AQM Mechanism with the Dropping Packet Function Based on the Answer of Several PI^α Controllers

Adam Domański[1], Joanna Domańska[2], Tadeusz Czachórski[2], Jerzy Klamka[2], Dariusz Marek[1], and Jakub Szyguła[1](✉)

[1] Institute of Informatics, Silesian University of Technology, Akademicka 16, 44-100 Gliwice, Poland
jakub.szygula@polsl.pl

[2] Institute of Theoretical and Applied Informatics, Polish Academy of Sciences, ul. Bałtycka 5, 44-100 Gliwice, Poland

Abstract. In this paper the performance of AQM mechanism based on three PI^α controllers and the impact of traffic self-similarity on network utilization are investigated with the use of discrete event simulation modelling. The queue is divided into several thresholds. Each segment of the queue is controlled by a different PI^α mechanism. We analyze in tests the length of the queue and the number of rejected packets. The results obtained by the proposed approach are compared to the results obtained for AQM mechanism based on single PI^α controller.

Keywords: AQM · Congestion control · Non-integer order PI^α controller

1 Introduction

The most important factor of the TCP/IP network traffic control is the rejection of packets arriving to an IP router to be queued and send then forward. At first, packets are queued following FIFO algorithm and rejected only when the whole buffer space used to queue the packets was already occupied. Since many years, the recommended by IETF active queue management (AQM) where packets are rejected following a certain algorithm, enhances the efficiency of transfers [20] and cooperates better with TCP congestion window mechanism in adapting the flows intensity to the congestion of the network [2].

In the classic RED algorithms (the basic AQM mechanism) the incoming packet is dropped according to the given by a predefined function. Usually, this function is linear and depends on the queue length [8,11,15].

Our previous works proposed to base the probability function on the answer of the PI^α controller [5–7,10,13]. The considered models were based on the controller with the non-integer integrate/derivative orders.

In this article we reconsider this problem by extending the controller to include variable parameters. Similarly to algorithm DSRED [22] (the well-known

The original version of this chapter was revised: the acknowledgement section has been removed. The correction to this chapter is available at https://doi.org/10.1007/978-3-030-21952-9_30

P. Gaj et al. (Eds.): CN 2019, CCIS 1039, pp. 400–412, 2019.
https://doi.org/10.1007/978-3-030-21952-9_29

variant of the RED algorithm) we divided the queue length into the three separate segments. For each segment we choose a different set of parameters (controller PI^{α} coefficients and integrate/derivative orders). The first two controllers are *weak* i.e. for high traffic load the bulk of packets are dropped due to maximal queue size exceeding. The third controller is *strong*: its main task is to counteract the buffer overloading. Such choice of controllers enables the incremental increase of controller response as a result of growth of the traffic load.

The remainder of the paper is organized as follows: Sect. 2 gives basic notions on active queue management and presents the DSRED algorithm, Sect. 3 presents briefly theoretical basis for PI^{α} controller. Section 4 discusses numerical results. Some conclusions are given in Sect. 5.

2 The RED and DSRED Algorithms

The RED algorithm was the solution which fundamentally changed the principles of discarding packets in a router queue. In the case of passive queue management newly incoming packets are dropped only when the buffer is totally full. In the case of RED queue packets are rejected earlier - when the queue length exceeds a planned level. The authors of the RED algorithm: Sally Floyd and Van Jacobson [15] suggested that the destiny of this type of mechanism is to cooperate with transport protocols and congestion control mechanisms based on the positive acknowledgment.

Its performance is based on a drop function giving the probability that a packet is rejected. In RED drop function there are two thresholds: Min_{th} and Max_{th}. The argument avg of this function is a weighted moving average queue length. If $avg < Min_{th}$, all packets are admitted. If $Min_{th} < avg < Max_{th}$, then dropping probability p increases linearly:

$$p = p_{max} \frac{avg - Min_{th}}{Max_{th} - Min_{th}}$$

The value p_{max} corresponds to a probability of packet rejection in the case of $avg = Max_{th}$. If $avg > Max_{th}$ then all packets are dropped. Efficient operation of the RED mechanism is dependent on the proper selection of its parameters. There were several works studying the impact of various parameters on the RED performance.

Many variations of the RED mechanism were developed to improve its performance. They can be classified according to the modification of the method of control variable or dropping packet function calculation and according to how to configure and set the parameters of the algorithm.

One of the possibilities is to increase the thresholds number in the queue. In the algorithm DSRED (Double-Slope RED) [22], the bufor is divided into four sections. Three thresholds K_l, K_m and K_h (usually $K_m = (K_l + K_h)/2$) and parameter γ determine two slopes of this drop function:

$$p(avg) = \begin{cases} 0 & \text{if} \quad avg < K_l \\ \alpha(avg - K_l) & \text{if} \quad K_l \leq avg < K_m \\ 1 - \gamma + \beta(avg - K_m) & \text{if} \quad K_m \leq avg < K_h \\ 1 & \text{if} \quad K_h \leq avg \leq N \end{cases}$$

where

$$\alpha = \frac{2(1-\gamma)}{K_h - K_l}, \qquad \beta = \frac{2\gamma}{K_h - K_l}.$$

The double slope function makes the algorithm more elastic (more parameters to fix); gentle at the beginning (for low congestion) drop function enhances throughput and reduces queue waiting times. The advantages of this algorithm authors presented in [4] (Fig. 1).

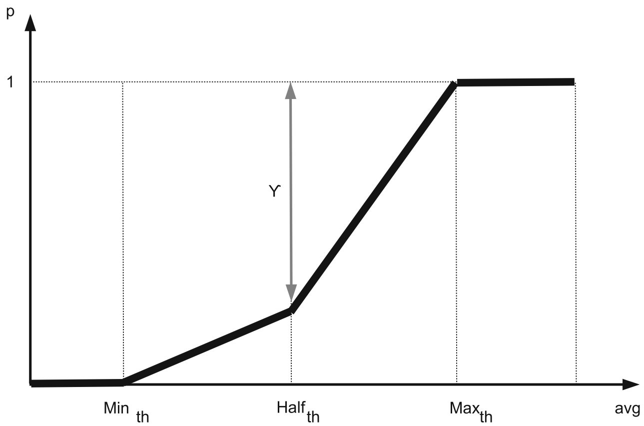

Fig. 1. The probability function of rejection the packet for the DSRED mechanism [4]

3 AQM Mechanism Based on Non-integer Order PI^α Controller.

Our papers [5–7,10] describe how to use the response from PI^α(non-integer integral order) to calculate the probability of packet loss. It is described by a formula:

$$p_i = max\{0, -(K_P e_k + K_I \Delta^\alpha e_k) \tag{1}$$

where K_P, K_I are tuning parameters, e_k is the error in current slot $e_k = Q_k - Q$, i.e. the difference between current queue Q_k and desired queue Q.

For standard PI controller (for $\alpha = -1$ and $\beta = 1$) the packet dropping probability is defined as follows:

$$p_i = max\{0, -(K_p e_i + K_i \sum_{j=i}^{0} e_j)\} \tag{2}$$

In this approach, the dropping probability depends on three parameters: the coefficients for the proportional and integral terms (K_p, K_i) and integrals (α) orders.

The Fractional Order Derivatives and Integrals (FOD/FOI) definitions unify the notions of derivative and integral to one differintegral definition. The most popular formulas to calculate differintegral numerically are Grunwald-Letnikov (GrLET) formula and Riemann-Liouville formulas (RL) [3,16,18].

Differintegral is a combined differentiation/integration operator. The q-differintegral of function f, denoted by $\Delta^q f$ is the fractional derivative (for $q > 0$) or fractional integral (if $q < 0$). If $q = 0$, then the q-th differintegral of a function is the function itself.

In the case of discrete systems (in the active queue management, packet drop probabilities are determined at discrete moments of packet arrivals) there is only one definition of differ-integrals of non-integer order. This definition is a generalization of the traditional definition of the difference of integer order to the non-integer order and it is analogous to a generalization used in Grunwald-Letnikov (GrLET) formula.

For a given sequence $f_0, f_1, ..., f_j, ..., f_k$

$$\triangle^q f_k = \sum_{j=0}^{k} (-1)^j \binom{q}{j} f_{k-j} \tag{3}$$

where $q \in R$ is generally a non-integer fractional order, f_k is a differentiated discrete function and $\binom{q}{j}$ is generalized Newton symbol defined as follows:

$$\binom{q}{j} = \begin{cases} 1 & \text{for } j = 0 \\ \dfrac{q(q-1)(q-2)..(q-j+1)}{j!} & \text{for } j = 1, 2, \ldots \end{cases} \tag{4}$$

Articles [5,7,10] show that using the non-integer order PI^{α} controller as AQM mechanism is more efficient in network congestion control than standard RED mechanism and improves the router performance. The approach proposed in the article divides the queue length into several segments and for each of them use a different set of controller coefficients. This solution should result in more flexible behavior of AQM mechanism independently of the network load or long-range dependence of the network traffic.

4 Packet Dropping Scheme Based on the Answer of the Three PI^{α} Controllers

The AQM algorithms drop packets following a dropping packet function. The choice of the proper coefficients of this function is not easy. These parameters may differ and depend on the network traffic profile.

This problem also exists in the case of the dropping packets functions based on the answers of the PI^{α} controllers. Our previous works show significant influence of the traffic parameters (intensity and self-similarity) on the choice of the optimal controller parameters. The AQM mechanism should change its parameters during operation as a result of traffic load. One of the possibilities is to change the parameters of the controller as a function of the queue occupancy.

This article presents the DSRED-like solution. We divide the queue length iwith the use of thresholds. Each segment of the queue is controlled by a different PI^{α} mechanism.

For queue length between 0 and 180 (packets) we use only one controller. For queue length from 180 to 220 the probability of packet dropping is a sum of answers of the first and second controller. When the queue occupancy exceeds 220 the probability is the sum of responses of all three controllers.

The packet dropping probability may be defined as follows:

$$p(q) = \begin{cases} p_1(q) & \text{if} \quad q < 180 \\ p_1(q) + p_2(q) & \text{if} \quad 180 \leq q < 220 \\ p_1(q) + p_2(q) + p_3(q) & \text{if} \quad 220 \leq q \end{cases}$$

where

p_1 - answer of the first controller,
p_2 - answer of the second controller,
p_2 - answer of the third controller.

All presented in this article results were obtained using the simulation model. The simulations were done using the Simpy Python packet. To accelerate the calculations the PI^{α} module was written in C language. During the tests, we analyzed the following parameters of the AQM transmission: the length of the queue and the number of rejected packets. The input traffic intensity $\lambda = 0.5$ was considered independently of the Hurst parameter. During the tests we changed the Hurst parameter of the input traffic within the range from 0.5 to 0.90. We use a fast algorithm for generating approximate sample paths for a fGn process, first introduced in [17]. After each trace generation the Hurst parameter was estimated with the use of popular self-similarity parameter estimators [9,12,14]: the R/S statistic, aggregated variance, periodogram as well known methods with a significant history of use in estimating LRD and wavelet based method, local Whittle's estimator as newer techniques. Traditional Hurst parameter estimators can be really biased [1,21]. Additionally, the different implementations of the same method may give varying results [19]. Only Hurst parameter estimator based on wavelets can be treated as unbiased and robust [21].

The Table 1 presents the estimations for sample generated trace with the assumed Hurst parameter. These results show that the assumed and estimated Hurst parameters are not the same. The obtained results changed for subsequent generated samples and differed depending on the method of estimating the Hurst parameter. For all differences in results, the dependence of the increase in the estimated Hurst parameter with the increase in the assumed parameter is clearly visible.

Table 1. Hurst parameter estimates for IITiS data traces

	H = 0.5	H = 0.6	H = 0.7	H = 0.8	H = 0.9
Estimator	Estimated Hurst parameter				
R/S method	0.6289	0.6638	0.7338	0.7486	0.7666
Aggregate variance method	0.5710	0.6710	0.7805	0.8785	0.9521
Periodogram method	0.5278	0.6383	0.7601	0.8735	0.9589
Whittle method	0.6889	0.7485	0.8021	0.8429	0.8565
Wavelet-based method	0.5872	0.6859	0.7893	0.8759	0.9337

The service time represents the time of a packet treatment and dispatching. In packet-switched networks it is the time required to transmit information. We have used discrete-time model, hence we have assumed that service-time distribution is geometric (which corresponds to Poisson traffic in case of continuous time models). The distribution of service time μ changed during the test.

The high traffic load was considered for parameter $\mu = 0.25$. The average traffic load we obtained for $\mu = 0.5$. Small network traffic was considered for parameter $\mu = 0.75$

The PI^α controllers coefficients and setpoints presents Table 2. The impact of controller parameters on the behavior of the AQM mechanism and packet dropping probability were described in [5,13]. In presented solution first and second controllers drop the some packets but mostly the queue size crosses the third threshold. When the queue size exceeded third threshold, the third controller begin to work. The third (strong) controller protects the queue against exceeding the maximum size.

Table 2. PI^α controllers coefficients

	K_p	K_i	α	Setpoint
1	0.0001	0.00040	−0.4	100
2	0.0001	0.00015	−0.5	180
3	0.0001	0.00035	−0.6	220

The distributions of the queue length present Figs. 2 and 3. The Tables 3, 4 and 5 present the detailed results. The results consider the high traffic load ($\mu = 0.25$). The Table 3 presents the results for the first controller. In our solution this controller works for the queue occupancy between 0 and 180. The controller parameters were chosen to maximize the queue length. However, the majority of packets are dropped by PI^α mechanism, several packets are dropped due to maximum queue length exceeding. The number of packets dropped by the queue increases with the Hurst parameter. The Table 4 presents the controller that starts when queue length exceeds 180. This controller is weak. The most packets are dropped by the queue. The third controller (Table 5) is very strong. All packets are discarded from the queue by controller mechanism. Obtained results confirmed the assumptions of the controllers behavior.

Table 3. PI^α controller, $\mu = 0.25$, $K_p = 0.0001$, $K_i = 0.0004$, $\alpha = -0.4$, setpoint = 100

Hurst	Avg. queue length	Packet drop by	
		PI^α	Queue
0.50	270.42	2492385	10134
0.60	268.90	2467277	30821
0.70	264.08	2374004	124992
0.80	246.98	2115155	384445
0.90	203.94	1744739	875155

Table 4. PI^α controller, $\mu = 0.25$, $K_p = 0.0001$, $K_i = 0.00015$, $\alpha = -0.5$, setpoint = 100

Hurst	Avg. queue length	Packet drop by	
		PI^α	Queue
0.50	296.95	1350549	1149714
0.60	295.94	1339802	1157844
0.70	292.22	1307309	1190066
0.80	275.13	1162557	1339373
0.90	222.05	875029	1735620

The proposed solution sums the behavior of all three presented above controllers. The packet dropping probability increases with assumed thresholds.

Tables 6, 7 and 8 present obtained results for different traffic intensity. The Table 6 presents the overloaded network. Although two first controllers drop most packets, the queue length exceeds the third threshold. The advantage of this solution is a small reaction of the first and second PI^α in the case of highly variable traffic. For $H = 0.90$ the most packets are dropped by third PI^α.

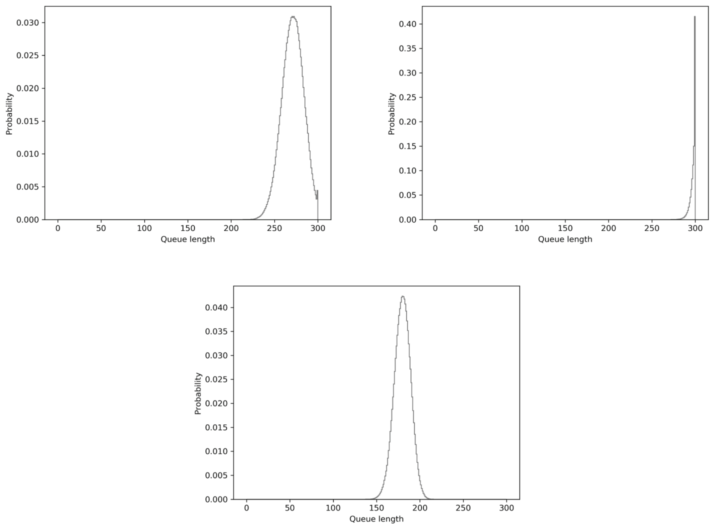

Fig. 2. Distribution of queue length for high traffic load ($\mu = 0.25$) and H = 0.5, left: $K_p = 0.0001$, $K_i = 0.00015$, $\alpha = -0.5$, right: $K_p = 0.0001$, $K_i = 0.0004$, $\alpha = -0.4$, center: $K_p = 0.0001$, $K_i = 0.00035$, $\alpha = -0.6$

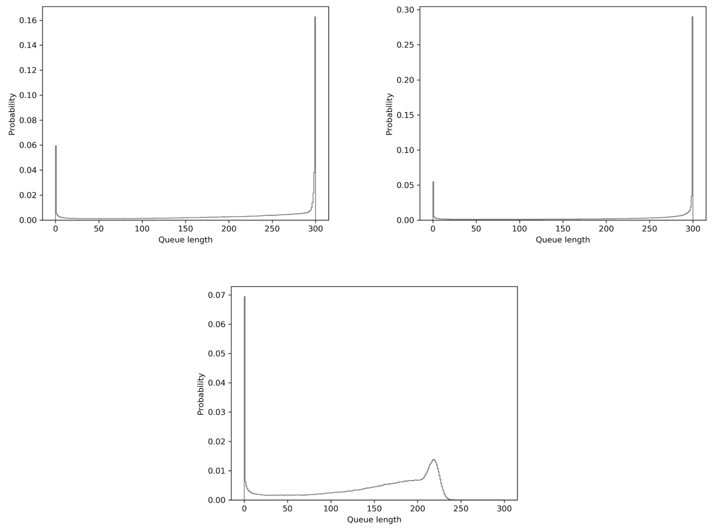

Fig. 3. Distribution of queue length for high traffic load ($\mu = 0.25$) and H = 0.9, left: $K_p = 0.0001$, $K_i = 0.00015$, $\alpha = -0.5$, right: $K_p = 0.0001$, $K_i = 0.0004$, $\alpha = -0.4$, center: $K_p = 0.0001$, $K_i = 0.00035$, $\alpha = -0.6$

Table 5. PI^{α} controller, $\mu = 0.25$, $K_p = 0.0001$, $K_i = 0.00035$, $\alpha = -0.6$, set-point = 100

Hurst	Avg. queue length	Packet drop by	
		PI^{α}	Queue
0.50	179.73	2499983	0
0.60	179.11	2498358	0
0.70	177.20	2498517	0
0.80	169.19	2504336	0
0.90	142.84	2638408	0

Independently of the degree of self-similarity, no packets are dropped by the queue.

The Table 7 presents the results in the case of the average traffic load. The average queue length does not exceed 80 packets (independently of the traffic self-similarity). However, the detailed results suggest that temporarily the queue length exceeds the third thresholds. The number of dropped packet by second and third controller grows with the degree od self-similarity. This phenomenon is caused by high variability of queue occupancy. This variability grows with Hurst parameter.

The average queue length for small network traffic (Table 8) is the largest in the case of $H = 90$. The queue length never exceeds the third threshold. All packets are dropped by two earlier controllers. The queue length exceeds the firt threshold only in case of degree of self-similarity (expressed in Hurst parameter) exceeds 0.8.

Table 6. Three PI^{α} controllers, $\mu = 0.25$

Hurst	Avg. queue length	Packet drop in stage			Sum of packet loss
		First	Second	Third	
0.50	199.42	62275	2324331	112980	2499586
0.60	199.78	126334	2148758	223179	2498271
0.70	199.32	251743	1758947	491208	2501898
0.80	190.33	317066	1229494	955963	2502523
0.90	158.95	204411	711370	1716164	2631945

Figure 4 presents distributions of the queue lengths depended on the traffic intensity and the degree of self-similarity.

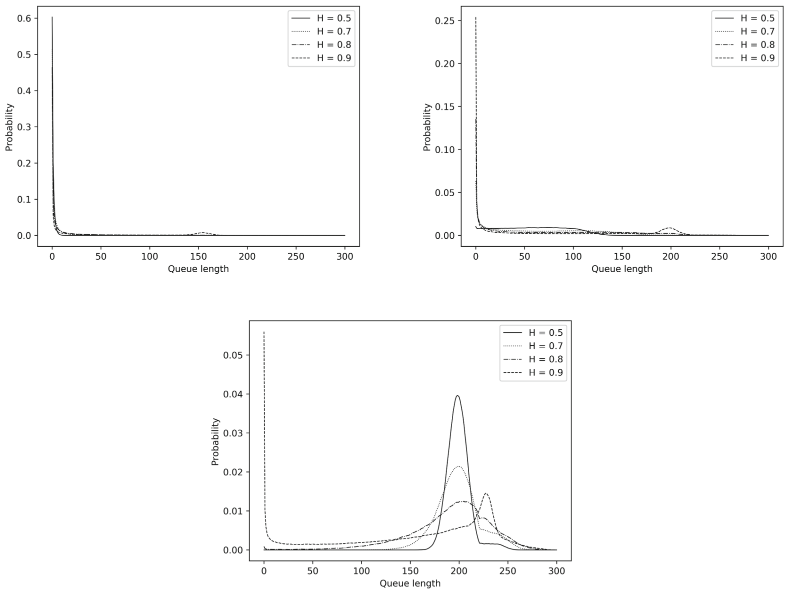

Fig. 4. The influence of degree of traffic self-similarity on queue distribution, three PI^{α} controllers, $\mu = 0.75$ (left), $\mu = 0.5$ (right), $\mu = 0.25$ (bottom)

Table 7. Three PI^{α} controllers, $\mu = 0.50$

Hurst	Avg. queue length	Packet drop in stage			Sum of packet loss
		First	Second	Third	
0.50	58.47	33866	0	0	33866
0.60	61.19	86436	62	0	86498
0.70	65.77	208121	20793	2045	230959
0.80	72.98	293810	218707	42700	555217
0.90	78.88	205968	840522	86260	1132750

Table 8. Obtained results for the input traffic intensity $\mu = 0.75$

Hurst	Avg. queue length	Packet drop in stage			Sum of packet loss
		First	Second	Third	
0.50	0.79	0	0	0	0
0.60	1.23	0	0	0	0
0.70	3.07	94	0	0	94
0.80	14.0	48981	53	0	49034
0.90	35.9	368562	1097	0	369659

5 Conclusions

The Internet Engineering Task Force (IETF) organization recommends that IP routers should use the active queue management mechanisms (AQMs). The basic algorithm for AQM is the RED algorithm. There are many modifications and improvements to the RED mechanism. One of these improvements is the calculation of the probability of packet loss using a PI^{α} controller. Our previous work has shown the advantage of this solution [5,10].

This paper introduces a new way of packet rejecting probability calculation based on the answer of three the non-integer order PI^{α} controllers. The additional controllers start to work when the queue occupancy exceeds the assumed threshold. The behavior of the proposed solution was also compared to the behavior of the queue controlled by a single PI^{α} controller. Obtained results show the advantage of such a solution. Individually, the PI^{α} controllers presented in the article are poorly adjusted to the network traffic. The first and the second controller did not work properly in the case of high traffic intensity. Most packets were dropped due to exceeding the maximum queue size. The reaction of the third controller was too strong. In the case of low traffic intensity the number of discarded packets was redundant. Only the combination of described above controllers allowed to design more flexible AQM mechanism.

Our article presents also the impact of the degree of self-similarity (expressed in the Hurst parameter) on the length of the queue and the number of rejected packets. Obtained results are closely related to the degree of self-similarity. The experiments are carried out for the four types of traffic ($H = 0.5, 0.7, 0.8, 0.9$). Additionally, we evaluate the number of dropped packets in assumed queue segments. This results allowed to select the desired parameters of the controller.

The results described in this article confirm that our approach increases the efficiency of the AQM mechanism based on the PI^{α} controller. In presented solution we refere mainly to the queue occupancy. In our future work we will focus on mechanisms based on the evaluation of the network traffic parameters and the selection of controller parameters according to the intensity or the self-similarity of the network traffic.

References

1. Abry, P., Veitch, D.: Wavelet analysis of long-range-dependent traffic. IEEE Trans. Inf. Theory **44**(1), 2–15 (1998). https://doi.org/10.1109/18.650984
2. Braden, B., et al.: Recommendations on queue management and congestion avoidance in the internet. Network Working Group - Request for Comments: 2309, IETF (1998)
3. Leszczyński, J., Ciesielski, M.: A numerical method for solution of ordinary differential equations of fractional order. In: Wyrzykowski, R., Dongarra, J., Paprzycki, M., Waśniewski, J. (eds.) PPAM 2001. LNCS, vol. 2328, pp. 695–702. Springer, Heidelberg (2002). https://doi.org/10.1007/3-540-48086-2_77

4. Domańska, J., Domański, A., Czachórski, T.: The drop-from-front strategy in AQM. In: Koucheryavy, Y., Harju, J., Sayenko, A. (eds.) NEW2AN 2007. LNCS, vol. 4712, pp. 61–72. Springer, Heidelberg (2007). https://doi.org/10.1007/978-3-540-74833-5_6
5. Domański, A., Domańska, J., Czachórski, T., Klamka, J.: Self-similarity traffic and AQM mechanism based on non-integer order $PI^{\alpha}D^{\beta}$ controller. In: Gaj, P., Kwiecień, A., Sawicki, M. (eds.) CN 2017. CCIS, vol. 718, pp. 336–350. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-59767-6_27
6. Domański, A., Domańska, J., Czachórski, T., Klamka, J., Marek, D., Szyguła, J.: GPU accelerated non-integer order $PI^{\alpha}D^{\beta}$ controller used as AQM mechanism. In: Gaj, P., Sawicki, M., Suchacka, G., Kwiecień, A. (eds.) CN 2018. CCIS, vol. 860, pp. 286–299. Springer, Cham (2018). https://doi.org/10.1007/978-3-319-92459-5_23
7. Domański, A., Domańska, J., Czachórski, T., Klamka, J., Szyguła, J.: The AQM dropping packet probability function based on non-integer order $PI^{\alpha}D^{\beta}$ controller. In: Ostalczyk, P., Sankowski, D., Nowakowski, J. (eds.) RRNR 2017. LNEE, vol. 496, pp. 36–48. Springer, Cham (2019). https://doi.org/10.1007/978-3-319-78458-8_4
8. Domański, A., Domańska, J., Czachórski, T.: Comparison of AQM control systems with the use of fluid flow approximation. In: Kwiecień, A., Gaj, P., Stera, P. (eds.) CN 2012. CCIS, vol. 291, pp. 82–90. Springer, Heidelberg (2012). https://doi.org/10.1007/978-3-642-31217-5_9
9. Domańska, J., Domański, A., Czachórski, T.: Estimating the intensity of long-range dependence in real and synthetic traffic traces. In: Gaj, P., Kwiecień, A., Stera, P. (eds.) CN 2015. CCIS, vol. 522, pp. 11–22. Springer, Cham (2015). https://doi.org/10.1007/978-3-319-19419-6_2
10. Domańska, J., Domański, A., Czachórski, T., Klamka, J.: The use of a non-integer order PI controller with an active queue management mechanism. Int. J. Appl. Math. Comput. Sci. **26**, 777–789 (2016). https://doi.org/10.1515/amcs-2016-0055
11. Domańska, J., Domański, A., Augustyn, D., Klamka, J.: A RED modified weighted moving average for soft real-time application. Int. J. Appl. Math. Comput. Sci. **24**(3), 697–707 (2014). https://doi.org/10.2478/amcs-2014-0051
12. Domańska, J., Domański, A., Czachórski, T.: Modeling packet traffic with the use of superpositions of two-state MMPPs. In: Kwiecień, A., Gaj, P., Stera, P. (eds.) CN 2014. CCIS, vol. 431, pp. 24–36. Springer, Cham (2014). https://doi.org/10.1007/978-3-319-07941-7_3
13. Domański, A., Domańska, J., Czachórski, T., Klamka, J., Marek, D., Szyguła, J.: The influence of the traffic self-similarity on the choice of the non-integer order PI^{α} controller parameters. In: Czachórski, T., Gelenbe, E., Grochla, K., Lent, R. (eds.) ISCIS 2018. CCIS, vol. 935, pp. 76–83. Springer, Cham (2018). https://doi.org/10.1007/978-3-030-00840-6_9
14. Estrada-Vargas, L., Torres Roman, D., Toral-Cruz, H.: A study of wavelet analysis and data extraction from second-order self-similar time series. Math. Probl. Eng. 1–14 (2013). https://doi.org/10.1155/2013/102834
15. Floyd, S., Jacobson, V.: Random early detection gateways for congestion avoidance. IEEE/ACM Trans. Netw. **1**(4), 397–413 (1993). https://doi.org/10.1109/90.251892
16. Miller, K., Ross, B.: An Introduction to the Fractional Calculus and Fractional Differential Equations. Wiley, New York (1993)
17. Paxson, V.: Fast, approximate synthesis of fractional Gaussian noise for generating self-similar network traffic. ACM SIGCOMM Comput. Commun. Rev. **27**(5), 5–18 (1997). https://doi.org/10.1145/269790.269792

18. Podlubny, I.: Fractional Differential Equations, vol. 198. Academic Press, San Diego (1999)
19. Ramirez-Pacheco, J., Torres-Román, D., Toral-Cruz, H., Estrada-Vargas, L.: High-performance tool for the test of long-memory and self-similarity. In: Simulation Technologies in Networking and Communications: Selecting the Best Tool for the Test. CRC Press/Taylor & Francis Group, pp. 93–114 (2014). https://doi.org/10.1201/b17650-6
20. Sawicki, M., Kwiecień, A.: Unexpected anomalies of isochronous communication over USB 3.1 Gen 1. Comput. Stand. Interfaces **49**, 67–70 (2017). https://doi.org/10.1016/j.csi.2016.08.010
21. Stolojescu, C., Isar, A.: A comparison of some Hurst parameter estimators. In: 13th International Conference on Optimization of Electrical and Electronic Equipment (OPTIM), pp. 1152–1157 (2012). https://doi.org/10.1109/OPTIM.2012.6231802
22. Zheng, B., Atiquzzaman, M.: DSRED: a new queue management scheme for the next generation internet. IEICE Trans. Commun. 242–251 (2000). https://doi.org/10.1093/ietcom/e89-b.3.764