Aminu Adamu
Yusuf Surajo
Muhammad T. Jafar

# SARED: SELF-ADAPTIVE ACTIVE QUEUE MANAGEMENT SCHEME FOR IMPROVING QUALITY OF SERVICE IN NETWORK SYSTEMS

**Abstract** *Considering the phenomenal growth of network systems, congestion remains a threat to the quality of the service provided in such systems; hence, research on congestion control is still relevant. The Internet research community regards active queue management (AQM) as an effective approach for addressing congestion in network systems. Most of the existing AQM schemes possess static drop patterns and lack a self-adaptation mechanism; as such they do not work well for networks where the traffic load fluctuates. This paper proposes a self-adaptive random early detection (SARED) scheme that smartly adapts its drop pattern based on a current network's traffic load in order to maintain improved and stable performance. Under light- to moderate-load conditions, SARED operates in nonlinear modes in order to maximize utilization and throughput, while it switches to a linear mode in order to avoid forced drops and congestion under high-load conditions. Our conducted experiments revealed that SARED provides optimal performance regardless of the condition of the traffic load.*

**Keywords** congestion control, AQM, RED, SARED, network systems, QoS

**Citation** Computer Science 22(2) 2021: 251–266

## 1. Introduction

The Internet has experienced rapid growth in recent years; Cisco has forecasted that several billion devices will be connected to the Internet in the near future [5]. Consequently, enormous amounts of valuable data will be transmitted over the Internet [1, 5, 14, 17, 20, 23]; as such, routers will be subjected to huge data traffic. Congestion is a key factor that affects the quality of service (QoS) that is provided in network systems [4, 11, 12]. Congestion occurs in a network when the amount of generated traffic exceeds the amount that the network's resources can handle. With current advancements in communication networks, congestion control has become a research focus. A tail-drop queue-management scheme was initially used in routers to prevent congestion. As data traffic is bursty in nature, tail-drop routers were equipped with fairly large queues in order to absorb traffic burstness and to maintain high link utilization. With tail-drop, sources will receive a congestion notification only when a queue is full. Generally, tail-drop routers were used with some shortcomings, such as large delay, overflow problems, global synchronization, and lock-out phenomena [3]. To address the observed problems of the tail-drop approach, active queue management (AQM) was later proposed and recommended for Internet routers [3]. Unlike tail-drop, the AQM scheme controls congestion by dropping or marking packets before a queue becomes full where packet dropping or marking events signal sources to decrease their transmission rates [3, 10].

The random early detection (RED) algorithm [9] is the most popular AQM scheme; it has been recommended by the Internet Engineering Task Force (IETF) for routers, and most router vendors have adhered to IETF's recommendation (e.g., Cisco implemented WRED) [21]. RED detects congestion early by computing an average queue length ($avg$), which is obtained by applying the exponentially weighted moving average (EWMA) procedure to the instantaneous queue length and probabilistically dropping packets based on the currently computed $avg$. Basically, RED uses two queue thresholds for packet drops; i.e., $min_{th}$, and $max_{th}$. If the computed $avg$ is below $min_{th}$, no packet will be dropped; however, if the $avg$ is between $min_{th}$ and $max_{th}$, the drop probability is adjusted based on the recently computed $avg$. To this end, the packet-dropping function is defined as a linear function of $avg$; it increases linearly from 0 to maximum packet-dropping probability $max_p$. Finally, all incoming packets will be dropped if the observed $avg$ is above $max_{th}$ [2, 9, 18, 21].

Although RED outperforms the tail-drop approach, it was discovered that the effect of RED's congestion management is seriously affected once a traffic load changes, since RED is highly sensitive to its parameter settings and network's traffic condition changes frequently; thus, constant tuning of RED's parameters is required to achieve the desired performance [2, 7, 15, 16, 18, 19, 24]. Additionally, RED's poor performance can also be associated with its linear drop function, which tends to be too aggressive at low loads and not aggressive enough when $avg$ approaches $max_{th}$ [19, 27].

To address the shortcomings of RED, several enhanced variants of RED have been proposed; e.g., gentle RED [7], nonlinear RED [27], adaptive RED [8], double-slope

RED [26], improved nonlinear RED [25], three-section RED [6], adaptive queue management with random dropping [13], and change trend queue management [22]. Despite all of these enhancements, the self-adaption problem of RED has yet to be properly addressed.

In this paper, a self-adaptive random early detection (SARED) algorithm is proposed. Unlike RED (and some of its enhanced variants), SARED does not merely consider $avg$ as a congestion indicator; it also considers the ratio of the total capacity demand of the current traffic to the available output (bottleneck) link's capacity as another congestion indicator (such a ratio is regarded as traffic load). Based on this information, SARED smartly adjusts its drop pattern. In SARED, $max_p$ is adapted based on an observed network's traffic-load (load) condition. SARED basically operates in two modes; i.e., linear, and nonlinear modes (also based on a currently observed load). With a high load, it switches to linear mode in order to be aggressive for avoiding congestion and forced packet drops while maximizing link utilization and avoiding global synchronization. At low and moderate loads, SARED operates in nonlinear modes, where its degree of nonlinearity is increased as a load decreases in order to be very gentle so as to avoid link under-utilization and maximize throughput. Subsequently, SARED works well in different load scenarios [19].

The rest of this paper is organized as follows. Section 2 presents related works, Section 3 presents the self-adaptive RED algorithm, and Section 4 presents SARED with a gentle slope. The results of the experiments that were conducted for the analysis of the proposed algorithm are presented in Section 5, and Section 6 concludes the paper.

## 2. Related works

The random early detection (RED) algorithm was proposed by S. Floyd and V. Jacobson in [9] as an active queue management (AQM) scheme. The RED algorithm works by detecting early congestion via average queue length ($avg$), which is obtained by using Equation (1):

$$avg = (1 - w)avg' + wq(t) \tag{1}$$

where $q(t)$ is the current queue length at time $t > 0$, $avg'$ is the previously obtained average queue length, and $w$ is the predefined weight parameter to compute the $avg$ (where $0 < w < 1$).

The obtained $avg$ is compared with the two queue's thresholds: minimum threshold ($min_{th}$) and maximum threshold ($max_{th}$). When the $avg$ is less than the minimum threshold, all of the incoming packets are allowed into the queue. When the $avg$ is greater than the maximum threshold, each arriving packet is dropped with a probability of 1. However, when the $avg$ is between the minimum and maximum thresholds, packets are dropped with probability as a function of $avg$, which increases linearly from 0 to maximum drop probability $max_p$. The RED drop function is given by Equation (2):

$$p_{RED}(avg) = \begin{cases} 0 & avg < min_{th} \\ \dfrac{avg - min_{th}}{max_{th} - min_{th}} \times max_p & min_{th} \leq avg < max_{th} \\ 1 & max_{th} \leq avg \end{cases} \tag{2}$$

Even though the RED algorithm provides better performance than the tail-drop approach, several works have revealed that RED suffers from some shortcomings, such as sensitivity to parameter settings and the lack of a self-adaptation mechanism [2, 7, 15, 16, 18, 19, 24].

In order to improve the throughput of RED, Floyd proposed gentle RED (GRED) in [7]. In GRED, another queue threshold ($2max_{th}$) was introduced after $max_{th}$. If the observed $avg$ is between $max_{th}$ and $2max_{th}$, then packets are dropped with probability. This increases linearly from $max_p$ to 1, thereby making it gentler than RED when $avg$ exceeds $max_{th}$.

In [8], Floyd et al. proposed adaptive RED as an enhanced version of RED. In adaptive RED (ARED), $max_p$ is adapted using an additive-increase multiplicative-decrease policy in order to keep the average queue length within a target range of halfway between $max_{th}$ and $min_{th}$. In ARED, $max_p$ is also constrained to remain within a range of 0.01 to 0.5. The implementation of ARED will lead to poor performance in a complex network environment where traffic intensity fluctuates rapidly.

In [26], Zheng and Atiquzzaman considered low throughput as an important limitation of RED and proposed double-slope RED (DS-RED). DS-RED uses two different drop probability distributions in order to perform better than RED. However, DS-RED operates almost similarly to GRED [7], as two linear drop functions were defined with different slopes in both of them. DS-RED inherits RED'S aggressiveness, as it relies on linear drop functions (and parameterization is still a problem in DS-RED).

In [27], Zhou et al. proposed nonlinear RED (NRED) by substituting the linear drop function defined in RED with a nonlinear drop function; apart from this alteration, all of the other features of RED are retained in NRED. Zhou et al. believed that the nonlinear drop function used in NRED makes it gentler than RED with low loads and more aggressive with heavy loads. However, NRED did not address RED's lack of self-adaptation problem and will perform poorly when used in very-high-load scenarios.

To improve the throughput of NRED, Zhang et al. proposed improved nonlinear RED (MRED) in [25]. MRED uses a nonlinear drop function when $avg$ is between $min_{th}$ and $max_{th}$; however, when $avg$ falls to between $max_{th}$ and $2max_{th}$, the drop probability increases linearly from $max_p$ to the maximum of 1 (as in GRED [7]). Basically, MRED is considered to be an improved version of NRED and GRED.

In [6], Feng et al. proposed three-section RED (TRED). TRED works by dividing the section of the queue from $min_{th}$ to $max_{th}$ into three sections. It was assumed that the load is low when *avg* falls in the first section, moderate when it falls within the middle section, and high when it falls within the last section. Subsequently, nonlinear drop functions are defined for the first and last sections, where a linear drop function is defined for the middle section. Results of the analysis conducted in [6] have shown that TRED improved throughput at low load and maintained low delays at high load.

Karmeshu et al. [13] believed that the use of *avg* to calculate dropping probability in terms of $min_{th}$ and $max_{th}$ (as done in RED and some of its enhanced versions) is ineffective during heavy-load situations, since the $max_{th}$ threshold will frequently be crossed (resulting in frequent packet dropping). As such, they proposed another AQM scheme called adaptive queue management with random dropping (AQMRD). In AQMRD, not only is the *avg* used to define packet drop probabilities but also its rate of change. This is done in order to make the queue size and delay low regardless of how frequently the traffic load changes. However, the dropping strategy introduced in AQMRD is very aggressive, which leads to poor link utilization and high loss rates.

In [22], Tang and Tan proposed another adaptive AQM, which they called change trend queue management (CT-AQM). CT-AQM works by predicting the change trend in an average queue length based on its rate of change coupled with the network environment to define the packet drop probabilities. The AQMRD proposed in [13] is very similar to CT-AQM, since both of them use the rate of change in average queue lengths. The results of the analysis conducted in [22] showed that CT-AQM was successful in lowering the loss rate and improving the throughput for different load situations. However, CT-AQM introduces high delays and allows more packets into a queue, which may lead to congestion when used in a complex network environment.

In this paper, self-adaptive RED (SARED) is proposed. Unlike other enhanced RED versions with static drop patterns, SARED considers the current traffic load and automatically adapts its drop pattern and $max_p$. SARED is based on nonlinear RED, and its exponent is regulated based on the observed network's load condition; i.e., as the load decreases, the value of the exponent is increased so as to achieve more throughput and avoid link under-utilization. However, when the load becomes high, SARED will switch to linear mode, subsequently becoming more aggressive in order to maintain stable performance.

## 3. Self-adaptive Random Early Detection algorithm

Many research findings have revealed that traditional RED and some of its enhanced variants cannot provide the desired performance for networks where the traffic load is ever fluctuating. However, traffic load is naturally unpredictable in networks; as such, there is a need to have an effective AQM scheme that will provide the desired performance regardless of how the traffic load changes. Most AQM schemes use the computed average queue length (*avg*) as a congestion indicator; based on this, the packet-dropping probability is defined while ignoring the network's traffic load, which directly affects the observed average queue length. In this paper, the self-adaptive random early detection (SARED) algorithm is proposed, which considers not only the computed average queue length as a congestion indicator but also the current traffic load; based on this information, the packet-dropping probability is defined. In SARED, the data-arrival rate from each flow (source) at time $t$ is considered; i.e., $\lambda_n(t)$, $n = 1, ..., N$, where $N$ is the number of active flows at time $t$ (Fig. 1). Then, the total data arrival rate at the router at time $t$ is given by Equation (3):

$$\lambda(t) = \sum_{n=1}^{N} \lambda_n(t) \tag{3}$$

Let $\mu$ be the bandwidth of the bottleneck link; then, the traffic load at time $t$ is denoted by $\rho(t)$ and expressed by Equation (4):

$$\rho(t) = \frac{\lambda(t)}{\mu} \tag{4}$$

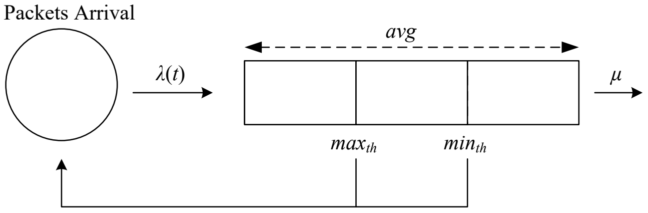

**Figure 1.** SARED's queue

If $\rho(t) < 1$, then the traffic load is low for the bottleneck link; therefore, packets will not accumulate in the queue. If this state is maintained for a long time, then link under-utilization will occur. If $\rho(t) \approx 1$, then the traffic load is moderate for the bottleneck link; hence, the performance will be optimal. However, if $\rho(t) > 1$, then the traffic load will be high for the bottleneck link, and packets will accumulate in the queue and wait to be sent. If this state is maintained for a long time, congestion and overflow may likely occur. SARED adapts its drop pattern based on the current network's load; i.e., for a high load, SARED operates in linear (aggressive) mode in order to avoid congestion and forced drops while maintaining high link utilization and avoiding global synchronization. For low to moderate loads, however, SARED operates in nonlinear (gentle) modes, where the its degree of nonlinearity depends on the observed load condition.

Note that, with SARED, different load states can also be formed based on the number of connections multiplexed over the bottleneck link.

In SARED, maximum drop probability $max_p$ is also defined based on the current network's load state (i.e., high $max_p$ for a high load and low $max_p$ for a low load), and average queue length ($avg$) is computed using Equation (1). If $avg < min_{th}$, then no packet will be dropped; if $avg \geq max_{th}$, then all of the arriving packets will be dropped with a probability of 1. However, if $min_{th} \leq avg < max_{th}$, then the packet-dropping probability increases linearly or nonlinearly from 0 to the current $max_p$. The drop function of SARED is presented in Equation (5), and an illustration of its packet-dropping probability curve is shown in Figure 2:

$$p_{SARED} = \begin{cases} 0 & avg < min_{th} \\ \left(\dfrac{avg - min_{th}}{max_{th} - min_{th}}\right)^{\lfloor n \rfloor} \times max_p & min_{th} \leq avg < max_{th} \\ 1 & avg \geq max_{th} \end{cases} \tag{5}$$

$$n = k^{\frac{1}{x}}, k \geq 2 \tag{6}$$

$$max_p = \frac{1}{\epsilon}(1 - k^{-x}) \tag{7}$$

where $k$ is a nonlinear index of the scheme, and $\epsilon > 0$ is the maximum drop probability regulator.

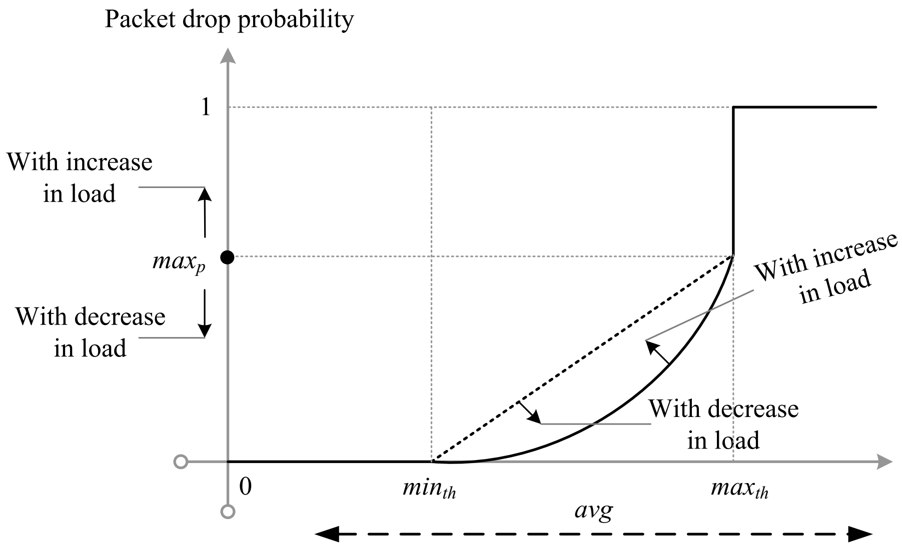

**Figure 2.** SARED's packet-dropping probability curve

The value of $x$ is varied based on network's load condition and used as a drop mode adapter.

The values of $x$ are defined for three load conditions (i.e., low, moderate, and high) as follows:

- $x < 1$ if $\rho(t) < 1$,
- $x \approx 1$ if $\rho(t) \approx 1$,
- $x \geq k$ if $\rho(t) > 1$.

Note that, with SARED, the load scenarios can be extended to more than three if desired.

SARED's drop behavior can be explained as follows:

Basically, SARED adapts its $max_p$ based on a network's load condition (a high $max_p$ for high loads, and a low $max_p$ for low loads) (Fig. 2). Additionally, when $avg$ falls within the $min_{th}$ and $max_{th}$ in SARED, a drop function that increases linearly or nonlinearly from 0 to the current $max_p$ is defined. Since the exponent of SARED's drop function is defined based on the load condition, its degree of nonlinearity is set to increase as the load becomes low. Conversely, as the load switches from low to moderate, its degree of nonlinearity is decreased, and when the load becomes high, SARED switches to linear mode. Subsequently, SARED can operate in two modes based on the load situation: linear, and nonlinear. It operates in nonlinear (gentle) modes for low to moderate loads and switches to linear (aggressive) mode for high loads (as presented in Algorithm 1). These features of SARED make it self-adaptive and will work well for a wide range of loads (see Table 1 for a complete list of the algorithm's parameters).

**Table 1**
Algorithm's parameters

| Saved variables | Fixed parameters | Other |
|---|---|---|
| $avg$: current average queue length | $w$: queue weight | $\lambda(t)$: current total incoming traffic flow (Mbps) |
| $avg'$: calculated previous average queue length | $min_{th}$: queue's minimum threshold | $\rho(t)$: current traffic load |
| $t_{queue\_idle\_time}$: start of queue idle time | $max_{th}$: queue's maximum threshold | $q(t)$: current queue length |
| $count$: packets since last marked packets | $\mu$: bottleneck link capacity (Mbps) | $f(t)$: linear function of time $t$ |
| $n$: exponent of nonlinear drop function | $k$: nonlinear index | $P_a$: current packet-marking probability |
| $x$: drop mode adapter | $\epsilon$: regulator of maximum drop probability | $max_p$: current maximum drop probability |

**Algorithm 1:** SARED

```
Data: $min_{th}$, $max_{th}$, $w$, $\mu$, $k$, $\epsilon$
initialization;
$avg \leftarrow 0$;
$count \leftarrow -1$;
$x \leftarrow 0$;
$n \leftarrow 0$;
$max_p \leftarrow 0$;
for each packet arrival do
    Return $\lambda(t)$ [Mbps];
    $\rho(t) \leftarrow \frac{\lambda(t)}{\mu}$;
    if $\rho(t) < 1$ then
        $x \leftarrow 0.25$;
        else if $\rho(t) \approx 1$ then
            $x \leftarrow 1$;
        else if $\rho(t) > 1$ then
            $x \leftarrow k$;
    end
    $n \leftarrow k^{\frac{1}{x}}$;
    $max_p \leftarrow \frac{1}{\epsilon}(1 - k^{-x})$;
    if queue is nonempty then
        $avg \leftarrow (1 - w)avg' + w \cdot q(t)$;
    else
        $m \leftarrow f(t - t_{queue\_idel\_time})$;
        $avg \leftarrow ((1 - w)^m \cdot avg')$;
    end
    if $avg < min_{th}$ then
        No packet drop;
        Set $count \leftarrow -1$;
        else if $min_{th} \leq avg < max_{th}$ then
            Set $count \leftarrow count + 1$;
            Calculate drop probability $P_a$;
            $P_b \leftarrow \left(\frac{avg - min_{th}}{max_{th} - min_{th}}\right)^{\lfloor n \rfloor} \times max_p$;
            $P_a \leftarrow \frac{P_b}{1 - count \cdot P_b}$;
            Mark arriving packet with $P_a$;
            Set $count \leftarrow 0$;
            Drop packet;
        else if $max_{th} \leq avg$ then
            Drop arriving packet;
            Set $count \leftarrow 0$;
        else
            $count \leftarrow -1$
        When queue becomes empty;
        Set $t_{queue\_idle\_time} \leftarrow t$;
    end
end
```

## 4. SARED with gentle slope

In SARED, the discontinuity of the packet-dropping probability from $max_p$ to 1 can be replaced with a gentle slope. In this case, the region of the queue from $min_{th}$ to $max_{th}$ should be extended to $2max_{th}$, and as $avg$ varies from $max_{th}$ to $2max_{th}$, then the packet-dropping probability should increase linearly from $max_p$ to 1 (as done in [7,25]). The drop function of SARED in this mode is presented in Equation (8) and depicted in Figure 3. Although the results that were obtained in [7,25] revealed that such an approach did not yield a significant performance improvement in RED, an analysis of SARED in this mode is reserved for future work.

$$p_{SARED} = \begin{cases} 0 & avg < min_{th} \\ \left(\dfrac{avg - min_{th}}{max_{th} - min_{th}}\right)^{\lfloor n \rfloor} \cdot max_p & min_{th} \leq avg < max_{th} \\ \left(\dfrac{avg - max_{th}}{max_{th}}\right) \cdot (1 - max_p) + max_p & max_{th} \leq avg < 2max_{th} \\ 1 & avg \geq 2max_{th} \end{cases} \tag{8}$$

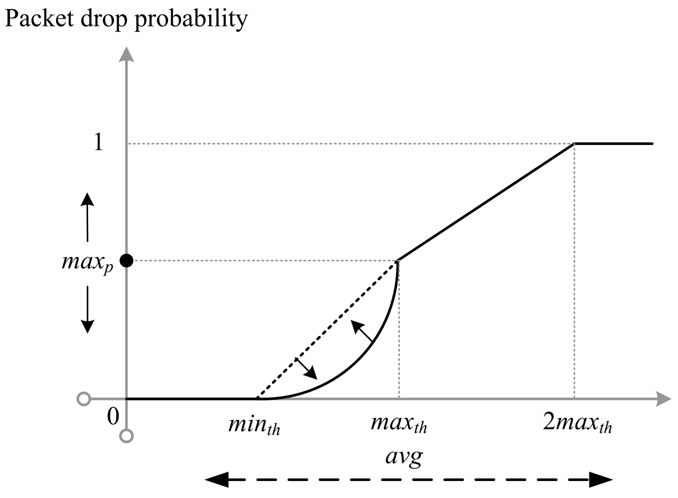

**Figure 3.** SARED's packet-dropping probability curve with gentle slope

## 5. Simulation experiments

Simulation experiments were conducted to validate the effectiveness of the SARED algorithm. The double dumbbell network topology presented in Figure 4 was used for simulation with an NS-2 simulator.

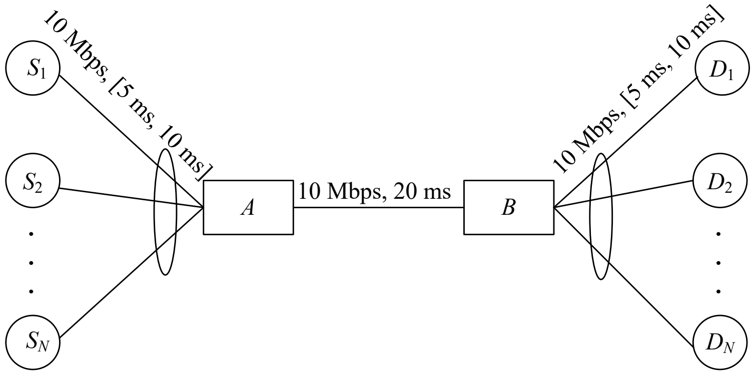

**Figure 4.** Simulation topology

The topology considers two routers ($A$ and $B$) connected via a bottleneck link that is shared by $N$ TCP flows that are generated by the FTP sources. The bottleneck link has a capacity of 10 Mbps and a propagation delay of 20 ms. The network's hosts are connected to the routers via links (each with a capacity of 10 Mbps), and their propagation delays are distributed between 5 and 10 ms. During the simulation, three levels of load scenarios were used: low, moderate, and high. For the **low-load scenario**, $N = 10$ flows were used; for the **moderate-load scenario**, $N = 50$ flows were used; and for the **high-load scenario**, $N = 150$ flows were used. An active queue-management scheme was implemented at Router $A$, whose queue capacity was 140 packets. New Reno TCP implementation was used. For a network with a given number of TCP flows, a 200-second simulation was conducted. For the computation of $\rho(t)$, the current data flow rate at the queue was obtained from the queue-monitoring object. The input parameters used for the performance analysis were $min_{th} = 20$, $max_{th} = 120$, $w = 0.002$, $k = 2$, and $\epsilon = k$ (Tab. 1).

$$x = \begin{cases} 0.25 & \text{if } \rho(t) < 0.75 \\ 1 & \text{if } \rho(t) \geq 0.75 \\ k & \text{if } \rho(t) \geq 1.25 \end{cases} \tag{9}$$

For the analyses of RED and TRED, $max_p = 0.1$ was used. The queue-length changes of RED over time in the low-, moderate-, and high-load scenarios are presented in Figures 5, 8, and 11, respectively, the queue-length changes of TRED over time in the three scenarios are presented in Figures 6, 9, and 12, respectively, and the queue-length changes of SARED over time in the three scenarios are presented in Figures 7, 10, and 13, respectively.

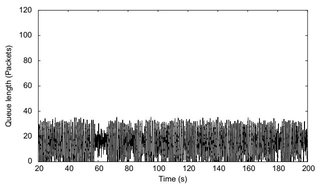

**Figure 5.** RED (Low load)

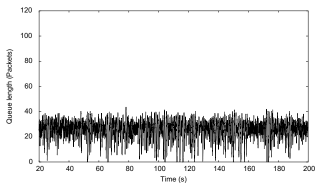

**Figure 6.** TRED (Low load)

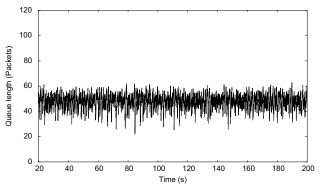

**Figure 7.** SARED (Low load)

The results presented in Figure 5 show that, for RED in the low-load scenario, there were several instances when the queue length was zero (0); this signaled poor link utilization. Figure 6 shows that, with TRED in the low-load scenario, there were some moments when the queue length was zero (0); this also indicated poor link utilization. The results presented in Figure 7 show that SARED tried to maximize the link utilization even at a low load by maintaining a queue length that was well above the $min_{th}$ threshold value. From the results presented in Figures 8 and 9, it can be observed that RED and TRED provided better link utilization in the moderate-load scenario; however, it can be noted that the average queue length of SARED (Fig. 10) at a moderate load was above that of RED and TRED. From Figures 11 and 12, it can be observed that, for RED and TRED in their respective high-load scenarios, the queue length hit the maximum threshold value ($max_{th}$) several times. This indicated several forced drops and, subsequently, global synchronization [2,15,24]. The results presented in Figure 13 show that SARED avoided forced drops in the high-load scenario by maintaining the queue at lengths that were below the maximum threshold value ($max_{th}$).

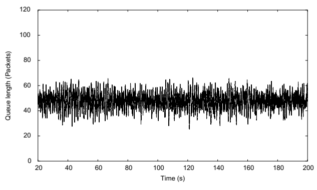

**Figure 8.** RED (Moderate load)

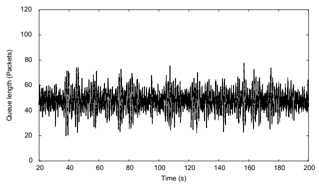

**Figure 9.** TRED (Moderate load)

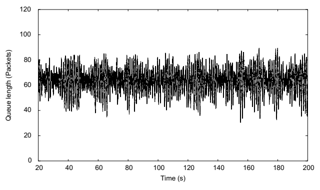

**Figure 10.** SARED (Moderate load)

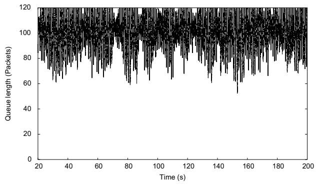

**Figure 11.** RED (High load)

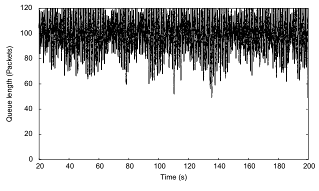

**Figure 12.** TRED (High load)

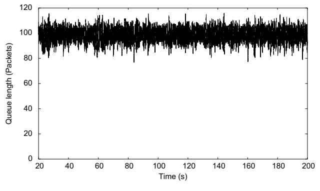

**Figure 13.** SARED (High load)

Table 2 presents the results of an analysis of the throughput. It can be seen that SARED tried to maximize the throughput even at a low load; however, the throughput of the RED, TRED, and SARED approached the maximum limits in their respective high-load scenarios.

**Table 2**
Throughput [Mbps]

| Load Scenario | RED | TRED | SARED |
|---|---|---|---|
| *Low* | 9.57 | 9.68 | 9.98 |
| *Moderate* | 9.94 | 9.95 | 9.99 |
| *High* | 9.99 | 9.99 | 9.99 |

Furthermore, an analysis of the delay was conducted. It can be observed from Table 3 that the delays of SARED in the low- and moderate-load scenarios were higher than those of RED and TRED. This is because SARED had a lower packet drop rate than RED and TRED in these load situations; hence, it allowed for more packets to accumulate in the queue. As such, the packets' queuing delay increased, which ultimately affected the overall delay that was encountered by the packets. However, at a high load, the delay observed with SARED was lower than that of RED and TRED. This is because SARED had a higher drop rate in that situation; hence, less packet accumulation in the queue. Subsequently, the packets experienced a lower delay with SARED than with RED and TRED at high loads.

**Table 3**
Delay [ms]

| Load Scenario | RED | TRED | SARED |
|---|---|---|---|
| *Low* | 81.43 | 89.76 | 120.13 |
| *Moderate* | 127.01 | 127.30 | 141.32 |
| *High* | 177.62 | 177.62 | 165.05 |

## 6. Conclusion

In this paper, a self-adaptive RED (SARED) algorithm was proposed. Unlike RED and some of its enhanced versions with static drop patterns, SARED considers the average queue length as well as the current traffic-load condition to adapt its maximum drop probability and drop pattern. At high loads, SARED operates in linear mode to avoid congestion and forced drops while maximizing link utilization and avoiding global synchronization. To prevent link under-utilization and improve the throughput at low and moderate loads, SARED operates in nonlinear mode, where its degree of nonlinearity is increased as its load decreases (and vice versa). These features of SARED make it work well in different load scenarios.

## References

[1] Atzori L., Iera A., Morabito G.: The internet of things: A survey, *Computer Networks*, vol. 54(15), pp. 2787–2805, 2010.

[2] Bonald T., May M., Bolot J.C.: Analytic evaluation of RED performance. In: *Proceedings IEEE INFOCOM 2000. Conference on Computer Communications. Nineteenth Annual Joint Conference of the IEEE Computer and Communications Societies (Cat. No.00CH37064)*, pp. 1415–1424, 2000.

[3] Braden B., Clark D., Crowcroft J., Davie B., et al.: Recommendations on Queue Management and Congestion Avoidance in the Internet. RFC2309, United States, April 1998, https://tools.ietf.org/html/rfc2309.

[4] Chaudhary P., Kumar S.: A Review of Comparative Analysis of TCP Variants for Congestion Control in Network, *International Journal of Computer Applications*, vol. 160(8), pp. 28–34, 2017.

[5] Cisco: *Cisco Visual Networking Index: Forecast and Trends, 2017-2022.* Cisco, https://twiki.cern.ch/twiki/pub/HEPIX/TechwatchNetwork/HtwNetworkDocuments/white-paper-c11-741490.pdf.

[6] Feng C.W., Huang L.F., Xu C., Chang Y.C.: Congestion Control Scheme Performance Analysis Based on Nonlinear RED, *IEEE Systems Journal*, vol. 11(4), pp. 2247–2254, 2017.

[7] Floyd S.: Recommendation on using the gentle_variant of RED. http://www.icir.org/floyd/red/gentle.html.

[8] Floyd S., Gummadi R., Shenker S.: Adaptive RED: An Algorithm for Increasing the Robustness of RED's Active Queue Management. Technical Report, UC, Berkeley, CA, 2001, https://www.icir.org/floyd/papers/adaptiveRed.pdf.

[9] Floyd S., Jacobson V.: Random early detection gateways for congestion avoidance, *IEEE/ACM Transactions on Networking*, vol. 1(4), pp. 397–413, 1993.

[10] Heinanen J., Baker F., Weiss W., Wroclawski J.: Assured forwarding PHB. RFC2597, June 1999, https://tools.ietf.org/html/rfc2597.

[11] Jacobson V.: Congestion avoidance and control, *ACM SIGCOMM Computer Communication Review*, vol. 25(1), pp. 157–187, 1995.

[12] Jain R.: Congestion control in computer networks: issues and trends, *IEEE Network*, vol. 4(3), pp. 24–30, 1990.

[13] Karmeshu S., Patel S., Bhatnagar S.: Adaptive mean queue size and its rate of change: Queue management with random dropping, *Telecommunication Systems*, vol. 65(2), pp. 287–295, 2017.

[14] Karthick G.S., Sridhar M., Pankajavalli P.B.: Internet of Things in Animal Healthcare (IoTAH): Review of Recent Advancements in Architecture, Sensing Technologies and Real-Time Monitoring, *SN Computer Science*, vol. 1(301), pp. 1–16, 2020.

[15] Korolkova A., Kulyabov D., Velieva T., Zaryadov I.: Essay on the study of the self-oscillating regime in the control system. In: *Communications of the European Council for Modelling and Simulation, Caserta, Italy*, pp. 473–480, 2019.

[16] May M., Bolot J., Diot C., Lyles B.: Reasons not to deploy RED. In: *1999 Seventh International Workshop on Quality of Service. IWQoS'99. (Cat. No.98EX354)*, pp. 260–262, 1999.

[17] Newton M., Kalman G.: Peer-to-Peer-Based Social Networks: A Comprehensive Survey, *SN Computer Science*, vol. 1(299), pp. 1–51, 2020.

[18] Patel S.: Performance analysis and modeling of congestion control algorithms based on active queue management. In: *Proceedings of 2013 International Conference on Signal Processing and Communication (ICSC)*, pp. 449–454, 2013.

[19] Plasser E., Ziegler T., Reichl P.: On the non-linearity of the RED drop function. In: *ICCC'02: Proceedings of the 15th International Conference on Computer Communication*, pp. 515–534, 2002.

[20] Sharma N., Rajput S., Dwivedi A., Shrimali M.: P-RED: Probability Based Random Early Detection Algorithm for Queue Management in MANET. In: *Advances in Computer and Computational Sciences. Advances in Intelligent Systems and Computing*, vol. 554, pp. 637–643, Springer, Singapore, 2018.

[21] Systems C.: Congestion Avoidance Overview. https://www.cisco.com/c/en/us/td/docs/ios/qos/configuration/guide/12_2sr/qos_12_2sr_book/congestion_avoidance.html.

[22] Tang L., Tan Y.: Adaptive Queue Management Based On the Change Trend of Queue Size, *KSII Transactions on Internet and Information Systems*, vol. 13(3), pp. 1345–1362, 2019.

[23] Varghese B., Wang N., Nikolopoulos D.S., Buyya R.: Feasibility of Fog Computing. https://arxiv.org/pdf/1701.05451.pdf.

[24] Velieva T.R., Korolkova A.V., Kulyabov D.S., Abramov S.A.: Parametric study of the control system in the TCP network. In: *Proceedings of 10th International Congress on Ultra Modern Telecommunications and Control Systems and Workshops (ICUMT)*, pp. 334–339, 2018.

[25] Zhang Y., Ma J., Wang Y., Xu C.: MRED: An Improved Nonlinear RED Algorithm. In: *International Conference on Computer and Automation Engineering (ICCAE 2011)*, vol. 44, pp. 6–11, 2012.

[26] Zheng B., Atiquzzaman M.: DSRED: improving performance of active queue management over heterogeneous networks. In: *ICC 2001. IEEE International Conference on Communications. Conference Record (Cat. No.01CH37240)*, vol. 8, pp. 2375–2379, 2001.

[27] Zhou K., Yeung K.L., Li V.O.K.: Nonlinear RED: A simple yet efficient active queue management scheme, *Journal of Computer Networks*, vol. 50, pp. 3784–3794, 2006.

## Affiliations

**Aminu Adamu**
Umaru Musa Yar'adua University, Katsina, Nigeria, amadamum@gmail.com

**Yusuf Surajo**
Federal University Dutsin-Ma, Katsina, Nigeria, ysurajo@fudutsinma.edu.ng

**Muhammad T. Jafar**
Federal University Dutsin-Ma, Katsina, Nigeria, mtukur@fudutsinma.edu.ng

**Received**: 18.10.2020
**Revised**: 16.01.2021
**Accepted**: 17.01.2021