Int. J. Appl. Math. Comput. Sci., 2021, Vol. 31, No. 1, 165–178
DOI: 10.34768/amcs-2021-0012

THE IoT GATEWAY WITH ACTIVE QUEUE MANAGEMENT

ADAM DOMAŃSKI [a], JOANNA DOMAŃSKA [b], TADEUSZ CZACHÓRSKI [b], JERZY KLAMKA [b], JAKUB SZYGUŁA [a,*], DARIUSZ MAREK [a]

[a]Department of Distributed Systems and Informatic Devices
Silesian University of Technology
ul. Akademicka 16, 44-100 Gliwice, Poland
e-mail: {adamd,jakub.szygula,dariusz.marek}@polsl.pl

[b]Institute of Theoretical and Applied Informatics
Polish Academy of Sciences
ul. Bałtycka 5, 44-100 Gliwice, Poland
e-mail: {joanna,tadek,jerzy.klamka}@iitis.pl

As the traffic volume from various Internet of things (IoT) networks increases significantly, the need for adapting the quality of service (QoS) mechanisms to the new Internet conditions becomes essential. We propose a QoS mechanism for the IoT gateway based on packet classification and active queue management (AQM). End devices label packets with a special packet field (type of service (ToS) for IPv4 or traffic class (TC) for IPv6) and thus classify them as priority for real-time IoT traffic and non-priority for standard IP traffic. Our AQM mechanism drops only non-priority packets and thus ensures that real-time traffic packets for critical IoT systems are not removed if the priority traffic does not exceed the maximum queue capacity. This AQM mechanism is based on the PI$^\alpha$ controller with non-integer integration order. We use fluid flow approximation and discrete event simulation to determine the influence of the AQM policy on the packet loss probability, queue length and its variability. The impact of the long-range dependent (LRD) traffic is also considered. The obtained results show the properties of the proposed mechanism and the merits of the PI$^\alpha$ controller.

Keywords: active queue management, PI controller, dropping packets, fractional calculus, IoT gateway.

1. Introduction

The number of deployed smart Internet of things (IoT) objects has increased significantly in recent years (Stoica *et al*., 2001). According to Cisco, more than 50 billion IoT devices will be connected to the Internet by 2020 (Miao *et al*., 2018), and network traffic management must meet the challenge of new requirements. The IoT environment consists of sensors and actuators that may require real-time transmission.

Detailed research on the traffic characteristics of the IoT network has not been conducted yet (Dymora and Mazurek, 2019). Shafiq *et al*. (2013) take a first look at the characteristics of machine-to-machine traffic, and it is stressed that this traffic has a much greater uplink-to-downlink ratio and generates synchronized traffic through aggregation. Hsu *et al*. (2017) propose a hybrid traffic generator which integrates big data and machine type communication traffic models. They underline that network traffic constitutes a combination of human-to-human and machine-to-machine traffic. Dymora and Mazurek (2019) offer the possibility of using the Hurst parameter analysis to detect anomalies in the IoT communication network.

The IoT devices are connected to the Internet using access communication protocols such as WiFi, LoraWAN, Bluetooth, Ethernet, or ZigBee. Data collected by sensors may be sent to servers (or actuators) connected to the Internet. Therefore, the crucial element is a special IoT gateway (see Fig. 1) that aggregates and processes data from sensors prior to transporting them via the Internet (Brun *et al*., 2018). Such a transmission model represents the best-effort service and does not guarantee the transmission quality.

The Internet Engineering Task Force (IETF)

*Corresponding author

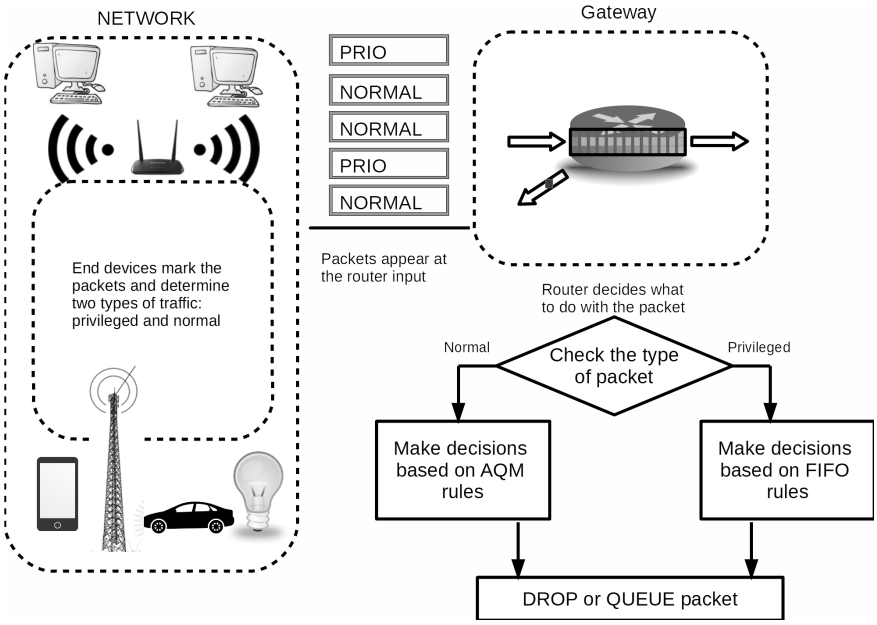


Fig. 1. IoT gateway: the node between the network of small devices and the Internet.

proposed two quality of service (QoS) architectures: integrated services (IntServ) and differentiated services (DiffServ). The implementation of the QoS architecture inside the IoT devices is challenging as the hardware capabilities of most of them are limited (Adjih *et al*., 2015). For such devices, the IETF proposed a special standard protocol 6LowPAN (IPv6 over low-power wireless personal area networks). The protocol classifies the traffic into time-sensitive and regular traffic specified in two priority bits. However, this solution requires the implementation of the protocol in all devices, and this may reduce its popularity.

Palattella *et al*. (2016) prove that it is not possible to provide a service level agreement (SLA) for IoT devices that use the ZigBee protocol, Bluetooth or WiFi as a network access communication protocol. This could be the reason why it is not possible to guarantee data transmission at predefined times (Kittipipattanathaworn and Nupairoj, 2014).

One of the mechanisms to guarantee QoS for real-time IoT communication between the access node and the cloud is the congestion control mechanism (Palattella *et al*., 2016). Halim *et al*. (2016) state that congestion control in IoT access nodes is still a big challenge that has not been fully addressed. According to Al-Kashoash *et al*. (2017), new congestion control algorithms for the 6LowPAN protocol should be developed.

This problem becomes serious with large scale IoT deployments where hundreds of thousands of IoT devices may be connected with a single access node. It is expected that the computational power of IoT devices will increase significantly along with the number of transmitted data. IoT devices are generating huge amounts of data traffic (Chandrashekhara and Veena, 2018). Chou *et al*. (2019) claim that the IoT-based systems of measuring the gain velocity generate 1.7 GB data per one smart home per day. The increasing amount of data traffic from different IoT networks will increase the complexity in the operation and management of legacy IP networks and may also degrade the QoS. When the IPv6 standard is used, all devices may receive their own unique IP addresses (Berte, 2018), and IoT traffic can thus be processed in this same way as the traditional IP traffic. Google Cloud, Microsoft Azure and Amazon AWS (three largest cloud companies) provide connections for MQTT (message queue telemetry transport) services, which means that these devices will be usually to the network and generate an important traffic.

It is not straightforward to master strict QoS guarantees in wireless networks because of resource allocation and management ability constraints in shared wireless media (Gubbi *et al*., 2012). However, the mechanisms allowing the achievement of a certain level of QoS via increasing the flexibility of the network traffic have proven to be very useful. Domańska *et al*. (2014) describe the idea of providing a certain level of QoS that meets the soft-real time requirements based on the congestion control mechanism implemented in the IoT gateway (analogously to the DiffServ solution).

1.1. Our contribution. This paper proposes an improvement in transmission conditions for IoT devices, based on the scheduling of IoT packets.

Our main goal is to implement the specific active

queue management (AQM) mechanisms in the IoT gateway. The IoT devices mark the packets using the "type of service" field (inside the IPv4 header), thus determining the type of a packet. In the case of the IPv6 protocol, the mechanism of determining the type of packet is the same (the "traffic class" field replaces the "type of service" functionality). We consider two types of traffic: privileged and normal. The decision about the type of transmission (normal or priority) is taken at the stage of IoT device configuration. The administrator must decide which data from which sensors must meet the SLA requirements while creating the network, and for them, he/she sets a high priority.

One of the functionalities of the IoT gateway is data routing, and therefore this gateway is called a router.

The proposed solution is based on two mechanisms built into the gateway (see Fig. 1): assessment of the type of packets (the division into priority and non-priority packages, according to the information stored in the type of service (ToS) field of IPv4 or in the traffic class of IPv6) and AQM. The network traffic is split into two streams. The packets from the normal traffic are controlled by the AQM mechanism. The packets belonging to the privileged stream share the queue with normal packets but are not under the control of AQM. The first goal is to reduce delays for sensitive traffic. This is achieved by the average queue size regulation using the AQM mechanism. In addition, because AQM drops packets of the normal stream, there is no loss in the priority traffic if the maximum queue size is not exceeded.

In the classic AQM algorithms (random early detection (RED)) (Floyd and Jacobson, 1993) and its modifications), the incoming packet is considered to be dropped according to an assumed loss probability function. This function is usually linear and depends on the moving average of the queue length.

The RED algorithm and its modifications (i.e., double slope RED (DSRED) (Zheng and Atiquzzaman, 2000) or nonlinear RED (NLRED) (Domańska *et al*., 2012)) are very sensitive to properties of the network traffic, such as the intensity or degree of the long-range dependence (LRD). Section 5 provides the explanation of the LRD concept. In the case of overloaded nodes such mechanisms are not able to maintain the intended queue length and very often the maximum queue size may be exceeded (Domański *et al.*, 2016; Domańska *et al.*, 2014; 2012). For this reason, they would not suit the proposed solution.

In our previous works, we proposed to base the loss probability function on the indications of the PI^{α} controller having non-integer integrate/derivative orders (Domańska *et al*., 2018). Domańska *et al*. (2016) demonstrate that using the non-integer order PI^{α} controller as an AQM mechanism is more efficient for network congestion control than a standard RED mechanism and improves the router's performance. Bingi *et al*. (2019) emphasize that the key feature of the fractional-order PID controller is its insensitivity to system parameters which ensures stable performance.

The remainder of the paper is organized as follows. Section 2 describes the related works. Section 3 presents theoretical background of the PI^{α} controller used in approximation and simulations. A comparison of PI^{α} controller to the FIFO scheduling is presented in Section 4. Section 5 provides simulation results of the open loop scenario in the presence of LRD traffic. Concluding remarks are presented in Section 6.

2. Related work

The problem of adapting real-time traffic to TCP/IP networks is studied frequently and a number of solutions have been proposed. The problem of real-time data transmission over the best-effort Internet is described by Wang (2009). Wijnants and Lamotte (2008) present the method for maintaining the network bandwidth for many client applications. Skeie *et al*. (2006) suggest using prioritization mechanisms to ensure the requirements of the IEEE 802.1D standard and indicate that the end nodes are mainly responsible for traffic latency. Some authors propose reservation schemes and feedback techniques to provide real-time guarantees.

The IETF groups propose the RSVP protocol, IntServ and DiffServ for providing better QoS control in IP networks. IntServ ensures end-to-end QoS by means of hop-by-hop resource reservation inside the IP network. IntServ is based on mathematical guarantees of bandwidth, delay, jitter, and it supports specific applications such as video streaming. IntServ provides individualized QoS guarantees to separate applications but needs serious changes in the IP network. This architecture requires the implementation of the RSVP protocol inside each core router, increasing its complexity. Totally different solutions are provided by DiffServ. This approach categorizes traffic into different classes and applies QoS parameters to these classes. It requires relatively little changes to the network and transport layers. The weakness of DiffServ is the lack of strict QoS guarantees.

On the other hand, some applications do not require strict QoS guarantees. The distinction between hard, firm and soft real-time systems is described by Stankovic and Rajkumar (2004). Many researchers study the problem of achieving soft, or statistical, and not only hard real-time QoS guarantees. Cucinotta *et al*. (2010) present examples of soft real-time applications in the industrial systems. Palopoli *et al*. (2000) discuss control applications using a soft-real-time environment. These are propositions of serious changes in the existing best-effort solution. The easiest way to ensure a suitable QoS over an IP network

is to increase bandwidth, but it is expensive. Alternative solutions postulate the provision of new service models and mechanisms. They use various mechanisms, such as AQM, resource reservation signalling or scheduling. The first one (AQM) is very useful in limiting the average queue size and thus controlling the delays in the queue.

The classic AQM mechanism RED is proposed by IETF and primarily described by Floyd and Jacobson (1993). It is based on a drop function giving the probability that a packet is rejected. The packet loss is a normal event in a computer network (Jiang *et al*., 2018). The argument of the packet dropping probability function is a weighted moving average queue length, acting as a low-pass filter. This average depends on the actual queue occupancy and a previously calculated value of the weighted moving average. The packet dropping probability is based on a linear function.

Despite the obvious advantages, RED also has drawbacks such as low throughput, unfair bandwidth sharing, the introduction of variable latency, deterioration of network stability. Therefore, there are numerous suggestions for improvement. Domańska *et al*. (2012) propose the NLRED algorithm, where the linear packet dropping function is replaced by a 3rd order polynomial function. For nonlinear drop functions, the packet dropping probability increases significantly when the queue length is close to the maximum value. Such an approach reduces the average queue size and decreases the transmission delays. The RED mechanism can also be replaced with any other controller. Hollot *et al*. (2001) describes a proportional-integral controller based on the low-frequency dynamics. Quet and Ozbay (2004) propose a robust controller, based on a known technique for H^∞ control of systems with time delays. The PI AQM controller by Hollot *et al*. (2001) is designed following the small-gain theorem. Easy implementation of PI AQM controllers in real networks results in a number of works (Michiels *et al*., 2006). The first application of the fractional-order PI controller as an AQM policy in a fluid flow model of a TCP connection is presented by Krajewski and Viaro (2014).

3. AQM mechanism based on the non-integer order PI$^\alpha$ controller

Fractional order derivatives and integrals (FODs/FOIs) generalize ordinary derivatives and integrals. Here a differintegral is a combined differential/integral operator. The most popular formulas for numerical calculation of differintegrals are the Grünwald–Letnikov and Riemann–Liouville formulas (Miller and Ross, 1993; Podlubny, 1999; Ciesielski and Leszczynski, 2002).

The α-differintegral of the function f, denoted by

$$\Delta^\alpha f, \tag{1}$$

is the fractional derivative (for $\alpha > 0$) or fractional integral (if $\alpha < 0$). If $\alpha = 0$, then the α-th differintegral of a function is the function itself.

The Grünwald–Letnikov definition of the n-th order continuous derivative is given by (Grünwald, 1867)

$$\frac{\mathrm{d}^n f}{\mathrm{d}t^n} = \lim_{h\to 0} \frac{1}{h^n} \sum_{r=0}^{n} (-1)^r \binom{n}{r} f(t-rh), \quad n = 1, 2, \ldots \tag{2}$$

The Grünwald–Letnikov definition of a derivative of fractional order α is given by

$$\frac{\mathrm{d}^\alpha f}{\mathrm{d}t^\alpha} = \lim_{h\to 0} \frac{1}{h^\alpha} \sum_{r=0}^{\infty} (-1)^r \binom{\alpha}{r} f(t-rh), \tag{3}$$

where $\binom{\alpha}{r}$ is the generalization of the binomial coefficient:

$$\binom{\alpha}{r} = \frac{\Gamma(\alpha+1)}{k!\Gamma(\alpha-k+1)} \tag{4}$$

and $\Gamma(x)$ is the gamma function:

$$\Gamma(x) = \lim_{n\to\infty} \frac{n! n^x}{x(x+1)(x+2)\ldots(x+n)}. \tag{5}$$

In active queue management, packet drop probabilities are determined at discrete moments of packet arrivals. There are a few definitions of the fractional discrete derivative (Abdeljawad *et al*., 2013; Abdeljawad, 2011). In this paper we use the Grünwald–Letnikov formula (Miller and Ross, 1993):

$$\triangle^\alpha f_k = \sum_{j=0}^{k} (-1)^j \binom{\alpha}{j} f_{k-j}, \tag{6}$$

where $\binom{\alpha}{j}$ is the generalization of the binomial coefficient:

$$\binom{\alpha}{j} = \begin{cases} 1 & \text{for } j = 0, \\ \frac{\alpha(\alpha-1)(\alpha-2)\ldots(\alpha-j+1)}{j!} & \text{for } j = 1, 2, \ldots, \end{cases} \tag{7}$$

For $\alpha = 1$ we obtain the formula for the difference of the first order (only two of the coefficients are non-zero),

$$\triangle^1 x_k = 1x_k - 1x_{k-1} + 0x_{k-2} + 0x_{k-3} \ldots. \tag{8}$$

For $\alpha = -1$ we obtain the sum of all the samples (the discrete integral of a first order equivalent).

$$\triangle^{-1} x_k = 1x_k + 1x_{k-1} + 1x_{k-2} + 1x_{k-3} \ldots \tag{9}$$

For a non-integer derivative and integral order, we obtain the weighted sum of all the samples, e.g.,

$$\triangle^{-1.2}x_k = 1x_k + 1.2x_{k-1} + 1.32x_{k-2} + 1.408x_{k-3}\ldots, \quad (10)$$

$$\triangle^{-0.8}x_k = 1x_k + 0.8x_{k-1} + 0.72x_{k-2} + 0.672x_{k-3}\ldots, \quad (11)$$

$$\triangle^{-0.4}x_k = 1x_k + 0.4x_{k-1} + 0.28x_{k-2} + 0.224x_{k-3}\ldots. \quad (12)$$

To determine the probability p_i of a packet loss at discrete time i, we calculate the response of PI$^\alpha$ given by

$$p_i = \min\{\{\max\{0, -(K_P e_i + K_I \Delta^\alpha e_i)\}, 1\}, \quad (13)$$

where K_P, K_I are tuning parameters, coefficients for the proportional and integral terms, respectively α is a non-integer integral order, e_i is the error in the current slot $e_i = q_i - q_0$, i.e., the difference between the current queue q_i and the intended queue q_0—this parameter denotes the queue size that should be maintained in the system. Of course, the size may temporarily be exceeded.

4. Fluid flow analysis of the AQM controller

The main goal of the fluid flow analysis presented below is modelling active queue management based on the response of the PI$^\alpha$ controller. The results display the influence of the parameters of the PI$^\alpha$ controller on the evolution of the TCP window. We compare these results with the TCP behaviour in the case of an FIFO queue and packet rejection due to exceeding the maximum queue size.

The model presented by Misra *et al.* (2000) shows the dynamics of the TCP protocol and allows obtaining the average values of key network performance parameters. The model is defined by the following differential equations:

$$W'(t) = \frac{1}{R(t)} - \frac{W(t)W(t-R(t))}{2R(t-R(t))}p(t-R(t)), \quad (14)$$

$$q'(t) = \frac{W(t)}{R(t)}N(t) - C, \quad (15)$$

where

- W is the expected TCP sending window size (packets),
- q is the expected queue length (packets),
- R is round-trip time $= q/C + T_p$ (sec),
- C is link capacity (packets/sec),
- T_p is propagation delay (sec),
- N is the number of TCP sessions (we consider the number of TCP streams passing through the communication node),
- p is packet drop probability.

The maximum values of q and W correspond to the buffer capacity and the maximum window size. The dropping probability p depends on the AQM algorithm.

The traffic composed of TCP and UDP streams has been considered by Czachórski *et al.* (2007). For this model, a single router supports N sessions, and each session is assumed to be either a TCP or a UDP session. Each TCP stream is a TCP-Reno connection (the most popular mechanism of network congestion control built in the TCP protocol), which is then extended to TCP/NCR(DCR) protocol for wired-wireless networks. Each UDP sender is a CBR (constant bit rate) (Domańska *et al.*, 2016) source.

In *passive* queue management (e.g., FIFO scheduling), packets coming to a buffer are rejected only if there is no space in the buffer to store them; the drop probability p takes values 0 (there is space in the buffer) or 1 (the buffer is full).

In the AQM mechanism based on the PI$^\alpha$ controller the drop probability p depends on the controller response.

The differential equations (14) and (15) are solved numerically. For numerical computations, we use software written in Python. In tests, we assume the following TCP connection parameters:

- transmission capacity of AQM router: $C = 0.075$,
- propagation delay for the i-th flow: $T_{p_i} = 2$,
- initial congestion window size for the i-th flow (measured in packets): $W_i = 1$,
- starting time for the i-th flow,
- the number of packets sent by the i-th flow.

The maximum queue size is set at a level of 30 packets, the PI$^\alpha$ controller setpoint is $q_0 = 10$ packets.

Domańska *et al.* (2017; 2016) discuss the influence of the controller's parameters on its behaviour, and it is suggested that an increase in the integral order (α) accelerates and strengthens the controller's response. Based on work of Domańska *et al.* (2019), three sets of the controller's parameters have been proposed. The experiment is repeated for an FIFO queue (CW1) and three example sets of controller tuning parameters: CW2 (a very strong controller), CW3 (a strong one), CW4 (a weak one); see Table 1. The controller with the same set of parameters is used in the open-loop scenario. Figure 2 presents the changes in the window size W. In the case of the FIFO queue (CW1), the TCP congestion window

Table 1. PI$^\alpha$ controller parameters.

	K_P	K_I	α
CW2	0.0001	0.0004	-1.2
CW3	0.0001	0.0014	-0.8
CW4	0.0001	0.005	-0.4

Table 2. Average queue length.

AQM	Avg. queue length
FIFO	18.1614
PI2	8.2467
PI3	8.4048
PI4	9.0096

Table 3. Estimated Hurst parameters obtained for sample fGn traces generated for the assumed Hurst parameter $H = 0.7$.

Trace number	Estimated Hurst parameter	Trace number	Estimated Hurst parameter
1	0.7279822	6	0.73197
2	0.7299411	7	0.7311628
3	0.7288566	8	0.7291909
4	0.7288566	9	0.7290085
5	0.7313482	10	0.7284157

increases until the maximum queue size is exceeded. In the case of AQM, we can influence the behaviour of the TCP stream by the choice of the PI$^\alpha$ parameters; see Table 2 for the obtained average queue lengths. The chosen set of PI$^\alpha$ parameters influences the average queue lengths and the number of dropped packets. An increase in the number of packets dropped by AQM should reduce the number of packets dropped by the FIFO. These phenomena affect the privileged position of the network traffic supported by FIFO. Figure 2 presents the behaviour of the PI$^\alpha$ controller in the case of a single TCP stream. This figure confirms that PI$^\alpha$ reduces the size of the TCP congestion window (CW2–CW4) and allows the evolution of the stream controlled by an FIFO queue (CW1).

The differences in TCP window evolutions (Fig. 2) and the obtained average queue lengths (Table 2) are not significant for the streams controlled by AQM. However, the experiment demonstrates that increasing the controller's integral order raises its aggressiveness. The average size of the queue decreases and speeds up the reaction to exceeding the setpoint. These differences will be shown more clearly in the next section.

5. AQM for privileged traffic flows under LRD traffic: The open loop discrete event simulation model

In this part an assessment of the proposed mechanism is presented. We examine how the LRD of the network traffic affects the obtained results.

There are some misunderstandings in the literature between the terms of long-range dependence and self-similarity. In network traffic modelling, we take into account time series. The aggregated sequence of the process $X(t)$ representing incremental process (e.g., in seconds) can be expressed as (Willinger *et al.*, 1994; Czachórski *et al.*, 2015)

$$X^{(m)}(k) = \frac{1}{m}\sum_{i=1}^{m} X((k-1)m+i), \quad k = 1, 2, \ldots, \tag{16}$$

where m is the level of aggregation. The process X is second-order self-similar if for all m the condition that the process $m^{1-H}X^{(m)}$ has the same variance and auto-correlation as the process X evolves. The process X is asymptotically second-order self-similar if the above condition is fulfilled as $m \to \infty$. The asymptotically second-order self-similar process is also called an LRD process (Gong *et al.*, 2005). H is the well-known Hurst parameter (Mandelbrot and Ness, 1968). It measures the intensity of long-range dependence (Taqqu and Teverovsky, 1998). The LRD corresponds to $1/2 < H < 1$ and short-range dependence corresponds to $H = 1/2$ (Abry and Veitch, 1998). Many researchers confirm that network traffic is long-range dependent.

During our experiments, the intensity of the LRD of the generated traffic is changed. The reason for this assumption is, first, the aforementioned scarcity of existing studies describing the characteristics of traffic generated by IoT devices and, secondly, the supposition that IoT devices can be connected to the Internet. Hence the traffic forwarded is a mixed one generated by normal network devices and IoT devices.

We use fractional Gaussian noise (fGn) as an example of an ideal traffic source model that displays LRD (Taqqu and Teverovsky, 1998). Nowadays, fGn is one of the most commonly used LRD source models in network performance evaluation. The fGn process is a zero mean, stationary Gaussian one. The autocovariance of the fGn process satisfies (Taqqu and Teverovsky, 1998)

$$\rho(i) = EX_jX_{j+i} = \frac{1}{2}|i+1|^{2H} + \frac{1}{2}|i-1|^{2H} - |i|^{2H}, \tag{17}$$

where $i \geq 0$ is the lag and X_j (for $j > 1$) denotes a series of observations. The synthetic generation of sample paths

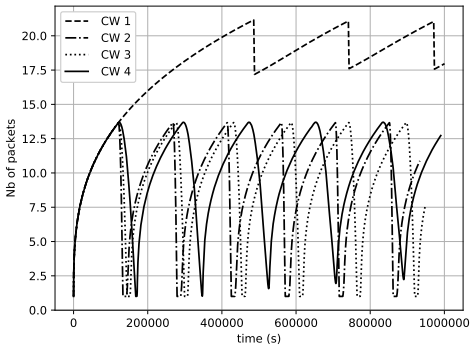


Fig. 2. TCP congestion window evolution.

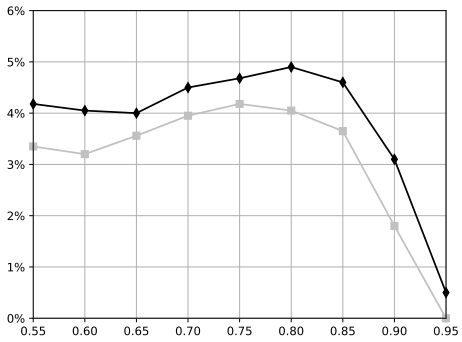
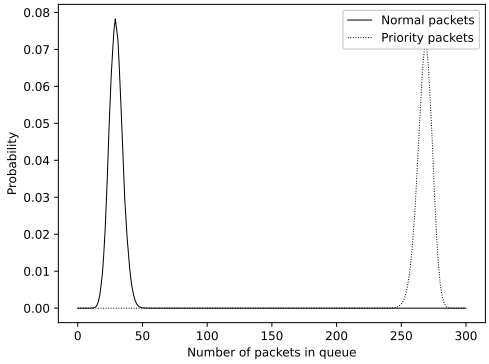


Fig. 3. Maximum and minimum differences between the assumed and estimated Hurst parameters.



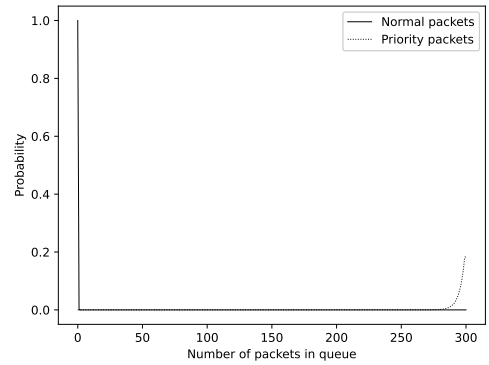


Fig. 4. Distribution of the queue length for high traffic load ($\mu = 0.2$) and $H = 0.5$, priority traffic 50%, $K_P = 0.0001$, $K_I = 0.0004$, $\alpha = -0.4$ (top), $K_P = 0.0001$, $K_I = 0.054$, $\alpha = -1.2$ (bottom).

(traces) of LRD processes is an important problem. In this paper, we use a fast algorithm for generating approximate sample paths for an fGn process, first introduced by (Paxson, 1997).

We have generated sample traces with the Hurst parameter with the range from 0.5 to 0.9. After each trace generation, the Hurst parameter is estimated with the use of the aggregated variance method. Domański *et al.* (2016) present the accuracy of the Hurst parameter. The fGn generator normally generates traffic with a slightly higher Hurst parameter (see Table 3). The difference between the assumed Hurst parameter and its estimate decreases for higher values of H; see Fig. 3. For our purposes, samples displaying the smallest difference possible have been chosen.

The simulation model of an appropriate AQM mechanism is prepared with the use of SimPy. SimPy is a process-based discrete-event simulation framework using the Python language. Its event dispatcher is based on Python's generators. SimPy is released under the MIT license (free software license originating at the Massachusetts Institute of Technology).

We investigate the influence of PI$^{\alpha}$ controller parameters on the performance of the proposed solution. We also study the impact of the degree of LRD on an examined mechanism. Open-loop simulations demonstrate the abilities of the proposed mechanism. The evaluation of the impact of the real Internet traffic on the performance of the proposed congestion control mechanism requires a large number of TCP streams. Such simulations would be extremely complex computationally. We replace a large number of TCP sources with one LRD source. Such a source reflects Internet traffic well enough. We call this model an open-loop (without loopback) one because the behaviour of the AQM mechanism does not directly affect the source (as in the case of TCP and conducted fluid-flow approximations). The intensity of the input traffic is chosen as $\lambda = 0.5$, independently of the Hurst parameter. The distribution of the service time is geometric. This service time represents the time of packet processing and dispatching. The parameter of the service time (μ) is changed during the test with the range of 0.2 to 0.8 with step 0.1. The highest traffic load is considered for parameter $\mu = 0.20$. The average traffic load is obtained for $\mu = 0.5$. A small network traffic is considered for

Table 4. Results for privilege and normal traffic, $K_P = 0.0001$, $K_I = 0.005$, $\alpha = -0.4$, $\mu = 0.5$, $H = 0.5$.

Priority packets [%]	Avg. queue length	Priority packet loss	Avg. priority packets waiting time	Normal packet loss	Avg. normal packets waiting time
90	78.89	0	15.94	26651	15.09
80	68.96	0	13.94	27835	13.57
70	65.25	0	13.21	30626	12.94
60	63.3	0	12.82	31565	12.61
50	61.62	0	12.48	31657	12.32
40	60.52	0	12.26	31310	12.13
30	59.1	0	11.97	31835	11.87
20	59.09	0	11.98	32533	11.87
10	59.51	0	12.06	33493	11.97

Table 5. Results for privilege and normal traffic, $K_P = 0.0001$, $K_I = 0.0014$, $\alpha = -0.8$, $\mu = 0.5$, $H = 0.5$.

Priority packets [%]	Avg. queue length	Priority packet loss	Avg. priority packets waiting time	Normal packet loss	Avg. normal packets waiting time
90	54.7	0	11.09	34823	10.3
80	50.86	0	10.32	36479	9.92
70	51.5	0	10.46	38802	10.19
60	50.8	0	10.33	40704	10.12
50	50.32	0	10.23	41027	10.07
40	49.43	0	10.05	39033	9.91
30	50.33	0	10.22	38740	10.11
20	49.7	0	10.1	40748	10.0
10	49.59	0	10.08	38319	9.99

Table 6. Results for privileged and normal traffic, $K_P = 0.0001$, $K_I = 0.0004$, $\alpha = -1.2$, $\mu = 0.5$, $H = 0.5$.

Priority packets [%]	Avg. queue length	Priority packet loss	Avg. priority packets waiting time	Normal packet loss	Avg. normal packets waiting time
90	52.56	0	10.67	38284	9.82
80	48.9	0	9.94	41007	9.54
70	46.75	0	9.51	43726	9.25
60	46.18	0	9.39	46527	9.21
50	44.16	0	8.98	45826	8.84
40	43.69	0	8.9	48887	8.78
30	43.27	0	8.82	48719	8.7
20	42.31	0	8.61	48372	8.53
10	42.49	0	8.65	49229	8.57

Table 7. Results for privileged and normal traffic, $K_P = 0.0001$, $K_I = 0.005$, $\alpha = -0.4$, $\mu = 0.3$, $H = 0.5$.

Priority packets [%]	Avg. queue length	Priority packet loss	Avg. priority packets waiting time	Normal packet loss	Avg. normal packets waiting time
90	298.23	1535285	99.45	465751	99.36
80	297.55	1077587	99.1	919513	98.98
70	296.1	642129	98.65	1355972	98.48
60	291.55	251945	97.19	1746799	96.72
50	274.9	25308	91.87	1975415	90.68
40	247.68	118	82.87	2000220	81.99
30	226.54	0	75.79	1999456	75.21
20	210.72	0	70.49	1998860	70.08
10	198.26	0	66.29	1996555	65.95

Table 8. Results for privileged and normal traffic, $K_P = 0.0001$, $K_I = 0.0004$, $\alpha = -1.2$, $\mu = 0.3$, $H = 0.5$.

Priority packets [%]	Avg. queue length	Priority packet loss	Avg. priority packets waiting time	Normal packet loss	Avg. normal packets waiting time
90	298.17	1497660	99.33	500386	33.8
80	297.29	1000816	99.1	999125	28.83
70	294.69	499998	98.24	1500041	23.18
60	206.7	14485	69.05	1985906	37.82
50	118.32	0	39.72	1999188	38.0
40	113.65	0	38.15	1998948	37.31
30	107.49	0	36.11	1999407	35.54
20	99.96	0	33.63	2003010	33.21
10	92.07	0	30.97	1999846	30.63

Table 9. Results for privileged and normal traffic, $K_P = 0.0001$, $K_I = 0.005$, $\alpha = -0.4$, $\mu = 0.5$, $H = 0.7$.

Priority packets [%]	Avg. queue length	Priority packet loss	Avg. priority packets waiting time	Normal packet loss	Avg. normal packets waiting time
90	119.47	55475	25.36	127615	18.14
80	100.82	14380	21.8	182473	17.04
70	89.73	2486	19.6	206357	16.38
60	81.8	365	17.94	214620	15.67
50	77.26	0	16.99	221871	15.27
40	73.34	0	16.14	224000	14.79
30	70.73	0	15.59	227465	14.47
20	68.52	0	15.13	229447	14.16
10	66.76	0	14.74	230807	13.91

parameter $\mu = 0.80$. During the test, we also change the intensity of the priority flow. This option is implemented by changing the ratio of priority packages to normal ones. We do so with a range of 10 to 90%.

During the tests, we analyzed the following parameters of the two traffic streams (privileged and normal): the length of the queue, queue waiting times and the number of rejected packets.

The tests are carried out for three different PI^α controller parameters. The controller parameters are presented in Section 4 (Table 1).

We carry out almost a thousand simulations, performed in parallel on a 64-core computer using the MPI for Python package mpi4py. Further on, the most interesting results are shown.

For the parameter μ greater than 0.5, the simulations concerned a lightly loaded system. The average queue lengths are small. The queue lengths regardless of the μ and Hurst parameters do not exceed 100 packets, and the packet losses are not noticeable. These results prove that, for an unloaded system, additional data transfer control mechanisms may not be needed.

The importance of the proposed mechanism grows with the increase of the buffer load. The first interesting results have been obtained for the μ parameter equal to 0.5. These results are presented in Tables 4–6. In the case of an average loaded system, there are no losses in the priority queue, regardless of the controller's parameters. However, for such sets of parameters, the queue exceeds the desired length and some packets from the normal stream are dropped.

As described earlier, the aggressiveness of the mechanism increases with the integration order of the controller. The number of dropped packets is largest for the controller with the integration order of 1.2 (Table 6). With an increase in the dropped packets in the regular queue, the waiting times in the priority queue are decreased.

In the next stages of the experiment, it is shown that the lack of losses and acceptable transmission delays in the privileged stream are possible to be maintained in a more loaded system as well. However, when the network is overloaded, the importance of the controller increases. In Table 7 results for $\mu = 0.3$, the Hurst parameter $H = 0.5$ and a controller CW4 are presented. This controller turns out to be too weak. For this controller, there will be no losses in the priority queue if the priority traffic does not exceed 30 percent of the whole traffic. When the desired queue length is exceeded, the reaction of the controller is insufficient, which causes such results to occur.

Table 8 shows the results for the same traffic ($\mu = 0.3$, $H = 0.5$) and the most aggressive controller (CW2). This mechanism enables no losses in the priority queue if the priority traffic does not exceed 50 percent of the

Table 10. Results for privileged and normal traffic, $K_P = 0.0001$, $K_I = 0.0014$, $\alpha = -0.8$, $\mu = 0.5$, $H = 0.7$.

Priority packets [%]	Avg. queue length	Priority packet loss	Avg. priority packets waiting time	Normal packet loss	Avg. normal packets waiting time
90	88.45	26094	19.18	195555	8.66
80	62.02	1763	13.8	247039	9.12
70	53.32	0	11.91	260362	9.39
60	50.22	0	11.21	265308	9.56
50	48.58	0	10.86	269166	9.6
40	47.48	0	10.62	272179	9.62
30	46.7	0	10.44	274334	9.62
20	46.49	0	10.39	275048	9.69
10	45.84	0	10.24	274040	9.63

Table 11. Results for privileged and normal traffic, $K_P = 0.0001$, $K_I = 0.0004$, $\alpha = -1.2$, $\mu = 0.5$, $H = 0.7$.

Priority packets [%]	Avg. queue length	Priority packet loss	Avg. priority packets waiting time	Normal packet loss	Avg. normal packets waiting time
90	87.44	26584	18.98	199828	8.51
80	60.61	1995	13.5	255293	8.9
70	51.18	0	11.47	271588	8.99
60	47.12	0	10.59	283824	8.95
50	44.66	0	10.05	294405	8.86
40	43.2	0	9.73	306666	8.81
30	41.3	0	9.31	315980	8.58
20	40.59	0	9.16	325563	8.55
10	39.71	0	8.95	339194	8.47

Table 12. Results for privileged and normal traffic, $K_P = 0.0001$, $K_I = 0.005$, $\alpha = -0.4$, $\mu = 0.5$, $H = 0.9$.

Priority packets [%]	Avg. queue length	Priority packet loss	Avg. priority packets waiting time	Normal packet loss	Avg. normal packets waiting time
90	126.51	787993	32.94	258152	17.55
80	121.9	569406	32.91	493763	17.84
70	117.03	373017	32.79	701160	18.12
60	112.23	204259	32.75	882029	18.5
50	107.07	71852	32.57	1026887	18.99
40	100.35	3655	31.36	1103586	19.91
30	93.15	0	29.18	1115966	20.7
20	87.57	0	27.42	1124161	20.92
10	83.16	0	26.08	1130126	20.81

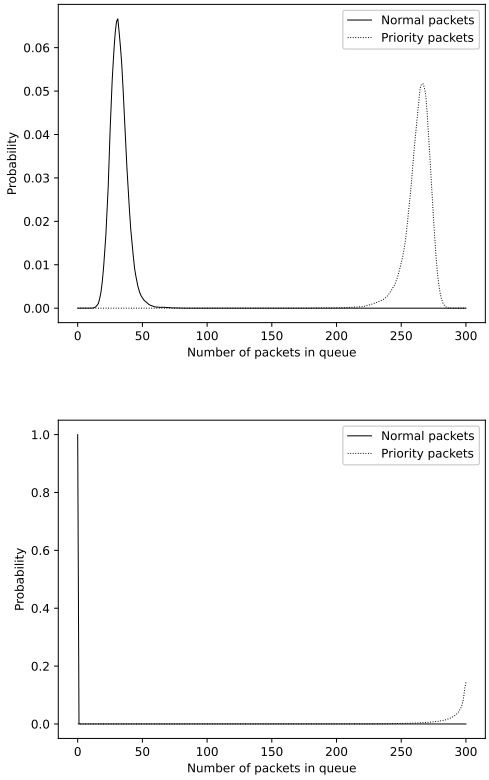


Fig. 5. Distribution of the queue length for high traffic load ($\mu = 0.2$) and $H = 0.7$, priority traffic 50%, $K_P = 0.0001$, $K_I = 0.0004$, $\alpha = -0.4$ (top), $K_P = 0.0001$, $K_I = 0.054$, $\alpha = -1.2$ (bottom).

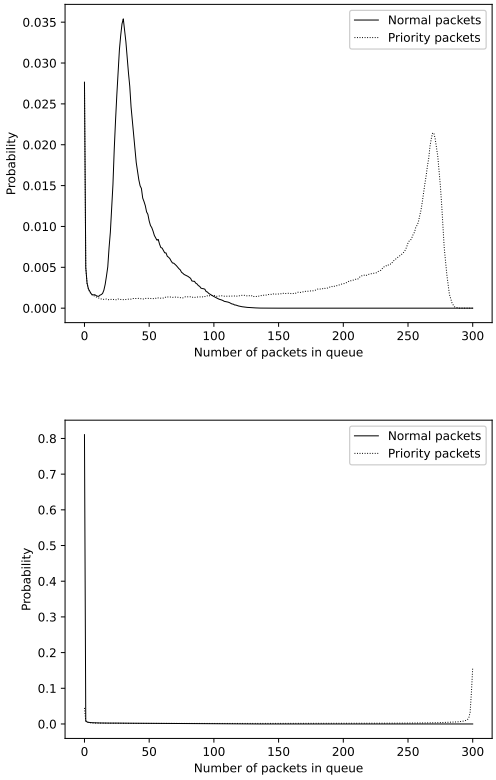


Fig. 6. Distribution of the queue length for high traffic load ($\mu = 0.2$) and $H = 0.9$, priority traffic 50%, $K_P = 0.0001$, $K_I = 0.0004$, $\alpha = -0.4$ (top), $K_P = 0.0001$, $K_I = 0.054$, $\alpha = -1.2$ (bottom).

whole traffic. When the same priority traffic exceeds the maximum gateway capacity, packets from the privileged stream might be lost. For $\mu = 0.2$ we do not present detailed results. In this case, we have a very heavily loaded system. For such traffic over 50 percent of packets are dropped. The first controller keeps no losses in priority traffic if it does not exceed 10 percent. Under the same conditions, the third controller counteracts losses if the priority traffic cannot exceed 30 percent. Minimization of losses in the priority queue is carried out by non-priority traffic starving. This situation is presented in Fig. 4. The figure presents the queue distributions in the case of equal proportion of priority and normal traffic. As can be seen, in the case of a very strong controller, all non-priority packets are lost.

The Hurst parameter $H = 0.5$ assumes no long-term dependencies in network traffic. A stormy characteristic of network traffic causes extreme fluctuations in queue occupancy. An increase in Hurst parameter enhances the number of lost packets and decreases the average queue length (Domańska *et al*., 2018).

Here, similar correlations are noticed. In the case of an average queue load ($\mu = 0.5$) and traffic without LRD ($H = 0.5$), losses in priority traffic do not occur (Tables 4–6). When $H = 0.7$, although the average numbers of incoming and outgoing packets are identical to each other, packets' losses appear. Tables 9–11 present the detailed results. The losses depend on the employed controller and the percentage of priority traffic.

In the best case, priority traffic cannot exceed 80 percent (see Tables 10 and 11). As can be noticed, the behaviours of the two controllers (CW2 and CW3) are very similar to each other. Domańska *et al*. (2016) explain this phenomenon. In the case of big traffic fluctuations, a delayed, weaker controller response may be beneficial.

In the case of traffic with a high degree of LRD (Hurst parameter $H = 0.9$) the priority traffic cannot exceed 40 percent (Table 14). When the CW4 controller is employed, it cannot exceed 30 percent (Tables 12). Given $H = 0.7$, the results obtained for the CW2 and CW3 controllers are similar (Tables 13 and 14).

The queue distributions (Figs. 5 and 6) show that losses in priority traffic are caused by exceeding the maximum node efficiency. In a critical load situation,

Table 13. Results for privileged and normal traffic, $K_P = 0.0001$, $K_I = 0.0014$, $\alpha = -0.8$, $\mu = 0.5$, $H = 0.9$.

Priority packets [%]	Avg. queue length	Priority packet loss	Avg. priority packets waiting time	Normal packet loss	Avg. normal packets waiting time
90	121.04	745453	31.91	329684	7.27
80	110.06	487667	30.55	627070	7.49
70	97.77	268818	28.78	883942	7.78
60	83.94	99407	26.32	1085505	8.25
50	59.54	384	19.25	1209211	8.84
40	48.8	0	15.34	1216461	10.16
30	47.39	0	14.95	1223627	10.97
20	46.12	0	14.58	1230641	11.39
10	44.94	0	14.19	1234818	11.59

Table 14. Results for privileged and normal traffic $K_P = 0.0001$, $K_I = 0.0004$, $\alpha = -1.2$, $\mu = 0.5$, $H = 0.9$.

Priority packets [%]	Avg. queue length	Priority packet loss	Avg. priority packets waiting time	Normal packet loss	Avg. normal packets waiting time
90	120.98	744590	31.88	332326	7.58
80	110.02	491038	30.53	630641	7.83
70	98.13	271577	28.86	891894	8.14
60	84.56	101826	26.53	1101742	8.51
50	59.27	86	19.11	1237992	9.1
40	49.62	0	15.78	1252821	10.34
30	47.61	0	15.21	1271436	11.13
20	45.72	0	14.61	1288994	11.48
10	43.62	0	13.88	1305141	11.48

all packets from non-priority traffic are dropped from the queue. This situation is clearly visible for the third, most powerful controller (CW2). In addition, it can be seen that, for that controller, the shape of the queue distribution is the same independently of the degree of traffic LRD (expressed in the Hurst parameter). This allows us to conclude that the proposed parameters of the controller are optimal regardless of long-term dependencies of network traffic.

6. Conclusion

This article proposed a mechanism allowing one to provide a certain level of QoS. This mechanism is based on determining the priority of packets and active queue management. The implementation of this mechanism in the gateway overcomes the limitations related to the lack of resources of IoT devices. The function implemented in IoT devices is only marking in the header of IP the type of the packet. In the article, we considered the behaviour of the gateway that supports two types of traffic: normal and priority. The queue is controlled by the AQM mechanism, but only packets from non-priority traffic are preventively dropped from the queue by AQM. Priority packets are dropped from the queue if the maximum queue size is exceeded.

Our AQM mechanism is based on the PI$^\alpha$ controller. Its quality highly depends on the proper selection of parameters. We tested three sets of controller parameters. The selected controllers have the same proportional term and differ by the integral term and the integral non-integer fractional order.

The proposed mechanism was investigated with the use of two methods. The behaviour of the PI$^\alpha$ controller and FIFO scheduling was shown using fluid flow approximation (closed-loop control). The operation of a complete mechanism was evaluated using simulation tests (an open loop scenario).

The fluid flow analysis showed the influence of the AQM mechanisms on a single TCP stream. It displayed the importance of controller parameters on the AQM behaviour and the TCP window evolution.

Our article presented the impact of the traffic intensity, the number of priority packets and the intensity of LRD (expressed in the Hurst parameter) on the length of the queue, queue waiting times and the number of rejected packets for both data streams considered. The results prove the usefulness of the PI$^\alpha$ controller for this application. The obtained results show that, if the priority traffic does not exceed the maximum gateway capabilities, it is possible to ensure no losses for priority packets.

Proper selection of the controller parameters is important in adaptation to various types of traffic. For a heavily loaded system, no loss of priority packets results in very aggressive rejection of normal packets. This feature can be very useful in the case of repelling DDoS attacks. Such functionality is performed by the third controller (see Table 1). In the case of a light load, better behaviour is displayed by the first controller, giving a smaller number of dropped packets. In our future work, we will focus on the implementation of the proposed mechanism in a real router. It will show the behaviour of the proposed mechanism in real Internet conditions.

Acknowledgment

This research was financed by the National Science Center under the grant no. 2017/27/B/ST6/00145.

References

Abdeljawad, T. (2011). On Riemann and Caputo fractional differences, *Computers and Mathematics with Applications* **62**(3): 1602–1611.

Abdeljawad, T., Baleanu, D., Jarad, F. and Agarwal, R.P. (2013). Fractional sums and differences with binomial coefficients, *Discrete Dynamics in Nature and Society* **2013**, Article ID: 104173.

Abry, P. and Veitch, D. (1998). Wavelet analysis of long-range-dependent traffic, *IEEE Transactions on Information Theory* **44**(1): 2–15.

Adjih, C., Baccelli, E., Fleury, E., Harter, G., Mitton, N., Noel, T., Pissard-Gibollet, R., Saint-Marcel, F., Schreiner, G., Vandaele, J. and Watteyne, T. (2015). FIT IoT-LAB: A large scale open experimental IoT testbed, *IEEE 2nd World Forum on Internet of Things (WF-IoT), Milan, Italy*, pp. 459–464.

Al-Kashoash, H.A.A., Amer, H.M., Mihaylova, L. and Kemp, A.H. (2017). Optimization-based hybrid congestion alleviation for 6LoWPAN networks, *IEEE Internet of Things Journal* **4**(6): 2070–2081.

Berte, D.-R. (2018). Defining the IoT, *Proceedings of the International Conference on Business Excellence, Bucharest, Romania*, Vol. 12, pp. 118–128.

Bingi, K., Ibrahim, R., Karsiti, M.N., Hassam, S.M. and Harindran, V.R. (2019). Frequency response based curve fitting approximation of fractional-order PID controllers, *International Journal of Applied Mathematics and Computer Science* **29**(2): 311–326, DOI: 10.2478/amcs-2019-0023.

Brun, O., Yin, Y. and Gelenbe, E. (2018). Deep learning with dense random neural networks for detecting attacks against IoT-connected home environments, *Procedia Computer Science* **134**: 458–463.

Chou, J.-J., Shih, C.-S., Wang, W.-D. and Huang, K.-C. (2019). IoT sensing networks for gait velocity measurement, *International Journal of Applied Mathematics and Computer Science* **29**(2): 245–259, DOI: 10.2478/amcs-2019-0018.

Chandrashekhara, G.B. and Veena, N.K. (2018). Routing in Internet of Things: Review, *Proceedings of the 2nd International Conference on Intelligent Computing and Control Systems, Madurai, India*, pp. 790–795.

Ciesielski, M. and Leszczynski, J. (2002). A numerical method for solution of ordinary differential equations of fractional order, *in* R. Wyrzykowski *et al.* (Eds), *Parallel Processing and Applied Mathematics,* Springer, Berlin/Heidelberg, pp. 695–702.

Cucinotta, T., Palopoli, L., Abeni, L., Faggioli, D. and Lipari, G. (2010). On the integration of application level and resource level QoS control for real-time applications, *IEEE Transaction on Industrial Informatics* **6**(4): 479–491.

Czachórski, T., Domańska, J. and Pagano, M. (2015). On stochastic models of internet traffic, *Communications in Computer and Information Science* **564**: 289–303.

Czachórski, T., Grochla, K. and Pekergin, F. (2007). Stability and dynamics of TCP-NCR(DCR) protocol in presence of UDP flows, *in* J. García-Vidal and L. Cerdà-Alabern (Eds), *Wireless Systems and Mobility in Next Generation Internet*, Springer, Berlin/Heidelberg, pp. 241–254.

Domańska, J., Augustyn, D.R. and Domański, A. (2012). The choice of optimal 3-rd order polynomial packet dropping function for NLRED in the presence of self-similar traffic, *Bulletin of the Polish Academy of Sciences: Technical Sciences* **60**(4): 779–786.

Domańska, J., Domański, A., Augustyn, D.R. and Klamka, J. (2014). A RED modified weighted moving average for soft real-time application, *International Journal of Applied Mathematics and Computer Science* **24**(3): 697–707, DOI: 10.2478/amcs-2014-0051.

Domańska, J., Domański, A., Czachórski, T. and Klamka, J. (2016). The use of a non-integer order PI controller with an active queue management mechanism, *International Journal of Applied Mathematics and Computer Science* **26**(4): 777–789, DOI: 10.1515/amcs-2016-0055.

Domańska, J., Domański, A., Czachórski, T. and Klamka, J. (2017). Self-similarity traffic and AQM mechanism based on non-integer order $PI^{\alpha}D^{\beta}$ controller, *in* P. Gaj *et al.* (Eds), *Communications in Computer and Information Science*, Springer International Publishing, Cham, pp. 336–350.

Domańska, J., Domański, A., Czachórski, T., Klamka, J., Marek, D. and Szyguła, J. (2018). The influence of the traffic self-similarity on the choice of the non-integer order PI^{α} controller parameters, *in* T. Czachórski *et al.* (Eds), *Communications in Computer and Information Science*, Springer International Publishing, Cham, pp. 76–83.

Domańska, J., Domański, A., Czachórski, T., Klamka, J., Szyguła, J. and Marek, D. (2019). AQM mechanism with the dropping packet function based on the answer of several PI^{α} controllers, *in* P. Gaj *et al.* (Eds), *Communications in Computer and Information Science*, Springer Verlag, Berlin, pp. 400–412.

Domański, A., Domańska, J. and Czachórski, T. (2016). The impact of the degree of self-similarity on the NLREDwM mechanism with drop from front strategy, *in* P. Gaj *et al.* (Eds), *Computer Networks*, Springer International Publishing, Cham, pp. 192–203.

Dymora, P. and Mazurek, M. (2019). Anomaly detection in IoT communication network based on spectral analysis and Hurst exponent, *Applied Sciences* **9**(24): 1–20.

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

Gong, W.-B., Liu, Y., Misra, V. and Towsley, D. (2005). Self-similarity and long range dependence on the internet: A second look at the evidence, origins and implications, *Computer Networks* **48**(3): 377–399.

Gubbi, J., Buyya, R., Marusic, S. and Palaniswami, M. (2012). Internet of Things (IoT): A vision, architectural elements, and future directions, *Future Generation Computer Systems* **29**(7): 1645–1660.

Grünwald, A.K. (1867). Uber "begrenzte" Derivationen und deren Anwendung, *Zeitschrift für Angewandte Mathematik und Physik*.

Halim, N.H.B., Yaakob, N.B. and Isa, A.B.A.M. (2016). Congestion control mechanism for Internet-of-Things (IOT) paradigm, *3rd International Conference on Electronic Design, Phuket, Thailand*, pp. 337–341.

Hollot, C., Misra, V., Towsley, D. and Gong, W. (2001). On designing improved controllers for AQM routers supporting TCP flows, *20th Annual Joint Conference of the IEEE Computer and Communications Society (INFOCOM 2001), Anchorage, USA,* Vol. 3, pp. 1726–1734.

Hsu, W., Li, Q., Han, X. and Huang, C. (2017). A hybrid IoT traffic generator for mobile network performance assessment, *International Wireless Communications and Mobile Computing Conference (IWCMC), Valencia, Spain,* pp. 441–445.

Jiang, T., Ammar, S.I., Chang, B. and Liu, L. (2018). Analysis of an N-policy GI/M/1 queue in a multi-phase service environment with disasters, *International Journal of Applied Mathematics and Computer Science* **28**(2): 375–386, DOI: 10.2478/amcs-2018-0028.

Kittipipattanathaworn, P. and Nupairoj, N. (2014). SLA guarantee real-time monitoring system with soft deadline constraint, *2014 11th International Joint Conference on Computer Science and Software Engineering (JCSSE), Chon Buri, Thailand,* pp. 52–57.

Krajewski, W. and Viaro, U. (2014). On robust fractional order PI controller for TCP packet flow, *BOS Conference: Systems and Operational Research, Warsaw, Poland*, pp. 493–505.

Mandelbrot, B. and Ness, J. (1968). Fractional Brownian motions, fractional noises and applications, *SIAM Review* **10**(4): 422–437.

Miao, D., Liu, L., Xu, R., Panneerselvam, J., Wu, Y. and Xu, W. (2018). An efficient indexing model for the fog layer of industrial internet of things, *IEEE Transactions on Industrial Informatics* **14**(10): pp. 4487–4496.

Michiels, W., Melchor-Aquilar, D. and Niculescu, S. (2006). Stability analysis of some classes of TCP/AQM networks, *International Journal of Control* **79**(9): 1136–1144.

Miller, K. and Ross, B. (1993). *An Introduction to the Fractional Calculus and Fractional Differential Equations*, Wiley, New York.

Misra, V., Gong, W. and Towsley, D. (2000). Fluid-based analysis of network of AQM routers supporting TCP flows with an application to RED, *Proceedings of the Conference on Applications, Technologies, Architectures and Protocols for Computer Communication, Stockholm, Sweden,* pp. 151–160.

Palattella, M.R., Dohler, M., Grieco, A., Rizzo, G., Torsner, J., Engel, T. and Ladid, L. (2016). Internet of things in the 5G era: Enablers, architecture, and business models, *IEEE Journal on Selected Areas in Communications* **34**(3): 510–527.

Palopoli, L., Abeni, L., Buttazzo, G., Conticelli, F. and Di Natale, M. (2000). Real-time control system analysis: An integrated approach, *21st IEEE Real-Time Systems Symposium, Orlando, USA*, pp. 131–140.

Paxson, V. (1997). Fast, approximate synthesis of fractional Gaussian noise for generating self-similar network traffic, *ACM SIGCOMM Computer Communication Review* **27**(5): 5–18.

Podlubny, I. (1999). *Fractional Differential Equations*, Academic Press, San Diego.

Quet, P. and Ozbay, H. (2004). On the design of AQM supporting TCP flows using robust control theory, *IEEE Transactions on Automatic Control* **49**(6): 1031–1036.

Shafiq, M.Z., Ji, L., Liu, A.X., Pang, J. and Wang, J. (2013). Large-scale measurement and characterization of cellular machine-to-machine traffic, *IEEE/ACM Transactions on Networking* **21**(6): 1960–1973.

Skeie, T., Johannessen, S. and Holmeide, O. (2006). Timeliness of real-time IP communication in switched industrial ethernet networks, *IEEE Transaction on Industrial Informatics* **2**(1): 25–39.

Stankovic, J. and Rajkumar, R. (2004). Real-time operating systems, *Real-Time Systems Journal* **28**(2–3): 237–253.

Stoica, I., Morris, R., Karger, D., Frans Kaashoek, M. and Balakrishnan, H. (2001). Chord: A scalable peer-to-peer lookup service for internet applications, *Proceedings of the 2001 Conference on Applications, Technologies, Architectures and Protocols for Computer Communications, San Diego, USA,* Vol. 31, pp. 149–160.

Taqqu, M. and Teverovsky, V. (1998). On estimating the intensity of long-range dependence in finite ANF infinite variance time series, *in* R. Adler *et al.* (Eds.), *A Practical Guide To Heavy Tails: Statistical Techniques and Applications*, Birkhauser, Boston, pp. 177–217.

Wang, B. (2009). *Priority and Real Time Data Transfer over the Best-Effort Internet*, VDM Verlag, Saarbrücken.

Wijnants, M. and Lamotte, W. (2008). Managing client bandwidth in the presence of both real-time and non real-time network traffic, *Proceedings of the 3rd International Conference on Communication Systems Software and Middleware (COMSWARE 2008), Bangalore, India*, pp. 442–450.

Willinger, W., Leland, W. and Taqqu, M. (1994). On the self-similar nature of traffic, *IEEE/ACM Transactions on Networking* **2**(1): 1–15.

Zheng, B. and Atiquzzaman, M. (2000). DSRED: A new queue management scheme for next generation networks, *25th Annual IEEE Conference on Local Computer Networks, Tampa, USA,* pp. 242–251.

Adam Domański, Ph.D., Eng., a professor of the Silesian University of Technology, works in the Department of Distributed Systems and Informatic Devices, Faculty of Automatic Control, Electronics and Computer Science, Silesian University of Technology. His main research interest regarding the computer networks domain is congestion control in packet networks.

Joanna Domańska, Ph.D., Eng., an institute professor, has been a member of the Computer Systems Modelling and Performance Evaluation Group of the Institute of Theoretical and Applied Informatics, Polish Academy of Sciences, Gliwice, Poland, since 1994, where she has been serving as the leader since 2018. Her main areas of research interest include performance modeling methods for computer networks, particularly the modeling of network traffic intensity and the quality of service (QoS) related problems.

Tadeusz Czachórski, PhD, Prof., is the head of the Institute of Theoretical and Applied Informatics of the Polish Academy of Sciences. His main areas of interest are mathematical and numerical methods for modelling and evaluation of computer networks.

Jerzy Klamka, Ph.D., Prof., a full member of the Polish Academy of Sciences, works in the Institute of Theoretical and Applied Informatics of the Polish Academy of Sciences. His main areas of research are controllability and observability of linear and nonlinear dynamical systems, and mathematical foundations of quantum computations. He is an author of monographs and numerous papers published in international journals.

Jakub Szyguła is a PhD student in the Department of Distributed Systems and Informatic Devices, Faculty of Automatic Control, Electronics and Computer Science at the Silesian University of Technology. His main research interests are related to algorithms of active queue management in computer networks.

Dariusz Marek is a PhD student in the Department of Distributed Systems and Informatic Devices, Faculty of Automatic Control, Electronics and Computer Science at the Silesian University of Technology. His main research interests are related to algorithms of active queue management in computer networks.

Received: 3 January 2020
Revised: 19 May 2020
Re-revised: 13 October 2020
Accepted: 20 October 2020