ISSN: 0067-2904

Double Function Random Early Detection (DFRED): A Revised RED-Oriented Algorithm

Samuel O. Hassan[1]*, Chigozirim Ajaegbu[2], Olakunle O. Solanke[1]
[1]*Department of Mathematical Sciences, Olabisi Onabanjo University, Ago-Iwoye, Nigeria*
[2]*School of Computing and Engineering Sciences, Babock University, Ilishan-Remo, Nigeria*

Received: 8/10/2021 Accepted: 24/10/2022 Published: 30/10/2023

Abstract

Dropping packets with a linear function between two configured queue thresholds in Random Early Detection (RED) model is incapable of yielding satisfactory network performance. In this article, a new enhanced and effective active queue management algorithm, termed Double Function RED (DFRED in short) is developed to further curtail network delay. Specifically, DFRED algorithm amends the packet dropping probability approach of RED by dividing it into two sub-segments. The first and second partitions utilizes and implements a quadratic and linear increase respectively in the packet dropping probability computation to distinguish between two traffic loads: low and high. The ns-3 simulation performance evaluations clearly indicate that DFRED algorithm significantly controls the average queue occupancy and yields a reasonable gain in end-to-end-delay under different network conditions.

Keywords: AQM; Congestion control, DFRED algorithm, Internet routers, Simulations**.**

1. Introduction

The emergence of the Internet has certainly led to the outstanding increase in technological innovations witnessed across the globe. Consequently, the rate of user's connectivity is also at a fast increase. The data traffic volume transmitted on the network from various sources will increase accordingly, thereby imposing an inevitable problem of network congestion [1–4]. Technically, congestion in a network happens when the aggregate of transmitted traffic load goes beyond the resource availability of network's supporting infrastructure (i.e., the router) thereby imposing limitations, such as network performance degradation and poor quality of network services [3; 5–7]. In the end, congestion may result in a collapse- when communication in the network ceases [8].

In the Internet, congestion control is considered a germane and in fact a research hotspot for scholars in the field of data communications and computer networks ([7], [9]). To ensure a good network performance, there is a need to effectively and properly manage the queue in the buffer of network routers [10].

The traditional router's algorithm, known as DropTail, a typical form of passive queue management (often abbreviated to "PQM") algorithm is challenged with limitations such as

*Email: samuel.hassan@oouagoiwoye.edu.ng

buffer overflow, lock-out phenomenon, long delay in data delivery, and global synchronization [11]. Thankfully, active queue management (often abbreviated to "AQM") algorithm has emerged as a better approach for tackling the challenges of network congestion and has since been recommended by the Internet Engineering Task Force (IETF) for implementation in Internet routers [12].

Typically, AQM algorithms monitors and detects congestion at an emerging moment and sends a congestion report (using dropped packets) to end-hosts so that the proportion of data traffic being transmitted is reduced. AQM seeks two important goals. One is to reduce delay by keeping the average queue size minimal. The other is to improve throughput performance by reducing the number of dropped packets.

There exists a plethora of research studies on AQM in the literature. Among them, one algorithm is the most recognized developed Floyd and Jacobson in [13]. The algorithm is termed Random Early Detection or (abbreviated to "RED"). Hitherto, RED has received enormous attention from scientists both in academia and industry. Using RED, each time a packet arrives at a router, the algorithm computes the average queue size (hereafter "avg") (Eq. (1)) by relying on an exponential weighted moving average (EWMA) function.

$$avg = (1 - \mathrm{w})avg' + (\mathrm{w} \times q) \tag{1}$$

Where: q specifies the current queue size; avg' specifies previously calculated average queue size; *w* refers to a pre-decided weight parameter required to calculate *avg*.
As for cases where the value of *avg* is less than a predetermined minimum threshold (hereafter "Q_{Tmin}"), then the packet will be admitted into the queue. As for cases where the value of *avg* lies between Q_{Tmin} and another pre-decided maximum queue's threshold (henceforth "Q_{Tmax}"), then it performs the process of computing the drop probability through a linear function that rises from *0* to the maximum drop probability (hereafter "*maxP*"). However, for cases where the value of *avg* exceeds Q_{Tmax}, then all incoming packets will be dropped. To summarize, the initial packet drop function (from now "P_b") for RED is expressed as follows:

$$P_b = \begin{cases} 0, & avg < Q_{Tmin} \\ maxP\left(\frac{avg - Q_{Tmin}}{Q_{Tmax} - Q_{Tmin}}\right), & avg \in [Q_{Tmin}, Q_{Tmax}) \\ 1, & Q_{Tmax} \leq avg \end{cases} \tag{2}$$

The final packet dropping probability (hereafter "P_a") is defined by:

$$P_a = \frac{P_b}{1 - count \times P_b} \tag{3}$$

where *count* specifies the number of arrived packets since the last packet dropped.
Although, the easy-to-understand mathematical formulation of RED model is well-recognized, research continues to evolve to determine the suitability of its linear relationship that exist between the average queue size and packet dropping probability [9; 14-16].

In this article, the major limitation of RED algorithm of interest is its incapability to keep the average queue size low especially when the network is heavily congested, thereby imposing the problem of large network delay. Effectively, the main contribution of this paper is to propose an appropriate and amended RED algorithm, termed Double Function Random Early Detection (DFRED). The proposed algorithm implements a mixture of nonlinear (that is, quadratic) and a linear packet dropping functions to improve delay by reasonably reducing the average queue size.

The remainder of the paper is arranged as follows: Section 2 focuses on discussion of related works. Next, Section 3 presents the proposed DFRED algorithm. Section 4 presents the simulation results. Finally, Section 5 concludes the paper.

2. Related Works

Four parameters are fixed in RED: Q_{Tmin}, Q_{Tmax}, *maxP*, and *w*. Among these four, Ryoo and Yang in [17] chose to dynamically adjust Q_{Tmax} and *w* to gain an improved delay and jitter, which are essential for traffics from real-time interactive services. The modified algorithm was termed state dependent RED or (SDRED).

An enhancement of RED, termed RED with reconfigurable maximum dropping probability or (RRMDP) algorithm introduced by Al-Allaf and Jabbar in [18] used a reconfiguration parameter to reconfigure the maximum packet dropping probability of RED. RRMDP was reported to gain a reduced delay and *avg*.

To improve upon RED, Q-Learning-based RED (QRED) algorithm was introduced by Su *et al.* in [19] that dynamically adjusts the maximum packet probability through the Q-Learning mechanism. QRED was reported to gain an increased throughput.

The extended double slope random early detection mechanism (ExRED) scheme by Prabhavat *et al.* in [20] is an enhanced flavor of RED scheme. Usually, RED drop packets when $Q_{Tmax} \leq avg$ (that is, drop probability is 1). However, ExRED developers believed that drop probability should increase slowly by using a second order polynomial packet dropping function so as to gain a reduced number of packet drop and consecutive packet drop.

The Smart RED's (SmRED) strategy presented by Paul *et al.* in [15] is also another extended version of RED algorithm. SmRED performs the process of computing the drop probability through a mixture of both quadratic and square root functions, for different degrees on network congestion. Moreover, SmRED was reported to gain improved link utilization at low traffic and a reduced end-to-end delay at high traffic, respectively. The SmRED dropping function is expressed in Eq. (4):

$$p_b = \begin{cases} 0 \quad , & avg < Q_{Tmin} \\ maxP\left(\frac{avg - Q_{Tmin}}{Q_{Tmax} - Q_{Tmin}}\right)^2 , & avg \in [Q_{Tmin}, Target) \\ maxP\sqrt{\frac{avg - Q_{Tmin}}{Q_{Tmax} - Q_{Tmin}}} \ , & avg \in [Target, Q_{Tmax}) \\ 1 \quad , & Q_{Tmax} \leq avg \end{cases} \tag{4}$$

where *Target* value in Eq. (4) is expressed in Eq. (5):

$$Target = Q_{Tmin} + \left(\frac{Q_{Tmax} - Q_{Tmin}}{2}\right) \tag{5}$$

In contrast to RED, RED-Exponential (RED_E) [16] algorithm performs the process of computing the drop probability by-means-of an exponential function that increases from zero to one cases when $avg \in [Q_{Tmin}, Q_{Tmax})$. Simulation results reported indicates that RED_E gained an improved performance in terms of reduction in average queue size and delay metrics especially at high congestion situation. Moreover, in RED_E's, there is no need for the involvement of *maxP* in the computation of dropping probability as expressed in Eq. (6):

$$P_b = \begin{cases} 0, & avg < Q_{Tmin} \\ \left(\frac{e^{avg} - e^{Q_{Tmin}}}{e^{Q_{Tmax}} - e^{Q_{Tmin}}}\right), & avg \in [Q_{Tmin}, Q_{Tmax}) \\ 1, & Q_{Tmax} \leq avg \end{cases} \tag{6}$$

In [21], the gentle RED (GRED) model was developed. The underlying strategy of GRED algorithm is the implementation of two drop functions (whereby both are linear). Dropping function for GRED is given in Eq. (7):

$$P_b = \begin{cases} 0, & avg < Q_{Tmin} \\ maxP\left(\frac{avg-Q_{Tmin}}{Q_{Tmax}-Q_{Tmin}}\right), & avg \in [Q_{Tmin}, Q_{Tmax}) \\ maxP + (1-maxP)\left(\frac{avg-Q_{Tmax}}{Q_{Tmax}}\right), & avg \in [Q_{Tmax}, 2Q_{Tmax}) \\ 1, & 2Q_{Tmax} \leq avg \end{cases} \quad (7)$$

To enhance the performance of GRED, Baklizi *et al* in [22] pursued to propose adaptive GRED (AGRED). The dropping function for AGRED is given in Eq. (8):

$$P_b = \begin{cases} 0, & avg < Q_{Tmin} \\ maxP\left(\frac{avg-Q_{Tmin}}{Q_{Tmax}-Q_{Tmin}}\right), & avg \in [Q_{Tmin}, Q_{Tmax}) \\ maxP + \left(\frac{1-maxP}{2}\right)\left(\frac{avg-Q_{Tmax}}{Q_{Tmax}}\right), & avg \in [Q_{Tmax}, 2Q_{Tmax}) \\ 1, & 2Q_{Tmax} \leq avg \end{cases} \quad (8)$$

To tackle the high number of packet drop limitation of RED algorithm especially when $Q_{Tmax} \leq avg$, the authors in [23] proposed to modify the packet dropping probability of RED such that it performs the process of computing the drop probability by-use-of an exponential function. The new algorithm was termed congestion control RED or (CoCo-RED). The drop function for CoCo-RED is given in Eq. (9):

$$p_b = \begin{cases} 0 & avg < Q_{Tmin} \\ maxP\left(\frac{avg-Q_{Tmin}}{Q_{Tmax}-Q_{Tmin}}\right) & avg \in [Q_{Tmin}, Q_{Tmax}) \\ \frac{maxP}{(e^{\frac{\ln(1/maxP)}{K-Q_{Tmax}}})^{Q_{Tmax}}} \times (e^{\frac{\ln(1/maxP)}{K-Q_{Tmax}}})^{avg} & avg \in [Q_{Tmax}, K) \end{cases} \quad (9)$$

where K in Eq. (9) specifies the maximum buffer capacity.

To improve the throughput performance of CoCo-RED, Suwannapong and Khunboa in [24] presented the enhanced CoCo-RED or (EnCoCo-RED) algorithm. Based on the congestion level, EnCoCo-RED, dynamically adjusts the Q_{Tmin} and Q_{Tmax} thresholds.

To improve upon GRED, Baklizi in [25] dynamically adjusted two thresholds namely: Q_{Tmin} and $2Q_{Tmax}$. The algorithm aimed at stabilizing the average queue size between Q_{Tmin} and $2Q_{Tmax}$. The algorithm obtained a performance gain with regards to queuing delay and packet loss rate.

Rather than using a linear drop function in RED, Zhou et al. in [9] performed the process of computing the drop probability via a quadratic function (expressed according to Eq. (10)) when $avg \in [Q_{Tmin}, Q_{Tmax})$. This enhancement to RED was termed nonlinear RED or (NLRED).

$$P_b = \begin{cases} 0, & avg < Q_{Tmin} \\ maxP'\left(\frac{avg-Q_{Tmin}}{Q_{Tmax}-Q_{Tmin}}\right)^2, & avg \in [Q_{Tmin}, Q_{Tmax}) \\ 1, & Q_{Tmax} \leq avg \end{cases} \quad (10)$$

where $maxP'$ was set as 1.5 × *maxP*. NLRED gained an improved throughput performance.

The flexible RED or (FXRED) algorithm proposed by Adamu *et al.* in [26] is yet another improved version of RED. In FXRED, avg and current traffic load condition were used as indicators for congestion. When $avg \in [Q_{Tmin}, \Delta)$. (where Δ is a mid-point threshold specified as $(Q_{Tmin}+Q_{Tmax})/2$), FXRED performs the process of computing the drop probability by-means-

of a nonlinear function at both low and moderate traffic loads. However, when $avg \epsilon [\Delta, Q_{Tmax})$. FXRED performs the process of computing the drop probability by-means-of a linear function at high traffic load. In FXRED, a mode selector γ is employed for switching between these two functions.

Authors in [27] developed an improved flavor of RED, named improved adaptive RED algorithm. Unlike RED, which performs the process of computing the drop probability by-means-of a linear function, the improved adaptive RED uses a cubic function expressed as follows:

$$P_b = \begin{cases} 0, & avg < Q_{Tmin} \\ maxP[\frac{3(avg-Q_{Tmin})^2}{(Q_{Tmax}-Q_{Tmin})^2} - \frac{2(avg-Q_{Tmin})^3}{(Q_{Tmax}-Q_{Tmin})^3}], & avg \epsilon [Q_{Tmin}, Q_{Tmax}) \\ 1, & Q_{Tmax} \leq avg \end{cases} \quad (11)$$

Upper threshold RED or (URED) algorithm presented by Patel in [28] performs the process of computing the drop probability by-means-of using double linear functions, however, at different traffic loads to gain an improved performance in terms of throughput and packet loss rate metrics.

The self-adaptive random early detection or (SARED) algorithm presented by Adamu *et al*. in [7] introduced a self-adaptive mechanism into the operations of RED such that it performs the process of computing the drop probability by-way-of nonlinear function when $avg \epsilon [Q_{Tmin};Q_{Tmax})$ at low and moderate traffic load conditions. However, when the traffic load gets heavy, FXRED performs the process of computing the drop probability by-use-of a linear function.

In [29], the double slope RED or (DSRED) was proposed to perform the process of computing the drop probability by using a mixture of two linear functions. The slopes for these two linear dropping functions are complementary and could be adjusted by using the mode selector γ. DSRED was reported to outperform RED in terms of queuing delay, normalized throughput, and packet drop.

In [10], linear RED or (LRED) utilizes an adaptive average queue size, which is a simplified congestion measure and a linear drop function that has a reduced number of operators. This way, RED's computational cost was reduced. It can therefore be deduced from the discussion so far that the effectiveness of an AQM mechanism is dictated, at least to a large extent by the dropping function implemented [30].

Unlike various highlighted existing RED enhancement algorithms, we propose a minimal adjustment to the packet dropping probability profile of RED algorithm. As it turns out, the proposed double function random early detection (DFRED) algorithm simply substitutes RED's linear drop function with a mixture of power-of-two drop function and a power-of-one drop function to gain a more effective congestion control. Other attributes of original RED are kept unchanged.

3. Double Function RED (DFRED) Algorithm

In this section, we present details of another nonlinear enhancement to RED, termed Double Function Random Early Detection (DFRED). DFRED's model depicted in Figure 1 distinguishes between a low and high traffic load scenarios. It exhibits the implementation of a

mixture of power-of-two drop function and a power-of-one drop function (i.e., quadratic and linear) both between Q_{Tmin} and Q_{Tmax} queue thresholds.

First, as is given in Eq. (12), DFRED performs the process of computing the drop probability by-way-of a power-of-two (or quadratic) function when the value of *avg* is between Q_{Tmin} and a new incorporated threshold, called *Target* (specified according to Eq. (13)) positions. Using this strategy, a low packet dropping probability will be gained at low traffic load to increase the network throughput.

$$P_b = 4maxP\left(\frac{avg-Q_{Tmin}}{Q_{Tmax}-3Q_{Tmin}}\right)^2 \tag{12}$$

$$Target = \left(\frac{Q_{Tmax}+Q_{Tmin}}{2}\right) - Q_{Tmin} \tag{13}$$

Secondly, as is given in Eq. (14), DFRED performs the process of computing the drop probability by-means-of a linear function particularly when the value of *avg* is between *Target* and Q_{Tmax} threshold positions. Using this linear growth approach, a high packet dropping probability will be gained at high traffic load to decrease network delay.

$$P_b = maxP + 2(1-maxP)\left(\frac{avg-Target}{Q_{Tmax}+Q_{Tmin}}\right) \tag{14}$$

It is easy to see that DFRED has three points in its packet dropping function curve: (Q_{Tmin}, 0), (*Target*, *maxP*), and (Q_{Tmax}, 1). Furthermore, Eq. (15) summarizes DFRED's strategy.

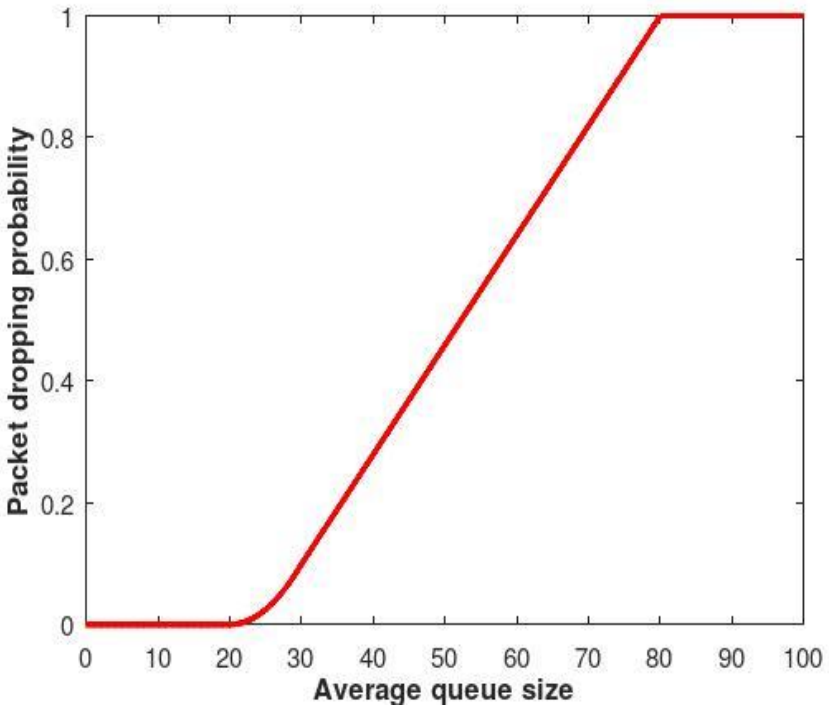

Figure 1: DFRED's dropping function

$$P_b = \begin{cases} 0 & avg < Q_{Tmin} \\ 4maxP\left(\frac{avg-Q_{Tmin}}{Q_{Tmax}-3Q_{Tmin}}\right)^2 & avg \in [Q_{Tmin}, Target) \\ maxP + 2(1-maxP)\left(\frac{avg-Target}{Q_{Tmax}+Q_{Tmin}}\right) & avg \in [Target, Q_{Tmax}) \\ 1 & Q_{Tmax} \leq avg \end{cases} \tag{15}$$

The pseudo-code for DFRED is expounded in Algorithm 1.

Algorithm 1 DFRED

1:Pre-decided parameters: Q_{Tmin}, Q_{Tmax}, *maxP*, and *w*
2:Set $avg = 0$
3:Set $Target = \left(\frac{Q_{Tmax}+Q_{Tmin}}{2}\right) - Q_{Tmin}$
4:For each packet arrival:
5:Calculate the average queue size avg(Eq. (1))
6:**if** ($avg = 0\ AND\ avg < Q_{Tmin}$) **then**
7:Accommodate the packet
8:**else if** ($avg \geq Q_{Tmin}\ AND\ avg < Target$) **then**
9:Implement the quadratic function (Eq. (12)) for the computation of P_b
10:Drop the arriving packet with the calculated probability
11:**else if** ($avg \geq Target\ AND\ avg < Q_{Tmax}$) **then**
12:Implement the quadratic function (Eq. (14)) for the computation of P_b
13:Drop the arriving packet with the calculated probability
14:**else if** $Q_{Tmax} \leq avg$ **then**
15:Packet is definitely dropped
16:**end if**

4. Simulation Results and Evaluation

The proposed DFRED algorithm is implemented using an open-source ns-3 simulator [31]. To investigate and demonstrate its effectiveness, simulation experiments were conducted based on the network topology shown in Figure 2 under two network traffic conditions namely, low and high loads. The performance of DFRED is compared to an existing algorithm - SmRED. Performance. Metrics of interest include (i) average queue size, (ii) delay, and (iii) throughput.

Referring to Figure 2, router A has the implementation for DFRED and SmRED algorithms (in a sequential pattern) while router B has implementation for DropTail algorithm. These routers perform a pivotal role of directing the flow of data packets. Individual sources (from S_1 to S_N) are connected to router A with 100 Mbps link capacity and 3 ms propagation delay. The two routers are connected with a bottleneck link having 10 Mbps as the capacity and 100 ms propagation delay. Router B is connected to the destination node (D) with 100 Mbps capacity and 3 ms propagation delay. TCP-NewReno algorithm was implemented. The buffer size is set to 250 packets. Total simulation duration is set to 100 seconds. Other parameter configurations are shown in Table 1. The values for Q_{Tmin} and Q_{Tmax} were both suggested by [15] while the values for *maxP* and *w* were both suggested by [13].

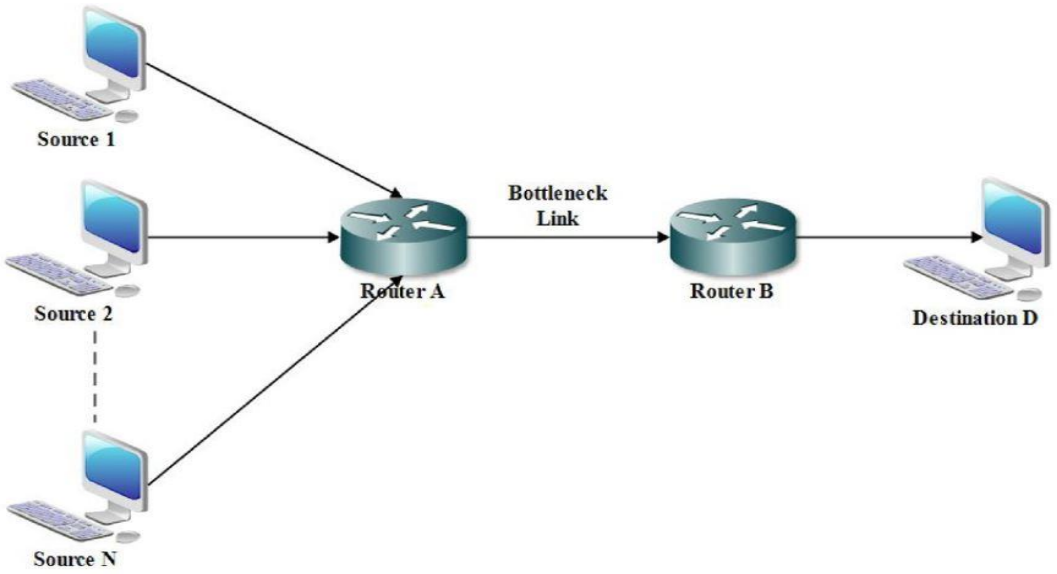

Figure 2: Network topology

Table 1: Simulation parameter settings

AQM algorithm	Parameter and value set
SmRED	Q_{Tmin}=20,*Target* =50, Q_{Tmax}=80, *maxP=0.1* and *w=0.002*
DFRED	Q_{Tmin}=20,*Target* =30, Q_{Tmax}=80, *maxP=0.1* and *w=0.002*

4.1 Scenario 1: Low traffic load

In this scenario, 5 TCP connections are involved. Table 2 presents the comparison between SmRED and DFRED algorithms based on the scenario under consideration.

Figure 3 shows the performance comparison of DFRED versus SmRED in terms of average queue size (in packets), delay (in milliseconds), and throughput (in Mb/s). Figure 3 (a) shows the change in average queue size over time for SmRED and DFRED algorithms. It can be easily seen from the figure that DFRED has a reduced and better control of the average queue size (which implies 51.7% reduction). Figure 3 (b) shows the delay. As seen in the figure, DFRED algorithm gained an outstanding performance (corresponding to 9.96% reduction) than SmRED with regards to delay. Figure 3 (c) shows the throughput. SmRED outperformed DFRED in terms of throughput (though at the expense of delay). Sm-RED has 14.15% gains in network throughput when compared to DFRED.

Table 2: Comparison of network performance under low traffic load

AQM algorithm	*Average queue size (Packets)*	*Delay (milliseconds)*	*Throughput (Mb/s)*
SmRED	12.2561	1.4065	9.4461
DFRED	5.9192	1.2664	8.2751

4.2 Scenario 2: High traffic load

50 TCP connections are involved in this scenario. Table 3 presents the comparison between SmRED and DFRED based on the scenario under consideration.

Figure 4 illustrates the simulation results for DFRED and SmRED in terms of average queue size (in packets), delay (in milliseconds) and throughput (in Mb/s). Figure 4 (a) shows the average queue size for both SmRED and DFRED algorithms. As seen in the figure, DFRED has the lowest average queue size (which implies a reduction by 60.1%). Figure 4 (b) depicts the delay of SmRED and DFRED algorithms. It can be observed that the proposed DFRED algorithm gained an outstanding performance than SmRED with regards to delay (which implies a reduction by 10.28%). Figure 4 (c) shows the throughput. SmRED outperformed DFRED in terms of throughput (though at the expense of delay).

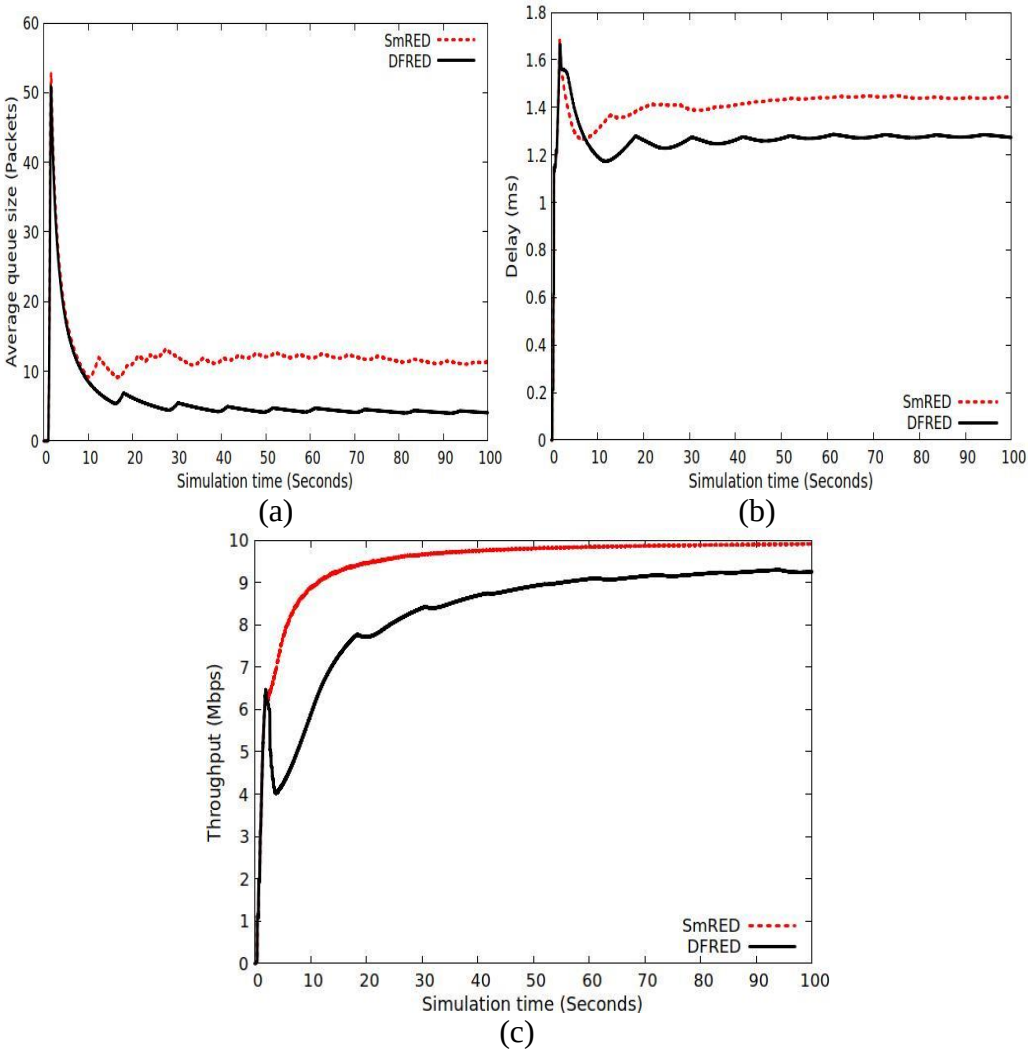

Figure 3: Comparison of DFRED with SmRED using average queue size (top left), delay (top right), and throughput (bottom) under low load condition

Table 3: Comparison of network performance under high traffic load

AQM algorithm	*Average queue size (Packets)*	*Delay (milliseconds)*	*Throughput (Mb/s)*
SmRED	35.5747	14.7500	9.8207
DFRED	14.9915	13.2337	9.3040

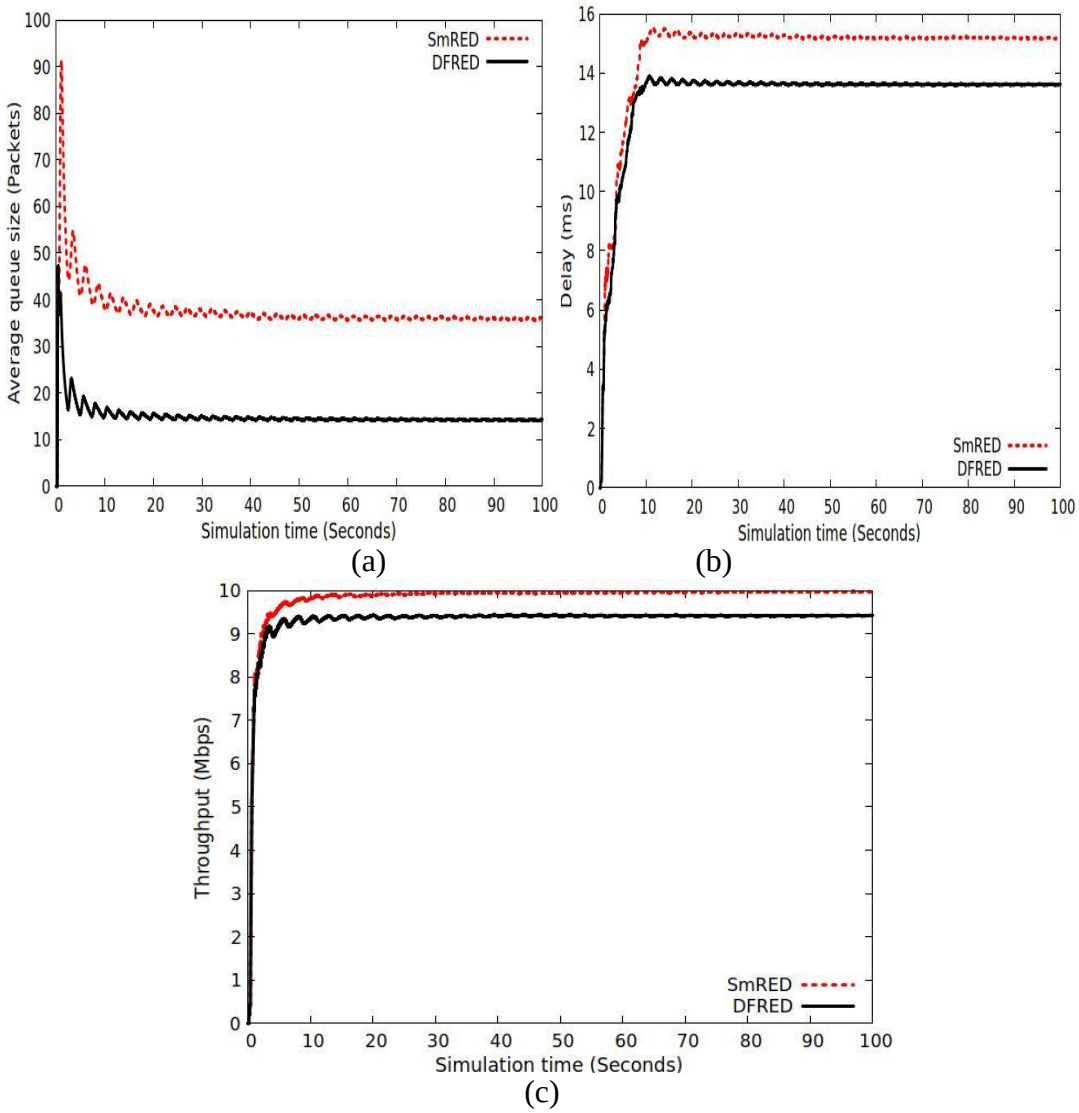

Figure 4: Comparison of DFRED with SmRED using average queue size (top left), delay (top right), and throughput (bottom) under high load condition

5. Conclusion

In this paper, DFRED AQM algorithm is proposed for Internet routers. DFRED modifies the dropping strategy of RED by employing a mix of power-of-two and power-of-one drop functions. DFRED algorithm was assessed and contrasted with SmRED (an enhanced RED algorithm) in a well-recognized ns-3 network simulator using performance metrics, such as average queue size, delay, and throughput. DFRED obtained shorter delay performance; however, this is at the expense of throughput under various network conditions. Therefore, the proposed DFRED could be easily implemented in Internet router as a replacement for RED even as the packet dropping probability profile moderately needs to be adjusted.

6. Disclosure and conflict of interest

The authors declare that they have no conflict of interest.

References

[1] S. S. Bisoyi, P. Mishra, and S. N. Mishra, "High Frequency Rule Synthesis in a Large Scale Multiple Database with MapReduce", International Journal of Electronics and Telecommunications, vol. 68, no. 2, pp. 177–186, 2022. (https://doi.org/10.24425 /ijet.2022. 139865).

[2] M. Q. Sultan, M. H. Jaber, and S. W. Shneen, "Proportional-Integral genetic algorithm controller for stability of TCP network", *International Journal of Electrical and Computer Engineering* vol. 10, no. 6, pp. 6225 – 6232, 2020. (https://doi.org/10.11591/ijece.v10i6.pp6225-6232).

[3] L. Pei, F. Wu, and S. Wang, "Periodic, Quasi-Periodic and Chaotic Oscillations in Two Heterogeneous AIMD/RED Network Congestion Models with State-Dependent Round-Trip Delays", *International Journal of Bifurcation and Chaos, vol.* 31, no. 6, pp. 2150124-1 – 2150124-21, 2021. (https://doi.org/10.1142/S0218127421501248).

[4] S. Sunassee, A. Mungur, S. Armoogum, and S. Pudaruth, "A Comprehensive Review on Congestion Control Techniques in Networking", 2021 2th International Conference on Computing Methodologies and Communication (ICCMC), pp. 305–312, 2021.

[5] A. Ahmed and N. Nasrelden, "New Congestion Control Algorithm to Improve Computer Networks Performance" In: Proceedings of the 28th International Conference on Innovative Trends in Computer Engineering (ITCE 2018), pp. 87–93, 2018. (https://doi.org /10.1109 /ITCE.2018.8316605).

[6] 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. (https://doi.org/10.1504/ijcat.2019.100298).

[7] A. Adamu, Y. Surajo, and M. T. Jafar, "SARED: Self-Adaptive Active Queue Management Scheme for Improving Quality of Service in Network Systems", *Journal of Computer Science,* vol. 22, no. 2, pp. 253 – 267, 2021. (https://doi.org/10.7494/csci.2021.22.2.4020).

[8] S. Ryu, C. Rump, and C. Qiao, "Advances in Internet Congestion Control. IEEE Communications Surveys and Tutorials" *IEEE Communications Surveys and Tutorials, vol.* 5, no. 1, pp. 28–39 2003.

[9] K. Zhou, K. L. Yeung, and V. O. K. Li, "Nonlinear RED: A Simple Yet Efficient Active Queue Management Scheme", *Journal of Computer Networks,* vol. 50, no. 18, pp. 3784-3794, (https://doi.org/10.1016/j.comnet.2006.04.007).

[10] A. A. Abu-Shareha, B. Al-Kasasbeh, Y. Qusai, Q. Y. Shambour, M. M. Abualhaj, M. A. Alsharaiah, and S. N. Al-Khatib, "Linear Random Early Detection for Congestion Control at the Router Buffer" *Informatica,* vol. 46, pp. 105–114,2022. (https://doi.org/10.31449/inf.v46i5.3966)

[11] C. Brandauer, G. Iannaccone, C. Diot, T. Ziegler, S. Fdida, and M. May, "Comparison of tail drop and active queue management performance for bulk-data and web-like internet traffic" In: Proceedings of the Sixth IEEE Symposium on Computers and Communications, pp. 122–129 2001. (https://doi.org/10.1109/ISCC.2001.935364)

[12] B. Braden, D. Clark, J. Crowcroft, B. Davies, S. Deering, D. Estrin, S. Floyd, V. Jacobson, G. Minshall, C. Patridge, L. Peterson, K. Ramakrishnan, S. Shenker, J. Wroclawski and L. Zhang. "Recommendations on Queue Management and Congestion Avoidance in the Internet" *RFC 2309,* 1998. (https://doi.org/10.17487/rfc2309)

[13] S. Floyd and V. Jacobson, "Random Early Gateway for Congestion Avoidance", *IEEE/ACM Transactions on Networking,* vol. 1, no. 4, pp. 397-413, 1993. (https://doi.org/10.1109 /90. 25 1892)

[14] Y. Zhang, J. Ma, Y. Wang, and C. Xu, "MRED: An Improved Nonlinear RED Algorithm" In: Proceedings of the 2011 International Conference on Computer and Automation Engineering (ICCAE 2011), vol. 44, no. 2, pp. 6–11, 2012. (https://doi.org/10.7763/IPCSIT.2012.V44.2).

[15] A. K. Paul, H. Kawakami, A. Tachibana and T. Hasegawa "An aqm based congestion control for enbrlc in 4g/lte network", *IEEE Canandian Conference on Electrical and Computer Engineering (CCECE),* 2016. (https://doi.org/10.21109/CCECE.2016.7726792).

[16] 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, 2020. (https://doi.org/10.1155/2020/8090468)

[17] I. Ryoo and M. Yang, "A State Dependent RED: An Enhanced Active Queue Management for Real-Time Interactive Systems", *Journal of IEICE Transactions on Communications,* vol. E89-B, no. 2, pp. 614 – 617, 2006.

[18] A. F. Al-Allaf and A. I. A. Jabbar A. I. A., "RED with Reconfigurable Maximum Dropping Probability", *International Journal of Computing and Digital Systems*, vol. 8, no. 1, pp. 61-72, 2019. (https://doi.org/10.12785/ijcds/080107)

[19] 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. (https://doi.org/10.3966/160792642018081904019)

[20] S. Prabhavat, R. Varakulsiripunth, and S. Noppanakeepong, (2002). "Throughput Improvement on RED Mechanism". *8th International Conference on Communication Systems, IEEE*, vol.1, pp. 599-603, 2002.

[21] S. Floyd, "Recommendation on using the "gentle" variant of RED" Available online: http://www.icir.org/oyd/red/gentle.html, 2000.

[22] M. Baklizi, H. Abdel-jaber, S. Ramadass, N. Abdullah, and M. Anbar, "Performance Assessment of AGRED, RED and GRED Congestion Control Algorithms" *Information Technology Journal*, vol. 11, no. 2 pp. 255–261, 2012. (https://doi.org/10.3923/itj.2012.255.261).

[23] C. Suwannapong and C. Khunboa, "Congestion Control in CoAP Observe Group Communication" *Sensors*, vol. 19, no. 3433, pp. 1–14, 2019. (https://doi.org/10.3390/s19153433)

[24] C. Suwannapong and C. Khunboa, Observe Group Communication" *Journal of Ad Hoc Networks*, vol. 112, pp. 1–10, 2021. (https://doi.org/10.1016/j.adhoc.2020.102377)

[25] M. Baklizi, "Stabilizing Average Queue Length in Active Queue Management Method" *International Journal of Advanced Computer Science and Applications* 10(3), 77–83 (2019)

[26] A. Adamu, V. Shorgin, S. Melnikov, and Y. Gaidamaka, "Flexible Random Early Detection Algorithm for Queue Management in Routers", *LNCS*, vol. 12563, pp. 196-208, 2020. (https://doi.org/10.1007/978-3-030-66471-8 16).

[27] Y. Bie, Z. Li, Z. Hu, and J. Chen, "Queue Management Algorithm for Satellite Network Based on Traffic Prediction" *IEEE Access*, vol. 10, pp. 54313–54324, 2022. https://doi.org/10.1109/ACCESS.2022.3163519

[28] C. M. Patel, "URED: Upper threshold RED an efficient congestion control algorithm," *2013 Fourth International Conference on Computing, Communications and Networking Technologies (ICCCNT)*,Tiruchengode, pp. 1-5, 2013. (https://doi.org/10.1109/ICCCNT.2013.6726469)

[29] B. Zheng and M. Atiquzzaman M., "DSRED: An Active Queue Management Scheme for the Next Generation Networks", *Proceedings 25th Annual IEEE Conference on Local Computer Networks, IEEE Computer Society*, pp. 242-251, 2000.

[30] Patel, S. and Karmeshu, "A New Modified Dropping Function for Congested Networks" *Wireless Personal Communication*, vol. 104, pp. 37–55, 2019. (https://doi.org/10.1007/s11277-018-6007-8)

[31] "The Network Simulator ns-3" https://www.nsnam.org