Journal of Computer Science

Original Research Paper

I-RED: An Improved Active Queue Management Algorithm

[1]Samuel Oluwatosin Hassan, [2]Adewole Usman Rufai, [3]Samson Ojo Ogunlere, [3]Olujimi Daniel Alao, [1]Lukman Adebayo Ogundele, [4]Michael Olugbenga Agbaje, [4]Aderonke Adelola Adegbenjo and [4]Shade Oluwakemi Kuyoro

[1]Department of Mathematical Sciences, Olabisi Onabanjo University, Ago-Iwoye, Nigeria
[2]Department of Computer Sciences, University of Lagos, Lagos, Nigeria
[3]Department of Information Technology, Babcock University, Ilisan-Remo, Nigeria
[4]Department of Computer Science, Babcock University, Ilisan-Remo, Nigeria

Article history
Received: 25-08-2021
Revised: 08-12-2022
Accepted: 10-12-2021

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

Abstract: Quality of Service (QoS) of Internet routers is still challenged with the issue of congestion. Active Queue Management (AQM) algorithms aimed at improving end-to-end delay of packets by keeping the average queue size small. This objective is yet to be fully accomplished, especially for interactive services. In this paper, an AQM algorithm named Improved-Random Early Detection (I-RED) algorithm based on the popular Random Early Detection (RED) is presented. I-RED deploys a combination of a nonlinear and a linear packet dropping functions. ns-3 simulation performance evaluations proved that I-RED effectively controls the average queue size and delay under light and heavy network traffic conditions. Replacing/upgrading the RED algorithm implementation in Internet routers (either software or hardware) requires minimal effort since only the packet dropping probability profile needs to be adjusted.

Keywords: Active Queue Management, Congestion Control, End-to-End Delay, I-RED, Simulation

Introduction

Essentially, in a computer network, congestion is said to occur when the aggregated amount of generated data traffic is greater than the network's resource buffer capacity (Abu-Shareha (2019); Adamu *et al*. (2021)).

Congestion has adverse effect on network performance leading to poor utilization, low throughput, large delay/jitter, high packet loss rate (Abdel-Jaber (2020); Adamu *et al*. (2020)). There is therefore a need to avoid congestion in order to have an improved Quality of Service (QoS) provided in network resources (Adamu *et al*., 2020).

Some of the shortcomings of the traditional Drop-Tail queue management algorithm are: buffer overflow, long delay in data delivery, lock-out phenomenon, and global synchronization. To address these problems, Internet Engineering Task Force (IETF) recommends the implementation of Active Queue Management (AQM) algorithm in Internet routers (Braden *et al*. (1998); Brandauer *et al*. (2001)).

AQM algorithm which differs from the traditional Drop-Tail queue management algorithm controls congestion by early detection of incipient congestion and sending feedback signals to end-hosts allowing them to reduce their rate of transmission before the router's buffer overflows (Aweya *et al*., 2001). Adamu *et al*. (2021) noted that research on AQM congestion control algorithm is necessary even in the current Internet, due to the ever increasing number of active users.

A lot of proposals on AQM algorithms leveraged on the simple, yet profound design of the famous Random Early Detection (RED) developed by Floyd and Jacobson (1993). The procedures for RED comprises of two sections: the computation of the average queue size (avg) and the decision of whether or not to accept an incoming packet. For the first section, avg (that is, the average number of packets in the buffer of the router) is estimated by the instantaneous queue length using an Exponential weighted Moving Average (EWMA) approach. For the second section, the avg is compared with two preset thresholds: the minimum threshold (denoted min_{TH}) and maximum threshold (denoted max_{TH}). Packets will be enqued if the avg is below the minT H; when avg varies between the two thresholds, the packet will be dropped randomly with a linear drop function that increases from 0 to a maximum packet dropping probability parameter (denoted maxP); all arriving packets are dropped if the avg is greater than max_{TH} threshold.

To improve network performance, there is a need for a congestion control algorithm that stabilizes the average

queue size. Unlike several proposals to increase the performance of RED algorithm, we propose a RED-based AQM algorithm called Improved RED (I-RED) which aimed at stabilizing and keeping the average queue size small which will in turn reduce the end-to-end delay of packets needed to ensure an improved QoS of Internet routers.

Related Works

The Random Early Detection (RED) algorithm developed by Floyd and Jacobson (1993) is indeed a profound congestion control algorithm for routers. Four parameters are important in RED, they include: the minimum threshold (denoted $\min_{TH}$), the maximum threshold (denoted $\max_{TH}$), the maximum packet dropping probability (denoted $\max_P$), and the queue weight (denoted w).

The average queue size (denoted avg) is computed by applying a low-pass filter with EWMA (Exponential weighted Moving Average) which acts to smooth out the burstiness of the instantaneous queue length according to Eq. (1):

$$avg = \begin{cases} (1-w)\times avg' + (w\times q) & q \neq 0 \\ (1-w)^m \times avg & otherwise \end{cases} \quad (1)$$

where:

q = Refers the current queue size
avg' = Refers to the computed previous average queue size
w = (which varies between 0 and 1) is a preset weight parameter to calculate avg; and
m = Refers to the idle time parameter computed as follows:

$$m = f\left(time - q\_idle\_time\right) \quad (2)$$

where, q_idle_time indicates the beginning of the queue idle time.

When avg is less than min_{TH}, then the arriving packet is accepted in the router's queue. However, if avg is between min_{TH} and max_{TH} then the packet will be dropped with a linear packet dropping function that rises from 0 to max_P. Lastly, if avg value is equal or greater than the max_{TH}, then the incoming packet will be dropped with a probability of 1. Hence, initial drop probability (P_b) function of RED is expressed as:

$$P_b = \begin{cases} 0, & \left| avg < \min_{TH} \right. \\ \max_P\left(\dfrac{avg - \min_{TH}}{\max_{TH} - \min_{TH}}\right), & \left| \min_{TH} \le avg < \max_{TH} \right. \\ 1, & \left| \max_{TH} \le avg \right. \end{cases} \quad (3)$$

Thus:

$$P_a = \frac{P_b}{1-\left(count \times P_b\right)} \quad (4)$$

where, P_a is the final packet dropping probability and count is the number of packets not dropped since the last dropped .

To address the parameter configuration issue of RED, Abdel-Jaber (2020) suggested an enhanced RED algorithm named RED-Exponential (RED_E) which does not require max_P in the packet dropping function (as expressed in Eq. (5)) when avg varies between the min_{TH} and max_{TH} queue. RED-E was reported to achieve a reduced end-to-end delay especially at heavy congestion state:

$$P_b = \begin{cases} 0, & \left| avg < \min_{TH} \right. \\ \left(\dfrac{e^{avg} - e^{\min} TH}{e^{\max} TH - e^{\min} TH}\right), & \left| \min_{TH} \le avg < \max_{TH} \right. \\ 1, & \left| \max_{TH} \le avg \right. \end{cases} \quad (5)$$

Adamu *et al*. (2021) developed the SARED (Self-Adaptive Random Early Detection) algorithm. SARED initiated a self-adaptive mechanism for RED algorithm such that when avg falls between min_{TH} and max_{TH}, packets are dropped with a nonlinear packet dropping function for a light and moderate traffic load conditions. However, when the avg falls between min_{TH} and max_{TH}, SARED changes to a linear mode for a high traffic load condition. At low and moderate traffic loads, SARED improved the throughput performance but at the expense of delay.

An introduction of self-adaptation mechanism for RED algorithm called Flexible RED (FXRED) algorithm was suggested by Adamu *et al*. (2020). FXRED employs both avg and the current traffic load condition as indicators for congestion. When avg size falls between min_{TH} and a midpoint threshold, FXRED deploys a nonlinear quadratic drop function for both low and moderate traffic loads in order to obtain an improved throughput and link utilization performance but at the expense of delay. Also, when avg falls between the mid-point threshold and max_{TH}, FXRED deploys a linear packet dropping function for high traffic load in order to obtain an improved delay.

By employing a fewer number of parameters for the calculation of the packet dropping probability, the Improved Gentle RED (IGRED) algorithm proposed by Abdel-Jaber *et al*. (2019) obtained an improved average queue size, average queuing delay and packet loss probability performance.

(EFRED) Enhancement of Fair Random Early Detection algorithm developed by Abdulkareem *et al*. (2015) aimed at achieving a low packet loss and reduced delay performance by employing hazard rate to determine packet dropping function for reducing packet dropping.

Abu-Shareha (2019) proposed the (DcRED) Delay-Controlled Random Early Detection algorithm which is an improved RED algorithm in order to achieve a reduced delay performance. DcRED determine the dropping rate by computing an estimated delay parameter with RED.

The underlying idea of the (BRED) Balanced–RED algorithm proposed by Anjum and Tassiulas (1999) is simply to achieve a fair bandwidth utilization between adaptive and non-adaptive traffic flows.

(Durresi *et al*., 2006) developed (LED) Load Early Detection algorithm prefers to compute the average traffic load as an indicator for congestion instead of *avg* (as common with RED algorithm). LED obtained an improved link utilization performance.

Floyd, (2000) observed that RED algorithm is too aggressive because packets are dropped when *avg* exceeds max_{TH}. Therefore, the (GRED) Gentle RED algorithm was proposed which extended RED by another threshold: ($2\times max_{TH}$). GRED utilizes two linear packet dropping functions. When avg is between min_{TH} and max_{TH}, packets are dropped linearly from 0 to *maxP*. However, if *avg* lies between max_{TH} and $2 \times max_{TH}$, packets are dropped linearly from *maxP* to 1. GRED obtained an increased throughput performance than RED.

The ModRED (Modified RED) algorithm was developed by Kachhad and Lathigara (2018) with the aim of achieving an improved throughput, goodput, packet delivery ratio and delay performance metrics. Depending on the incoming traffic, ModRED divides the packet dropping probability of RED algorithm into three sections and utilizes the (AIMD) Additive Increase Multiplicative Decrease algorithm.

MRED was developed by Koo *et al*. (2001) in order to improve throughput and delay. MRED computes the packet dropping probability by using a stepwise function based on the link history and packet loss information when *avg* varies between the min_{TH} and max_{TH} thresholds.

The (HRED) (Half-way RED) algorithm was developed by Hamadneh *et al.* (2019) is an idea to address the parameter configuration problem of RED algorithm. HRED provides an enhancement to the packet dropping probability calculation and developed a nonlinear dropping probability. HRED obtained an improved performance in terms of link utilization, throughput and packet loss rate metrics.

Ismail *et al*. (2014) proposed the ENRED (Enhanced RED) algorithm with the aim of reducing the packet loss rate and delay performance metrics by working on the computation of *avg*.

Patel and Karmeshu (2019) chose to modify the dropping function of RED, especially when avg lies between min_{TH} and max_{TH} in order to achieve an improved network performance, such that:

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

In which:

$$p_1 = \max_p \left(\frac{avg - \min_{TH}}{\max_{TH} - \min_{TH}} \right) \tag{7}$$

QRTRED algorithm was proposed by Jamali *et al.* (2014) to address the fixed parameter setting issue of RED. The min_{TH} and max_{TH} of RED are dynamically configured by the algorithm based on one *QRT* metric which estimates network condition from occupancy of the router's buffer. QRTRED achieved an improved performance in terms of link utilization.

Patel (2013) developed the URED (Upper threshold RED) algorithm as an extension for RED. URED introduced a queue threshold between min_{TH} and max_{TH} of RED algorithm and utilizes two linear packet dropping functions in order to achieve an improved performance in terms of throughput and packet loss rate metrics.

The Smart Red(SmRED) algorithm proposed by Paul *et al*. (2016) is an enhanced RED algorithm. SmRED initiates another threshold, *Target* (Eq. 6) between min_{TH} and max_{TH}. SmRED deploys a combination of a nonlinear (quadratic) and a linear packet dropping functions. SmRED was reported to achieve an improved end-to-end delay at heavy traffic and an increased link utilization at light traffic:

$$Target = \min_{TH} + \frac{\max_{TH} + \min_{TH}}{2} \tag{8}$$

ExRED (Extended Double Slope Random Early Detection mechanism) developed by Prabhavat *et al*. (2002) aimed at providing a tolerance scheme for RED by introducing a 2^{nd} order polynomial drop function when *avg* exceeds max_{TH}. ExRED was reported to achieve an increased throughput.

Abdel-Jaber *et al*. (2015) developed two analytical models based on RED namely, RED-Exponential and RED-Linear which utilizes instantaneous queue length as the congestion detector in lieu of average queue size.

The underlying idea of SDRED (State Dependent RED) algorithm proposed by Ryoo and Yang (2006) is to dynamically configure the max_{TH} and queue weight parameters of RED. SDRED achieved a reduction in queue delay and jitter.

Su *et al*. (2018) proposed the Q-Learning-based RED (QRED) algorithm which dynamically modify max_P by using the Q-Learning approach. QRED was reported to obtain an improved throughput performance.

MRED developed by Zhang *et al.* (2012) is an improved GRED algorithm in the sense a nonlinear quadratic packet drop function is used when the avg falls between the min_{TH} and max_{TH} (instead of a linear packet drop function as used by GRED). MRED obtained an increased throughput and a reduced packet loss rate.

Double Slope RED (DSRED) was developed Zheng and Atiquzzaman (2000) by in order to address the low throughput weakness of RED. The algorithm extended RED by introducing a mid-point threshold between min_{TH} and max_{TH} and deploys a combination of two linear packet dropping functions. The slopes of these two functions are complementary and adjustable through a mode selector.

The NLRED (Nonlinear Random Early Detection) algorithm was developed by Zhou *et al*. (2006). When the *avg* is greater than min_{TH} but less than max_{TH} (in comparison with RED), NLRED drops packet using a nonlinear quadratic drop function. The gentle increase in the packet dropping probability of NLRED is aimed at reducing the number of packet dropped at light traffic load thereby achieving an improved throughput.

One of the important goals of AQM is to keep the average queue size small which will in turn positively impact the end-to-end delay of packets needed for interactive services. Despite all the highlighted proposals, not much has been achieved in this area.

In this study, we propose an extension to RED algorithm to improve the QoS of Internet routers by effectively keeping the average queue size thereby resulting in a reduced end-to-end delay.

The Proposed Improved RED (I-RED) Algorithm

The proposed algorithm is named Improved-Random Early Detection (denoted I-RED) which is based on the RED algorithm is shown in Fig. 1. I-RED modified RED's packet dropping function by dividing the section between min_{TH} and max_{TH} thresholds of the queue into two parts in order to distinguish between two traffic load situations namely, light and heavy.

Similar to RED, for every packet that arrive the router's queue, I-RED measures congestion by computing the average queue size (*avg*) by using Eq. (1).

The control function for dropping packets in I-RED can be described as follows:

(a) If *avg* is found to exist between 0 and min_{TH} threshold, then the incoming packet will be admitted into the queue. That is:

$$P_b = 0 \tag{9}$$

(b) If *avg* value is found to be higher than the min_{TH} threshold but less than *Target* threshold, then the packet is dropped with probability:

$$Target = 2\left(\frac{\min_{TH} + \max_{TH}}{3}\right) \tag{10}$$

$$P_b = 9\max_P\left(\frac{avg - \min_{TH}}{2\max_{TH} - \min_{TH}}\right)^2 \tag{11}$$

The quadratic drop function in Eq. (9) is meant to ensure a slow increase in the packet dropping probability from 0 to *maxP* for smaller average queue size when congestion is not too serious.

(c) If *avg* is found to be higher than the *Target* threshold but less than max_{TH} threshold, then the incoming packet is dropped with probability:

$$P_b = \max_P + 3\left(1 - \max_P\right)\left(\frac{avg - Target}{\max_{TH} - 2\min_{TH}}\right) \tag{12}$$

The linear drop function in Eq. (10) is meant to ensure a fast increase in the packet dropping probability from max_P to 1 for larger average queue size when congestion is very serious.

(d) However, if *avg* value is found to be equal or greater than max_{TH} threshold, then the arriving packet will be forced dropped with a probability of one. That is:

$$P_b = 1 \tag{13}$$

The pseudocode for I-RED algorithm is presented in Algorithm 1.

Algorithm 1 I-RED Algorithm

```
For each packet arrival into I-RED router buffer do
Compute the average queue size avg according to Eq. (1)
if (avg < min_TH) then
    Packet is admitted
else if (min_TH ≤ avg < Target) then
    Compute the packet drop probability using a quadratic function according to Eq. (9)
    With the calculated probability, drop the arriving packet
else if (Target ≤ avg < max_TH) then
    Compute the packet drop probability using a linear function according to Eq. (10)
    With the calculated probability, drop the arriving packet
else if (max_TH ≤ avg) then
    Arriving packet is dropped
end if
```

Simulation and Performance Analysis

In this section, we implement I-RED algorithm in ns-3 simulation tool and compare its performance with the NLRED algorithm under two traffic load conditions namely, light and heavy. The network topology used to test both algorithms is depicted in Fig. 2. The network topology has the following configuration parameters: *N* connecting TCP sources, two routers (A having I-RED and NLRED implementations while B has Drop-Tail implementation), one sink. Each of the sources is connected to router A with a link rate of 100 *Mbps* and 3 *ms* propagation delay time. Similarly, the destination node D has a capacity of 100 *Mbps* with 3 *ms* propagation delay time. The bottleneck link has a constraint of 10 *Mbps* with 10 *ms* propagation delay time. Packet size is set as 1000 bytes. Buffer size is set to 25 packets. Simulation duration is 100 sec.

Unless otherwise stated I-RED and NLRED parameters are configured to as: min_{TH} = 3 (as suggested by Abdel-Jaber (2020)), *Target* = 8, max_{TH} = 9 (as suggested by Abdel-Jaber (2020)), *maxP* = 0.1 (as suggested by Floyd and Jacobson (1993)), and W_q = 0.002 (as suggested by Floyd and Jacobson (1993)).

Light Traffic Load

Figure 3 (a)-(c) presents the simulation results for I-RED and NLRED algorithms under light traffic load condition (having 5 TCP connecting sources) using performance metrics (such as average queue size, delay and throughput). Using 5 TCP flows to represent light traffic load is consistent with those reported in Pan *et al*. (2013), Schepper *et al*. (2016) and Jain *et al*. (2018).

Figure 3 (a) illustrates the average queue size for the duo algorithms. It can be seen that I-RED evidently reduces the average queue size better than NLRED. I-RED attains an initial peak of 6.7508 while NLRED attains 6.8007. However, both algorithms controls the burst and regulates the oscillation in the average queue size. I-RED attains a mean value of 2.4836 for instantaneous average queue size while NLRED attains 2.5891. The reason is simple: when *avg* is near the *Target* value, the packet dropping probability of I-RED is higher than NLRED.

The delay result for the duo algorithms is presented in Fig. 3 (b). It can be seen that the delay of I-RED is also satisfactorily lower than NLRED. I-RED attains 0.4875 as a mean value of delay while NLRED obtained 0.4931.

The throughput result is depicted in Fig. 3 (c). As illustrated, I-RED achieved a mean value of 9.9887 while NLRED obtained 9.9825.

Analysis of the performance metrics is further shown in Table 1.

Heavy Traffic Load

Figure 4 (a)-(c) presents the simulation results for I-RED and NLRED algorithms under heavy traffic load condition (having 50 TCP connecting sources) using performance metrics (such as average queue size, delay and throughput). Similarly, using 50 TCP flows to represent heavy traffic load is consistent with those reported in Pan *et al*. (2013), Schepper *et al*. (2016) and Jain *et al*., (2018).

Figure 4 (a) illustrates the average queue size for the duo algorithms. It can be seen that I-RED satisfactorily performed better than NLRED algorithm in terms of average queue size. I-RED attains an initial peak of 5.3879 while NLRED attains 5.7027. However, both algorithms controls the burst and regulates the oscillation in the average queue size. I-RED attains a mean value of 5.1351 for instantaneous average queue size while NLRED attains 5.4866. The reason is simple: when *avg* is near max_{TH}, the packet dropping probability of I-RED is higher than NLRED.

The delay result for I-RED and NLRED algorithms is presented in Fig. 4 (b). It can be seen that the delay of I-RED is also clearly lower than NLRED. I-RED attains 5.3588 as a mean value of delay while RED obtained 5.3329.

The throughput result is depicted in Fig. 4 (c). As illustrated, I-RED achieved a mean value of 10.1016 while NLRED obtained 10.1333.

Analysis of the performance metrics is further shown in Table 2.

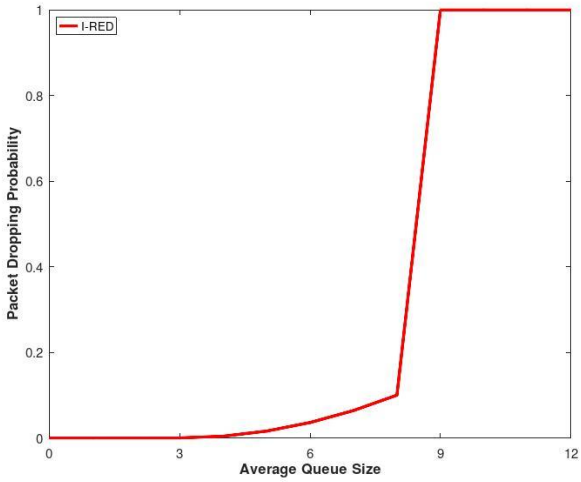


Fig. 1: I-RED's packet dropping probability

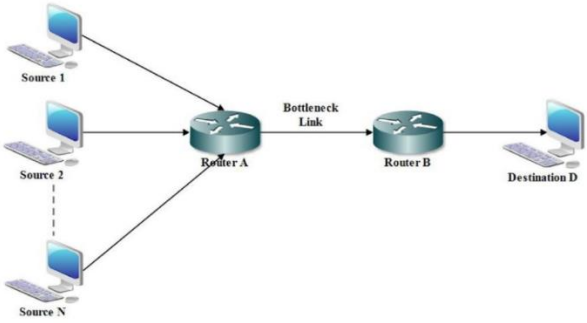


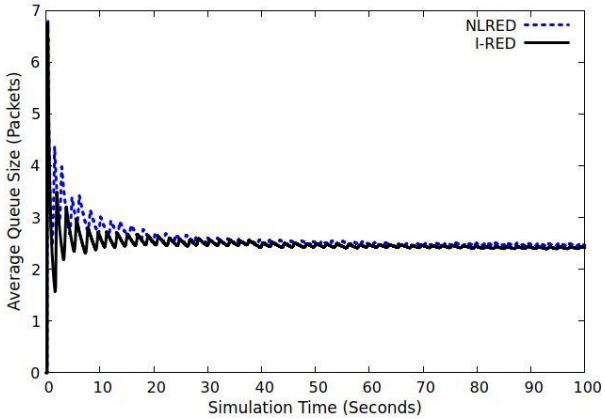
Fig. 2: Network topology

Table 1: Light traffic performance analysis

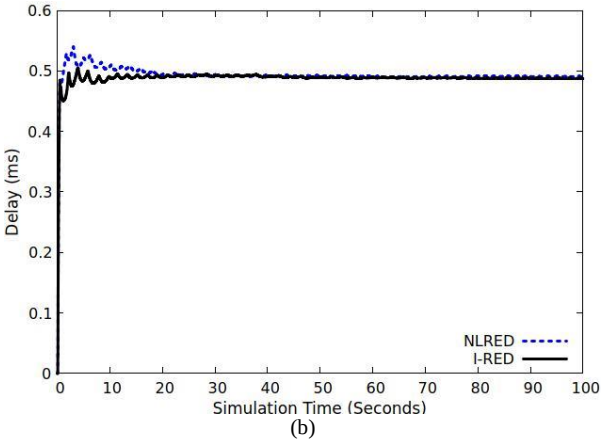
Algorithm	Average queue Size (Packets)	Delay (ms)	Throughput (Mbps)
NLRED	2.5891	0.4931	9.9825
I-RED	2.4836	0.4875	9.9887

Table 2: Heavy traffic performance analysis

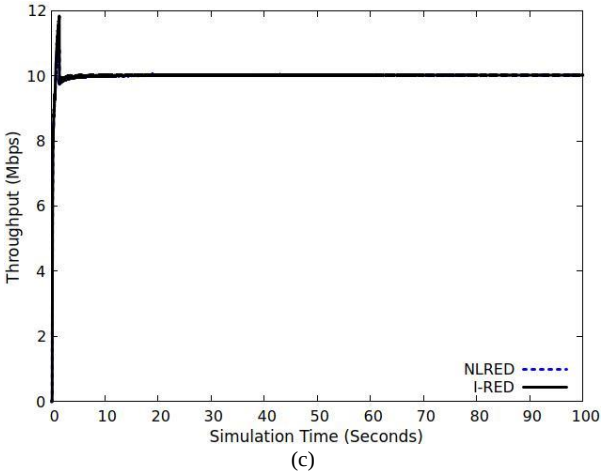
Algorithm	Average queue Size (Packets)	Delay (ms)	Throughput (Mbps)
NLRED	5.4866	5.3329	10.1333
I-RED	5.1351	5.3588	10.1016



(a)

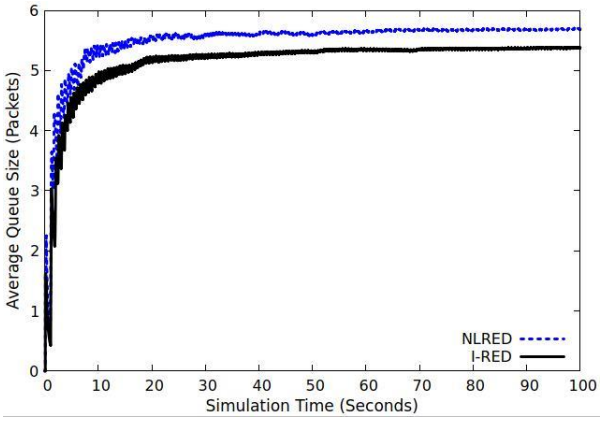


(b)

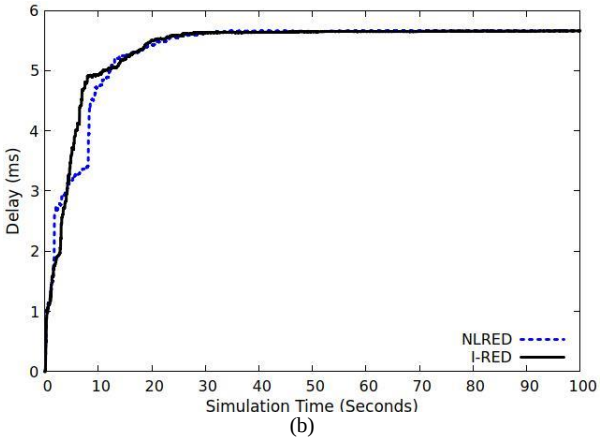


(c)

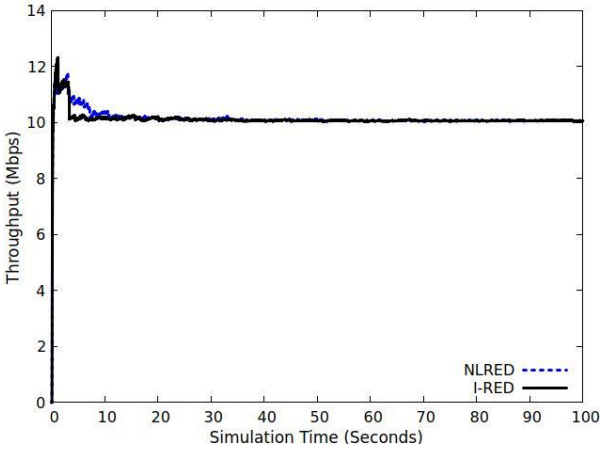
Fig. 3: Light traffic condition: Average queue size graph



(a)



(b)



(c)

Fig. 4: Heavy traffic condition: Average queue size graph

Conclusion

In this study, Improved-RED (I-RED) AQM algorithm is suggested as an improvement over RED algorithm. I-RED deploys a combination of a nonlinear (quadratic) with a linear packet dropping functions in

order to distinguish between two traffic load situations namely, light and heavy. Simulations conducted in ns-3 using a small buffer size capacity shows that I-RED evidently performed better than NLRED in terms of average queue size and a comparable throughput performance metrics both at light and heavy traffic load scenarios. Going forward, we intend to implement the proposed I-RED algorithm in Linux kernel (which will in turn be embedded in a software router) and compare its performance with other AQM algorithms on a real network.

Acknowledgement

We would like to sincerely appreciate the anonymous reviewers for their comments which has improved the work.

Author's Contributions

Samuel Oluwatosin Hassan: Problem formulation, mathematical modeling, literature review, and editing.

Adewole Usman Rufai: Literature review, writing and supervision

Samson Ojo Ogunlere: Literature review and simulkation

Olujimi Daniel Alao: Writing and mathematical modeling

Lukman Adebayo Ogundele: Literature review and supervision

Michael Olugbenga Agbaje: Simulation data analysis

Aderonke Adelola Adegbenjo: Mathematical modeling and writing

Shade Oluwakemi Kuyoro: Literature review, writing and formatting

Ethics

This article is an original research paper. There are no conflict of interest and no ethical issues that may arise after the publication of this manuscript.

References

Abdel-Jaber, H. (2020). An exponential active queue management method based on random early detection. Journal of Computer Networks and Communications, 2020. doi.org/10.1155/2020/8090468

Abdel-jaber, H., Shehab, A., Barakat, M., & RASHAD, M. (2019). IGRED: An Improved Gentle Random Early Detection Method for Management of Congested Networks. Journal of Interconnection Networks, 19(02), 1950004. doi.org/10.1142/S021926591950004X

Abdulkareem, M., Akil, K., Kalakech, A., & Kadry, S. (2015). Efred: Enhancement of fair random early detection algorithm. International Journal of Communications, Network and System Sciences, 8(07), 282. doi.org/10.4236/ijcns.2015.87028

Abu-Shareha, A. A. (2019). Controlling Delay at the Router Buffer Using Modified Random Early Detection. Int. J. Comput. Netw. Commun, 11, 63-75. doi.org/10.5121/ijcnc.2019.11604

Adamu, A., Surajo, Y., & Jafar, M. T. (2021) SARED: A Self-Adaptive Active Queue Management Scheme for Improving Quality of Service in Network Systems. Computer Science, 22(2). doi.org/10.7494/csci.2021.22.2.4020

Adamu, A., Shorgin, V., Melnikov, S., & Gaidamaka, Y. (2020, September). Flexible Random Early Detection Algorithm for Queue Management in Routers. In International Conference on Distributed Computer and Communication Networks (pp. 196-208). Springer, Cham. doi.org/10.1007/978-981-10-3773-3_62

Anjum, F. M., & Tassiulas, L. (1999). Balanced-RED: An algorithm to achieve fairness in the Internet. Maryland univ college park inst for systems research. https://apps.dtic.mil/sti/citations/ADA439654

Aweya, J., Ouellette, M., & Montuno, D. Y. (2001). A control theoretic approach to active queue management. Computer networks, 36(2-3), 203-235. doi.org/10.1016/S1389-1286 (00)00206-1

Brandauer, C., Iannaccone, G., Diot, C., Ziegler, T., Fdida, S., & May, M. (2001, July). 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 (pp. 122-129). IEEE. doi.org/10.1109/ISCC.2001.935364

Schepper, K., Bondarenko, O., Tsang, I. J., & Briscoe, B. (2016, December). Pi2: A linearized aqm for both classic and scalable tcp. In Proceedings of the 12th International on Conference on emerging Networking EXperiments and Technologies (pp. 105-119). doi.org/10.1145/2999572.2999578

Durresi, A., Barolli, L., Sridharan, M., Chellappan, S., & Jain, R. (2006). Load Early Detection (LED): A Congestion Control Algorithm Based on Routers' Traffic Load. IPSJ Digital Courier, 2, 94-107. doi.org/10.2197/ipsjdc.2.94

Floyd, S. (2000). Recommendation on using the gentle variant of RED. www. icir. org/floyd/red/gentle. html. https://ci.nii.ac.jp/naid/10017501130/

Floyd, S., & Jacobson, V. (1993). Random early detection gateways for congestion avoidance. IEEE/ACM Transactions on networking, 1(4), 397-413. doi.org/10.1109/90.251892

Hamadneh, N., Obiedat, M., Qawasmeh, A., & Bsoul, M. (2019). HRED, an active queue management algorithm for TCP congestion control. Recent Patents on Computer Science, 12(3), 212-217. doi.org/10.2174/2213275912666181205155828

Ismail, A. H., El-Sayed, A., Elsaghir, Z., & Morsi, I. Z. (2014). Enhanced random early detection (ENRED). International Journal of Computer Applications, 92(9). doi.org/10.1155/2011/87234, 2011

Jain, V., Mittal, V., Shravya, K. S., & Tahiliani, M. P. (2018). Implementation and validation of BLUE and PI queue disciplines in ns-3. Simulation Modelling Practice and Theory, 84, 19-37. doi.org/10.1016/j.simpat.2018.01.002

Jamali, S., Alipasandi, N., & Alipasandi, B. (2014). An improvement over random early detection algorithm: A self-tuning approach. Journal of Electrical and Computer Engineering Innovations (JECEI), 2(2), 57-61. https://jecei.sru.ac.ir/article_242.html

Kachhad, K., & Lathigara, A. (2018). ModRED: Modified RED an efficient congestion control algorithm for wireless networks. International Research Journal of Engineering and Technology (IRJET), 5(5), 1879-1884.

Koo, J., Song, B., Chung, K., Lee, H., & Kahng, H. (2001). MRED: A new approach to random early detection. Proceedings 15th International Conference on Information Networking (pp. 347-352). IEEE. https://ieeexplore.ieee.org/abstract/document/905450

Pan, R., Natarajan, P., Piglione, C., Prabhu, M. S., Subramanian, V., Baker, F., & VerSteeg, B. (2013, July). PIE: A lightweight control scheme to address the bufferbloat problem. In 2013 IEEE 14th international conference on high performance switching and routing (HPSR) (pp. 148-155). IEEE. https://ieeexplore.ieee.org/abstract/document/6602305

Patel, C. M. (2013, July). URED: Upper threshold RED an efficient congestion control algorithm. In 2013 Fourth International Conference on Computing, Communications and Networking Technologies (ICCCNT) (pp. 1-5). IEEE. doi.org/10.1109/ICCCNT.2013.6726469

Paul, A. K., Kawakami, H., Tachibana, A., & Hasegawa, T. (2016, May). An AQM based congestion control for eNB RLC in 4G/LTE network. In 2016 IEEE Canadian Conference on Electrical and Computer Engineering (CCECE) (pp. 1-5). IEEE. doi.org/10.1109/CCECE.2016.7726792

Prabhavat, S., Varakulsiripunth, R., & Noppanakeepong, S. (2002, November). Throughput improvement on RED mechanism. In The 8th International Conference on Communication Systems, 2002. ICCS 2002. (Vol. 1, pp. 599-603). IEEE. https://ieeexplore.ieee.org/abstract/document/1182545

Ryoo, I., & Yang, M. (2006). A state dependent RED: An enhanced active queue management scheme for real-time internet services. IEICE transactions on communications, 89(2), 614-617. doi.org/10.1093/ietcom/e89-b.2.614

Su, Y., Huang, L., & Feng, C. (2018). QRED: A Q-learning-based active queue management scheme. Journal of Internet Technology, 19(4), 1169-1178. https://jit.ndhu.edu.tw/article/view/1734

Zheng, B., and Atiquzzaman, M. (2000, November). DSRED: An active queue management scheme for next generation networks. In 25th Annual IEEE Conference on Local Computer Networks LCN 2000 (pp. 242-251).

Zhang, J., Xu, W. and Wang, L. (2011). An improved adaptive queue management algorithm based on nonlinear smoothing. Elsevier, 15, 2369-2373.

Zhang, Y., Ma, J., Wang, Y., & Xu, C. (2012). MRED: an improved nonlinear RED algorithm. In International Conference Proceedings on Computer and Automation Engineering (ICCAE 2011) (Vol. 44, pp. 6-11).

Zhou, K., Yeung, K. L., & Li, V. O. (2006). Nonlinear RED: A simple yet efficient active queue management scheme. Computer Networks, 50(18), 3784-3794. doi.org/10.1016/j.comnet.2006.04.007