Flexible Random Early Detection Algorithm for Queue Management in Routers

Conference Paper · September 2020

DOI: 10.1007/978-3-030-66471-8_16

CITATIONS
3

READS
175

4 authors, including:

Aminu Adamu

20 PUBLICATIONS **23** CITATIONS

SEE PROFILE

Yuliya Gaidamaka

Peoples' Friendship University of Russia (RUDN University)

117 PUBLICATIONS **591** CITATIONS

SEE PROFILE

Flexible Random Early Detection Algorithm for Queue Management in Routers

Aminu Adamu[1(✉)], Vsevolod Shorgin[2], Sergey Melnikov[3], and Yuliya Gaidamaka[2,3]

[1] Umaru Musa Yar'adua University, Katsina, Nigeria
amadamum@gmail.com
[2] Federal Research Center "Computer Science and Control" of the Russian Academy of Sciences, Moscow, Russia
vshorgin@ipiran.ru, gaydamaka-yuv@rudn.ru
[3] Peoples' Friendship University of Russia (RUDN University), Moscow, Russia
melnikov@linfotech.ru

Abstract. With recent advancements in communication networks, congestion control remains a research focus. Active Queue Management (AQM) schemes are normally used to manage congestion in routers. Random Early Detection (RED) is the most popular AQM scheme. However, RED lacks self-adaptation mechanism and it is sensitive to parameter settings. Many enhancements of RED were proposed and are yet to provide stable performance under different traffic load situations. In this paper, AQM scheme called Flexible Random Early Detection (FXRED) is proposed. Unlike other RED's enhancements with static drop patterns, FXRED recognizes the state of the current network's traffic load and auto tune its drop pattern suitable to the observed load situation in order to maintain stable and better performance. Results of the experiments conducted have shown that regardless of traffic load's fluctuation, FXRED provides optimal performance and efficiently manages the queue.

Keywords: Network · AQM · RED · Flexible RED · Congestion control

1 Introduction

Congestion is one of the major phenomena that affect the quality of service (QoS) provided in networks, when congestion occurred, poor utilization, high delay, high packet loss rate and low throughput would be experienced [1]. Congestion remains a threat to performance in existing and especially future

The publication has been prepared with the support of the "RUDN University Program 5–100" (recipient Yu. Gaidamaka, conceptualization). The reported study was funded by RFBR, project numbers 18-07-00576 and 20-07-01064 (recipient V. Shorgin, validation and visualization).

V. M. Vishnevskiy et al. (Eds.): DCCN 2020, LNCS 12563, pp. 196–208, 2020.
https://doi.org/10.1007/978-3-030-66471-8_16

generation networks [2–6]. To ensure better network performance, congestion must be avoided. Active Queue Management (AQM) was introduced to control congestion and works by sending early congestion notification alert to end devices; subsequently end devices upon reception of such notification will learn to reduce their transmission rates before overflow occurs in the network routers. AQM also aimed at increasing throughput by reducing the number of packets dropped, managing queue lengths to absorb short-term congestion, providing a lower interactive delay and avoiding global synchronization by randomly dropping packets across all flows [7].

Random Early Detection (RED) algorithm proposed in [8] was one of the early generations of AQM schemes and it was recommended by Internet Engineering Task Force (IETF) to be used in Internet routers for congestion control [7], as such, some router vendors have implemented RED algorithm as a default congestion control algorithm in their product; Cisco implemented WRED [9]. RED algorithm controls congestion by actively dropping incoming packets probabilistically. Traditional RED algorithm is associated with four basic parameters, which are the queue's thresholds (min_{th} and max_{th}), average queue length (avg), maximum packet drop probability (max_p) and weighted parameter (w). The average queue length is computed using the Exponentially Weighted Moving Average (EWMA) method applied to the instantaneous queue length. If the computed avg is below the min_{th} threshold, no packet will be dropped and if avg is above max_{th} threshold, all the incoming packets will be dropped with probability 1, however, if avg is between thresholds min_{th} and max_{th}, then packet drop probability increases linearly to the maximum drop probability max_p [8].

Even though it was proven that RED is superior to conventional Tail drop algorithm [7], however, RED lacks self-adaptation mechanism and it is sensitive to parameter settings [10]. To address the shortcomings of RED, many enhanced versions of RED were proposed, such as Gentle RED [10], Dynamic RED [11], Double Slope RED [12], Nonlinear RED [13], Improved Nonlinear RED [14], Three Section RED [15], Adaptive Queue Management with Random Dropping [16], Change Trend Queue Management [17], etc. However, with RED or some of its enhanced versions, the effect of congestion management will be seriously affected once the traffic load changed [15–20].

In this paper, an AQM scheme termed as Flexible RED (FXRED) is proposed. Unlike RED and some of its enhanced variants, in addition to avg, FXRED considers the current traffic load as congestion indicator, i.e. FXRED recognizes the state of the current network's traffic load and adapts a drop pattern suitable to the observed load situation. FXRED increases its drop rate as load becomes high in order to avoid overflow and congestion, conversely, as load becomes low, FXRED decreases its drop rate in order to maximize link utilization and throughput. The scheme proposed in this paper could also be used to create queue-based service policies for network systems as in [21–23].

The rest of this paper is organized as follows. Section 2 presents the related works and the proposed Flexible RED algorithm is presented in Sect. 3. Results of the experiments conducted for the analysis of the proposed algorithm are presented in Sect. 4 and Sect. 5 concludes the paper.

2 Related Works

Traditionally, Active Queue Management (AQM) schemes are used to control congestion in routers. The most popular AQM is Random Early Detection (RED) algorithm proposed by S. Floyd and V. Jacobson in [8]. RED achieves the desired goals of AQM scheme by dividing the router's queue into three sections using threshold values min_{th} and max_{th} and performs its operations in two phases.

1^{st} **Phase of RED operations:** computation of the average queue length *avg*.

Instead of using instantaneous queue length, RED continuously computes the average queue length (avg) using a simple exponentially weighted moving average.

2^{nd} **Phase of RED operations:** decision on whether to accept or drop an incoming packet. In order to make a decision on whether to accept or drop an incoming packet, the computed avg in the first phase is compared with queue's threshold values, i.e. min_{th} and max_{th}. If $avg < min_{th}$, then all incoming packets will be accepted, if $min_{th} \leq avg < max_{th}$, then packets are dropped randomly across all the flows with probability which increases linearly as a function of avg to the maximum drop probability max_p, however, if $avg \geq max_{th}$, then all incoming packets will be dropped with probability 1.

Even though it was proven that RED is superior to conventional Tail drop algorithm, however, RED suffers from some shortcomings such as parameterization problem (i.e. setting RED parameters and their tuning in order to achieve better performance based on current load condition), low throughput, large delay/jitter, unfairness to connections, and induces instability in the network [15–20].

In order to improve the throughput of RED, Floyd in [10] proposed Gentle RED (GRED). In GRED, the region of the queue from min_{th} to max_{th} was further extended to $2max_{th}$. If the avg falls between min_{th} and max_{th}, packet dropping probability increases linearly from 0 to max_p (as done in traditional RED), further, if the avg falls within max_{th} and $2max_{th}$, then packet dropping probability increases linearly from max_p to 1, thereby making GRED more gentler than RED.

Zheng and Atiquzzaman in [12] proposed Double-Slope RED (DS-RED). In DS-RED, a mid-point (mid_{th}) was introduced between min_{th} and max_{th}, when avg is between min_{th} and mid_{th}, the drop probability is defined by a linear function, when avg is between mid_{th} and max_{th}, drop probability also increases linearly (with different slope). This is done in order to reduce the aggressiveness of RED and to improve link utilization. However, DS-RED's behavior is similar to that of Gentle RED [10].

The aggressiveness of RED and some of its enhanced versions made Zhou et al. to propose a Non-linear RED (NRED) in [13]. In NRED, the linear drop function of RED is replaced with a non-linear quadratic drop function. However, under high load conditions, NRED will cause overflow and forced packet drops.

Zhang et al. in [14] proposed MRED which is an improved nonlinear RED. MRED differs from GRED in the sense that if avg is between min_{th} and max_{th},

the drop probability is defined by a nonlinear function. However, if *avg* falls between max_{th} and $2max_{th}$, the drop probability increases linearly to the maximum of 1 as in GRED.

In [15], Feng et al. proposed Three Section RED (TRED). Unlike other versions of RED where a mid_{th} is introduce between min_{th} and max_{th}, in TRED, two mid-points $mid_{th}(1)$ and $mid_{th}(2)$ were introduced in order to capture three load scenarios, i.e. low, moderate and high. When *avg* is between min_{th} and $mid_{th}(1)$ (corresponding to the low load state), packets are dropped with probability defined by a nonlinear drop function, when *avg* falls between $mid_{th}(1)$ and $mid_{th}(2)$ (moderate load), packets are dropped with probability which increases linearly and when *avg* falls between $mid_{th}(2)$ and max_{th} (high load), then packet drop probability is defined by another nonlinear function. Although results of the analysis conducted in [13] have shown that TRED improved throughput at low load and maintained low delays at high load, however, parameterization is still a problem in TRED.

Karmeshu et al. in [16] proposed another AQM scheme called Adaptive Queue Management with Random Dropping (AQMRD). In AQMRD, *avg* and its rate of change (*davg*) are used to optimally define the drop function, such that the queue length as well as the delay remains low regardless of how frequently the traffic load changes. However, dropping strategy introduced in AQMRD is so aggressive which leads to poor link utilization and high loss rate.

Another Adaptive AQM was proposed by Tang and Tan called Change Trend Queue Management (CT-AQM) [17]. CT-AQM predicts the change trend in average queue length based on its rate of change as well as the network environment to define packet drop function. AQMRD proposed in [16] is to some extent similar to CT-AQM since both of them use the rate of change in average queue length. Results of the analysis conducted in [17] have shown that CT-AQM was successful in lowering loss-rate and improved throughput for different load situations. However, CT-AQM introduces high delay and allows more packets into queue which may lead to congestion when used in a complex network environment.

In this paper, a Flexible Random Early Detection (FXRED) algorithm is proposed. Unlike the existing enhanced RED schemes, FXRED considers the current traffic load condition and adapts a suitable drop pattern. If the observed traffic load is low, FXRED gently drops packets since in that situation congestion is not a threat, however, when traffic load becomes high, FXRED will switch to aggressive drop pattern in order to avoid congestion and overflow.

3 Flexible Random Early Detection Algorithm

In the proposed Flexible RED (FXRED), the router's queue with finite capacity is divided into four (4) segments (A, B, C and D) via threshold values min_{th}, Δ and max_{th} (Fig. 1), where $\Delta = \frac{1}{2}(min_{th} + max_{th})$.

In FXRED, the data arrival rate from each flow at time t is considered, i.e. $\lambda_n(t)$, $n = 1, ..., N$, where N is the number of active flows at time t. Then, the

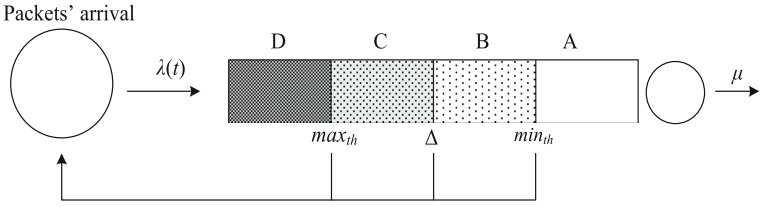

Fig. 1. FXRED's queue

total data arrival rate (from all the active flows) at the router at time t is given by Eq. (1).

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

Let μ be the bandwidth of the bottleneck link, then the traffic load at time t is denoted by $\rho(t)$ and expressed by Eq. (2).

$$\rho(t) = \frac{\lambda(t)}{\mu}. \tag{2}$$

Given (2), at time $t > 0$, the system can be in three (3) possible load states as follows:

State-1: $\rho(t) < 1$,
State-2: $\rho(t) \approx 1$,
State-3: $\rho(t) > 1$.

If the system is in *state*-1 ($\rho(t) < 1$), i.e. for the bottleneck link the traffic load is light, therefore, packets will not accumulate in the queue, if the system remains in this state for a long time, link underutilization will occur. If the system is in *state*-2 ($\rho(t) \approx 1$), the traffic load is moderate for the bottleneck link, hence, performance is optimal. However, if the system is in *state*-3 ($\rho(t) > 1$), the traffic load is high for the bottleneck link, 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. Note that with FXRED, network's load states can also be formed based on the *number of sources multiplexed over the bottleneck link*.

FXRED recognizes the current traffic load state and adapts a suitable drop pattern. If the network is in *state*-1, FXRED gently drops packets in order to maximize link utilization and throughput, however, if the network is in *state*-3, FXRED aggressively drops packets in order to avoid congestion and overflow.

Just like in traditional RED and its other variants, in FXRED, average queue length (avg) is computed via Eq. (3).

$$avg = (1 - w)avg' + wq(t), \tag{3}$$

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

If $avg \in [0, min_{th})$ (Segment A), then the drop probability is zero, i.e. no packet will be dropped. If $avg \in [min_{th}, \Delta)$ (Segment B), then FXRED proposes a nonlinear drop function expressed by Eq. (4).

$$p_{d(1)} = 2^{\lfloor k \rfloor} \left(\frac{avg - min_{th}}{max_{th} - min_{th}} \right)^{\lfloor k \rfloor} \cdot (1 - \epsilon), \tag{4}$$

where k and ϵ are expressed by Eqs. (5) and (6) respectively,

$$k = c^{\frac{1}{\gamma}}, c \geq 2. \tag{5}$$

$$\epsilon = c^{-\gamma}. \tag{6}$$

In this case, the value of $c \geq 2$ was chosen in order to turn the operation of FXRED in Segment B of the queue into nonlinear mode at low and moderate load states, where γ is the drop mode regulator and is expressed for the defined three network states as follows (if needed, more states can be added in FXRED):

$$\begin{cases} \gamma < 1 & \text{for } \textbf{State-1}, \\ \gamma \approx 1 & \text{for } \textbf{State-2}, \\ \gamma \geq c & \text{for } \textbf{State-3}. \end{cases} \tag{7}$$

If $avg \in [\Delta, max_{th})$ (Segment C), then FXRED proposes a linear drop function expressed by Eq. (8).

$$p_{d(2)} = 2\epsilon \left(\frac{avg - \Delta}{max_{th} - min_{th}} \right) + (1 - \epsilon). \tag{8}$$

Lastly, if $avg \geq max_{th}$ (Segment D), then all incoming packets will be dropped with probability 1. The general drop function of FXRED is presented in Eq. (9) and depicted in Fig. 2.

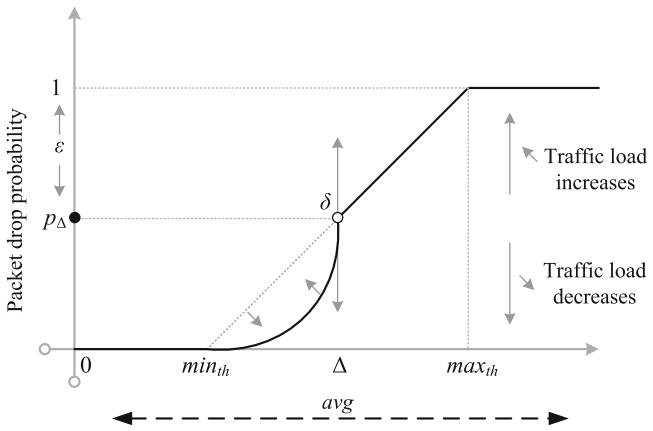

Fig. 2. FXRED's packet dropping probability curve

$$p_d = \begin{cases} 0 & avg < min_{th}, \\ 2^{\lfloor k \rfloor} \left(\frac{avg - min_{th}}{max_{th} - min_{th}} \right)^{\lfloor k \rfloor} \cdot (1 - \epsilon) & min_{th} \leq avg < \Delta, \\ 2\epsilon \left(\frac{avg - \Delta}{max_{th} - min_{th}} \right) + (1 - \epsilon) & \Delta \leq avg < max_{th}, \\ 1 & avg \geq max_{th}. \end{cases} \tag{9}$$

Algorithm 1: FXRED

1. Input: min_{th}, Δ, max_{th}, w, μ, c
2. Initialization:
3. $avg \leftarrow 0$
4. $count \leftarrow -1$
5. **for each** packet arrival **do**
6. Return $\lambda(t)$ [Mbps]
7. $\rho(t) \leftarrow \frac{\lambda(t)}{\mu}$
8. **if** $\rho(t) < 1$ **then**
9. $\gamma = 0.25$
10. **else if** $\rho(t) \approx 1$ **then**
11. $\gamma = 1$
12. **else if** $\rho(t) > 1$ **then**
13. $\gamma = c$
14. **end if**
15. $k \leftarrow c^{\frac{1}{\gamma}}$
16. $\epsilon \leftarrow c^{-\gamma}$
17. Compute the average queue length
18. **if** the queue is nonempty **then**
19. $avg \leftarrow (1 - w)avg' + w \cdot q(t)$
20. **else**
21. $m \leftarrow f(t - t_{queue\_idel\_time})$
22. $avg \leftarrow ((1 - w)^m \cdot avg')$
23. **end if**
24. **if** $avg < min_{th}$ **then**
25. No packet drop
26. Set $count \leftarrow -1$
27. **else if** $min_{th} \leq avg < \Delta$ **then**
28. Set $count \leftarrow count + 1$
29. Calculate packet drop probability P_a
30. $P_b \leftarrow 2^{\lfloor k \rfloor} \left(\frac{avg - min_{th}}{max_{th} - min_{th}} \right)^{\lfloor k \rfloor} \cdot (1 - \epsilon)$
31. $P_a \leftarrow \frac{P_b}{1 - count \cdot P_b}$
32. Mark the arriving packet with P_a
33. Set $count \leftarrow 0$
34. Drop the packet
35. **else if** $\Delta \leq avg < max_{th}$ **then**
36. Set $count \leftarrow count + 1$
37. Calculate packet drop probability P_a
38. $P_b \leftarrow 2\epsilon \left(\frac{avg - \Delta}{max_{th} - min_{th}} \right) + (1 - \epsilon)$
39. $P_a \leftarrow \frac{P_b}{1 - count \cdot P_b}$
40. Mark the arriving packet with P_a
41. Set $count \leftarrow 0$
42. Drop the packet
43. **else if** $max_{th} \leq avg$ **then**
44. Drop the arriving packet
45. Set $count \leftarrow 0$
46. **else** $count \leftarrow -1$
47. When router's queue becomes empty
48. Set $t_{queue\_idle\_time} \leftarrow t$
49. **end if**
50. **end for**

Table 1. FXRED's parameters

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

4 Simulation Experiments

To analyze the performance of the proposed algorithm, experiments were conducted using NS2 simulator. For the simulation, a network with a double dumbbell topology (Fig. 3) was used, where N TCP flows generated by N sources shared a bottleneck router R_1 (where AQM was implemented), the considered network also comprised of N sinks. The two routers R_1 and R_2 were connected via a bottleneck link with capacity 10Mbps and propagation delay of 20 ms.

Hosts were connected to routers via links with capacity 10 Mbps and propagation delay of 10 ms. The TCP data senders were of the New-Reno type. Different levels of traffic load were created by varying the number of flows across the bottleneck link.

In the experiments, number of flows N was increased from 5 to 95 and each point of the simulation result was obtained after 200s simulation. The queue capacity of router R_1 was 100 packets. The current value of $\rho(t)$ was obtained by applying the current value of $\lambda(t)$ received from queue monitoring object to the bottleneck link capacity. The queue threshold values used were $min_{th} = 25$, $max_{th} = 75$, where $\Delta = \frac{(max_{th}+min_{th})}{2}$, $c = 2$ and $w = 0.002$ (Table 1). The values of γ for three different load states are given by (10).

$$\gamma = \begin{cases} 0.25 & \text{if } \rho(t) < 0.75, \\ 1 & \text{if } \rho(t) \geq 0.75, \\ c & \text{if } \rho(t) \geq 1.25. \end{cases} \tag{10}$$

In this paper, the performance of FXRED is compared with one of the recently proposed enhanced RED versions, i.e. TRED. Since it was discovered in

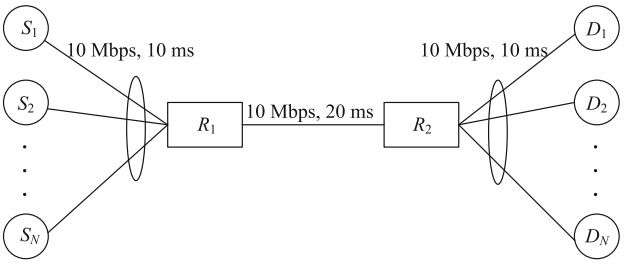

Fig. 3. Simulation topology

[15] that TRED provides better performance than RED, therefore, in this paper RED is not considered. For the analysis of TRED, $max_p = 0.1$ was used. Results obtained from the experiments conducted are presented in Fig. 4, 5, 6 and 7.

Graph presented in Fig. 4 has shown that at low load, with TRED the observed average queue length is slightly above the min_{th} value and as load becomes high the observed average queue length of TRED approaches the max_{th} as exhibited by RED (as observed in [18] and [19]). Although at low load TRED improved link utilization (better than RED [15]), however, at very high load states, TRED cannot cope and will incur forced packet drops and global synchronization [19]. On the other hand, since FXRED adapts its drop pattern based on load, at low load, FXRED adapts a drop pattern that will maintain average queue length well above the min_{th} in order to maximize link utilization and throughput, however, with increase in load, FXRED adapts a drop pattern that will maintain average queue length well below max_{th} in order to maintain stable performance. It can also be observed from graph presented in Fig. 4 that TRED works well at moderate load scenarios.

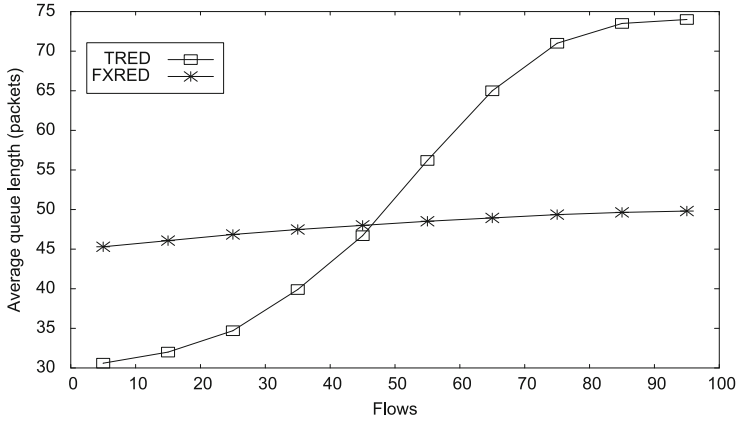

Fig. 4. Average queue length vs. number of flows

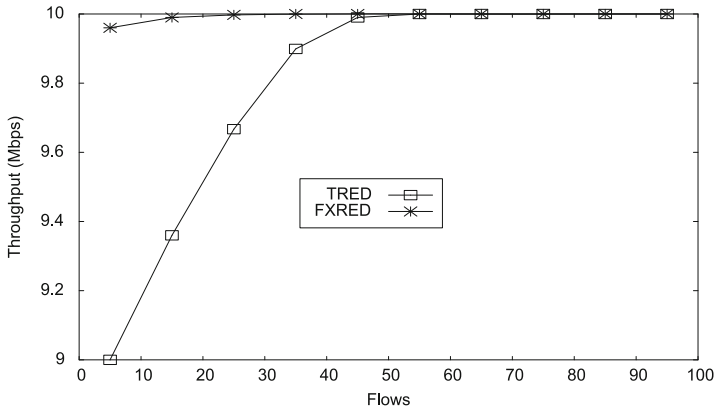

Fig. 5. Throughput vs. number of flows

Results presented in Fig. 5 revealed that FXRED maximizes throughput regardless of load, unlike TRED, at low load the bottleneck link is not fully utilized.

Graph presented in Fig. 6 has shown that delay encountered with FXRED at low load is higher than that of TRED since at low load, FXRED allows more packets to accumulate in the queue than TRED does (Fig. 4), hence, packets will encounter larger queuing delay with FXRED than with TRED, which will finally have impact on the end-to-end delay. However, at high load, the delay observed with FXRED is lower than that of TRED since in high load situation, FXRED has higher drop rate and subsequently sources will send fewer packets (Fig. 4).

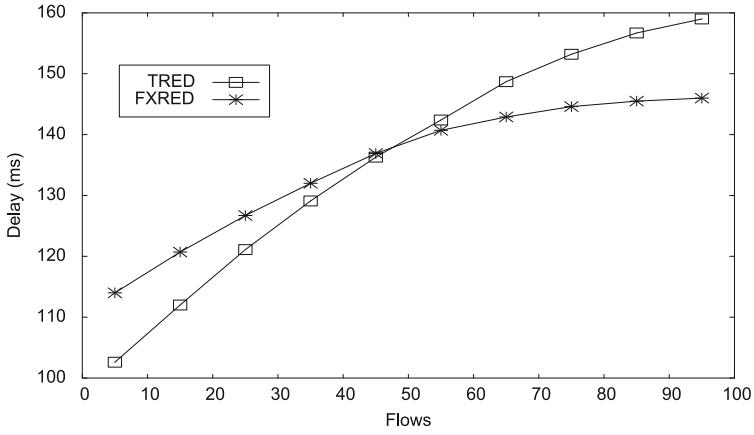

Fig. 6. Delay vs. number of flows

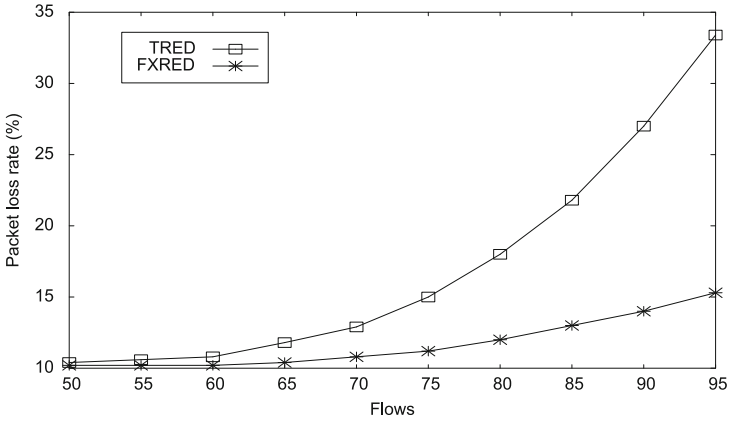

Fig. 7. Packet loss rate vs. number of flows

Finally packet loss rate was analyzed for moderate to high load situations and the results have shown that at high load, TRED has higher packet loss rates than FXRED (Fig. 7), this is because the observed packets losses with FXRED are solely due to its drop policy, while the observed packets losses with TRED are both due to its drop policy and forced packet drops.

5 Conclusion

In this paper, a Flexible Random Early Detection (FXRED) algorithm is proposed. Unlike other enhanced RED schemes with static drop patterns, FXRED considers the state of the current network's traffic load and adapts a suitable drop pattern. If the observed load is low, FXRED adapts a drop pattern that will gently drop packets since in that situation congestion is not a threat and it adapts a drop pattern that will aggressively drop packets when load becomes high in order to avoid congestion and overflow.

References

1. Chaudhary, P., Kumar, S.: A review of comparative analysis of TCP variants for congestion control in network. Int. J. Comput. Appl. **160**, 28–34 (2017)
2. Sharma, N., Rajput, S.S., Dwivedi, A.K., Shrimali, M.: P-RED: probability based random early detection algorithm for queue management in MANET. In: Bhatia, S.K., Mishra, K.K., Tiwari, S., Singh, V.K. (eds.) Advances in Computer and Computational Sciences. AISC, vol. 554, pp. 637–643. Springer, Singapore (2018). https://doi.org/10.1007/978-981-10-3773-3_62
3. Dmitry, K., et al.: Mobility-centric analysis of communication offloading for heterogeneous Internet of Things devices. Green Internet Things (IoT) Enabling Technol. Arch. Perform. Des., 1–10 (2018). https://doi.org/10.1155/2018/3761075

4. Vishnevsky, V., Dudin, A., Kozyrev, D., Larionov, A.: Methods of performance evaluation of broadband wireless networks along the long transport routes. In: Vishnevsky, V., Kozyrev, D. (eds.) DCCN 2015. CCIS, vol. 601, pp. 72–85. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-30843-2_8
5. Samouylov, K.E., Abaev, P.O., Gaidamaka, Y.V., Pechinkin, A.V., Razumchik, R.V.: Analytical modelling and simulation for performance evaluation of sip server with hysteretic overload control. In: 28th European Conference Proceedings on Modelling and Simulation, ECMS, pp. 603–609 (2014)
6. Gaidamaka, Y., Pechinkin, A., Razumchik, R., Samouylov, K., Sopin, E.: Analysis of an $M|G|1|R$ queue with batch arrivals and two hysteretic overload control policies. Int. J. Appl. Math. Comput. Sci. **24**(3), 519–534 (2014)
7. Braden, B., et al.: Recommendations on Queue Management and Congestion Avoidance in the Internet. RFC2309, United States, April 1998
8. Floyd, S., Jacobson, V.: Random early detection gateways for congestion avoidance. IEEE/ACM Trans. Netw. **1**, 397–413 (1993)
9. Cisco Systems, Congestion Avoidance Overview. https://www.cisco.com/c/en/us/td/docs/ios-xml/ios/qos_conavd/configuration/xe-16/qos-conavd-xe-16-book/qos-conavd-oview.html. Accessed 10 Sept 2020
10. Floyd, S.: Recommendation on Using the Gentle Variant of RED. https://www.icir.org/floyd/red/gentle.html. Accessed 10 Sept 2020
11. Cheng, M., Wang, H., Yan, L.: Dynamic RED: a modified random early detection. J. Comput. Inf. Syst. **7**, 5243–5250 (2011)
12. Zheng, B., Atiquzzaman, M.: DSRED: improving performance of active queue management over heterogeneous networks. In: IEEE ICC Proceedings, vol. 8, pp. 2375–2379 (2006)
13. Zhou, K., Yeung, K.L., Li, V.O.K.: Nonlinear RED: a simple yet efficient active queue management scheme. J. Comput. Netw. **50**, 3784–3794 (2006)
14. Zhang, Y., Ma, J., Wang, Y., Xu, C.: MRED: an improved nonlinear RED algorithm. In: International Conference Proceedings on Computer and Automation Engineering (ICCAE 2011), vol. 44, pp. 6–11 (2012)
15. Feng, C.W., Huang, L.F., Xu, C., Chang, Y.C.: Congestion control scheme performance analysis based on nonlinear RED. IEEE Syst. J. **11**, 2247–2254 (2017)
16. Karmeshu, Patel, S., Bhatnagar, S.: Adaptive mean queue size and its rate of change: queue management with random dropping. Telecommun. Syst. **11**, 287–295 (2017)
17. Tang, L., Tan, Y.: Adaptive queue management based on the change trend of queue size. KSII Trans. Internet Inf. Syst. **13**, 1345–1362 (2019)
18. Plasser, E., Ziegler, T., Reichl, P.: On the non-linearity of the RED drop function. In: Proceedings of ICCC, pp. 515–534 (2002)
19. Korolkova, A.V., Kulyabov, D.S., Velieva, T.R., Zaryadov, I.S.: Essay on the study of the self-oscillating regime in the control system. Commun. Eur. Counc. Model. Simul. Caserta Italy, pp. 473–480 (2019). https://doi.org/10.7148/2019-0473
20. Velieva, T.R., Korolkova, A.V., Kulyabov, D.S., Abramov, S.A.: Parametric study of the control system in the TCP network. In: 10th International Congress Proceedings on Ultra Modern Telecommunications and Control Systems, Moscow, Russian Federation, pp. 334–339 (2019). https://doi.org/10.1109/ICUMT.2018.8631267
21. Pavlotsky, O.E., Bobrikova, E.V., Samouylov, K.E.: Comparison of LBOC and RBOC mechanisms for SIP server overload control. In: Galinina, O., Andreev, S., Balandin, S., Koucheryavy, Y. (eds.) NEW2AN/ruSMART -2018. LNCS, vol. 11118, pp. 247–254. Springer, Cham (2018). https://doi.org/10.1007/978-3-030-01168-0_23

22. Kim, C., Dudin, S., Dudin, A., Samouylov, K.: Multi-threshold control by a single-server queuing model with a service rate depending on the amount of harvested energy. Perform. Eval. **127**(128), 1–20 (2018). https://doi.org/10.1016/j.peva.2018.09.001
23. Efrosinin, D., Gudkova, I., Stepanova, N.: Performance analysis and optimal control for queueing system with a reserve unreliable server pool. Commun. Comput. Inf. Sci. 1109, 109–120 (2019). https://doi.org/10.1007/978-3-030-33388-1_10