International Journal of Computing and Digital Systems
ISSN (2210-142X)
Int. J. Com. Dig. Sys.14, No.1 (Jul-23)

http://dx.doi.org/10.12785/ijcds/140117

A Triplex Region - Random Early Detection (TR-RED) Algorithm for Active Queue Management in Internet Routers

Samuel O. Hassan[1], Olakunle O. Solanke[1], Chigozirim Ajaegbu[2], Tola J. Odule[1], Khadijha-Kuburat A. Abdullah[1], Oyebola Akande[3] and Semiu A. Ayinde[4]

[1]*Department of Mathematical Sciences, Olabisi Onabanjo University, Ago-Iwoye, Nigeria*
[2]*School of Computing & Engineering Sciences, Babcock University, Ilishan-Remo, Nigeria*
[3]*Department of Computer Science, Babcock University, Ilishan-Remo, Nigeria*
[4]*Department of Basic Sciences, Babcock University, Ilishan-Remo, Nigeria*

Received 17 Aug. 2022, Revised 6 May. 2023, Accepted 8 May. 2023, Published 1 Jul. 2023

Abstract: Internet traffic is growing at an explosive proportion due to ever-rising innovations in technology. Consequently, the massive growth in the number of Internet users has also led to the problem of network congestion. When congestion occurs, the quality of network service can no longer be guaranteed. One of the widely-known active queue management (AQM) algorithms performed in Internet router is the Random Early Detection (RED). RED itself certainly has a deficiency of applying same linear drop function for varying network loads, resulting in poor quality of network service. This paper presents a new type of amendment to RED, named as Triplex Region - Random Early Detection (TR-RED) algorithm to cope with the mentioned shortcoming. In TR-RED, three dropping functions are implemented as an alternative to RED's one and only linear drop function. Differently from RED, the working principle for the proposed TR-RED aims at providing different drop action (that is to say square, linear and exponential) for unique levels of congestion in the network. Moreover, using ns-3 network simulator, we conduct three simulation experiments to benchmark TR-RED with two known enhanced RED-based AQM algorithms under three distinctive scenarios. Simulation results clearly indicated that TR-RED is a promising algorithm which offers a surpassing and indeed remarkable performance gain in delay across all scenarios considered. Interestingly, to upgrade from original RED to the proposed TR-RED only involves simple implementation effort. This is because only little modification is needed in the dropping profile of RED's algorithm implementation.

Keywords: Dropping function, Network congestion, Random early detection, Simulation, TR-RED AQM algorithm

1. Introduction

The term "congestion" in the framework of computer networks and communications has been described as a situation whereby the amount of data traffic transmitted over a network transcends the usable buffer capability of the network's resource ([1]-[?]).Congestion is a weighty issue for scientist in communications system and indeed computer networks. Some of the disintegrating effects of congestion in a network includes: heightened delay, reduced throughput, higher packet loss rate, large average queue size and even network collapses, meaning a situation whereby nearly no packet is delivered to the destination ([?], [?]- [?]).

Generally speaking, there are two well-known approaches for tackling the problem of network congestion. One is end-to-end control mechanism which is executed by Transport Control Protocol (TCP). The other is network-assisted control mechanism which is executed in intermediate machines (that is to say routers) ([?], [?]-[?]). However, implementing congestion control at the router has been adjudged as most effective due to its pivotal responsibility in the network ([?], [?], [?], [?], [?]).

Furthermore, congestion control techniques on routers can be classified into two main categories. One is Passive Queue Management (hereafter referred to as "PQM") algorithm, while the other class is Active Queue Management (hereafter referred to as "AQM") algorithm. One important responsibility of these algorithms is to perform the role of queuing management by either accepting or rejecting an incoming packet at router's buffer [?].

The simple DropTail queuing is a full-featured representative of PQM. This algorithm rejects incoming packet at the tail-end of a buffer once the full capacity is reached without any congestion notice to transmitting sources. Contrasting to DropTail queuing, AQM simply refers to router-based algorithms which detects congestion at its inception and then sends an early warning signal (by-way-of packet dropping) to transmitting end-sources before the router's queue is filled-up. With this, end-sources better controls their sending rates ([?], [?]-[?]).AQM algorithms must seek

E-mail address: {*samuel.hassan, solanke.olakunle, tola.odule, abdullah.adebisi*}*@oouagoiwoye.edu.ng*, {*ajaegbuc, akandeo, ayindes*}*@babcock.edu.ng*

to accomplish an important goal: shortening the average queue size, and correspondingly, the end-to-end delay is kept minimal [?].

The better-known Random Early Detection (hereafter referred to as "RED") algorithm was introduced by scientist in [?]. The 30-year-old RED has gained prominence as a foremost AQM algorithm which has inspired several scientists to develop newer and more robust AQM algorithms. RED is composed of two phases. The first phase involves determining the status of congestion by-means-of computing a congestion indicator - average queue size (hereafter referred to as "Q_{ave}") by using an exponential weighted moving average (EWMA) function - Eq. 1:

$$Q_{ave} = (1 - w_q) \times Q'_{ave} + (w_q \times q) \tag{1}$$

in which:
Q'_{ave} identifies the Q_{ave} computed previously, $w_q \in [0, 1]$ refers to the averaging weight and q identifies the current queue size.

The second phase involves determining whether to enqueue or reject the packet. Mathematically, the working principle for RED can be expressed as follows:

$$P_b = \begin{cases} 0, & \text{if } Q_{ave} < TH^{min} \\ P_{max} \times (\frac{Q_{ave} - TH^{min}}{TH^{max} - TH^{min}}), & \text{if } TH^{min} \leq Q_{ave} < TH^{max} \\ 1, & \text{if } Q_{ave} \geq TH^{max} \end{cases} \tag{2}$$

where:
TH^{min} and TH^{max} identifies the minimum and maximum queue size thresholds respectively; P_{max} indicates the maximum dropping probability at TH^{max} and P_b identifies the initial dropping probability:

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

where:
P_a signifies the final packet drop probability and *count* signifies the number of undropped packets since the last dropped packets.

Eq. 2 implies that: (i) RED drops no packet when Q_{ave} is below TH^{min}, (ii) randomly drops packet with a probability obtained using a singly linear function when Q_{ave} is below TH^{max} but is above TH^{min}, and (iii) drops packet with a probability of one when Q_{ave} is larger than TH^{max}.

There has been an increase in scholarly publications which investigates the performance of RED with a view to developing a more effective algorithm to address network congestion. Past studies include Linear RED (LRED) [?], Exponential RED (RED_E) [?], BetaRED [?], Nonlinear RED (NLRED) [?], Smart RED (SmRED) [?], RED-Quadratic Linear (RED-QL) [?], Three section RED (TRED) [?], Gentle RED (GRED) [?], Modified Dropping RED [?], Quadratic RED (QRED) [?], RED-Linear Exponential (RED-LE) [?], Average queue length and change rate-RED (AC-RED) [?], Quadratic Exponential RED (QERED) [?], Double Slope RED (DSRED) [?], to name but a few.

These mentioned algorithms has attempted to provide different drop functions to take action on varying levels of network congestion, however, have not achieved satisfactory results. In this paper, a new RED-based variant (with improved dropping mechanism) named as Triplex Region - Random Early Detection (TR-RED), is proposed to offer a desirable and efficient congestion control for routers.

The remaining part of this article is structured as follows. In Section 2, related works are discussed. Section 3 is devoted to description of the proposed algorithm. Section 4 presents simulation results. Finally, Section 5 presents the conclusion.

2. Related Works

Differing from RED, LRED [?] computes and utilizes an adaptive estimated average queue size as congestion measure. This approach was claimed to better simplify the mathematical formula for computing Q_{ave}. The algorithm also goes further to utilize a linear drop function (albeit with a reduced computational cost). LRED offers lower dropping rate.

NLRED [?] computes its dropping probability by singly using a square function when Q_{ave} lies between TH^{min} and TH^{max}. This alternate method yields a superior throughput performance than RED. To summarize:

$$P_b = \begin{cases} 0, & \text{if } Q_{ave} < TH^{min} \\ P'_{max} \times (\frac{Q_{ave} - TH^{min}}{TH^{max} - TH^{min}})^2, & \text{if } TH^{min} \leq Q_{ave} < TH^{max} \\ 1, & \text{if } Q_{ave} \geq TH^{max} \end{cases} \tag{4}$$

in which

$$P'_{max} = 1.5 \times P_{max} \tag{5}$$

To support different traffic loads, SmRED [?] involves the composite of two drop functions. SmRED computes dropping probability for low load using a square function. However, for high loads, dropping probability was computed using a square root function. SmRED demonstrated a sensible trade-off in throughput and delay admirable metrics. To summarize:

$$P_b = \begin{cases} 0, & \text{if } Q_{ave} < TH^{min} \\ P_{max} \times (\frac{Q_{ave} - TH^{min}}{TH^{max} - TH^{min}})^2, & \text{if } TH^{min} \leq Q_{ave} < Target \\ P_{max} \sqrt{\frac{Q_{ave} - TH^{min}}{TH^{max} - TH^{min}}}, & \text{if } Target \leq Q_{ave} < TH^{max} \\ 1, & \text{if } Q_{ave} \geq TH^{max} \end{cases} \tag{6}$$

in which

$$Target = TH^{min} + (\frac{TH^{max} - TH^{min}}{2}) \tag{7}$$

In RED_E [?], a nonlinear (that is to say exponential) function was singly employed as a choice drop function

when Q_{ave} lies between TH^{min} and TH^{max}. This enhanced approach effectively leaves out the requirement for P_{max} parameter overhead in deciding the dropping probability. To summarize:

$$P_b = \begin{cases} 0, & \text{if } Q_{ave} < TH^{min} \\ (\frac{e^{Q_{ave}} - e^{TH^{min}}}{e^{TH^{max}} - e^{TH^{min}}}), & \text{if } TH^{min} \leq Q_{ave} < TH^{max} \\ 1, & \text{if } Q_{ave} \geq TH^{max} \end{cases} \quad (8)$$

BetaRED [**?**] is an improved enhancement to RED. In BetaRED, beta distribution function was singly employed as an alternative to the linear drop function between TH^{min} and TH^{max}. BetaRED offered better stability.

The underlying idea behind RED-QL [**?**] is that it combines both squaring and linear dropping functions as an alternative to the lone linear drop function applied in RED, particularly between the TH^{min} - TH^{max} operating region. RED-QL attained a lower delay metric.

Also to support different traffic loads, TRED [**?**] utilizes the composite of three functions for the computation of the dropping probability. In particular, nonlinear function (with an exponent of 3 - Eq. 9) was used for low load, a linear function - Eq. 10 was used for a moderate load. Lastly, another nonlinear function (with an exponent of 3 - Eq. 11) was used for high load.

$$P_b = 9P_{max}(\frac{Q_{ave} - TH^{min}}{TH^{max} - TH^{min}})^3 \quad (9)$$

$$P_b = \frac{P_{max} \times (Q_{ave} - TH^{min})}{(TH^{max} - TH^{min})} \quad (10)$$

$$P_b = P_{max} + 9P_{max}(\frac{Q_{ave} - TH^{max}}{TH^{max} - TH^{min}})^3 \quad (11)$$

GRED [**?**], a newer variant of RED involves the addition of an extra threshold: $2 \times TH^{max}$. In GRED, a linear drop function is followed immediately by another linear function. GRED computes the needed dropping probability by using Eq. 12 when Q_{ave} lies between TH^{max} and $2 \times TH^{max}$. GRED maintains the same drop function when Q_{ave} lies between TH^{min} and TH^{max}. The algorithm obtained a higher throughput.

$$P_b = P_{max} + (1 - P_{max})(\frac{Q_{ave} - TH^{min}}{TH^{max}}) \quad (12)$$

Researchers in [**?**] introduced another modification to RED in order improve its delay. Specifically, RED was not allowed to expressly drop packets after computing the dropping probability by using the linear function. The modified algorithm involves one more step. This is explained as follows:

$$p_b = 1 - \left[P_1 \times \frac{-log(p_1)}{count + 1}\right] \quad (13)$$

in which

$$P_1 = \frac{P_{max} \times (Q_{ave} - TH^{min})}{(TH^{max} - TH^{min})} \quad (14)$$

QRED [**?**] employ a square function (using two approaches - Eqs. 15 and 16) for computing the dropping probability when Q_{ave} lies between TH^{min} and TH^{max}.

$$P_b = (\frac{Q_{ave} - TH^{min}}{K - TH^{min}})^2 \quad (15)$$

or

$$P_b = 1 - (\frac{K - Q_{ave}}{K - TH^{min}})^2 \quad (16)$$

where K identifies the buffer size.

By employing two dropping function curves - linear and exponential when Q_{ave} exists on the interval [TH^{min}, TH^{max}], RED-LE [**?**] obtained a lessened end-to-end delay performance. The drop action for RED-LE can be summarized as follows:

$$P_b = \begin{cases} 0, & \text{if } Q_{ave} < TH^{min} \\ 2P_{max}(\frac{Q_{ave} - TH^{min}}{TH^{max} - TH^{min}}), & \text{if } TH^{min} \leq Q_{ave} < Target \\ e^{log(P_{max})(\frac{2(TH^{max} - Q_{ave})}{TH^{max} - TH^{min}})}, & \text{if } Target \leq Q_{ave} < TH^{max} \\ 1, & \text{if } Q_{ave} \geq TH^{max} \end{cases} \quad (17)$$

in which

$$Target = \frac{TH^{max} + TH^{min}}{2} \quad (18)$$

Authors in [**?**] developed the AC-RED algorithm which determines its packet dropping probability in accordance with a composite of quadratic and power functions. This was done to obtain a lower packet loss rate and lessened delay.

In order to have the average queue size at a minimal stage, QERED [**?**] decides its packet dropping probability by combining the utilization of a square function and an exponential function (given in Eq. 19) when Q_{ave} exists in the range from $TH^{max} to TH^{max}$.

$$P_b = \begin{cases} 0, & \text{if } Q_{ave} < TH^{min} \\ 9P_{max}(\frac{Q_{ave} - TH^{min}}{TH^{max} + TH^{min}})^2, & \text{if } TH^{min} \leq Q_{ave} < Target \\ e^{log(P_{max})\left[\frac{3(TH^{max} - Q_{ave})}{2(TH^{max} - 2TH^{min})}\right]}, & \text{if } Target \leq Q_{ave} < TH^{max} \\ 1, & \text{if } Q_{ave} \geq TH^{max} \end{cases} \quad (19)$$

in which

$$Target = TH^{min} + (\frac{TH^{min} + TH^{max}}{3}) \quad (20)$$

DSRED [**?**] is another RED variant that increases its throughput. In DSRED, a linear drop function is immediately followed with another linear drop function. Both functions are were within the interval of TH^{min} and TH^{max}.

The main innovation of this paper is as follows. On the one hand, a new variant of RED is introduced which operates three distinct drop functions for supporting varying traffic loads; on the other hand, the proposed strategy is compared with two advanced algorithms yielding significant advantages.

3. Triplex Region - Random Early Detection Algorithm

The proposed Triplex Region - Random Early Detection (TR-RED) algorithm incorporates a composite of three packet dropping functions as an alternative to the one and only linear drop function of RED.

In TR-RED, a quadratic growth is followed by a (rapid) linear growth, which is in turn, followed by (a more rapid) exponential growth. Four coordinates are used in the packet dropping function. They are: (i) $\left(TH^{min}, 0\right)$, (ii) $\left(\frac{TH^{max}+2TH^{min}}{3}, \frac{P_{max}}{3}\right)$, (iii) $\left(\frac{2TH^{max}+TH^{min}}{3}, P_{max}\right)$, and (iv) $(TH^{max}, 1)$.

Three regions are also evident in TR-RED's packet dropping function. The first one is found between TH^{min} and $\frac{TH^{max}+2TH^{min}}{3}$. When Q_{ave} lies in this partition, low traffic load is believed to occur. As a result, the drop function (that is to say square) to be applied is given in Eq. 21:

$$P_b = 3P_{max}(\frac{Q_{ave} - TH^{min}}{TH^{max} - TH^{min}})^2 \tag{21}$$

The second one is found between $\frac{TH^{max}+2TH^{min}}{3}$ and $\frac{2TH^{max}+TH^{min}}{3}$. When Q_{ave} lies in this section, moderate traffic load is believed to occur. Consequently, the drop function (that is to say linear) to be applied is given as follows:

$$P_b = P_{max}\left[\frac{6Q_{ave} - (TH^{max} + 5TH^{min})}{3(TH^{max} - TH^{min})}\right] \tag{22}$$

The third one is found between $\frac{2TH^{max}+TH^{min}}{3}$ and TH^{max}. When Q_{ave} lies in this portion, high traffic load is believed to occur. Therefore, the drop function (that is to say exponential) to be applied is given as follows:

$$P_b = e^{log(P_{max})\left[\frac{3(TH^{max}-Q_{ave})}{(TH^{max}-TH^{min})}\right]} \tag{23}$$

In effect, the working principle for TR-RED is summarized in Eq. 24. Therefore, conceding that $TH^{min} = 30$ packets, $P_{max} = 0.1$ (as recommended by authors in [**?**]), and $TH^{max} = 90$ packets (which is $3 \times TH^{min}$, following the suggestion of authors in [**?**]), the envisioned TR-RED's dropping function is depicted in Figure 1. The pseudo-code in Algorithm 1 goes further to explain the core steps involved in TR-RED.

$$P_b = \begin{cases} 0, & \text{if } Q_{ave} < TH^{min} \\ 3P_{max}(\frac{Q_{ave}-TH^{min}}{TH^{max}-TH^{min}})^2, & \text{if } TH^{min} \le Q_{ave} < \frac{TH^{max}+2TH^{min}}{3} \\ P_{max}\left[\frac{6Q_{ave}-(TH^{max}+5TH^{min})}{3(TH^{max}-TH^{min})}\right], & \text{if } \frac{max+2min}{3} \le Q_{ave} < \frac{2TH^{max}+TH^{min}}{3} \\ e^{log(P_{max})\left[\frac{3(TH^{max}-Q_{ave})}{(TH^{max}-TH^{min})}\right]}, & \text{if } \frac{2TH^{max}+TH^{min}}{3} \le Q_{ave} < TH^{max} \\ 1, & \text{if } Q_{ave} \ge TH^{max} \end{cases} \tag{24}$$

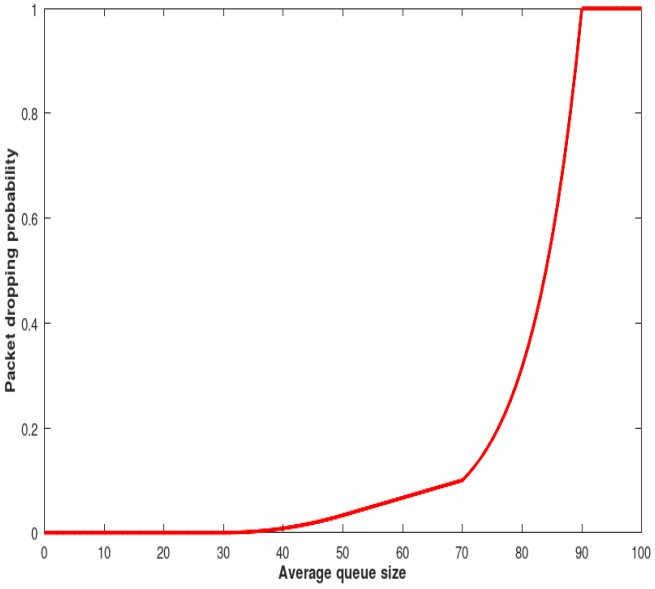


Figure 1. TR-RED's drop probability function curve

Algorithm 1 Pseudo-code of the TR-RED

1: Pre-decided parameters: P_{max}, TH^{min}, TH^{max}, w_q
2: Set $Q_{ave} = 0$
3: For each packet arrival:
4: Calculate the average queue size Q_{ave} Eq. 1
5: **if** $Q_{ave} < TH^{min}$ **then**
6: Enqueue packet
7: **else if** $TH^{min} \le Q_{ave} < \frac{TH^{max}+2TH^{min}}{3}$ **then**
8: Compute dropping probability P_b based on Eq. 21 (that is to say quadratic function)
9: Drop packet according to the calculated probability
10: **else if** $\frac{max+2min}{3} \le Q_{ave} < \frac{2TH^{max}+TH^{min}}{3}$ **then**
11: Compute dropping probability P_b based on Eq. 22 (that is to say linear function)
12: Drop packet according to the calculated probability
13: **else if** $\frac{2TH^{max}+TH^{min}}{3} \le Q_{ave} < TH^{max}$ **then**
14: Compute dropping probability P_b based on Eq. 23 (that is to say exponential function)
15: Drop packet according to the calculated probability
16: **else if** $Q_{ave} \ge TH^{max}$ **then**
17: Drop packet
18: **end if**

4. Simulation Results

We implement TR-RED algorithm in ns-3 simulator [**?**], wherein scripting was done using C++ programming language. Figure 2 depicts the network topology which has only one bottleneck link formed between the two intermediate machines: R_1 and R_2. Transmitting sources (that is to say S_1 to S_N) are individually connected to R_1 while the receiver's node (that is to say D) is connected to R_2. The bottleneck link is configured to use a bandwidth of 10 Mbit/s and a propagation delay of 100 ms. Other access links (either for connecting transmitting sources to R_1 or for connecting R_2 with D) is configured to use a bandwidth of 100 Mbit/s and a propagation delay of 25 ms. All source nodes are configured to implements TCP NewReno algorithm. Buffer size is set to 250 packets, while packet size is set to 1000 bytes. The queue of R_1 is controlled by AQM algorithms (that is to say TR-RED, TRED and NLRED), however in a sequential order. Unless otherwise stated, simulation time is set to 100 seconds and w_q to 0.002 (as recommended by authors in [**?**]).

To verify the performance of the proposed TR-RED, three simulation experiments was performed based on Figure 2, namely (i) low load (having 5 TCP transmitting sources) (supported by authors in [**?**]-[**?**]), (ii) moderate load (having 20 TCP transmitting sources), and (iii) high load (having 50 TCP transmitting sources) (supported by authors in [**?**]-[**?**]).

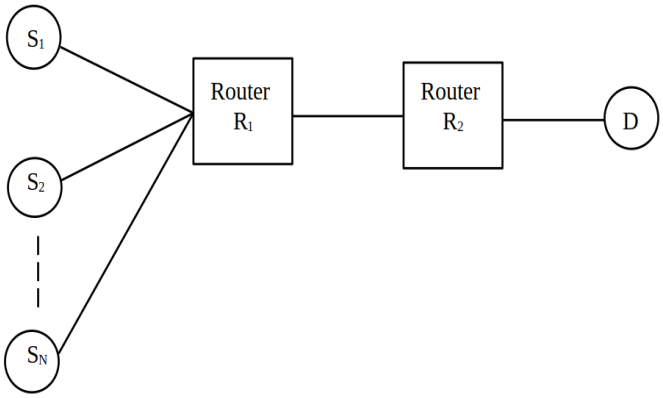
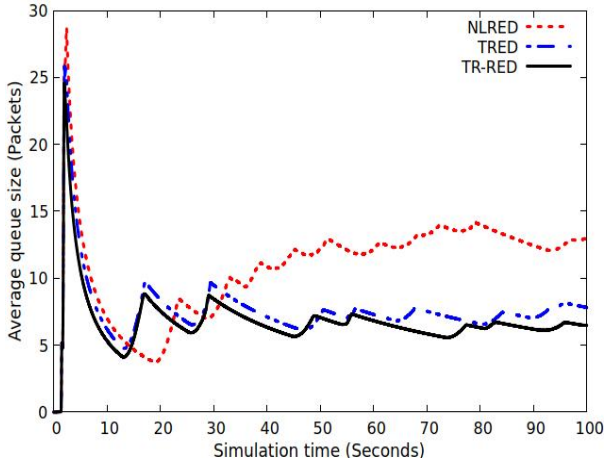


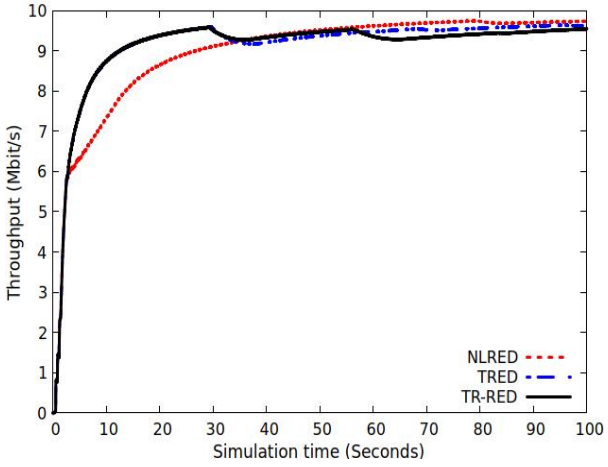
Figure 2. Network topology

A. *First scenario - low load*

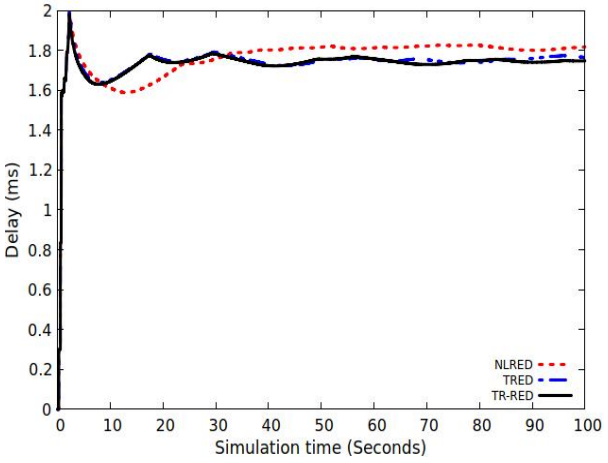
According to Figure 3(a), the initial maximum height attained by TR-RED was lowest among the three comparison algorithms. This indicates the superiority of TR-RED in controlling the queue size. From Figure 3(b), TRED obtained the highest result, although the performance of TR-RED was better than NLRED. The result shown in Figure 3(c) indicates that TR-RED has ability to effectively improve delay. It can be observed that TR-RED traded-off throughput for the purpose of achieving an advantage in terms of delay. Comparison results for the three algorithms is further illustrated in Table I.



(a) Average queue size



(b) Throughput



(c) Delay

Figure 3. Comparison for TR-RED with TRED and NLRED algorithms under low traffic circumstance

TABLE I. Comparison results of TR-RED with improved AQM algorithms (under low load)

Evaluation criteria	AQM Algorithm		
	TR-RED	TRED	NLRED
Avg. queue size (in packets)	6.7627	7.6035	10.8891
Avg. delay (in milliseconds)	1.7280	1.7343	1.7613
Avg. throughput (in Mbit/s)	9.0796	9.1186	8.9528

TABLE II. Comparison results of TR-RED with improved AQM algorithms (under moderate load)

Evaluation criteria	AQM Algorithm		
	TR-RED	TRED	NLRED
Avg. queue size (in packets)	15.7018	16.9022	29.6405
Avg. delay (in milliseconds)	7.2050	7.2252	7.7957
Avg. throughput (in Mbit/s)	9.3974	9.5031	9.2665

TABLE III. Comparison results of TR-RED with improved AQM algorithms (under high load)

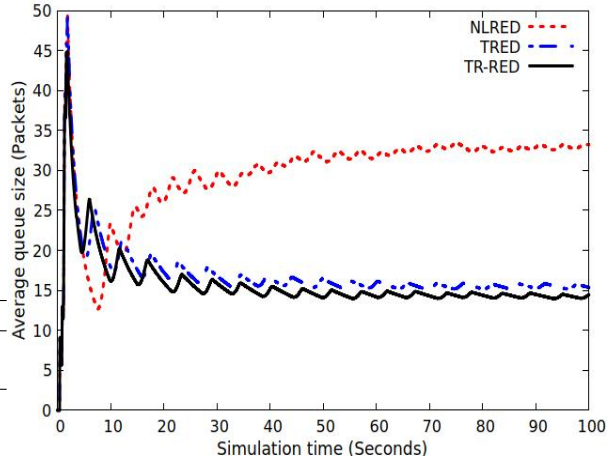
Evaluation criteria	AQM Algorithm		
	TR-RED	TRED	NLRED
Avg. queue size (in packets)	25.4275	30.6343	53.3537
Avg. delay (in milliseconds)	17.9899	18.3761	20.3817
Avg. throughput (in Mbit/s)	9.3448	9.5708	9.7532

B. Second scenario - moderate load

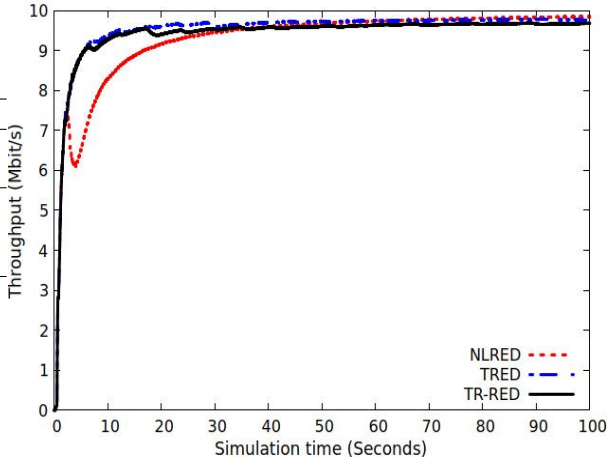
As Figure 4(a) shows, the performance of TR-RED outperforms other improved RED approaches, indicating that TR-RED is a more effective strategy to control the queue size. As depicted in Figure 4(b), TRED provides better throughput performance compared to both TR-RED and NLRED. The delay diagram for TR-RED and other two algorithms is plotted in Figure 4(c). According to the graph, TR-RED provides a good reduced delay in comparison with TRED and NLRED. Again, it can be seen that TR-RED traded-off throughput gains in order to achieve delay advantages. Comparison results for the three algorithms is further provided in Table II.

C. Third scenario - high load

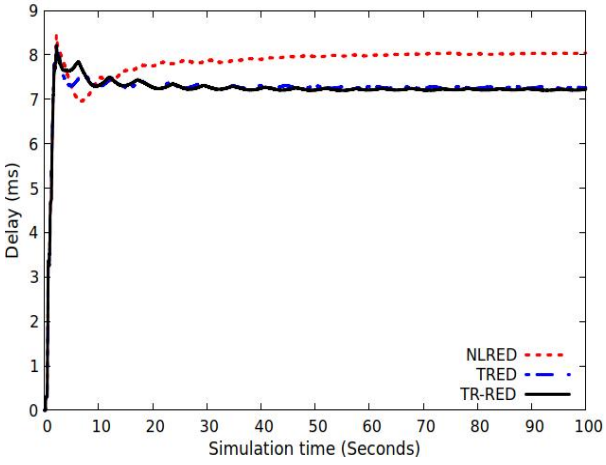
The result plotted in Figure 5(a) shows that TR-RED achieved a substantial and indeed a good performance compared with TRED and NLRED algorithms in terms of average queue size. The result for throughput is shown in Figure 5(b). It reveals that the performance of NLRED has a more significant advantage over TR-RED and TRED. Figure 5(c) shows that TR-RED obtained a remarkable delay performance when compared with the advanced algorithms - TRED and NLRED. These results shows that TR-RED traded-off throughput gains for delay advantages. Comparison results for the three algorithms is further illustrated in Table III.



(a) Average queue size

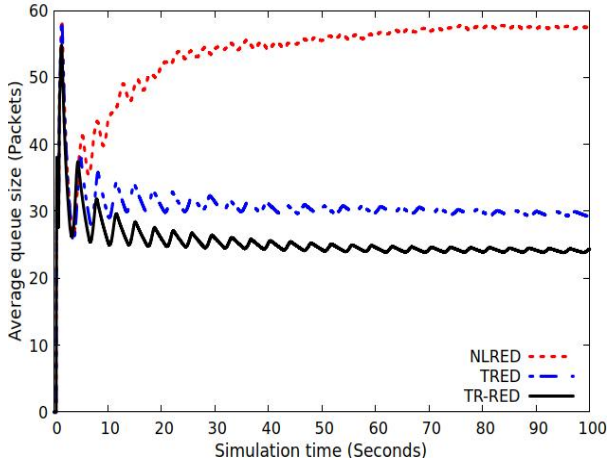


(b) Throughput

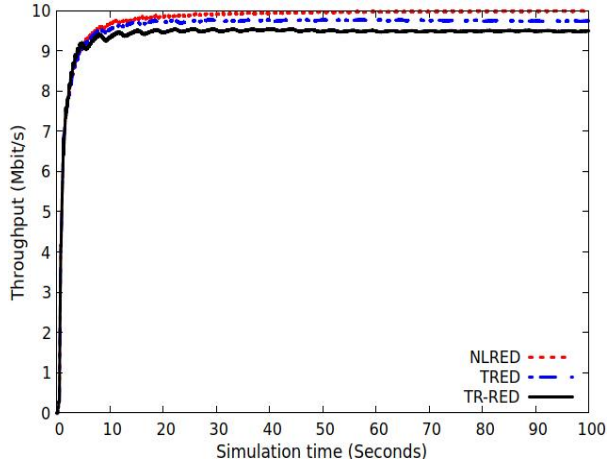


(c) Delay

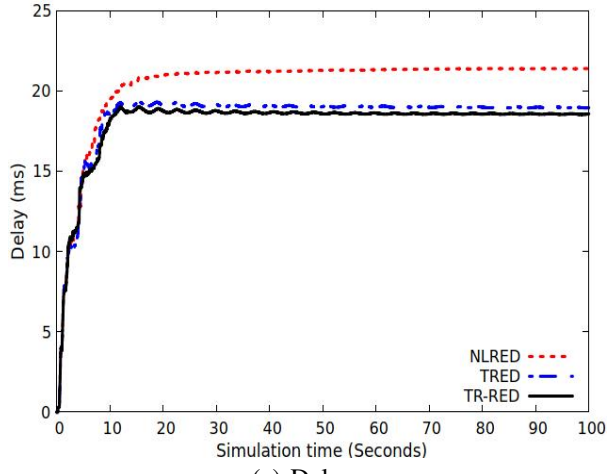
Figure 4. Comparison for TR-RED with TRED and NLRED algorithms under moderate traffic circumstance



(a) Average queue size



(b) Throughput



(c) Delay

Figure 5. Comparison for TR-RED with TRED and NLRED algorithms under high traffic circumstance

5. Conclusion

A new type of modified RED algorithm, termed Triplex Region - RED (TR-RED) which combines a square, linear and exponential drop functions, is proposed in this paper. Moreover, TR-RED was compared with two other better-known AQM algorithms in ns-3 simulation software. Simulation results shows that TR-RED yielded competitive advantage in handling various traffic loads. What's more, it can be concluded that TR-RED traded-off throughput benefit for delay gains. In follow-up research, TR-RED will be implemented in Linux kernel and in turn embedded in a software router for possible deployment on a personal computer in a real network circumstance. Its performance will be assessed and compared with some selected AQM algorithms.

References

[1] N. Kaur and R. Singhai, "Congestion control scheme using network coding with local route assistance in mobile adhoc network," *International Journal of Computer Applications in Technology*, vol. 60, no. 3, pp. 242–253, 2019.

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 in the Department of Mathematical Sciences (Computer Science Unit), Olabisi Onabanjo University, Ago-Iwoye, Nigeria. His research interests spans Computational Mathematics, computer networks and communications, Mathematical modeling and simulation, and Internet congestion control.

Olakunle O. Solanke is a senior lecturer in the Department of Mathematical Sciences (Computer Science Unit), Olabisi Onabanjo University, Ago-Iwoye, Nigeria. He obtained his PhD degree in Computer Science in 2016. He has been involved in researches in the area of network intrusion detection, network congestion control and information security.

Chigozirim Ajaegbu is an Associate Professor in Babcock University, Ilishan-Remo, Nigeria. His research interest includes wireless networking, humanized computing and information technology. He is also an associate editor and reviewer of some high ranking journals.

Tola J. 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 Multi-party Computation, with collaborations in Data Mining and Machine Learning-based security protocols.

Khadijha-Kuburat A. Abdullah received a M.Sc and Ph.D degrees in Computer Science from University of Ibadan, Ibadan, Oyo State, Nigeria. Currently, She is a Lecturer in the Department of Mathematical Sciences (Computer Science Unit), Olabisi Onabanjo University, Ago-Iwoye, Nigeria. Her research spans Artificial Intelligence, Machine learning, Natural Language Processing and Semantic Information.

Oyebola Akande is a lecturer in the Department of Computer Science, Babcock University, Ilishan-Remo, Ogun State, Nigeria. She has B.Tech in Computer Science from Ladoke Akintola University of Technology, M.Sc and Ph.D in Computer Science from the University of Ibadan.

Semiu A. Ayinde is a Lecturer in the Department of Basic Sciences, Babcock University, Nigeria. He has a PhD in Mathematics.