International Journal of Computing and Digital Systems
ISSN (2210-142X)
Int. J. Com. Dig. Sys. 8, No.1 (Jan-2019)

http://dx.doi.org/10.12785/ijcds/080107

RED with Reconfigurable Maximum Dropping Probability

Ahmad F. AL-Allaf[1] and A. I. A. Jabbar[2]

[1]*Dept. of Computer Eng., Technical college-Mosul, Northern Technical University, Mosul, IRAQ*
[2]*Dept. of Electrical Eng., College of Engineering, University of MOSUL, Mosul, IRAQ*

Received 24 Jun. 2018, Revised 9 Nov. 2018, Accepted 2 Dec. 2018, Published 1 Jan. 2018

Abstract: The continued demand for improved quality of service for the Internet network is an incentive to find new methods and means to reduce the impact of congestion on the network. The Random Early Detection (RED) algorithm is one of the commonly used algorithms in Internet routers to reduce the problem of congestion. However, network performance and QoS cannot be improved based on fixed and predefined parameters in RED algorithms. Therefore, many researchers have looked to use the adjustment approach (adaptation methods) which does not depend on determining the level of congestion on predefined assumptions. In this paper, an improved RED algorithm for multimedia traffic is proposed by using reconfigurable approach to redefining one of the RED parameters, maximum dropping probability (max_p), with changing the traffic over the network. The proposed algorithm, named as "RED with Reconfigurable Maximum Dropping Probability (RRMDP)", aims to reduce the average queue size and then queuing delay time without sacrificing in the packet dropping rate and link utilization. OPNET simulation is offered to support the idea. The result of the simulation shows a significant improvement in the average queue size and queuing delay comparing to the RED and ARED algorithms.

Keywords: ARED, AQM, Maximum Dropping Probability, Reconfigurable Technique.

1. INTRODUCTION

Congestion Control has become an active search area due to the diversity and development of real-time and multimedia traffic. In the last decade, many multimedia applications have emerged, such as video and audio conferencing (VOIP), video on demand (VOD), video streaming and so on. These applications have spread and the number of Internet users has increased rapidly, which could lead to congestion. This increase in the complexity of the Internet creates difficulty for TCP to control congestion and improve the quality of Internet service (QoS) [1].

The congestion usually occurs in the packet switching network as a result of exceeding the number of packets requiring transmission of the number of packets that the network can manage. It can cause degeneration in network performance, such as a long end to end delays, and loss of high packets, with the possibility of collapse due to congestion in resending lost packets in TCP protocol [2]. The collapse of congestion which first discovered by John Nagel in 1984 [3], has alert researchers about the severity of congestion and the need to find methods to control congestion and queue management.

Tail Drop (TD) was one of the earliest strategies, applied by TCP networks, to solve the congestion problem. However, TD has many drawbacks[4] such as Full Queue, Lock Out, Global Synchronization, and Bias against bursty traffic. To overcome these problems, a set of methods called "Active queue management (AQM) [5]" have been developed mainly to control congestion in networks early, and achieve a high quality of service for traffic loads. Unlike the TD method, which usually starts dropping packets after the overflow of the router buffers, the AQM methods begin dropping packets in early stages, hence this enables the sources to reduce their transmitting rates early before the router buffers become completely full. Examples of these methods are RED [6], Adaptive RED (ARED) [7], Gentle (GRED) [8], Dynamic RED (DRED) [9], Random Exponential Marking (REM) [10], BLUE [11], Generalized Random Early Evasion Network (GREEN) [12], Proportional

E-mail: Ahmadfalih64@gmail.com, Aiajabbar1@yahoo.com

Integral (PI) [13] and others. In recent years, two new AQM methods (Proportional Integral controller Enhanced (PIE) [14,15] and Controlled Delay (CoDel) [16,17]) have been developed by the IETF AQM working group to solve the full buffer problem "bufferbloat" in the network by reducing the queuing delay in routers.

PIE uses a classic proportional-integral controller to preserve the average queuing delay close to the desired target level. CoDel was designed to improve the overall performance of the RED algorithm by addressing some of its essential misconceptions. Both algorithms try to keep configuration parameters to a minimum, self-tune, and control queue delay around a target value. Also, both exhibit high latency during transient congestion periods. Grigorescu et al. [18], evaluate the performance of PIE and CoDel algorithms in simulated rural broadband networks. The authors compare those two algorithms with the ARED. They noted that the performance of ARED is comparable to that of PIE and CoDel in constant capacity links. The results of simulations also show that PIE performs better than CoDel in terms of packet loss rates affecting video quality.

Some of the equipment designers believe that they should implement one of these new algorithms, to make their AQM easier to manage. However, European Commission recommended in its project named RITE (Reducing Internet Transport Latency)[19], that improving already existing AQM such as RED better than start on a complete new AQM reimplementation. Since AQM algorithms are usually implemented in hardware, and reimplementation of a new algorithm can involve considerable development cost and upheaval. Further, there is a little performance difference between different AQMs—the solely existence of any AQM at all gives most of the benefit compared to no AQM. Then the advantages of developing AQM can be made available quickly to all existing users, without having to wait for the next round of hardware purchases[19]. Kuhn et al.[20] conclude in their paper, that there is no single overall best AQM scheme, as each scheme proposes a specific trade-off.

In general, AQM algorithms can reduce the number of packets dropped by routers and provide greater capacity to absorb traffic bursts without dropping packets while reducing end to end packets delays [21]. One of the early methods of AQM is the RED method proposed in the early 1990s by Floyd and Jacobson[22] and was subsequently adapted by the IETF. Though it has outstanding advantages [6], several drawbacks are clarified. The performance of RED is sensitive to the traffic load and to its parameters and poor adaptation to highly dynamic network loads [23]. Also, the RED performance is not good when the average queue length is close to the maximum threshold, where the maximum dropping probability increases sharply to 1. The RED mechanism was originally developed to work in conjunction with transport-layer congestion control protocols such as TCP. However, X. Wang [24] concludes in his thesis, that RED is also able to control the queuing delay and jitter of UDP based traffic under different load conditions and can provide a good solution to quality degradation of real-time traffic under congested network.

This paper proposes a modified RED algorithm, named "RED with Reconfigurable Maximum Dropping Probability (RRMDP)", for improving the QoS of real-time traffic over IP network. The proposal adopts the point of view that says: It is difficult to get improved performance and the required QoS by depending on a predefined, fixed parameters of the RED algorithm. Therefore, the proposed model reconfigures, dynamically, the maximum dropping probability parameter of the RED algorithm depending on the traffic load. It aims to enhance the RED performance with respect to the queuing delay and the average queue size with minimal changes to the original RED.

2. PREVIOUS WORKS

One of the most important RED algorithm problems is the sensitivity of the algorithm to its parameters minimum threshold (min_{th}), the maximum threshold (max_{th}), the maximum packet dropping probability (max_p), and queue weight (q_w)). Many papers suggest different methods to address this problem.

Feng et al. [25] have proposed a three-section RED (TRED) to overcome the issue of RED parameters setting and their tuning, particularly for heavy loads. In TRED, the queue size limits (maximum and minimum thresholds) was divided into three equal sections. The, dropping probability is calculated according to the dropping function used in each of the three different sections. Nonlinear dropping functions are used in first and third sections of the queue size limit, while the second section uses a linear dropping function. TRED is able to improve the throughput at low loads and maintains low delays at high loads. Alemu et al.[26] change maximum dropping probability dynamically by a multiplicative factor to improve the predictability of performance measures such as queue delay and delay jitter. The multiplicative factor is equal to the average weighted queue size at the current interval (*ˆKcur*) on the average weighted queue size value at the previous interval (*ˆKprev*): *change rate = ˆKcur / ˆKprev.*

Chen et al. [27], proposed the RED-Restraint algorithm, to maintain the queue length around a stable target value. When the current queue length value is far from the target value, the RED-Restraint algorithm adjusts the dropping probability to stabilize the queue size at the target value. It differs from the original RED using the actual queue length instead of the average queue length in RED. In [28], A. Dhamodaran et al. proposed an Adaptive Queue Management Scheme for Flexible Dual TCP/UDP Streaming Protocol to improve the Quality of Experience (QoE). This work use additive-increase/multiplicative-decrease (AIMD) algorithm based on the UDP packet loss rate and the TCP rebuffering to adaptive adjusting the TCP threshold.

AL-Raddady et al.[29], propose a model for adaptive RED which depends on varying the slope of the dropping probability according to the variation of incoming arrival rate and average mean queue length. The simulation results show a similar throughput, lower delay compared with RED and ARED and lower variation in average queue size than ARED and RED for a specific target delay. A Q-learning method, enhanced with fuzzy inference, was used by Masumzadeh et al. [30] to provide RED with self-adjustment and improved system performance. The proposed system is a congestion control approach based on the fuzzy system, with the ability to adapt the traffic to address the key issue of the RED algorithm, a parameter sensitivity.

A proportional-differential-type feedback controller called Novel-PD was proposed by Bisoy et al. [31], as new active queue management (AQM) to adjust the queue length with small oscillation. It also regulates the packet drop probability dynamically by using the measured current queue length and the differential error signals. The self-converting RED algorithm was developed in [32] by Feng et al. to modify the max_p parameter based on the average queue size. If the average queue size (avg_Q) fluctuates around a min_{th}, the max_p value is reduced by a constant value to minimize dropping of the packets. Conversely, if avg_Q fluctuates around max_{th}, the max_p value is increased to increase dropping of packets so as to maintain avg_Q between min_{th} and max_{th}. As an extension of the RED self-configuration, the Adaptive RED (ARED) was developed by Floyd et al. [33] to improve the performance of RED and reduce fluctuations in queue size. Unlike the RED self- configuration, ARED designed to maintain an avg_Q in the target range between min_{th} and max_{th}, and therefore, max_p changed in constant values accordingly.

To prevent the queuing delay from growing beyond a critical value over large RTT paths, Nicolas et al.[34] propose a Maximum and Average queuing Delay with PIE (MADPIE) scheme . This proposal extending PIE scheme by adding deterministic drops at controlled intervals. This proposal roughly simulating CoDel's drop policy. FQ Codel [35] is a hybrid scheme combining flow scheduling with active queue management that aims to reduce queueing delay. It consists of a set of queues; one queue per flow based on CoDel. A deficit round-robin scheduler is used to decides from which queue a packet should be dequeued. The algorithm works to preserve the queue size at a small value and providing isolation for low-rate traffic.

Patel et al. [36] proposed an adaptive queue management algorithm with random dropping (AQMRD). This work incorporates information about the average queue size and its rate of change to adaptively changing threshold level that falls in between the minimum and maximum thresholds. Kamal et al.[37] propose an adaptive method for dynamic threshold adjustment in the RED algorithm to increase the throughput and reduce the packet loss and delay. The minimum threshold was set using an expression that was derived for a given burst size, while the maximum thresholds are changed dynamically based on traffic conditions and buffer size. Based on traffic condition, the thresholds are adjusted dynamically and therefore they call the algorithm "Adaptive RED with Dynamic Thresholds Adjustment (ARDTA)".

Finally, In [38], I. Jarvinen et al., compare the performance of the CoDel and PIE, against a variant RED algorithm called HRED (Harsh RED). They study the AQM behavior during load transients with the common traffic types of today such as Web transactions. They found that CoDel auto-tuning does not scale well with the load. Also, they found that PIE with default parameters does not work with low-rate links because PIE fails to measure departure rate within a single update interval. The authors discover that HRED is better than PIE and CoDel when more than a few simultaneous flows share the bottleneck link.

3. RANDOM EARLY DETECTION

RED as one of the widely used active queue management algorithms, proposed by S. Floyd[6], and recommended by IETF [39]. It aims to avoid congestion by controlling the average queue size at the router; avoid global synchronization, and to react against bursty traffic. The RED congestion control mechanism observes the avg_Q for the output queue and sends feedback to responsive flows before the queue overflows to notify them before the congestion starts. It uses two methods to notify these sources of traffic: By dropping packets or

marking packets. When a new packet arrives, the router calculates the average queue size and decides to mark/drop the packet with a dropping probability. The dropping probability P_d is a linear function based on the average queue size and the queue thresholds. The average queue size (avg_Q) is computed using a low pass filter with an exponential weighted moving average(EWMA), as shown in (1):

$$avg_Q = (1 - w_q) * avg_Q + w_q * q \quad (1)$$

Where avg_Q and q are the average queue length and instantaneous queue length respectively and w_q is the weight of average queue size. The calculated average queue size is then compared with two thresholds: a minimum and a maximum threshold (min_{th} and max_{th}) in order to compute the marking/dropping probability P_d according to the following equation[6]:

$$p_d = \begin{cases} 0 & avg_Q \le min_{th} \\ 1 & avg_Q \ge max_{th} \\ \left(\frac{avg_Q - min_{th}}{max_{th} - min_{th}}\right) * max_p & min_{th} < avg_Q < max_{th} \end{cases} \quad (2)$$

max_p is the maximum packet dropping probability. The probability that a packet is marked from a particular connection is roughly proportional to that connection's share of the bandwidth at the router. As avg_Q varies from min_{th} to max_{th}, the packet-marking probability p_d varies linearly from 0 to max_p. The final drop-probability P_a is calculated as:

$$P_a = P_d / (1\text{-}count * P_d) \quad (3)$$

where the *count* is the number of packets since last marked/dropped packet. The final packet dropping probability p_a slowly increases with increasing *count* since the last dropped packet, in order to prevent consequent dropping of packets and ensuring a fair distribution of the dropping on the flows. The recommended setting for RED parameters are[6]: max_{th} is set to at least $2*min_{th}$, max_p=0.1, and w_q = 0.002, which they believe can fit any traffic characteristics.

The studies on RED performance revealed that although RED can improve TCP performance under certain parameter settings and network circumstances, the basic RED algorithm is still subject to several problems, such as sensitivity to its control parameters, in addition, the bandwidth unfairness, and low throughput[37]. To overcome the problems of the RED algorithm, researchers proposed several variants of RED.

4. ADAPTIVE RED

One of the ways to improve the performance of RED and to overcome the problem of RED sensitivity to its parameters is the use of adaptivity which has been the subject of broad research studies ever since RED was proposed. The most well-known proposal is Adaptive-RED (ARED)[33] proposed by Floyd et al. which is an improved version to the original ARED[32]. Unlike RED, the ARED attempts to stabilize the average queue size around a targeted value through a dynamic modification of the maximum packet dropping probability (max_p) parameter, in fixed step size at a fixed time interval. Based on the observed average queue length, the max_p is adapted to maintain the queuing delay within a target range and to keep the target queue size within a range between min_{th} and max_{th}. Through simulation, it has been demonstrated that ARED tends to be more stable than RED. This leads to a more predictable and consistent Round Trip Time (RTT) and therefore a smaller variance in the RTTs[7]. The flowchart of the ARED algorithm is shown in Fig. 1, where the time slot is set to 0.5 seconds, and the target average queue size (T_{avgQ}) is restricted to the following interval: {min_{th} + 0.4*(max_{th}-min_{th}), min_{th} + 0.6*(max_{th}-min_{th})}.

avg_Q is computed as in the RED. According to ARED algorithm, when the avg_Q is larger than the target queue value, the ARED algorithm becomes more aggressive by increasing the max_p by a fixed additive factor α, with a default value equal to *min (0.01, max_p/4)*, so that avg_Q decreased and then become close to the target value. Conversely, when the avg_Q is smaller than the target queue value, the ARED becomes more

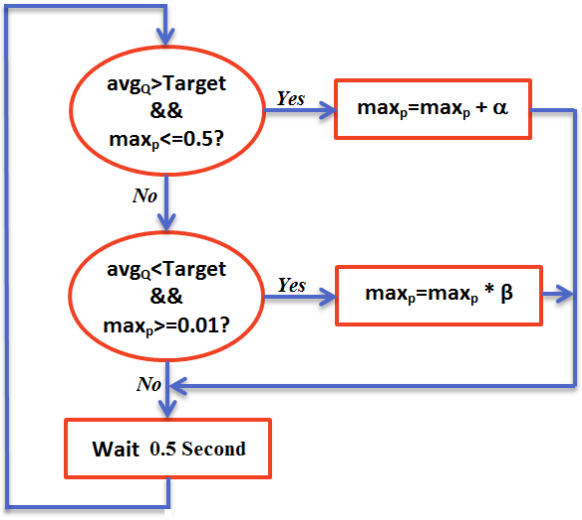

Figure1. Flowchart of max_p in the ARED algorithm[40]

conservative by decreasing the max_p by a constant multiplicative factor β, with a default value equal to 0.9, so that the sources can send more packets and then the avg_Q is increased toward the *target* queue value.

In the original RED paper, there are strategies about how to select the values of these parameters which depend on the delay rather than traffic load. ARED is less sensitive to the parameter than the other versions of RED and requires a little change in the original RED design. However, the setting of α or β values are to prevent the changing in average queue length from above to below the target range or from below to above the target range, respectively, within a single slot.

5. PROPOSED RRMDP ALGORITHM

RED's main drawbacks are the difficulty of controlling of its parameters w_q, max_{th}, $min_{th,}$ and max_p. Studies on this subject indicate the sensitivity of RED parameters to the congestion level [26]. However, some researcher[32, 41] have concluded that there is no single set of RED parameters that operate under different traffic scenarios. The choice of the weight parameter (w_q) in the calculation of the avg_Q would depend on the traffic profile. Choosing a small value for w_q gives a low weighting to the value of the current queue and so tends to smooth and burstiness in the traffic. On the other hand, Choosing a relatively large value for w_q gives a more sensitive response to a bursty profile.

The max_p and max_{th} are important parameters because they are responsible for determining the intensity of the congestion control in the RED algorithm and have a direct effect on the calculation of the P_d. In addition, the performance of RED is greatly influenced by the choosing of the max_p. If maxp is too small, the number of packet drops becomes less and hence, cannot prevent the queue overflow. If max_p is too high, the number of packet drops becomes large and significantly affects the throughput. Therefore, several approaches have been proposed to reduce RED's sensitivity to its parameters. One approach is to use quantitative models as in [42], to predict the RED parameters based on network variables such as RTT, number of flows and link bandwidth. Other works like [43] use a control theoretic analysis to model the parameter setting of RED.

However, it is still difficult to retrieve accurate information about network variables from local notes because network load is generally unknown by the RED routers. Therefore, many researchers have begun to use the adaptive approach that they can adjust the RED parameters according to the traffic load and at the same time does not depend on the traffic type as well as it can infer the traffic load from the average queue size. This paper also uses the adaptation method to improve the RED performance. However, it differs from other work as in [26,32,33,44] in the way that the parameters are modified. In the proposed approach, a simple multiplicative factor called γ, represented by a linear equation, is added to reconfigure the max_p based on the traffic load and target average queue size, without significantly sacrificing in the packet dropping rate or throughput.

In RED, the probability of dropping P_d is linearly increased with the increase of average queue size between the minimum and maximum thresholds. When the min_{th} and max_{th} thresholds are fixed, the slope of the dropping probability line depends on the maximum dropping probability (max_p). In ARED, static adjustment factors are added to the max_p that do not reflect the actual change rate of the traffic load. Since when the change in the traffic load is slight, there is no need to apply a sharp change to the max_p that can cause oscillations in the queue size [26]. To achieve acceptable QoS we must maintain the average queue size (avg_Q) at a halfway, with graceful fluctuation between the min_{th} and max_{th} threshold values[40]. This also helps to be the probability of packets dropping is not very high.

The reconfiguration factor γ, that is used to reconfigure the maximum drop probability max_p depends on the:

- current average queue length that reflects the change in arrival rate,
- minimum and maximum thresholds of the queue, and
- target average queue length that set at the mid-way between the min_{th} and max_{th} thresholds as in[33].

Equation (4) and (5) shows the modification that was added to the dropping probability equation:

$$P_d = \left(\frac{avg_Q - min_{th}}{max_{th} - min_{th}}\right) * (max_p * \gamma) \quad (4)$$

$$\gamma = (4 + 8*(avg_Q - Target)/(max_{th} - min_{th})) \quad (5)$$

Where the Target is the target average queue size and equal (max_{th} + min_{th})/2. The choosing numbers 4 and 8 in (5) can be justified as follow: Assuming $max_{th}=2*min_{th}$ (as recommended in [6]), then Target = $3min_{th}/2$.

According to (4) and (5), if $avg_Q = min_{th}$, this makes the probability of dropping the packets (P_d) equal to 0. This result is consistent with the condition of (2)

($Pd = 0 \;\; if \;\; avg_Q \leq min_{th}$). On the other hand, if the $avg_Q = max_{th}$, then the max_p(RRMDP) = max_p (RED) * 8. If max_p (RED) = 0.1, then the value of max_p(RRMDP) will become 0.8. This case represents a heavy load state where the probability of dropping the packets jumps to 1(when $avg_Q > max_{th}$). Again, this result is consistent with the condition of (2) ($Pd = 1 \;\; if \;\; avg_Q \geq max_{th}$). Note that in other research, the value of max_p reaches to 0.75 in [26], or 0.5 in [33].

However, this upper bound situation in the proposed model, will not occur as the avg_Q is stabilized around the target average queue size. Finally, when the avg_Q equal the target, the multiplier factor (see (5)) will be 4 which set max_p(RRMDP) to 0.4. By this modification, the change in the value of the avg_Q is directly translated by a factor γ as a change with the same amount in the value of the max_p. Since the minimum and maximum thresholds and target average queue length are set as a constant then the slope of the dropping probability curve varies depending on the change in the average queue size (incoming arrival rate).

The proposed model defers from the adaptive RED (ARED) as follows:
It has the advantage that it uses less number of parameters than ARED model, which is the main problem in the original RED algorithm. In addition, the proposed algorithm updates the maximum dropping probability with the arrival of each new packet while the adaptive RED algorithm performs the update at fixed intervals. Another difference is that ARED uses fixed values in the increment or decrement the maxp in each update, while in the proposed algorithm, the increment or decrement is adaptive and depends on the average queue size. Except the modification described above, other details and parameter setting of RRMDP are the same as an original RED algorithm.

The proposed RRMDP calculates changes in the maximum dropping probability (if any) after each arrival on a basis of the linear equation reflecting the change in the slope of the probability(see (4)). However, it is possible to introduce the nonlinearity in (1) as shown in (6):

$$P_d = \left(\frac{avg_Q - min_{th}}{max_{th} - min_{th}}\right) * \left[(max_p * \gamma)\right]^n \qquad (6)$$

Where n is the degree of nonlinearity. Note that the first part of the P_d equation changes linearly from 0 to 1 when avg_Q changes from min_{th} to max_{th}, while the second part of the equation is also changing linearly, but from 0 to 0.8 when avg_Q changes from min_{th} to max_{th}, i.e., the slop is different from the second part. So the change in Pd (the multiplication of both parts) will be non-linear as shown in Fig. 2, assuming that min_{th}=100, max_{th}=300, T=(max_{th}+min_{th})/2=200, max_p=0.1.

Fig. 3 shows the effect of multiplying γ by the max_p (using the same setting for RED parameters as described above) at different nonlinearity degree (n):

$$(max_p * \gamma)^n = (0.1 * (4+8*(avg_Q-200)/200))^n \qquad (7)$$

Fig. 4 shows a MATLAB comparison of P_d with avg_Q for algorithms RED, ARED, and RRMDP (with different nonlinearity degrees), using the same setting for RED parameters as described above):

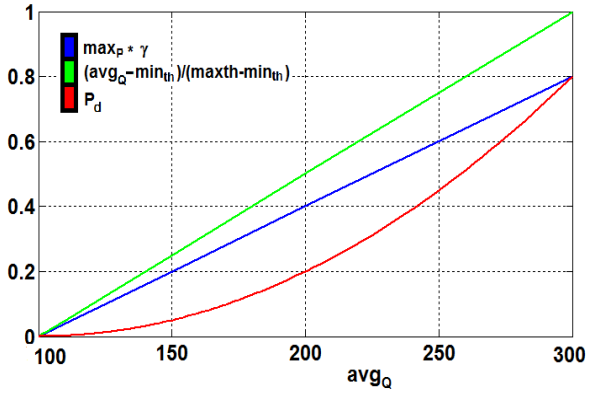

Figure 2. Effect of introducing the γ factor to the P_d equation

$$p_d = ((avg-100)/200) * (0.1*(4+8*(avgQ-200)/200))^n \qquad (8)$$

As is clear from the Fig. 4, although the area under the curve of P_d in RED is less compared in the case of ARED and RRMDP, in the case of RED, the avgQ approach to the max_{th} value before the case of ARED and RRMDP. In other words, the RED begins to drop packets at probability one before the ARED and RRMDP. The RRMDP reduces the average queue size and then the queuing delay more than ARED and RED, as will illustrated later. Also, it is noted that with increasing n to 2 or 3, the curve of P_d falls down leading to reducing the probability of dropping. While when n is one or less, the P_d value will be increased and the probability of dropping is increased.

6. Results and Discussions

The proposed algorithm is implemented in OPNET modeler 14.5 software and its performance is compared with the RED and ARED algorithms. However, RED is one of the default AQM algorithms in OPNET, whereas ARED is not.

Therefore, ARED and RRMDP have been included in the OPNET software as new AQM algorithms which require some changes in the OPNET's source code. The

simulations were done on the network topology shown in Fig. 5, consisting of five traffic sources, five sinks, and two routers. The link between the two routers is the bottleneck link. All the five traffic sources are connected to the router A using 10 Mb/s link.

The loads in the traffic sources are activated at different simulation time to study the performance of the proposed system under different traffic load and thus different levels of congestion on the bottleneck link. The parameters of RED, ARED, and RRMDP are set as min_{th}=100, max_{th}=300, w_q=0.002, and max_p=0.1. The max_{th} is set three times the min_{th} as recommended in[6].

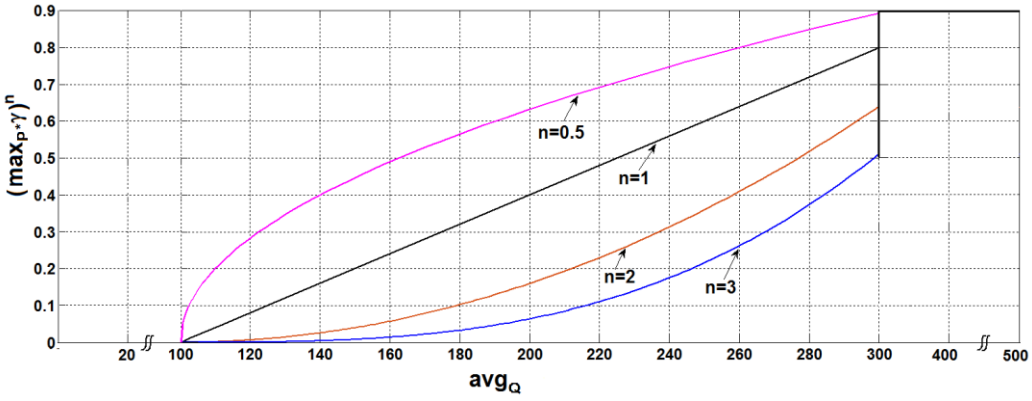

Figure 3. The relationship between $(max_p * \gamma)^n$ and avg_Q

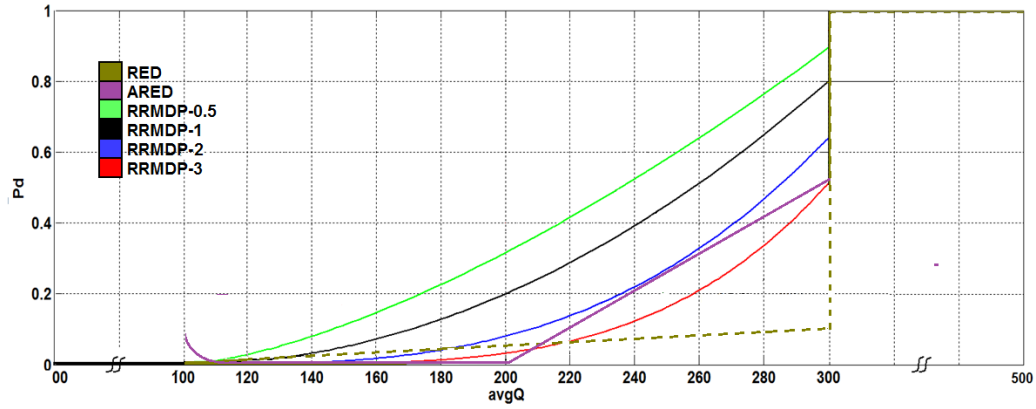

Figure 4. Dropping probability in RED, ARED, and RRMDP(at different nonlinearity order) evolution.

The traffic sources are only UDP traffic (videoconferencing traffic) with the characteristics as indicated in Table (1). Type of service is defined in the application properties which will be used in the simulation. There are eight types of services [5]: Best effort(lowest priority), Background, Standard, Excellent effort, Streaming multimedia, Interactive multimedia, Interactive voice, and Reserved. These TOS field categories difference priority of data, that is a network device could process important data first and less important after that.

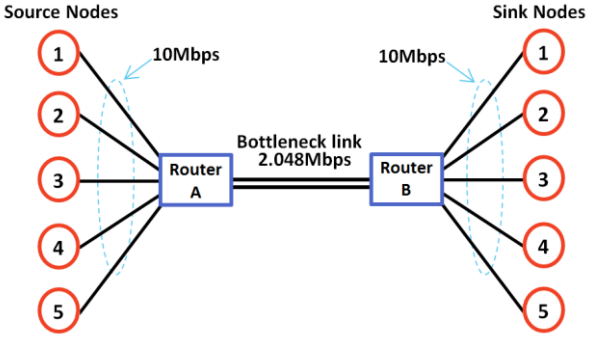

Figure 5. Network topology of the simulated scenarios

Figs. 6-10 show the results of simulations for the average queue size, delay, jitter, packet loss and the probability of dropping with the time when the traffic load increasing from %75 to %200 of the full load. In these figures, the regions labeled A-E represent the value of traffic load expressed as a ratio between the generating traffic load (bits/sec) to the capacity of the output link (Full load). Where; A=75%, B=100%, C=110%(light congestion status), D=150%, and E=200% (heavy congestion status). The simulations were implemented when the γ's equation is a linear (RRMDP-1) and nonlinear for the value of *n* equal to 0.5, 2 and 3, as RRMDP-0.5, RRMDP-2, and RRMDP-3, respectively.

The results of comparing the algorithms: RED, ARED, and RRMDP (with different nonlinearity degree) in the Figs. 6-10, demonstrate that the three algorithms give the same performance when the load is less than the full load. Fig. 6 shows that as the traffic load increases, RRMDP (at n=0.1 and n=1) tends to control the average queue size better than RED and ARED. The average queue size of the RED algorithm approaches quickly to the max_{th} values after the load increase beyond the full load. While in the ARED algorithm the average queue size stays below the max_{th} until the load is increased to double the full load. In the case of the proposed algorithm, the average queue size remains below the max_{th} values and around the target value even when the load is increased to double the full load. This means that the RRMDP algorithm (at n-0.5 and n=1) and by associating the change in max_p with the change in the load (expressed by the avg_Q) can maintain the average queue size around the target value even in the case of heavy congestion.

It is obvious from Fig. 7 that the queuing delay in RRMDP-1 (and RRMDP-0.5) algorithm was improved

TABLE 1 CHARACTERISTICS OF THE TRAFFIC SOURCES

Traffic source No.	1	2	3	4	5
Frame/sec	30 frame/sec	30 frame/sec	30 frame/sec	30 frame/sec	30 frame/sec
Frame size	6554 byte/frame	2185 byte/frame	874 byte/frame	3495 byte/frame	4369 byte/frame
Starting time(sec)	100	300	400	500	600
Ending time(sec)	700	700	700	700	700
Type of service	streaming traffic	excellent effort	standard	background	streaming traffic
Traffic in bit/sec	1572960b/s	524400b/s	209760b/s	838800b/s	1048560b/s
Accumulated traffic	1572960 b/s	2097360 b/s	2307120 b/s	3145920 b/s	4194480 b/s
%of full load	%75	%100	%110	%150	%200

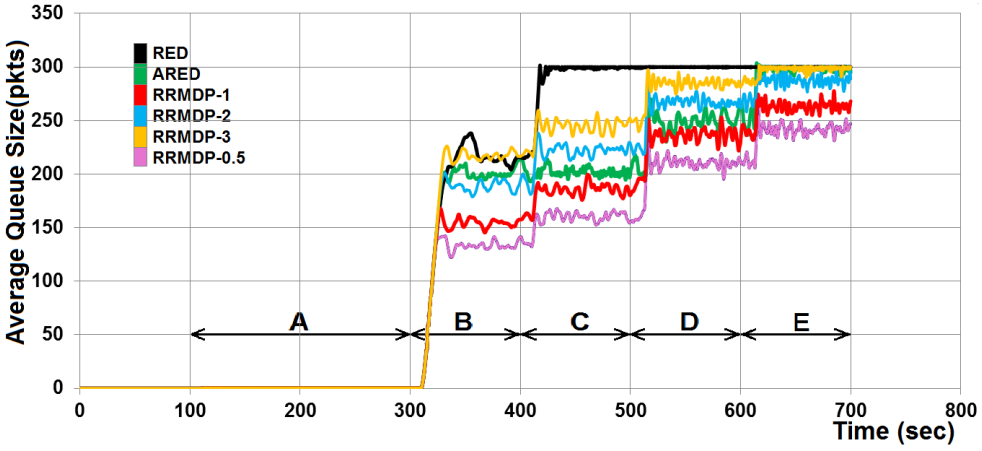

Figure 6. Average queue size (avg_Q) under different traffic load with the time

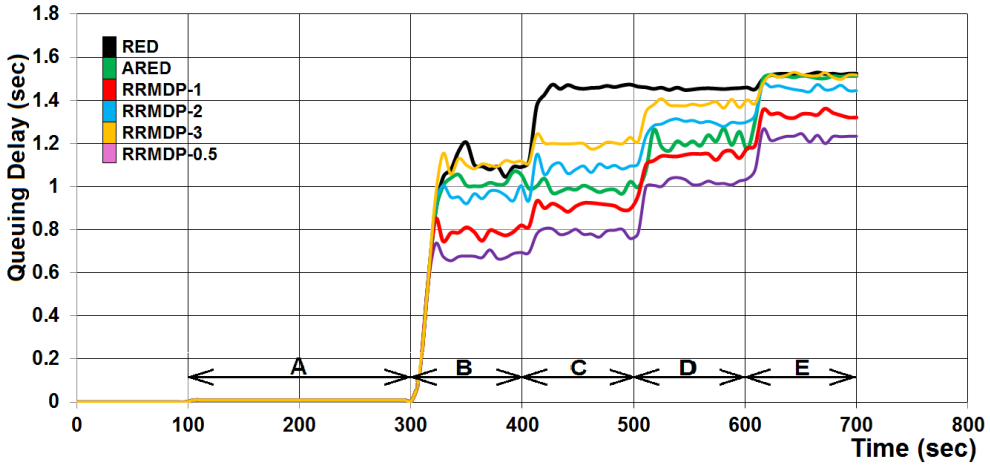

Figure 7. Queuing delay (sec) under different traffic load with the time

compared to the RED or ARED algorithms. This is because of that avg_Q in the RRMDP-1 (and RRMDP-0.5) algorithm is better controlled than either RED or ARED.

On the other hand, regardless of the congestion situation, the proposed algorithm shows a slight increase in the amount of packet dropping compared to the RED and ARED (see Fig. 8). The reason is that although the RED's router buffer drops packets at the beginning less than ARED and the RRMDP as shown from P_d curves in Fig. 4, it overflows as early as ARED and the RRMDP routers. In other words, the RED router starts dropping packets at a probability one before that of ARED and RRMDP.

Table 2 illustrate the simulation data of average packet drop under the three algorithms. It is known that performance in the queuing system will decline with the increased load. This certainly applies to queuing delay but from the surprising features of the delay jitter that this behavior may be not accurate [45, 46]. This can be seen from Fig. 9 where the jitter decreases when the load increases especially after traffic increased periods. This can be explained as follows:

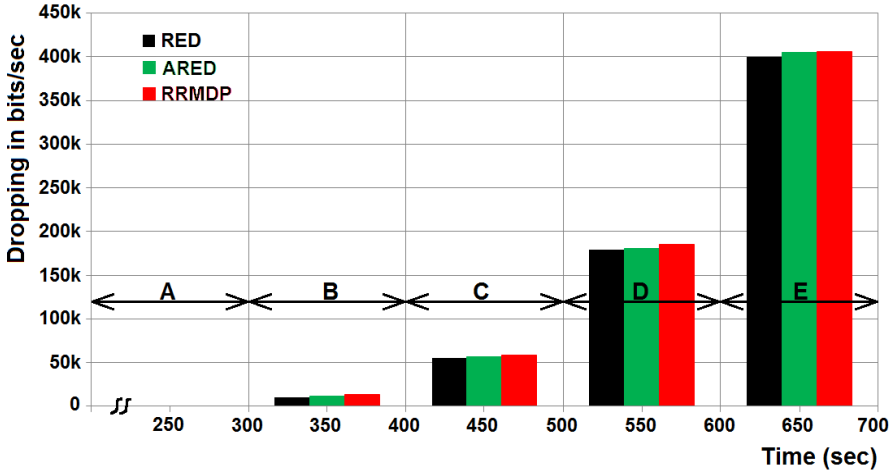

Figure 8. Average dropping (in bits/sec) under different traffic load with the time

TABLE 2 SIMULATION DATA OF AVERAGE PACKET DROP UNDER THE THREE ALGORITHMS

Time (sec)	% of Full Load	ARED-Drop (bits/s)	RRMDP-Drop (bits/s)	RED-Drop(bits/s)
0	75	0	0	0
350	100	11252.95	13241.64	10655.08
450	110	56372.33	58938.11	55509.1
550	150	180226.6	184895.4	178932.4
650	200	405397.3	405608.4	399503.5

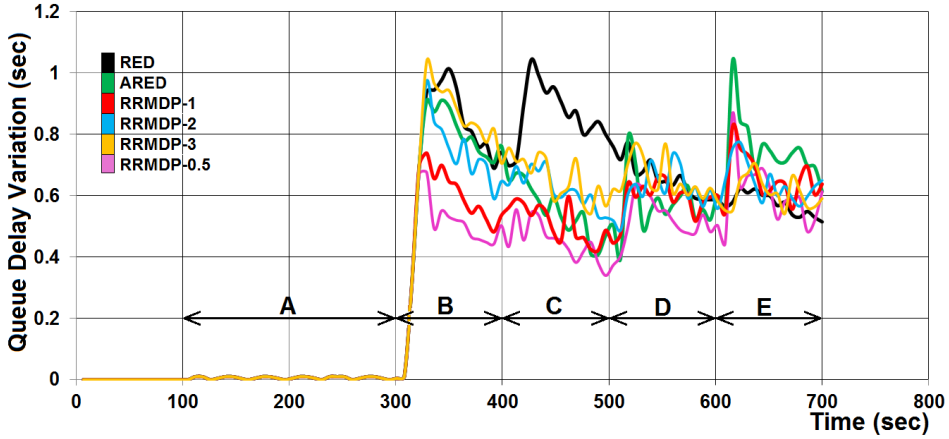

Figure 9. Queue delay variation (sec) under different traffic load with the time

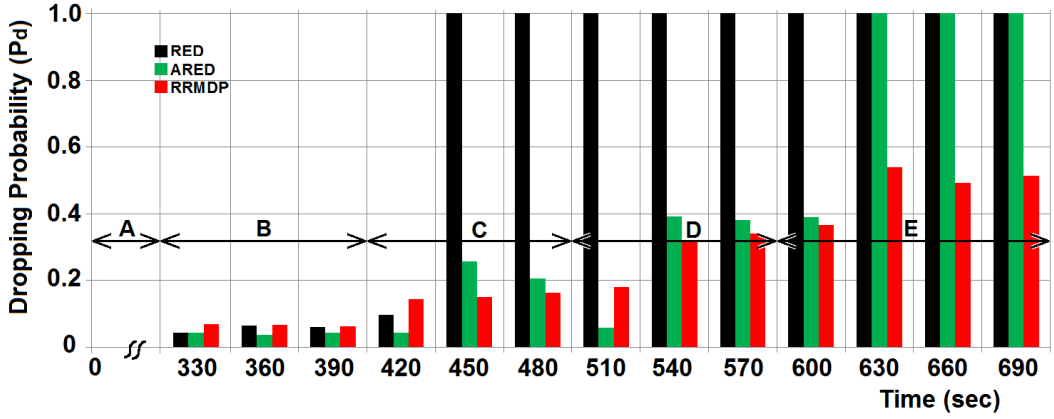

Figure 10. Dropping probability under RED, ARED, and RRMDP algorithms with the time

When the queue is almost empty, the arrived packets do not have to wait often (time before 300 sec in Fig. 9). Whereas when the queue is full, almost all packets must wait. In this case, most of the variation in delay (jitter) is due to service time only. In general, it is unlikely that the jitter, in this case, will be smaller than in an unloaded network. Also, the bursty effects at the beginning of every traffic change period will cause to sudden increase in the jitter, since many packets have to wait to be transmitted, and after that, the jitter starts to decrease gradually with the time, as depicted in Fig. 9.

Finally, the nonlinearity of the modified max_p allows more packet bursts to pass when the average queue size is small and drops more packets when the average queue size becomes large. This helps to improve the performance of the router. Note that there is an improvement in the delay and average queue size when n = 0.5. This improvement is reduced with increasing of n value, and this can be understood by looking at the shape of P_d curves in Fig. 5, which illustrate the change in P_d when the avg_Q varied smoothly from min_{th} to max_{th}.

Fig. 10 shows the dropping probability P_d with the time according to the scenario described above. RRMDP is successful in maintaining the same throughput as ARED and RED regardless of the traffic loading. Additionally, the utilization of the output link is 100% under all algorithm. Where throughput is defined as the number of bits or packets successfully received or transmitted over the link per second, while the link utilization represents the ratio between the amount of data carried on the link to the link's capacity[47]. This because these applications use UDP protocol which does not respond to the congestion notification, therefore UDP will continue to transmit at line rate regardless of latency or losses in the connection and the link's capacity is fully consumed.

7. CONCLUSION

In this paper, an improved Adaptive RED AQM mechanism is proposed which reconfigure the maximum dropping probability parameter (max_p) based on traffic load to enhance the network performance in terms of average queue size and queuing delay. The proposed algorithm, named RRMDP, is the same as the original

RED except that the max_p in packet dropping probability function is multiplied by a reconfiguration parameter (γ) that change max_p according to the traffic load expressed by average queue length. The results of the simulation show that the RRMDP has a more controllable average queue size, which lead to lower queuing delay without affecting the link utilization and throughput. This controllable average queue size keeps the router queue away from buffer overrun even in the case of severe congestion.The added reconfigurable parameter makes the packet dropping probability more gently at the onset of the congestion, and a much more aggressive dropping probability when the congestion becomes more prominent. In addition, the results of simulation show that the proposed algorithm reduces queue delay without significantly increasing the packet loss rate or decreasing the link utilization. Finally, the proposed modification is limited, making it easier to incorporate them into the RED routers with reduced cost comparing to the implementation of a new AQM algorithm.

REFERENCES

[1] V. Jacobson, "Congestion Avoidance and Control," in Proceedings of ACM/SIGCOMM, pp. 314-329, 1988.

[2] R. Alasem, "Intelligent Active Queue Management Schemes for Congestion Control in TCP/IP Networks," Ph.D. Thesis, University of Bradford, 2007.

[3] M. Welzel, "Network Congestion Control: Managing Internet Traffic," Wiley Series in Communications Networking & Distributed System, West Sussex, England, September 2005.

[4] J. Wang, An adaptive active queue management algorithm in Internet, M.Sc. thesis, Université Du Québec À Chicoutimi, 2006.

[5] R. Adams, "Active Queue Management: A Survey," IEEE Communications Surveys & Tutorials, Vol. 15, Issue 3, 2013.

[6] S. Floyd, and V. Jacobson, "Random Early Detection Gateways for Congestion Avoidance," IEEE/ACM Transactions on Networking, vol. 1, issue 4, pp. 397-413, August 1993.

[7] R. La, P. Ranjan, and E. Abed, "Analysis of adaptive random early detection (ARED)," Teletraffic Science and Engineering, Vol. 5, 2003, pp.1271-1280

[8] S. Floyd, "Recommendations on using the gentle variant of RED," May 2000, Available at http://www.aciri.org/floyd/red/gentle.html.

[9] J. Aweya, M. Ouellette, and D. Montuno, "A Control Theoretic Approach to Active Queue Management," Computer Net., vol. 36, issue 2-3, pp. 203-35, July 2001.

[10] S. Athuraliya, S. Low, V. Li., and Q. Yin, "REM: Active Queue Management," IEEE Network, vol. 15, no. 3, pp. 48-53, May 2001.

[11] W. Feng, K. Shin, and D. kandlur, "The Blue Active Queue Management Algorithms," IEEE/ACM Transactions on Networking, vol. 10, issue 4, pp. 513-528, August 2002.

[12] W. Feng, A. Kapadia, and S. Thulasidasan, "GREEN: Proactive Queue Management over a Best-Effort Network," The Proceeding of IEEE Global Telecommunications Conference, GLOBECOM '02, vol. 2, pp. 1774-1778, Taipei, Taiwan, November 2002.

[13] C. Hollot, V. Misra, D. Towsley, and W. Gong, "On Designing Improved Controllers for AQM Routers Supporting TCP Flows," *Proceedings of the IEEE INFOCOM 2001*, vol. 3, April, pp. 1726–1734. Anchorage, Alaska, USA.

[14] R. Pan, P. Natarajan, C. Piglione, M. Prabhu, V. Subramanian, F. Baker, and B. VerSteeg, "PIE: A lightweight control scheme to address the Bufferbloat problem," in Proceedings of IEEE HPSR, Taipei, Jul. 2013.

[15] R. Pan, P. Natarajan, F. Baker, and G. White, "Proportional Integral Controller Enhanced (PIE): A Lightweight Control Scheme to Address the Bufferbloat Problem." RFC 8033 (Experimental), Feb 2017.

[16] K. Nichols and V. Jacobson, "Controlling Queue Delay," ACM Queue, vol. 10, no. 5, pp. 20:20–20:34, May 2012.

[17] K. Nichols, V. Jacobson, A. McGregor, and J. Iyengar, "Controlled Delay Active Queue Management." RFC 8289, Jan 2018.

[18] E. Grigorescu, C. Kulatunga, and G. Fairhurst, "Evaluation of the impact of packet drops due to AQM over capacity limited paths," *Network Protocols (ICNP), 2013 21st IEEE International Conference on*. IEEE, 2013. p. 1-6.

[19] P. Hurtig, G. Fairhurst, B. Briscoe, K. Schepper, A. Petlund, N. Khademi, N. Kuhn, and I. Learmonth, RITE:Reducing Internet Transport Latency, European Commission, Project No.317700, November 16, 2015

[20] N. Kuhn, D. Ros, A Bagayoko, C. Kulatunga, G. Fairhurst, and N. Khademi, "Operating ranges, tunability, and performance of CoDel and PIE," Computer Communications, Vol.103, 2017, PP.74-82.

[21] F. Baker, & G, Fairhurst, IETF Recommendations Regarding Active Queue Management, (No. RFC 7567), 2015.

[22] S. Floyd, "RED: Discussions of Setting Parameters," November 1997. Available at:

http://www.aciri.org/floyd/REDparameters.txt.

[23] M. Christiansen, K. Jeffay, D. Ott and F. Smith, "Tuning RED for Web Traffic," IEEE/ACM Transaction on Networking. 2001; vol.9, no.3, pp. 249-264.

[24] X. Wang, Active Queue Management for Real-time IP Traffic, Ph.D. thesis, Department of Electronic Engineering, University of London, October 2006.

[25] C. Feng, L. Huang, C. Xu, and Y. Chang, "Congestion control scheme performance analysis based on nonlinear RED," *Journal of IEEE Systems*, *PP*(99), 2015, pp.1–8.

[26] T. Alemu, & A. Jean-Marie, " Dynamic configuration of RED parameters," In Global Telecommunications Conference, 2004. GLOBECOM'04. IEEE, Vol. 3, pp. 1600-1604.

[27] J. Chen, C. Hu, and Z. Ji, "Self-tuning random early detection algorithm to improve performance of network transmission," Mathematical Problems in Engineering, Vol. 2011, 2011 .

[28] A. Dhamodaran, K. Gatimu, and B. Lee, "Adaptive Queue Management Scheme for Flexible Dual TCP/UDP Streaming Protocol," MMEDIA 2017 : The Ninth International Conferences on Advances in Multimedia, April 23 - 27, 2017 - Venice, Italy.

[29] F. AL-Raddady, and M. Woodward, "A New Adaptive Congestion Control Mechanism for the Internet Based on RED," 21st International Conference on Advanced Information Networking and Applications Workshops (AINAW'07), IEEE, Canada, 2007.

[30] S. Masoumzadeh, K. Meshgi, S. Ghidari and G. Taghizadeh, "FQL-RED: an adaptive scalable schema for active queue management," INTERNATIONAL JOURNAL OF NETWORK MANAGEMENT, Vol. 21, Issue 2, March/April 2011, pp.147–167.

[31] S. K. Bisoy, and P. K. Pattnaik, "Design of feedback controller for TCP/AQM networks," *Engineering Science and Technology, an International Journal* Vol. 20, No. 1, 2017, PP.116-132.

[32] W. Feng, D. Kandlur, D. Saha, and K. Shin, "A Self-Configuring RED Gateway," In Proceedings of IEEE INFOCOM'99. Eighteenth Annual Joint Conference of the IEEE Computer and Communications Societies, volume 3, pages1320–1328 vol.3, Mar 1999.

[33] S. Floyd, R. Gummadi, and S. Shenker, "Adaptive RED: an algorithm for increasing the robustness of RED's active queue management," Technical report, ICSI, August 2001, available at: http://www.icir.org/˜floyd.

[34] N. Kuhn, and R. David, "Improving PIE's performance over high-delay paths," *arXiv preprint arXiv: 1602. 00569* (2016).

[35] T. Høiland-Jørgensen, P. McKenney, D . Taht, J. Gettys, and E. Dumazet, "Flowqueue-codel," IETF Draft, Oct. 2015.

[36] S. Patel, and S. Bhatnagar, "Adaptive mean queue size and its rate of change: queue management with random dropping," *Telecommunication Systems* Vol. 65, No. 2, 2017, PP.281-295.

[37] A. Kamal, and M. Murshed, "Adaptive RED with Dynamic Threshold Adjustment," Research report, Iowa State University, pp. 1-42, 2005.

[38] I. Järvinen, M. Kojo, "Evaluating CoDel, PIE, and HRED AQM techniques with load transients," In: *Local Computer Networks (LCN), 2014 IEEE 39th Conference on*. IEEE, 2014. PP. 159-167.

[39] B. Braden , D. Clark, J. Crowcroft, B. Davie, S. Deering, D. Estrin, S. Floyd, V. Jacobson, G. Minshall, C. Partridge, and L. Peterson, "Recommendations on queue management and congestion avoidance in the Internet," 1998.(No. RFC 2309).

[40] A. Sungur, TCP – Random Early Detection (RED) mechanism for Congestion Control, M.Sc. thesis, College of Computing & Information Sciences, Rochester Institute of Technology, Rochester, NY, 2015

[41] W. Chen, and S. Yang, " The mechanism of adapting RED parameters to TCP traffic," Computer Communications, Vol.32, No.13, 2009, pp.1525-1530.

[42] M. Ali, Modified Random Early Detection (RED) Technique Using Various Congestion Indicators, M.Sc. thesis, Department of Computer Information Systems, Faculty of Information Technology, Middle East University, January – 2017

[43] C. Hollot, V. Misra, D. Towsley, and W. Gong. "A control theoretic analysis of RED," In Proc. IEEE INFOCOM'01, 2001

[44] W. Feng, D. Kandlur, D. Saha, and K. Shin, "Techniques for eliminating packet loss in congested TCP/IP networks," University of Michigan, Tech. Rep. CSE-TR-349-97, November 1997.

[45] H. Dahmouni, A. Girard, B. Sansò, "An analytical model for jitter in IP networks," annals of telecommunications-annales des telecommunications, Vol. 67, No.2, 2012, pp. 81-90.

[46] H. Dahmouni, H. Elghazi, D. Bonacci, B. Sansò, and A. Girard, "Improving QoS of all-IP generation of pre-WiMax networks using delay-jitter model," Journal of Telecommunication, Vol. 2, No.2, 2010, pp. 99–103.

[47] A. Sethi and Y. Hnatyshin, The Practical OPNET User Guide for Computer Network Simulation, Taylor & Francis Group, USA, 2013.

Ahmad Al-Allaf received the B.Sc. degree in electrical engineering and M.Sc. degree in electronics and communication from the department of electrical engineering, University of Mosul, Iraq. He has published 15 research papers in the areas of computer networks, fault tolerance computing, reconfigurable computing, and parallel processing. From Nov 2000 to June 2006 he was worked as a lecturer at dept. of computer science, Benghazi University (Benghazi-Libya). From July 2006 and up to date he is a lecturer with the dept. of computer engineering at Northern Technical University-Technical college/Mosul, Iraq. He is currently pursuing his Ph.D. degree in computer networks. His research interests include reconfigurable computing, the design of computer network devices, and parallel processing.

A.I.A. Jabbar received the B.Sc. and M.Sc. degree in electrical engineering from Mosul university in 1975 and 1979 respectively, and the Ph.D. degree in communication engineering from the University of Bradford in 1989. His interesting field of research has been concentrated in the area of protocol design for mobile and computer networks. He is now working as a professor in Mosul university and supervising computer and LTE networks Ph.D. projects.