ ,  $Target = 2 \left( \frac{max_{Th} + min_{Th}}{3} \right) - min_{Th}$
3. Upon each packet arrival do
4. Calculate average queue size  $ave$
5. **if** ( $ave < min_{Th}$ ) **then**
6. Enqueue incoming packet
7. **else if** ( $min_{Th} \leq ave < Target$ ) **then**
8. Compute the drop probability  $p_a$  (Eq. (5)) based on  $p_b$  (given in Eq. (22))
9. Drop arriving packet with the calculated probability
10. **else if** ( $Target \leq ave < max_{Th}$ ) **then**
11. Compute the drop probability  $p_a$  (Eq. (5)) based on  $p_b$  (given in Eq. (23))
12. Drop arriving packet with the calculated probability
13. **else if** ( $max_{Th} \leq ave$ ) **then**
14. Arriving packet is dropped
15. **end if**

### 3. RESULTS AND DISCUSSION

In this section, we implement RED-QL AQM algorithm using an open-source ns-3 network simulator [32]. The performance of RED-QL is evaluated and contrasted against three representative AQM algorithms namely: RED, NLRED, and SmRED under light, moderate, and heavy network traffic load scenarios. These scenarios are in line with the recommendation given in RFC 7948 [5].

#### 3.1. Simulation setup

For all our simulation scenarios considered in this article, the simulation setup (dumbbell topology depicted in Figure 2) used consists of  $N$  TCP sender nodes ( $S_1$  to  $S_N$ ), one router, and one receiver node – R. All TCP senders are connected to the router with a link capacity of 100 Mbps and a propagation delay time of 5 ms. The bottleneck link, which lies between the router and R has a link capacity of 10 Mbps and a propagation delay time of 100 ms. Packet size is configured as 1000 bytes. The buffer size is specified as 250 packets. TCP NewReno implementation was used. Simulation time is set as 100 seconds. The number of senders is varied to yield different levels of congestion on the bottleneck link. All our simulation experiments uses the following input parameter values:  $min_{Th} = 30$  packets,  $max_{Th} = 90$  packets,  $max_P = 0.1$ , and  $w_q = 0.002$ .

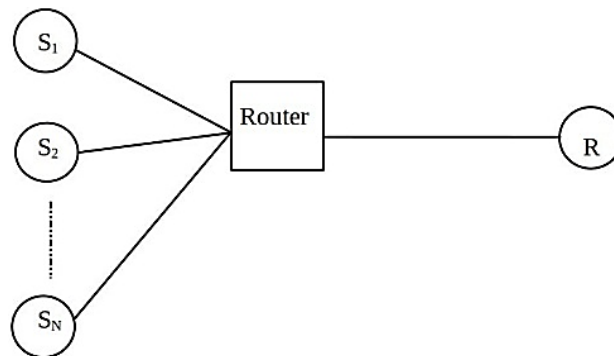


Figure 2. Simulation network topology

#### 3.2. Simulation 1: light TCP traffic scenario

In this scenario, we assess the performance of RED, NLRED, SmRED and RED-QL under a light congestion situation. In particular, the number of TCP flows are 5 (similar to those reported in [33]–[35]). The average queue size is plotted in Figure 3(a). We observe that the initial burst in RED reaches 36.62, NLRED goes up to 36.93, SmRED reaches 37.52 while RED-QL goes up to 29.80. So, RED-QL performs better than other AQM algorithms in controlling the burst. As Table 1 shows, it is clear that RED-QL substantially controls the queue size better than RED, NLRED, and SmRED algorithms. Thus, the queue size is stabilized better with RED-QL.

Figure 3(b) plots the delay for RED, NLRED, SmRED and RED-QL algorithms. The analysis is presented in Table 2. It is evidently clear that RED-QL offers an acceptable performance in terms of delay when compared with other AQM algorithms.

Figure 3(c) presents the throughput plot for the four algorithms under consideration. The analysis is further provided in Table 3. It can be observed that NLRED achieved the highest throughput when compared with others. Moreover RED-QL slightly performs better than SmRED. This is because RED-QL improves the delay at the expense of throughput under this scenario.

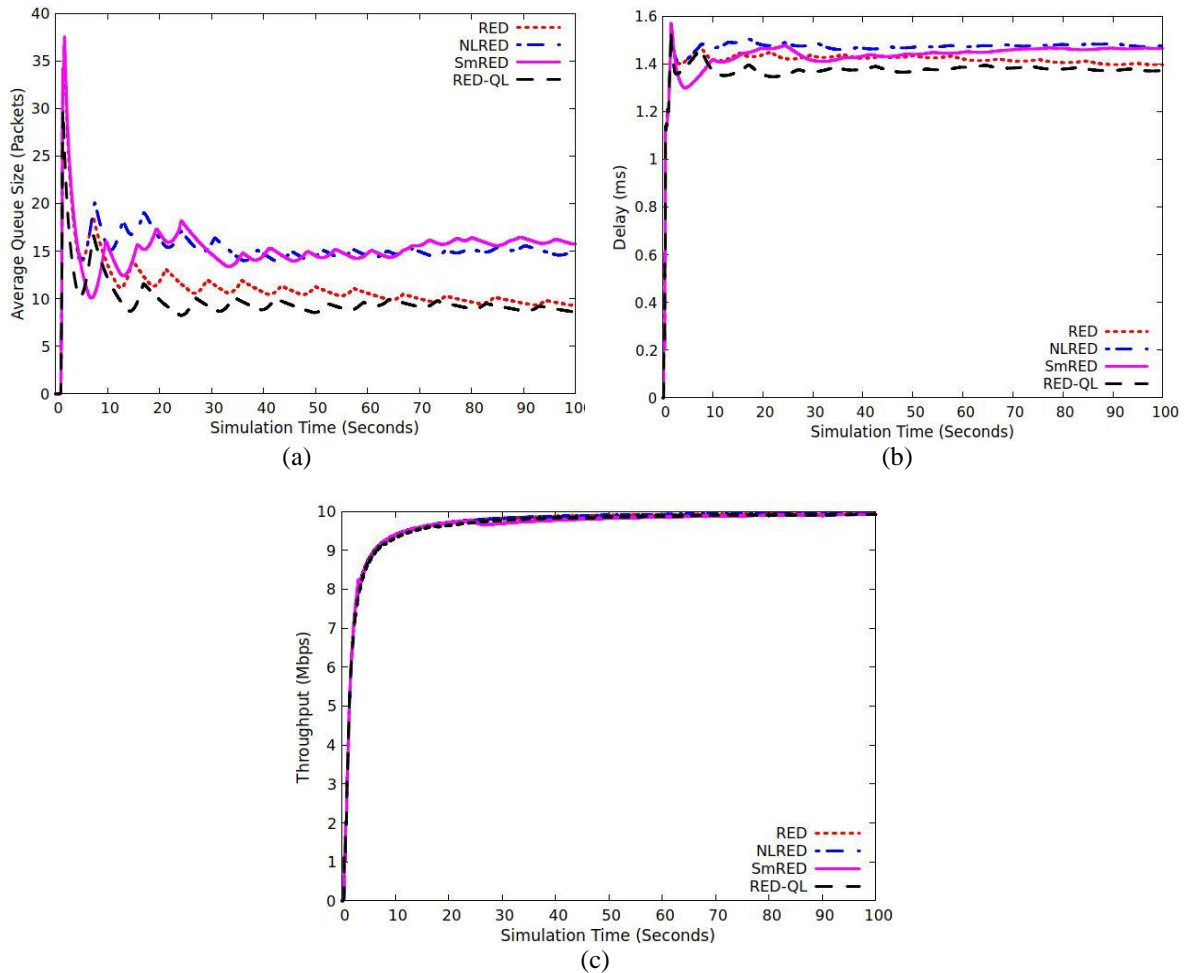


Figure 3. Light traffic condition (a) average queue size graph, (b) delay graph, and (c) throughput graph

Table 1. Performance comparison of average queue size (packets)

	RED	NLRED	SmRED	RED-QL
Light	11.22	15.38	15.19	9.68
Moderate	20.92	24.38	22.02	15.20
Heavy	39.77	36.75	39.33	21.74

Table 2. Performance comparison of delay (ms)

	RED	NLRED	SmRED	RED-QL
Light	1.41	1.46	1.43	1.37
Moderate	5.73	5.78	5.61	5.41
Heavy	14.91	14.56	14.81	13.56

Table 3. Performance comparison of throughput (Mbps)

	RED	NLRED	SmRED	RED-QL
Light	9.63	9.63	9.58	9.59
Moderate	9.73	9.72	9.62	9.31
Heavy	9.86	9.74	9.85	9.35

### 3.3. Simulation 2: moderate TCP traffic scenario

In this scenario, we increase the number of TCP connections to 20 to indicate a moderate congestion situation. Figure 4(a) plots the average queue size for RED, NLRED, SmRED, and RED-QL algorithms under a moderate congestion situation. The burst noticed at the early stage in RED goes up to 56.80, NLRED reaches to 56.70, SmRED goes up to 56.84 while RED-QL reaches to 48.00. Clearly, RED-QL exhibits better performance in burst control. Further analysis of the instantaneous average queue size is presented in Table 1. It can be observed from the table that RED-QL obtained the lowest value. This simply goes to show that RED-QL reduces and stabilizes the queue size better than RED, NLRED, and SmRED algorithms under this scenario.

Figure 4(b) depicts the delay plot for RED, NLRED, SmRED, and RED-QL algorithms under consideration. The analysis is presented in Table 2. It can be observed the RED-QL algorithm still outperforms other AQM algorithms. Figure 4(c) depicts the throughput plot for the four algorithms under consideration. It can be seen in Table 3 that RED-QL obtained the least value when compared with RED, NLRED, and SmRED algorithms. Again, this result implies that RED-QL offers an improved delay performance at the expense of throughput under this scenario.

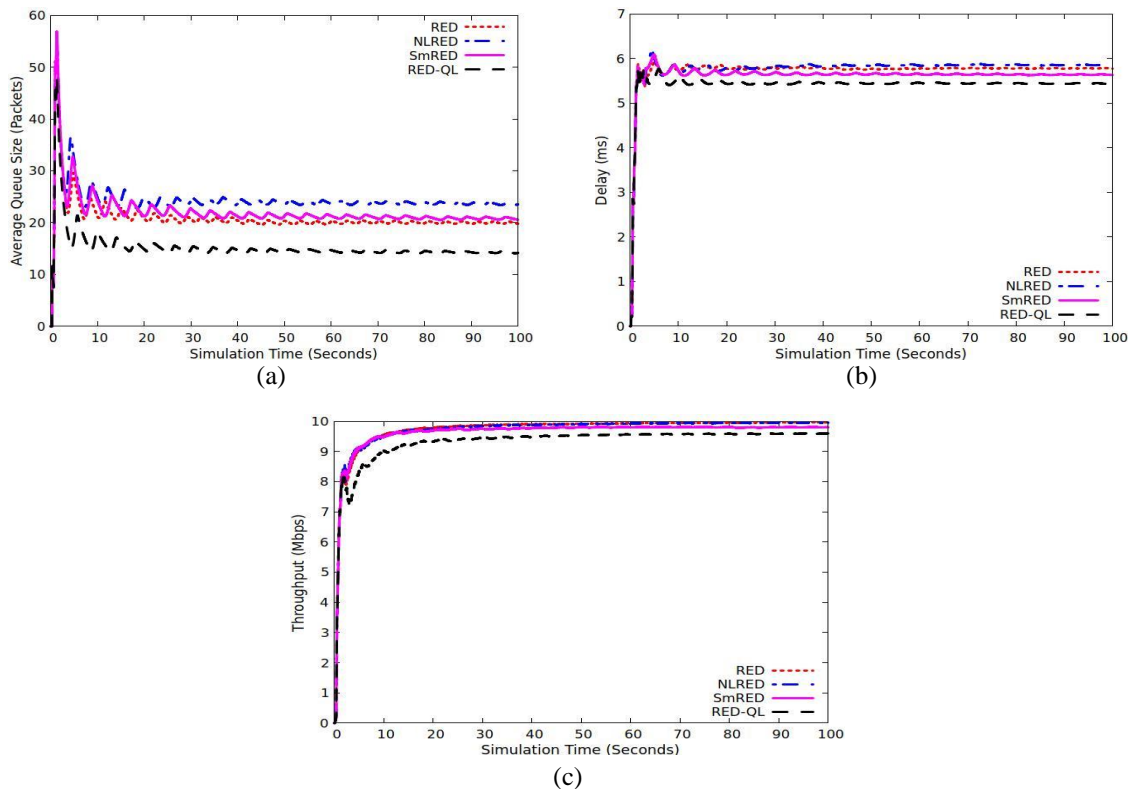


Figure 4. Moderate traffic condition (a) average queue size graph, (b) delay graph, and (c) throughput graph

### 3.4. Simulation 3: heavy TCP traffic scenario

In this scenario, we further increase the number of TCP connections to 50 (similar to those reported in [33]–[35]) to indicate a heavy congestion situation. Figure 5(a) indicates the average queue size for RED, NLRED, SmRED, and RED-QL algorithms. A high initial upsurge is noticed at the beginning for all the algorithms. RED got a peak of 60.89, NLRED goes up to 61.28, SmRED goes up to 65.82 while RED-QL got 51.72. Still, the proposed RED-QL algorithm obtained the least initial peak of the burst. Therefore, compared to RED, NLRED, and SmRED algorithms, RED-QL provides a significant reduction of the average queue size as shown in Table 1. This is because at heavy congestion situation, RED-QL utilizes a higher packet dropping probability to ease congestion than other three algorithms.

Figure 5(b) depicts the delay for RED, NLRED, SmRED, and RED-QL algorithms. Furthermore, Table 2 provides the analysis. It can be observed that RED-QL outperformed others by achieving the least value. This is because RED-QL is more aggressive at dropping packets particularly with heavy traffic load condition when network congestion is serious. Figure 5(c) depicts the throughput performance. As presented in the analysis of Table 3, RED offers a significant throughput result than other three algorithms.



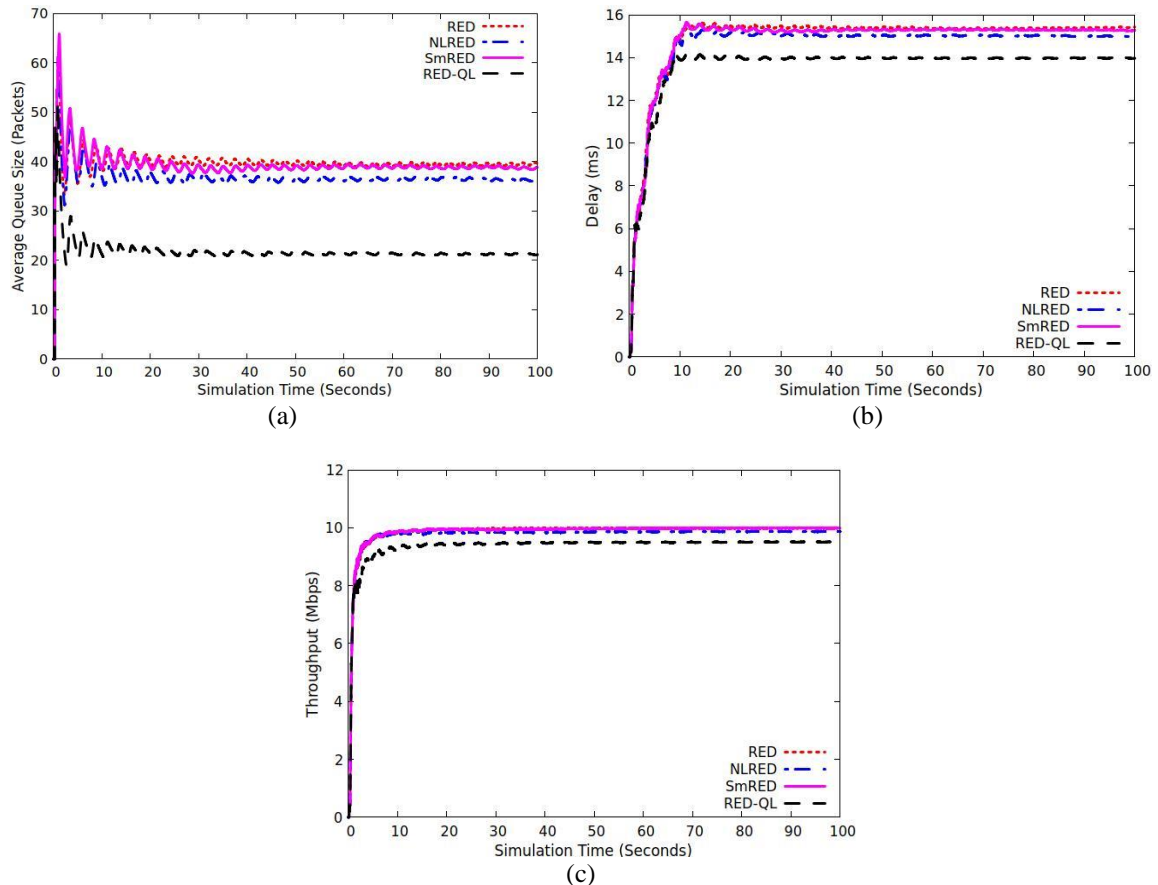


Figure 5. Heavy traffic condition (a) average queue size graph, (b) delay graph, and (c) throughput graph

#### 4. CONCLUSION

In this article, we have described a new AQM algorithm named RED-QL, a RED-based algorithm with an improved packet dropping probability function. Accordingly, RED-QL utilizes a quadratic together with a linear packet dropping functions to distinguish between light-to-moderate and heavy traffic loads respectively. RED-QL was shown to substantially outperform three existing algorithms namely RED, NLRED, and SmRED algorithms in terms of end-to-end delay by keeping the average queue size small at the expense of throughput in all simulation experiments conducted in an open-source network simulation environment. In our future work, we intend to implement RED-QL for active queue management in the radio link control (RLC) layer of long term evolution (LTE) cellular networks.




#### REFERENCES

- [1] 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, p. 5137, Dec. 2021, doi: 10.11591/ijece.v11i6.pp5137-5143.
- [2] 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*, vol. 19, no. 1, p. 241, Jul. 2020, doi: 10.11591/ijeecs.v19.i1.pp241-247.
- [3] 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, p. 6225, Dec. 2020, doi: 10.11591/ijece.v10i6.pp6225-6232.
- [4] J. Shi, Y. B. Leau, K. Li, and J. H. Obid, "A comprehensive review on hybrid network traffic prediction model," *International Journal of Electrical and Computer Engineering (IJECE)*, vol. 11, no. 2, p. 1450, Apr. 2021, doi: 10.11591/ijece.v11i2.pp1450-1459.
- [5] D. Ros, "Characterization guidelines for active queue management (AQM)," Jul. 2016. doi: 10.17487/RFC7928.
- [6] E. Natsheh, A. B. Jantan, S. Khatun, and S. Subramaniam, "Fuzzy active queue management for congestion control in wireless ad-hoc," *The International Arab Journal of Information Technology*, vol. 4, no. 1, pp. 50–59, 2007.
- [7] F. Baker and G. Fairhurst, "IETF recommendations regarding active queue management," Jul. 2015. doi: 10.17487/RFC7567.
- [8] B. Braden *et al.*, "Recommendations on Queue Management and Congestion Avoidance in the Internet," Apr. 1998. doi: 10.17487/rfc2309.
- [9] 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.
- [10] 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.
- [11] 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.
- [12] Z. Albayrak and M. Çakmak, “A Review: Active queue management algorithms in mobile communication,” *International Conference on Cyber Security and Computer Science*, no. October 2018, pp. 180–184, 2018.
- [13] J. M. Amigó, A. Giménez, O. Martínez-Bonastre, and J. Valero, “Internet congestion control: from stochastic to dynamical models,” *Stochastics and Dynamics*, vol. 21, no. 03, p. 2140009, May 2021, doi: 10.1142/S0219493721400098.
- [14] A. Giménez, M. A. Murcia, J. M. Amigó, and ..., “New RED-type TCP-AQM algorithms based on beta distribution drop functions,” *arXiv preprint arXiv ...*, no. Query date: 2022-03-12 00:05:45, 2022, doi: 10.48550/arXiv.2201.01105.
- [15] S. Floyd, “Recommendation on using the gentle variant of RED,” 2000. <http://www.icir.org/floyd/red/gentle.html>.
- [16] 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, vol. 2016-October, pp. 1–5, doi: 10.1109/CCECE.2016.7726792.
- [17] A. Paul, H. Kawakami, A. Tachibana, and T. Hasegawa, “Effect of AQM-based RLC buffer management on the eNB scheduling algorithm in LTE network,” *Technologies*, vol. 5, no. 3, p. 59, Sep. 2017, doi: 10.3390/technologies5030059.
- [18] 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.
- [19] Y. Zhang, J. Ma, Y. Wang, and C. Xu, “MRED: an improved nonlinear RED algorithm,” vol. 44, no. Iccae 2011, pp. 6–11, 2012.
- [20] X. Lin, X. Chang, and J. K. Muppala, “VQ-RED: an efficient virtual queue management approach to improve fairness in infrastructure WLAN,” in *The IEEE Conference on Local Computer Networks 30th Anniversary (LCN'05)*, 2005, vol. 2005, pp. 7 pp. – 638, doi: 10.1109/LCN.2005.136.
- [21] B. Abbasov and S. Korukoğlu, “An active queue management algorithm for reducing packet loss rate,” *Mathematical and Computational Applications*, vol. 14, no. 1, pp. 65–72, Apr. 2009, doi: 10.3390/mca14010065.
- [22] 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.
- [23] A. F. AL-Allaf *et al.*, “RED with reconfigurable maximum dropping probability,” *International Journal of Computing and Digital Systems*, vol. 8, no. 1, pp. 61–72, Jan. 2019, doi: 10.12785/ijcds/080107.
- [24] 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.
- [25] 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.
- [26] L. Lu, Y. Xiao, S. Woo, and K. Kim, “Nonlinear AQM for multiple RED routers,” in *2008 Third International Conference on Convergence and Hybrid Information Technology*, Nov. 2008, vol. 2, pp. 122–127, doi: 10.1109/ICCIT.2008.68.
- [27] J. Sun, K.-T. Ko, G. Chen, S. Chan, and M. Zukerman, “Adaptive drop-tail: a simple and efficient active queue management algorithm for internet flow control,” 2003, pp. 1261–1270.
- [28] Suwannapong and Khunboa, “Congestion control in CoAP observe group communication,” *Sensors*, vol. 19, no. 15, p. 3433, Aug. 2019, doi: 10.3390/s19153433.
- [29] 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.
- [30] 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.
- [31] 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 Computing*, vol. 06, no. 03, p. 1550028, Sep. 2015, doi: 10.1142/S1793962315500282.
- [32] N. S. Foundation, “The network simulator - ns-3,” 2012. <http://www.nsnam.org/>.
- [33] R. Pan *et al.*, “PIE: a lightweight control scheme to address the bufferbloat problem,” in *2013 IEEE 14th International Conference on High Performance Switching and Routing (HPSR)*, Jul. 2013, pp. 148–155, doi: 10.1109/HPSR.2013.6602305.
- [34] K. De Schepper, O. Bondarenko, I.-J. Tsang, and B. Briscoe, “PI<sup>2</sup>: a linearized AQM for both classic and scalable TCP,” in *Proceedings of the 12th International Conference on emerging Networking EXperiments and Technologies*, Dec. 2016, pp. 105–119, doi: 10.1145/2999572.2999578.
- [35] V. Jain, V. Mittal, S. K. S., and M. P. Tahiliani, “Implementation and validation of BLUE and PI queue disciplines in ns-3,” *Simulation Modelling Practice and Theory*, vol. 84, pp. 19–37, May 2018, doi: 10.1016/j.simpat.2018.01.002.

## 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 spans computational mathematics, computer networks and communications, Mathematical modeling and simulation, and Internet congestion control. He can be contacted at email: samuel.hassan@oouagoiwoye.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 email: onwaocha@noun.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 started his career in academia at Olabisi Onabanjo University, Ago-Iwoye as a Graduate Assistant (2000-2002), Assistant Lecturer (2002-2005), and Lecturer II (2005-2006). He later joined the University of Lagos in 2006 and rose through the ranks to become a Senior Lecturer. Dr. Rufai 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 email: arufai@unilag.edu.ng






**Tola John Odule**    had his Ph.D. (Computer Science) from Babcock University Ilishan-Remo, Ogun State, Nigeria in 2016. He is currently a Senior Lecturer in the Department of Mathematical Sciences (Computer Science Unit), Olabisi Onabanjo University, Ago-Iwoye. His research interests are mainly related to applied cryptography/computer and information security, distributed algorithms and secure multiparty computation, with collaborations in data mining and machine learning-based security protocols. He has quite a number of publications in peer-reviewed international and local journals. He can be contacted at email: tola.odule@oouagoiwoye.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 email: ta.enem@afit.edu.ng



**Lukman Adebayo Ogundele**    obtained his Ph.D. (in Computer Science) from Federal University of Technology, Akure, Nigeria. Currently, He is a Lecturer in the Department of Mathematical Sciences (Computer Science Unit), Olabisi Onabanjo University, Ago-Iwoye, Nigeria. His research interests are cyber security and information technology policy. He can be contacted at email: ogundele.lukman@oouagoiwoye.edu.ng



**Suleiman Abu Usman**    holds an MSc in Information Communication Technology from the National Open University, Kaduna, Nigeria. He has several years' experience of teaching Computer Science and cyber security courses at the Airforce Institute of Technology, Kaduna and has previously worked with Budget and Planning Commission as a Principal Data Processing Officer. He is a full member of the Nigeria Computer Society (NCS). His areas of interest are machine learning, cyber security, and software engineering. He can be contacted at email: usman.sule@afit.edu.ng