Curvilinear RED: An Improved RED Algorithm for Internet Routers

Conference Paper · September 2014

DOI: 10.2316/P.2014.813-025

2 authors:

Ayodeji Oluwatope
Obafemi Awolowo University
47 PUBLICATIONS **142** CITATIONS

SEE PROFILE

Samuel Hassan
Olabisi Onabanjo University
1 PUBLICATION **0** CITATIONS

SEE PROFILE

CURVILINEAR RED: AN IMPROVED RED ALGORITHM FOR INTERNET ROUTERS

Samuel Hassan
Department of Computer Science and Engineering
Obafemi Awowolo University
Ile-Ife, Osun State, Nigeria
email: samueltosinhassan@gmail.com

Ayodeji Oluwatope
Department of Computer Science and Engineering
Obafemi Awowolo University
Ile-Ife, Osun State, Nigeria
email: aoluwato@oauife.edu.ng

ABSTRACT
Random Early Detection (RED), an active queue management (AQM) scheme has been known to address the problem of network congestion in Internet routers. However, RED suffers from large delay which can be partly traced to the single linear packet drop function it deploys. In this paper, we present a new RED-based AQM scheme called Curvilinear Random Early Detection (CLRED) scheme which modified the single linear dropping function of RED with a two-segment (that is, a quadratic and a linear) dropping functions to address the reported drawback. Simulation carried out in network simulator 3 (NS-3) confirms that the proposed CLRED scheme achieved a significant reduction in the average queue size when compared with the classical RED scheme. Therefore, the existing RED implementations can be upgraded/replaced with the proposed CLRED scheme so as to help RED overcome its large delay drawback.

KEY WORDS
Active queue management, Random Early Detection, Nonlinear RED, Congestion control, Internet, Routers.

1 Introduction

The primary role of a router is to switch packets from the input links to output links through buffer. Apart from forwarding the packets, routers are involved for controlling the congestion in the network [9].

It has been shown by Floyd and Jacobson in [6] that routers can be deployed to handle network congestion, owing to the fact that the most effective detection of congestion was believed to occur in the gateway (that is, the router) itself. In the light of this, Tanenbaum and Wetherall in [8] are of the opinion that since congestion occurs within the network, it is the network layer (particularly the router) that directly experiences it and must ultimately determine what to do with the excess packets.

Most of the existing Internet routers play a passive role in congestion control and are known as "drop tail" routers ([1]; [12]). The traditional technique for managing router queue lengths is to set a maximum length (in terms of packets) for each queue, accept packets for the queue until the maximum length is reached, then reject (drop) subsequent incoming packets until the queue decreases because a packet from the queue has been transmitted. This technique is known as drop tail, since the packet that arrived most recently (i.e., the one on the tail of the queue) is dropped when the queue is full [3].

Active Queue Management (AQM) scheme was proposed as a solution for preventing losses due to buffer overflow. The idea behind active queue management is to detect incipient congestion early and convey notification to the end-hosts, allowing them to back-off before queue overflow and sustained packet loss occur [2].

Random Early Detection (RED) [6], an AQM scheme has been recommended by the Internet Engineering Task Force (IETF) as a default approach to adddress the problem of network congestion in Internet routers [3]. RED was originally proposed by Floyd and Jacobson to overcome the lock-out and full queue problems of Drop-Tail (DT) (passive queue management scheme) routers in handling network congestion.

The RED algorithm is the most popular and extensively studied active queue management schemes [12]. Due to its popularity, RED (or its variants) has been implemented by many router vendors in their products (e.g. Cisco implemented weighted RED, WRED) ([1]); [12]).

The main contribution of this study is in developing a new RED-based scheme called *Curvilinear RED* which keeps the average queue of routers small (one of the most important goals of AQM according to [3]) and as such results in a reduced/lower delays of packets across the network.

The rest of the paper is organized as follows: Section 2 describes the RED algorithm in detail and a review of some RED-based enhancement AQM schemes, such as GRED, DSRED, NLRED, and MRED. In Section 3, we describe our proposed Curvilinear Random Early Detection (CLRED) algorithm. In Section 4, we present the simulation configuration and results obtained from the comparison between the proposed CLRED algorithm with the RED algorithm using NS-3. Section 5 contains the conclusion of the work and suggestions for future work.

2 Previous Works

This section presents the Random Early Detection scheme, being the most profound AQM scheme and review of some recent improvement on it.

2.1 Random Early Detection (RED)

The operation of the RED scheme is quite simple but profound. For each new packet that arrives the gateway, RED computes the average queue size (avg) (meaning, the average number of packets in the buffer of the router) using a low pass-pass filter with Exponential Weighted Moving Average (EWMA) (that is, equation (1)) and compares the result with two preset thresholds: the minimum threshold (Min_{th}) and the maximum threshold (Max_{th}).

$$avg = ((1 - W_q) \times avg') + (W_q \times q) \quad (1)$$

Where q is the current queue size; avg' is the calculated previous average queue size; and W_q is a pre-defined weight parameter to calculate avg.

If the calculated average queue size is found to be less than the Min_{th}, then the packet is dropped with probability 0 (that is, allowed into the gateways buffer). If the calculated average queue size is found to exceed the Max_{th}, then the packet is dropped with probability 1. However, if the average queue size is found to be a value between the Min_{th} and Max_{th}, then the packet is dropped linearly from 0 to Max_p with probability:

$$P_b = Max_p(\frac{avg - Min_{th}}{Max_{th} - Min_{th}}) \quad (2)$$

Where P_b is the initial packet dropping probability and Max_p is the maximum drop probability.

Thus,

$$P_a = (\frac{P_b}{1 - Count.P_b}) \quad (3)$$

Where P_a is the final packet dropping probability and count is the number of arrived packets since the last dropped.

Therefore, the dropping function $P_d(avg)$ of RED can be expressed as:

$$P_d(avg) = \begin{cases} 0 & avg < Min_{th} \\ Max_p(\frac{avg - Min_{th}}{Max_{th} - Min_{th}}) & Min_{th} \leq avg < Max_{th} \\ 1 & Max_{th} \leq avg \end{cases} \quad (4)$$

Figure 1 depicts the dropping function of the RED model. Therefore, the pseudocode for the general RED algorithm is presented in Algorithm 1.

2.2 Gentle RED

In [5], Floyd observed that in RED, when the calculated average queue size, avg exceeds Max_{th} all incoming packets are dropped. This was believed to be too aggressive and results in low throughput.

Algorithm 1 Pseudocode for RED Algorithm (Courtesy of [6]

For each packet arrival
calculate the average queue size avg
if $Min_{th} \leq avg < Max_{th}$ **then**
  calculate probability p_a
  with probability p_a:
  mark the arriving packet
else if $Max_{th} \leq avg$ **then**
  mark the arriving packet
end if

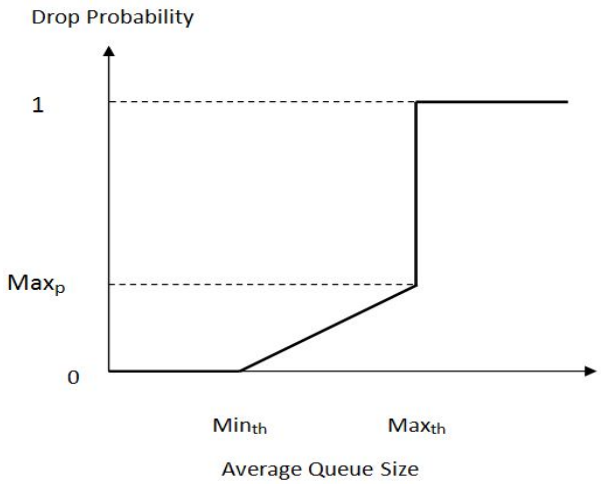

Figure 1. RED Dropping Function

Therefore, the congestion avoidance scope (that is, the region: Min_{th} and Max_{th}) of RED was further extended by another threshold, $2 \times Max_{th}$ which also uses a linear dropping function from Max_p to 1 thereby making it more gentle than the original RED. This improved version of RED is called *Gentle RED* (or GRED) and is depicted in Figure 2.

The detailed pseudocode for GRED algorithm is presented in Algorithm 2.

2.3 Double-Slope Random Early Detection

In [11], Zheng and Atiquzzaman reported that the RED scheme suffers from performance problems, such as low throughput, large delay or jitter, and inducing instability in networks. Of these problems, low throughput was considered to be the most important.

To address this downside of RED, a two-linear dropping function called *Double-Slope Random Early Detection* (or DSRED) scheme which are complementary and adjustable by using a mode selector γ was proposed.

The drop function $P_d(avg)$ of DSRED can be ex-

Algorithm 2 Detailed Pseudocode for GRED Algorithm

Initialization:
$avg \leftarrow 0$
$count \leftarrow -1$
For each packet arrival,
Calculate the average queue size avg
if the queue is non-empty **then**
  $avg \leftarrow ((1 - W_q) \times avg') + (W_q \times q)$
else $m \leftarrow f(time - q\_idle\_time)$
  $avg \leftarrow ((1 - W_q)^m \times avg')$
end if
if $Min_{th} \leq avg < Max_{th}$ **then**
  Increment count
  Calculate the packet drop probability P_a
  $P_b \leftarrow Max_p \times ((avg - Min_{th})/(Max_{th} - Min_{th}))$
  $P_a \leftarrow P_b/(1 - count.P_b)$
  mark the arriving packet with probability P_a:
  $count \leftarrow 0$
  Drop the packet
else if $Max_{th} \leq avg < 2 \times Max_{th}$ **then**
  Set $count \leftarrow count + 1$
  Calculate the packet drop probability P_a
  $P_b \leftarrow (1 - Max_p) \times ((avg - Maxth)/Max_{th})) + Max_p$
  $P_a \leftarrow P_b/(1 - count.P_b)$
  mark the arriving packet with probability P_a:
  Set $count \leftarrow 0$
  Drop the packet
else if $2 \times Max_{th} \leq avg$ **then**
  Drop the arriving packet
  Set $count \leftarrow 0$
else $count \leftarrow -1$
  when the buffer of the router becomes empty
  Set $q\_idle\_time \leftarrow time$
end if

Saved Variables:
avg: current average queue size
avg': calculated previous average queue size
q_idle_time: start of the queue idle time
count: packets since last marked packet

Fixed parameters:
W_q: queue weight
Min_{th}: minimum threshold for queue
Max_{th}: maximum threshold for queue
Max_p: maximum value for P_b

Other:
P_a: current packet-marking probability
q: current queue size
$time$: current time
$f(t)$: a linear function of the time t

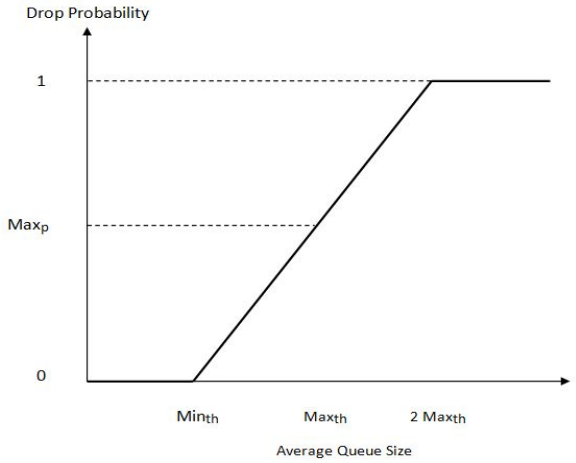

Figure 2. GRED Dropping Function

pressed as follows:

$$P_d(avg) = \begin{cases} 0 & avg < K_l, \\ \alpha(avg - K_l) & K_l \leq avg < K_m, \\ 1 - \gamma + \beta(avg - K_m) & K_m \leq avg < K_h, \\ 1 & K_h \leq avg \leq N. \end{cases} \quad (5)$$

where $\alpha = (\frac{2(1-\gamma)}{K_h - K_l})$; $\beta = (\frac{2\gamma}{K_h - K_l})$

In DSRED, K_m can be readily interpreted to be the mid-point between the minimum threshold K_l and the maximum threshold K_h while N is the size of the buffer.

2.4 Non-linear Random Early Detection

In [12], Zhou *et al.,* believes that the throughput performance of RED is not stable. According to the scholars, when the traffic load is very light and RED parameters are aggressively set or when the traffic load is very heavy and the parameters are tenderly set, the throughput is low. Also, the observed instability issue was claimed to be attributed to at least in part, the linear packet dropping function adopted by RED which tends to be too aggressive at light load and not aggressive enough when the average queue size approaches the Max_{th}.

Based on the foregoing, the authors proposed a modified RED algorithm called "Non-linear Random Early Detection" (NLRED) scheme which simply replaced the linear packet dropping function in RED with a non-linear (that is, a quadratic) function while retaining other features of the RED algorithm.

The calculation for dropping packet for the NLRED model is as given below:

$$P_d(avg) = \begin{cases} 0 & avg < Min_{th}, \\ (\frac{avg - Min_{th}}{Max_{th} - Min_{th}})^2 Max'_p & Min_{th} \leq avg < Max_{th}, \\ 1 & Max_{th} \leq avg. \end{cases} \quad (6)$$

Where Max'_p was set to 1.5 times of Max_p of RED.

2.5 MRED

In [10], Zhang *et al*., acknowledged that in RED, the linear dropping function though simple and easy to calculate, when average queue size approaches either of Min_{th} and Max_{th}, the loss rate is unreasonable.

To address the reported problem, the authors proposed an improved RED algorithm called "MRED" which uses two dropping functions: a non-linear from 0 to Max_p and a linear from Max_p to 1. The MRED model is quite similar to the GRED model except that the first linear dropping function used in the region: Min_{th} and Max_{th} of GRED was replaced with a quadratic dropping function.

The pseudocode for the MRED algorithm is presented in Algorithm 3.

Algorithm 3 Pseudocode for MRED Algorithm (Courtesy of [10]

Initialization:
$avg = 0$
$count = -1$
For each packet arrival, calculate the average queue size avg
if $^{Min_{th}<=avg<Max_{th}}$ **then** calculate the packet drop probability P_b
$P_b = Max_p \times (avg^2 - Min_{th}^2)/(Max_{th}^2 - Min_{th}^2)$
mark the arriving packet with probability P_a
else if $^{Max_{th}<=avg<2Max_{th}}$ **then** calculate the packet drop probability P_b
$P_b = (1-Max_p)\times(avg-Max_{th})/Max_{th}+Max_p$
mark the arriving packet with probability P_a
else if $2Max_{th} <= avg < Buffer\ Size$ **then**
mark the arriving packet
end if

These foregoing improvements $(2.2 - 2.5)$ are no doubt a pointer to the fact that the single linear dropping function used in RED is inadequate to address network congestion.

In this study, we took a new look at the RED scheme with particular interest in the single linear packet dropping function it deploys and therefore developed an improved RED-based model which exploits the combination of a quadratic and a linear packet dropping functions in an attempt to address the delay drawback in RED.

3 Curvilinear RED (CLRED) Algorithm

The proposed scheme is called Curvilinear Random Early Detection (CLRED) and is as depicted in Figure 3. For each packet arrival, CLRED computes the average queue size (avg) just like RED using a low pass-pass filter with Exponential Weighted Moving Average (EWMA) (that is, equation (1)).

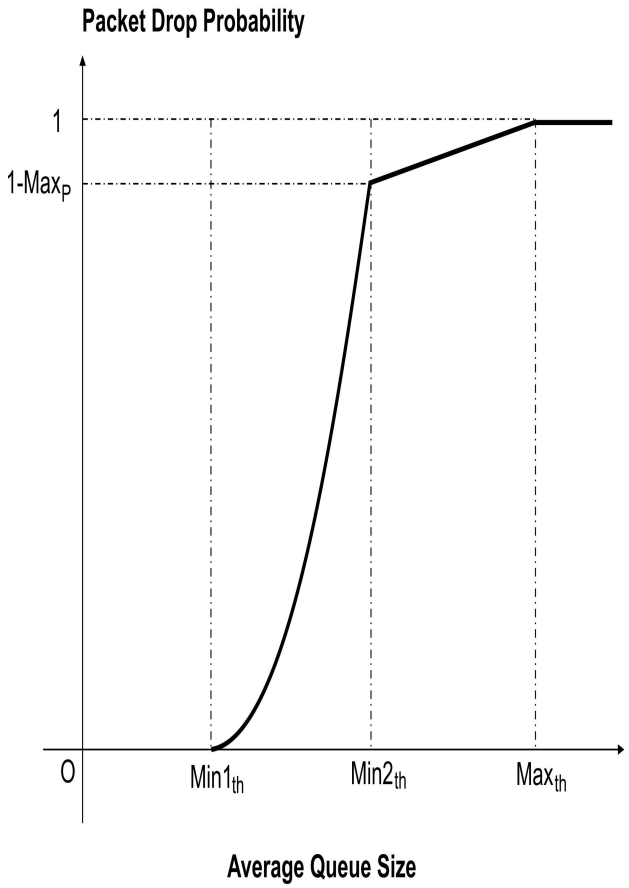

Figure 3. CLRED Dropping function

The dropping function $P_d(avg)$ of CLRED is presented as follows:

$$P_d(avg) = \begin{cases} 0, & avg < Min1_{th}, \quad P_b = 4(1 - Max_p)(\frac{avg - Min1_{th}}{Max_{th} - Min1_{th}})^2 \\ 4(1 - Max_p)(\frac{avg - Min1_{th}}{Max_{th} - Min1_{th}})^2, & Min1_{th} \leq avg < Min2_{th}, \\ (1 - Max_p) + 2Max_p(\frac{avg - Min2_{th}}{Max_{th} - Min1_{th}}), & Min2_{th} \leq avg < Max_{th}, \\ 1, & Max_{th} \leq avg. \end{cases} \quad (7)$$

Where, $Min1_{th}$ is the first minimum threshold (same as Min_{th} for RED); Max_{th} is the maximum threshold (same as Max_{th} of RED); and $Min2_{th}$ is the mid-point threshold between $Min1_{th}$ and Max_{th}. Therefore,

Algorithm 4 Pseudocode for CLRED Algorithm

```
Initialization:
avg ← 0
count ← −1
For each packet arrival,
Calculate the average queue size avg
if the buffer of the router is non-empty then
    avg ← ((1 − W_q) × avg′) + (W_q × q)
else m ← f(time − q_idle_time)
    avg ← ((1 − W_q)^m × avg′)
end if
if avg < Min1_th then
    No packet drop
    Set count ← −1
else if Min1_th ≤ avg < Min2_th then
    Set count ← count + 1
    Calculate the packet drop probability P_a
    P_b ← 4(1 − Max_p) × ((avg − Min1_th)/(Max_th − Min1_th))^2
    P_a ← P_b/(1 − count.P_b)
    mark the arriving packet with probability P_a:
    count ← 0
    Drop the packet
else if Min2_th ≤ avg < Max_th then
    Set count ← count + 1
    Calculate the packet drop probability P_a
    P_b ← (1 − Max_p) + 2Max_p((avg − Min2_th)/(Max_th − Min1_th))
    P_a ← P_b/(1 − count.P_b)
    mark the arriving packet with probability P_a:
    Set count ← 0
    Drop the packet
else if Max_th ≤ avg then
    Drop the arriving packet
    Set count ← 0
else count ← −1
    when the buffer of the router becomes empty
    Set q_idle_time ← time
end if
```

(i.) if avg is found to be value between 0 and $Min1_{th}$, then the packet will not be dropped.

(ii.) if avg is found to be a value between $Min1_{th}$ and $Min2_{th}$, then the packet is dropped with probability P_a. That is, $P_a = P_b/(1 - count.P_b)$

(iii.) similarly, if avg is found to be a value between $Min2_{th}$ and Max_{th}, then the packet is dropped with probability P_a (as in ii.). However,

$$P_b = (1 - Max_p) + 2Max_p(\frac{avg - Min2_{th}}{Max_{th} - Min1_{th}})$$

(iv.) lastly, if avg is found to exceed Max_{th}, then the packet will be dropped.

The pseudocode for CLRED algorithm is presented in Algorithm 4.

4 Simulation and Evaluation

We performed packet-level simulation using network simulator version 3 (NS−3)[7]. The network simulated is depicted in Figure 4. The topology consists of N TCP subnet sources, one router (or gateway) and one TCP sink. Each of the sources is connected to the router using a link rate of 100 Mbps with 1 ms propagation delay time. The bottleneck link between the router and the sink is 10 Mbps with 10 ms propagation delay time. The capacity of the bottleneck link is set to 10 Mbps while others are set to 100 Mbps, so that only the link between the router and the sink result in congestion. The 10 ms propagation delay time of the bottleneck link was chosen so as to indicate a large delay across the network.

The CLRED algorithm was implemented on the router. By varying the number of sources N, different levels of traffic load and thus different levels of congestion are produced on the bottleneck link.

In this simulation, 100 TCP sources were run for 100 seconds from randomly selected nodes in source 1, source 2, . . . , source N. The values for flow N are: 10, 20, 30, 40, 50, 60, 70, 80, 90 and 100. The NS−3 default packet size of 1000 bytes was set for each packet. The router has a buffer size of 100 packets. The parameter settings [4] used for the simulation are presented in Table 1 for both RED and CLRED.

Result shown in Figure 5 confirms that CLRED possesses capability to keep smaller the average queue of the packets in the router small better than RED. It is evident from the figure that the average queue size of CLRED converges to a much stable value even faster than RED.

Table 1. Parameter settings for RED and CLRED

RED		CLRED	
Parameters	**Values**	**Parameters**	**Values**
Min_{th}	10 packets	$Min1_{th}$	10 packets
Max_{th}	30 packets	$Min2_{th}$	20 packets
W_q	0.002	Max_{th}	30 packets
Max_p	0.1	W_q	0.002
		Max_p	0.1

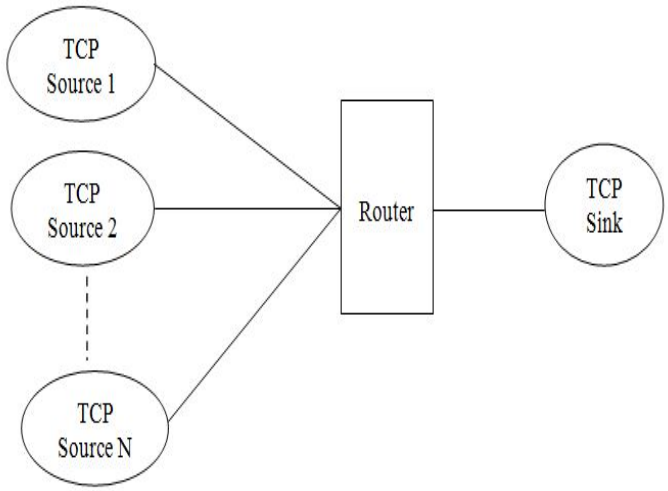

Figure 4. Network Topology

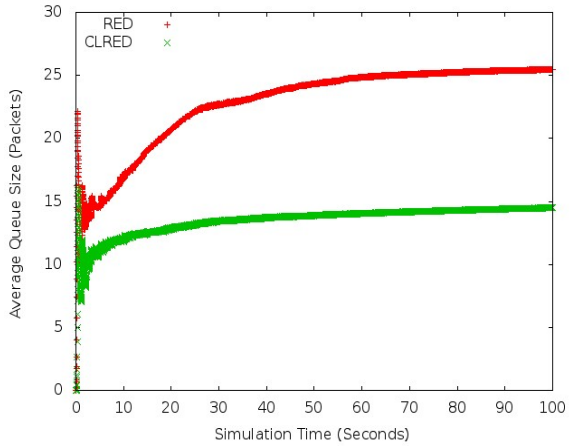

Figure 5. Average Queue Size vs. Simulation Time using CLRED and RED

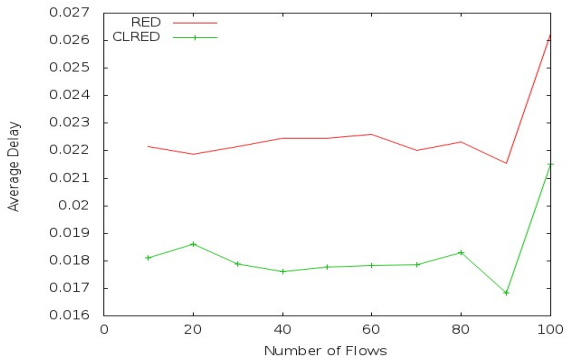

Figure 6. Average Delay vs. Number of Flows using CLRED and RED

In Figure 6, the average delay experienced by the packets was examined as the number of TCP flow increases for the proposed CLRED scheme and RED scheme. It is obvious that CLRED achieved a much smaller average delay than RED. This performance gain can be attributed to be a result of the CLRED reduction of the RED's average queue size depicted in Figure 5.

5 Conclusion and Future Work

This study has identified large delay as the major downside of the RED scheme and a new RED-based AQM scheme named Curvilinear Random Early Detection (CLRED) scheme which deploys a combination of a quadratic and a linear dropping functions instead of a single linear dropping function used in RED has been proposed and evaluated to address the reported drawback.

The proposed CLRED algorithm was evaluated and compared with RED algorithm in NS-3 using performance metrics, such as average queue size and average delay.

From this study, it is believed that the observed large reduction in the average queue size was made possible through the combination of the quadratic and a linear dropping functions that CLRED deploys. Therefore, CLRED algorithm would be suitable for deployment in routers especially for handling network traffic emanating from delay sensitive real-time multimedia applications.

In future work, we plan to compare the performance of the CLRED scheme with other recently developed RED-based AQM schemes, such as GRED, DSRED, NLRED, and MRED; further test the CLRED scheme using an emulator; study the fairness issue of the CLRED scheme with regards to mixed flows.

6 Acknowledgment

We acknowledge the contributions of Prof. S. S. Ọkọ́ya of the Department of Mathematics, Ọbáfẹ́mi Awólọ́wọ̀ University, Ilé-Ifẹ̀, Nigeria.

7 References

1 B. Abbasov and S. Korukoglu, Effective RED: An Algorithm to Improve RED's Performance by Reducing Packet Loss Rate, *Elsevier, Journal of Network and Computer Applications*, 43, 2009, 703-709.

2 J. Aweya and M. Ouellette and D. Y. Montuno and A. Chapman, A control Theoretic Approach to Active Queue Management, *Elsevier, Journal of Networks*, 36, 2001, 203-235.

3 B. Braden and D. Clark and J. Crowcroft and S. Davie and S. Deering and D. Estrin and S. Floyd and and V. Jacobson and G. Minshal and C. Patridghe and L. Peterson and K. Ramakrishman and S. Shenka and J. Wrockawski and L. Zhou, Recommendations on Queue Management and Congestion Avoidance in the Internet, *RFC 2309*, 1998.

4 S. Floyd, RED: Discussions of Setting Parameters, *Available online at http://www.icir.org/ oyd/RED-parameters.txt*, (Date accessed: 3rd August, 2013), 1997.

5 S. Floyd, Recommendation on using the "gentle_" variant of RED, *Avail- able online at http://www.icir.org/ oyd/red/gentle.html*, (Date accessed: 16th August, 2013), 2000.

6 S. Floyd and V. Jacobson, Random Early Gateway for Congestion Avoidance, *IEEE/ACM Transactions on Networking*, 1(4), 1993, 397-413.

7 Network Simulator: NS−3, *Available online at http://www.nsnam.org/.*

8 A. S. Tanenbaum and D. J. Wetherall, *Computer Networks*, **Chapter 5:** *(The Network Layer)* (Pearson Education, Inc., publishing, 2011), *392.*

9 G. Thiruchelvi and J. Raja, A survey on Active Queue Management Mechanism, *IJCSNS International Journal of Computer Science and Network Security*, 8(12), 2008, 130-145.

10 Y. Zhang and J. Ma and Y. Wang and C. Xu, MRED: An Improved Nonlinear RED Algorithm, *International Proceedings of Computer Science and Information Technology*, 44, 2012, 6-11.

11 B. Zheng and M. Atiquzzaman, DSRED: A New Queue Management Scheme for the Next Generation Internet, *IEICE Trans. Commu.*, E89-B3, 2006, 764-774.

12 K. Zhou and K. L. Yeung and V. O. K. Li, Nonlinear RED: A Simple Yet Efficient Active Queue Management Scheme, *Elsevier, Journal of Computer Networks*, 50, 2006, 3784-3794.