Politechnika Śląska
Wydział Automatyki, Elektroniki
i Informatyki

mgr inż. Jakub Szyguła

ADAPTACYJNE ALGORYTMY ZARZĄDZANIA NATĘŻENIEM RUCHU W SIECI INTERNET

Rozprawa doktorska napisana pod kierunkiem
dr hab. inż. Adama Domańskiego, Prof. PŚ

Gliwice 2022

Spis treści

Lista publikacji zawartych w niniejszej rozprawie doktorskiej

1. Domański, A.; Domańska, J.; Czachórski, T.; Klamka, J.; Marek, D.; Szyguła, J. AQM mechanism with the dropping packet function based on the answer of several PI^{α} controllers. 26th International Conference on Computer Networks (CN 2019), Communications in Computer and Information Science, Springer International Publishing, vol. 1039, s. 400-412, 2019. https://doi.org/10.1007/978-3-030-21952-9_29.

2. Szyguła, J.; Domański, A.; Domańska, J.; Czachórski, T.; Marek, D.; Klamka, J. AQM mechanism with neuron tuning parameters. 12th Asian Conference on Intelligent Information and Database Systems (ACIIDS 2020). Lecture Notes in Artificial Intelligence, vol. 12034, s. 299-311. Springer, Cham. https://doi.org/10.1007/978-3-030-42058-1_25.

3. Marek, D.; Szyguła, J; Domański, A.; Domańska, J.; Filus, K.; Szczygieł, M. Adaptive Hurst-Sensitive Active Queue Management. Entropy 2022, vol. 24(3), 418. https://doi.org/10.3390/e24030418.

4. Szyguła, J.; Domański, A.; Domańska, J.; Marek, D.; Filus, K.; Mendla, S. Supervised Learning of Neural Networks for Active Queue Management in the Internet. Sensors 2021, vol. 21(15), 4979. https://doi.org/10.3390/s21154979.

5. Domański, A.; Domańska, J.; Czachórski, T.; Klamka, J.; Szyguła, J; Marek, D. The IoT gateway with active queue management. International Journal of Applied Mathematics and Computer Science, vol. 31 (1), s. 165-178. https://doi.org/10.34768/amcs-2021-0012.

Lista pozostałych publikacji

1. Filus, K.; Domański, A.; Domańska, J.; Marek, D.; Szyguła, J. Long-Range Dependent Traffic Classification with Convolutional Neural Networks Based on Hurst Exponent Analysis. Entropy 2020, vol. 22(10), 1159. https://doi.org/10.3390/e22101159.

2. Domański, A.; Domańska, J.; Filus, K.; Szyguła, J.; Czachórski, T. Self-Similar Markovian Sources. Appl. Sci. 2020, vol. 10(11), 3727. https://doi.org/10.3390/app10113727.

3. 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. 9th International Conference on Non-integer Order Calculus and its Applications, Łódź, Poland. Lecture Notes in Electrical Engineering, Springer International Publishing, vol. 496, s. 36-48, 2019, https://doi.org/10.1007/978-3-319-78458-8_4.

4. Marek, D.; Domański, A.; Domańska, J.; Czachórski T.; Klamka J.; Szyguła J. Combined diffusion approximation - simulation model of AQMs transient behavior. Computer Communications 2021, vol. 166, s. 40-48. https://doi.org/10.1016/j.comcom.2020.11.014

5. Marek, D.; Domański, A.; Domańska, J.; Szyguła, J.; Czachórski, T.; Klamka, J. Diffusion Model of a Non-Integer Order PI^{γ} Controller with TCP/UDP Streams. Entropy 2021, vol. 23, 619. https://doi.org/10.3390/e23050619.

6. Czajkowski, A.; Remiorz, L.; Pawlak, S.; Remiorz, E.; Szyguła, J.; Marek, D.; Paszkuta, M.; Drabik, G.; Baron, G.; Paduch, J.; Antemijczuk, O. Global Water Crisis: Concept of a New Interactive Shower Panel Based on IoT and

Cloud Computing for Rational Water Consumption. Appl. Sci. 2021, vol. 11, 4081. https://doi.org/10.3390/app11094081.

7. Domański, A.; Domańska, J.; Czachórski, T.; Klamka, J.; Szyguła, J.; Marek, D. Diffusion approximation model of TCP NewReno congestion control mechanism. Conference on Modelling Methods in Computer Systems, Networks and Bioinformatics, 14-15 October 2019, Paris. Springer Nature Computer Science, vol. 1 (43). https://doi.org/10.1007/s42979-019-0032-x.

8. 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. 32nd International Symposium on Computer and Information Sciences (ISCIS), Communications in Computer and Information Science, Springer International Publishing, vol. 935, s. 76-83, 2018. https://doi.org/10.1007/978-3-030-00840-6_9.

9. 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. 25th International Conference on Computer Networks (CN 2018), Communications in Computer and Information Science, Springer International Publishing, vol. 860, s. 286-299, 2018. https://doi.org/10.1007/978-3-319-92459-5_23.

10. Remiorz, L.; Czajkowski, A,; Pawlak, S.; Remiorz, E.; Szyguła, J.; Drabik, G.; Marek, D.; Antemijczuk, O.; Paduch, J.; Baron, G.; Paszkuta, M. The concept of an interactive shower panel in terms of the assumptions of Industry 4.0. Contemporary problems of power engineering and environmental protection 2020, Politechnika Śląska, s.7-14, ISBN 978-83-950087-9-5.

11. Czajkowski, A.; Remiorz, L.; Pawlak, S.; Remiorz, E.; Szyguła, J. Methodology of modeling the detection of limescale deposits in sanitary installations. 21st International Multidisciplinary Scientific GeoConference SGEM 2021, 7-10 December, 2021, International Multidisciplinary Scientific GeoConference & EXPO SGEM, vol. 3.2, 2021, Sofia, STEF92 Technology, ISBN 978-619-7603-32-3. https://doi.org/10.5593/sgem2021V/3.2/s12.18.

Podsumowanie dorobku naukowego

Podsumowanie dorobku naukowego:

- Liczba publikacji: 16
- Liczba patentów: 1
- h-index (Cytowania Scopus): 3
- h-index (Cytowania WoS): 3
- Sumaryczny IF: 21.249
- Sumaryczny SNIP: 10.009
- Sumaryczny CiteScore: 37.4
- Sumaryczna punktacja MNiSW: 1 045

Udział w projektach badawczych:

1. Opracowanie systemowego oprogramowania oraz aplikacji mobilnej wraz z obsługą działania serwisu/systemu miejskiej wypożyczalni pojazdów hybrydowych i elektrycznych na obszarze Katowic, Okres uczestnictwa w projekcie: 01.12.2017 – 28.02.2018, Numer Projektu: NB-217/RM4/2017.

2. Prace badawczo-rozwojowe nad innowacyjnym wielofunkcyjnym urządzeniem sanitarnym, Okres uczestnictwa w projekcie: 02.01.2018 – 31.03.2021, Numer Projektu: POIR.01.01.01-00-0810/16.

3. Wykorzystanie operatorów ułamkowego rzędu do sterownia przeciążeniami sieci Internet, Okres uczestnictwa w projekcie: 20.06.2018 – 19.06.2022, Numer Projektu: 2017/27/B/ST6/00145.

4. Innowacyjna technologia tworzenia wydarzeń multimedialnych opartych o walki dronów z synergią między poziomami: wirtualnym, rozszerzonym i fizycznym, Okres uczestnictwa w projekcie: 01.06.2021 - 31.12.2021, Numer projektu: POIR.01.02.00-00-0160/20.

5. Opracowanie technologii inteligentnego roju rekonfigurowalnych dronów i jej weryfikacja na przykładzie pokazu dronów i inspekcji farm fotowoltaicznych, Okres uczestnictwa w projekcie: 10.01.2022 - 31.07.2022, Numer projektu: POIR.04.01.04-00-0078/20-01.

6. Technologia do bezpiecznego i niezawodnego dostarczania profesjonalnych przekazów kontrybucyjnych audio/wideo na żywo przy zachowaniu minimalnego możliwego opóźnienia, Okres rozpoczęcie prac w projekcie: 01.08.2021, Numer Projektu: POIR.01.01.01-00-1896/20-00.

7. Automated Guided Vehicles integrated with Collaborative Robots for Smart Industry Perspective, Okres rozpoczęcie prac w projekcie: 01.2022, Numer projektu: NOR/POLNOR/CoBotAGV/0027/2019 00.

Patent:

1. Sposób wykrywania wewnętrznego osadu kamiennego w układach hydraulicznych oraz urządzenie do realizacji tego sposobu. Numer patentu: PL 240 385, Data ogłoszenia patentu: 28.03.2022, Twórcy wynalazku: Sebastian Pawlak, Leszek Remiorz, Eryk Remiorz, Jakub Szyguła, Adrian Czajkowski, Gabriel Drabik, Dariusz Marek, Oleg Antemijczuk, Jarosław Paduch, Grzegorz Baron, Marcin Paszkuta.

Streszczenie

Niniejsza rozprawa rozszerza i porządkuje wiedzę na temat adaptacyjnych mechanizmów aktywnego zarządzania kolejką w sieciach komputerowych. Implementacja przedstawionych rozwiązań powinna wydatnie zwiększyć efektywność transmisji w sieci Internet, poprzez optymalizację poziomu zajętości kolejki oraz średniego czasu oczekiwania pakietu w kolejce. Rozprawę tę stanowi zbiór pięciu opublikowanych artykułów naukowych, w których przedstawiono zaproponowane modele adaptacyjnych mechanizmów aktywnego zarządzania kolejką oraz zawarto wyniki przeprowadzonych badań naukowych.

Zakres niniejszej pracy jest następujący: na początku przedstawiono opublikowane artykuły będące głównym osiągnięciem naukowym rozprawy doktorskiej. Następnie wypisano pozostałe publikacje, które są dodatkowym osiągnięciem pośrednio związanym z realizacją dysertacji oraz podsumowano dorobek naukowy.

W rozdziale 1 opisano zagadnienia wprowadzające w tematykę sieci komputerowych oraz algorytmów zarządzania natężeniem ruchu w sieci Internet, a następnie omówiono cel, zakres i tezę niniejszej rozprawy.

Rozdział 2 przedstawia najważniejsze zagadnienia teoretyczne pracy, które dotyczą tematyki transmisji danych w sieci Internet. W kolejnych podrozdziałach omawiane są następująco: zasady współpracy protokołu TCP oraz mechanizmu AQM (podrozdział 2.1), metody zarządzania przeciążeniami w protokole TCP (podrozdział 2.2), model aproksymacji Fluid-Flow wykorzystany do oceny współpracy mechanizmów AQM z protokołem TCP (podrozdział 2.3), cechy samopodobieństwa ruchu sieciowego (podrozdział 2.4), a także aktualny stan wiedzy związany z mechanizmami aktywnego zarządzania kolejką AQM (podrozdział 2.5).

W rozdziale 3 omówiono zbiór pięciu artykułów naukowych, w których to zaproponowano adaptacyjne mechanizmy aktywnego zarządzania kolejką. W rozdziale tym przytoczono również najważniejsze rezultaty wykonanych prac badawczych oraz dokładnie przedstawiono wkład własny autora.

W pracy przedstawionej w rozdziale 4 zaproponowano model mechanizmu aktywnego zarządzania kolejką, który do wyznaczania wartości prawdopodobieństwa odrzucenia pakietu wykorzystuje odpowiedź z trzech różnych kontrolerów PI^{α}.

W publikacji zawartej w rozdziale 5 skupiono się na doborze parametrów mechanizmu aktywnego zarządzania kolejką, w oparciu o techniki uczenia maszynowego oraz uczenia przez wzmocnienie.

Rozdział 6 przedstawia adaptacyjny mechanizm AQM, który dostosowuje się do intensywności ruchu oraz stopnia samopodobieństwa. Zaproponowane rozwiązanie również oparte jest o sieci neuronowe oraz techniki uczenia przez wzmocnienie.

W rozdziale 7 zaprezentowano adaptacyjny mechanizm aktywnego zarządzania kolejką, który oparto o metody uczenia nadzorowanego.

Rozdział 8 skupia się na modelu AQM dedykowanym dla transmisji w przypadku ruchu urządzeń Internetu Rzeczy, a zwłaszcza dla ochrony jego priorytetowych danych.

Wyniki przeprowadzonych prac, które zaprezentowano w rozdziałach 4 - 8 dowodzą postawionej w tej pracy tezie. Stworzenie adaptacyjnych mechanizmów zarządzania pakietami, dostosowujących się nie tylko do natężenia ruchu sieciowego, ale również do zależności długoterminowych, może znacząco zwiększyć efektywność transmisji w sieciach komputerowych.

Wnioski końcowe z przeprowadzonych badań, podsumowanie rozprawy oraz możliwe kierunki dalszych prac badawczych opisano w rozdziale 9.

Abstract in English

This dissertation presents the knowledge of adaptive Active Queue Management mechanisms in computer networks. Implementation of the presented solutions optimizes the transmission efficiency on the Internet. This allows for optimizing the queue occupancy level and the average packet waiting time in the transmission buffer. This dissertation is a collection of five published research peer-reviewed articles. They present the proposed models of adaptive Active Queue Management mechanisms and contain the results from research work.

The scope of this dissertation is as follows: first, there is a list containing published articles that are the main scientific achievements of this dissertation is presented. Then a list containing other publications and a summary of all scientific achievements are described.

Chapter 1 refers to the topic of computer networks and Active Queue Management mechanisms. It discusses the dissertation's purpose and thesis.

Section 2 presents the main theoretical issues of data transmission in the Internet network. The following subsections are discussed: principles of cooperation between TCP protocol and Active Queue Management mechanism (subsection 2.1), methods of congestion control in TCP protocol (subsection 2.2), Fluid-Flow approximation model used to evaluate cooperation of AQM mechanisms with TCP protocol (subsection 2.3), self-similarity characteristics of network traffic (subsection 2.4) and the current State Of The Art related to Active Queue Management mechanisms (subsection 2.5).

Subsection 3 presents a collection of five peer-reviewed research articles that propose novel adaptive Active Queue Management mechanisms. This chapter also describes obtained results and explains in detail the author's own contributions.

The paper presented in chapter 4 proposes a model of the Active Queue Management mechanism that uses the response from three different PI^{α} controllers to calculate the packet loss probability.

The paper described in chapter 5 focuses on selecting the Active Queue Ma-

nagement mechanism parameters based on Machine Learning and Reinforcement Learning.

The article in section 6 presents an adaptive AQM mechanism that adapts to the traffic intensity and the degree of self-similarity. The proposed solution is also based on Neural Networks and Reinforcement Learning.

The chapter 7 refers to the article that proposes an adaptive Active Queue Management mechanism based on supervised learning methods.

The chapter 8 presents a paper that focuses on the model of Active Queue Management, dedicated to the transmission for Internet of Things devices, especially for protecting its priority data.

The results presented in chapters 4 - 8 proves thesis stated in this dissertation. Creating adaptive Active Queue Management mechanisms that adjust not only to network traffic intensity but also to Long-Range traffic dependencies can significantly increase the computer network transmission efficiency.

The final conclusions from the research, the thesis summary and possible directions for further research are described in section 9.

Rozdział 1

Wstęp

Rozwój technologii informatycznych oraz nieustanny wzrost liczby użytkowników posiadających nieprzerwany dostęp do Internetu, spowodował napotkanie nowych problemów. Pierwotnie sieć Internet wykorzystywana była do przesyłania danych niewielkiego rozmiaru. Problem transferu dużych zbiorów danych oraz treści multimedialnych pojawił się z początkiem XXI wieku. Jednym z podstawowych problemów, które należy rozwiązać w sieciach komputerowych jest zagadnienie zapobiegania wystąpienia przeciążeń w transmisji danych. Początkowo problem przeciążeń w sieciach komputerowych był rozwiązywany w sposób taki, w którym to węzły samodzielnie monitorowały swoje przeciążenia. W przypadku ich zaobserwowania i detekcji wysyłały o tym fakcie informację bezpośrednio do nadajników. Zgodnie z założeniem, po odebraniu takiej informacji nadajnik zmniejszał prędkość nadawania, co z kolei przyczyniało sie do zniwelowania danego przeciążenia. Do najpopularniejszych rozwiązań tego typu należy zaliczyć mechanizm tzw. bitu ostrzegawczego (ang. warning bit), który stosowany był przez firmę IBM. Druga najpopularniejsza metoda oparta była o tzw. pakiety tłumiące (ang. choke packets) [1]. W przypadku szybkich sieci mechanizmy te okazały się jednak zbyt wolne i informacja o przeciążeniach docierała do nadajnika zbyt późno. Z tej przyczyny mechanizmy te przestały być powszechnie stosowane. Dlatego też w kolejnych latach zdecydowano się na stworzenie pewnych mechanizmów, w których to nadajnik sam decyduje jak prędko ma wysyłać dane. Mechanizmy mające zapewnić wspomniane postulaty zostały wbudowane w protokół warstwy transportowej sieci Internet. Są one szerzej opisane w licznych dokumentach RFC (ang. Request for Comments).

Pierwszym autorem, który opracował zasady sterowania przeciążeniami w protokole TCP był Van Jacobson w 1988 roku [2]. Natomiast pierwszą w pełni

opisaną metodą sterowania przeciążeniami oraz przygotowaną do wdrożenia w sieci Internet, był mechanizm TCP Tahoe. Rozwiązanie to było wielokrotnie modyfikowane i rozwijane w kolejnych latach. W zależności od klasy sieci przewodowych oraz bezprzewodowych powstało wiele różnych wersji algorytmów zarządzania oknem przeciążenia w protokole TCP. Na dzień dzisiejszy najpopularniejszym i najszerzej stosowanym mechanizmem jest TCP New Reno. Mechanizm ten będzie opisany w rozdziale 2.2 niniejszej rozprawy.

W sieci Internet mechanizm przeciwdziałania występowania przeciążeń realizowany jest poprzez wbudowanie w protokół transportowy TCP pewnych specyficznych mechanizmów zarządzania transmisją. Od czasu powstania pierwszego mechanizmu zarządzania oknem przeciążenia powstało wiele nowych wersji tego rozwiązania, które dedykowane są zarówno dla sieci homogenicznych, w których transmisja odbywa się za pomocą połączeń kablowych [3] lub bezprzewodowych [4], a także dla sieci heterogenicznych, gdzie trasa składa się z połączeń przewodowych oraz radiowych [5].

Na etapie projektowania protokołu TCP jego twórcy zdawali sobie sprawę z istoty zagrożenia, jakie może się pojawić gdy system odbiorcy nie będzie w stanie w odpowiednim czasie pobierać danych z bufora, do którego kierowane są pakiety wysyłane od nadawcy. Z tego powodu w protokole TCP uwzględniono mechanizm okna odbiornika (ang. Receiver Window). Określa on w precyzyjny sposób jaką ilość danych może wysyłać nadawca bez informacji zwrotnej o aktualnie dostępnym miejscu w kolejce, otrzymanej od systemu odbiorcy. W przypadku braku otrzymania takiego potwierdzenia transmisja zostanie wstrzymana. Celem okna odbiornika jest dostosowanie prędkości przesyłu do jego własnych możliwości.

Mechanizm okna przeciążenia (ang. Congestion Window) pozwala odbiorcy uchronić się przed przepełnieniem kolejki, a więc przed utratą danych. Co istotne w tym przypadku nadawca nie otrzymuje wprost informacji o wolnym miejscu w buforze, a wszelkie decyzje podejmowane są tylko na podstawie liczby otrzymanych potwierdzeń. Maksymalna liczba pakietów, które mogą zostać wysłane przez nadawcę nie może przekroczyć wartości zarówno okna odbiorcy, jak też przeciążenia. Okno przeciążenia ma więc na celu dostosowanie prędkości przesyłu do możliwości sieci.

Jednym z negatywnych skutków istnienia protokołów zarządzania oknem przeciążenia może być zjawisko globalnej synchronizacji. Wartym odnotowania jest, że organizacja IETF (ang. Internet Engineering Task Force) mając na uwadze wcześniej przytoczone zagrożenia, poleca stosowanie algorytmów aktywnego zarządzania kolejką (ang. Active Queue Management). Mają one na celu zapobie-

ganie niebezpieczeństwu występowania wspomnianego wcześniej zjawiska - globalnej synchronizacji oraz utraty łączności. Mają także na uwadze zwiększenie efektywności transmisji. Koncepcja ta związana jest z prewencyjnym odrzucaniem pakietów, nawet gdy w buforze wciąż znajduje się jeszcze wolne miejsce. Z kolei samo prawdopodobieństwo odrzucenia pakietu wzrasta wraz z poziomem zajętości kolejki. Pakiety odrzucane są losowo, skutkiem czego większość użytkowników nie zaobserwuje jakichkolwiek problemów związanych z przesyłem danych.

Wyróżnić można dwa podstawowe mechanizmy zarządzania zajętością kolejki w transmisji w sieci Internet. Pierwszym z nich jest metodologia pasywnego odrzucania danych. Natomiast do drugiej grupy rozwiązań zaliczamy mechanizmy aktywnego zarządzania kolejkami.

W przypadku metod statycznego zarządzania kolejką pakiety wyrzucane są dopiero po przepełnieniu bufora. W tym przypadku jedynym sposobem zarządzania średnim czasem oczekiwania pakietu w kolejce jest dobór maksymalnego dopuszczalnego rozmiaru długości kolejki. W przypadku przepełnienia bufora dodatkową wadą jest zjawisko, w którym pakiety są gubione w większej ilości. To natomiast skutkuje pojawieniem się problemu globalnej synchronizacji, która to omówiona będzie dokładnie w dalszych rozdziałach niniejszej rozprawy.

Mechanizmy aktywnego zarządzania kolejką AQM (ang. Active Queue Management) bazują natomiast na stałej obserwacji łącza, czyli stopnia zapełnienia bufora. Na tej podstawie podejmowane są następnie decyzje o wcześniejszym - prewencyjnym odrzuceniu pakietu. Pakiety te są wyrzucane losowo zgodnie z wyliczoną wartością funkcji prawdopodobieństwa. Pozwala to zwiększyć przepustowość sieci i zapewnić sprawiedliwy dostęp do łącza.

Pierwszym pełnoprawnym i zarazem wciąż najpopularniejszym algorytmem aktywnego zarządzania kolejką AQM jest mechanizm RED (ang. Random Early Detection), zaproponowany w 1993 roku przez Sally Floyd i Van Jacobson'a [6]. Mechanizm ten pozwala uzyskać rozsądny, akceptowalny poziom długości kolejki, a liczba odrzuconych prewencyjnie pakietów rośnie wraz ze wzrostem zapełnienia bufora. Ukazały się również prace, w których starano się zwiększyć wydajność tego mechanizmu [7]. Z kolei inne podejście do tego zagadnienia zaobserwowano w artykule [8], gdzie rozważano bardziej szczegółową analizę historii zajętości kolejek oraz zmianę w sposobie implementacji średniej ważonej długości kolejki. W pracy tej wykazano, że taka koncepcja może również znacząco poprawić wydajność transmisji.

Zastosowanie mechanizmu RED oraz jego licznych modyfikacji pozwala zmniejszyć poziom opóźnień w transmisji. Konieczne jest jednak właściwe dobra-

nie parametrów mechanizmu aktywnego zarządzania kolejką [9]. W przeciwnym przypadku, system TCP/RED staje się niestabilny [10]. Naturalnym następstwem tej sytuacji było pojawienie się wersji adaptacyjnych mechanizmu RED, które to automatycznie dostosowują parametry algorytmu w zależności od aktualnego natężenia ruchu. Pierwszym tego typu zaproponowanym rozwiązaniem był mechanizm ARED (ang. Adaptive Random Early Detection) [11]. W tym przypadku modyfikacja polega na dostosowaniu funkcji, za pomocą której wyznaczane jest prawdopodobieństwo odrzucenia pakietu w taki sposób, aby jej wartość oscylowała pomiędzy zdefiniowanym progiem minimalnym oraz maksymalnym. Przeprowadzone badania dowiodły, że rozwiązanie to zmniejsza liczbę odrzucanych pakietów oraz poziom zmienności opóźnień występujących w transmisji [12]. Rozwiązanie to niesie za sobą jednak również pewne wady. W pracy [13] zwrócono uwagę, że dostosowanie funkcji prawdopodobieństwa odrzucenia pakietu jest niestety czasochłonne. W związku z tym adaptacja do nieustannie zmieniających się warunków w sieci Internet nie odbywa się w sposób wystarczający. Dlatego też w ostatnich latach ukazało się wiele kolejnych prac, w których autorzy starali się zaproponować różne, nowe modyfikacje algorytmu RED, mając na uwadze zwiększenie efektywności transmisji. Swoista popularność prac prowadzonych w obszarze modyfikacji mechanizmów TCP oraz AQM wynika również z faktu, że z uwagi na rozległość sieci Internet, najlepszym sposobem wprowadzania usprawnień jest stosowanie mechanizmów separowalnych, które to implementować można na jednej krańcówce sieci, czyli w pojedynczych routerach oraz stacjach nadawczych. Efekty wspomnianych prac szerzej opisane zostały w rozdziale 2.4 niniejszej rozprawy.

Z drugiej strony, wszystkie algorytmy adaptacyjne, które znane są z literatury, uzależniają się od intensywności ruchu, który to z kolei powiązany jest z poziomem zajętości kolejki. Inną zaobserwowaną i bardzo istotną własnością ruchu, wpływającą na jego dynamikę i właściwości, są cechy samopodobieństwa, które pomijane były przez wiele lat w pracach nad mechanizmami mającymi na celu zwiększenie prędkości transmisji.

Zależności długoterminowe zaobserwowano po raz pierwszy w połowie XX wieku, kiedy to Sir H.E. Hurst pracował nad budową tamy na rzece Nil. Posiadał on zapisy zawierające zmiany linii brzegowych z przeszło 800 lat. Wtedy to też po przeprowadzeniu dokładnej analizy posiadanych danych Hurst wykazał, że poziom wody w rzece nie jest procesem całkowicie losowym, niezależnym od przeszłości. Swoje wnioski oparł na całkowicie innowacyjnej koncepcji, w stosunku do metod stosowanych przez współczesnych hydrologów. Wykorzystał on opublikowaną przez Alberta Einsteina 1908 roku pracę na temat ruchów Browna.

Hurst zaproponował, by poziom lustra wody w projektowanym zbiorniku przedstawić za pomocą położenia wspomnianej cząstki Browna, natomiast sumę całorocznych wpływów wody jako skoki i zmiany tej cząstki [14, 15]. Przyjęcie powyższych założeń pozwoliło wykazać istnienie długoterminowej pamięci zdarzeń w przypadku nieskończenie długich szeregów czasowych [16]. Pomimo faktu, że terminy samopodobieństwa oraz zależności długoterminowych bywają używane zamiennie, należy pamiętać, że nie są one dokładnie tym samym [17].

Analiza prac z zakresu samopodobieństwa ruchu sieciowego, pozwala stwierdzić, że jest ono jedną z cech, która znacząco wpływa na stopień zajętości kolejki oraz właściwości transmisji [18, 19] i koniecznym jest uwzględnianie jego wpływu podczas przeprowadzania analizy służącej ocenie mechanizmów aktywnego zarządzania kolejką. Wyniki prac przeprowadzonych przez naukowców z Bellcore [20] jednoznacznie wykazały samopodobną cechę ruchu pakietów w sieci. Praca ta stanowiła motywację pod liczne badania, w których wykazano istotny wpływ stopnia samopodobieństwa na ruch w transmisji TCP [21]. Potwierdzono również jego występowanie w sieciach rozległych WAN (ang. Wide Area Network) [22]. O cechach samopodobieństwa ruchu w łączu transmisyjnym mówimy zazwyczaj wtedy, gdy jest on traktowany całościowo. Jednak w przypadku, gdy analizie poddany zostanie wyłącznie ruch poszczególnych usług i aplikacji, to również będzie się on cechować podwyższoną wartością stopnia samopodobieństwa [23]. Analiza danych rzeczywistego ruchu sieciowego, zgromadzona przez naukowców z Bellcore [20], a także danych ze zbiorów CAIDA [24] (Centrum analizy danych sieci Internet na Uniwersytecie Kalifornijskim w San Diego), pozwala stwierdzić, że na tym samym łączu transmisyjnym obserwuje się ruch o różnym stopniu samopodobieństwa. Wszelkie próby pominięcia powyższych reguł sprawiają, że w zasadzie niemożliwe staje się poprawne oszacowanie stopnia zajętości bufora [25].

Prowadzone w tej pracy badania nad adaptacyjnymi algorytmami zarządzania natężeniem ruchu mają wykazać, czy dobór i modyfikacja parametrów tego mechanizmu w czasie rzeczywistym pozwoli na zwiększenie prędkości transmisji w sieci Internet. Pośrednim celem niniejszej rozprawy jest więc optymalizacja poziomu zajętości kolejki, a co za tym idzie średniego czasu oczekiwania pakietu w kolejce.

Pomimo ukazania się licznych prac, w których rozważane są algorytmy aktywnego zarządzania kolejką, według mojej najlepszej wiedzy nie istnieją inne adaptacyjne mechanizmy, które skalują się i adaptują do pierwszorzędnych cech ruchu, czyli do jego natężenia, a także do zależności długoterminowych i cech samopodobieństwa ruchu sieciowego.

W ramach tej pracy zostaną zaproponowane pewne mechanizmy, a ich ewaluacja odbędzie się na drodze badań symulacyjnych oraz badań analitycznych. Do prawidłowej oceny mechanizmów aktywnego zarządzania kolejką oraz wydajności sieci komputerowej konieczne jest stworzenie odpowiednich modeli mechanizmów sieciowych, a także rzeczywiste odzwierciedlenie ruchu danych. Modele analityczne oparte o teorie kolejkowania są często spotykane w pracach badawczych z zakresu tematyki sieci komputerowych. Z drugiej strony, podążając za wnioskami autorów prac [26, 27, 28, 29] stosowanie symulacji sieciowych pozwala znacząco usprawniać rozwiązania sieciowe.

Celem niniejszej pracy jest przedstawienie nowego podejścia do adaptacyjnych mechanizmów aktywnego zarządzania kolejką. Standardowo algorytmy te zmieniają swoje parametry dostosowując się do natężenia ruchu sieciowego. Zaproponowane podejście ma na celu dobór tychże parametrów zarówno w zależności od natężenia ruchu, jak również stopnia jego samopodobieństwa wyrażonego za pomocą parametru Hursta. Wprowadzenie tej modyfikacji powinno pozwolić na optymalizację poziomu zajętości kolejki, a co za tym idzie również średniego czasu oczekiwania pakietu w buforze. Optymalizacja kolejki w routerze powinna wydatnie zwiększyć efektywność transmisji w sieciach komputerowych.

Do oceny skuteczności zaproponowanych rozwiązań w przypadku modelu pętli otwartej (ang. open loop) wykorzystano metody symulacyjne. Modelowanie zachowania protokołów TCP i ich współpracy z mechanizmami aktywnego zarządzania kolejką oparto o model aproksymacji Fluid-Flow [30, 31]. W rozpatrywanym przypadku są to badania modelu pętli zamkniętej (ang. closed loop).

Tezę niniejszej rozprawy zdefiniowano w sposób następujący:

Teza. *Stworzenie adaptacyjnych mechanizmów zarządzania pakietami, dostosowujących się nie tylko do natężenia ruchu sieciowego, ale również do zależności długoterminowych, może znacząco zwiększyć efektywność transmisji w sieciach komputerowych.*

Rozdział 2

Teoria i analiza transmisji w sieci Internet

2.1 Transmisja sieciowa oparta o współpracę protokołu TCP oraz mechanizmu AQM

Stworzenie protokołu TCP (ang. Transmission Control Protocol) w 1974 roku można uznać za wydarzenie stające się swoistym motorem napędowym ewolucji sieci Internet, które to umożliwiło jej dynamiczny rozwój. Aplikacje internetowe wymagały po pierwsze niezawodnego dostarczania danych, a po drugie by wysyłane one były we właściwej kolejności. Kolejnym wyzwaniem było zapobieganie skutkom wystąpienia potencjalnych awarii. Wszystkie te problemy znalazły rozwiązanie właśnie w postaci zastosowania i wdrożenia protokołu TCP, czyli protokołu sterowania transmisją [31]. Za podstawowy mechanizm protokołu TCP uznać należy potwierdzanie odbioru danych przez odbiornik oraz zegar retransmisji. Z czasem, w związku z zaobserwowaniem zróżnicowania pod względem wydajności poszczególnych węzłów, koniecznym stało się wprowadzenie mechanizmów zarządzania przeciążeniami. Wspomniana sytuacja, a także pojawianie się coraz większych wymagań pod względem szybkości transmisji, wymusiło powstawanie szeregu algorytmów sterowania przeciążeniami. Z kolei pojawienie się sieci bezprzewodowych zapoczątkowało nową gałąź rozwoju, mającą na celu dostosowanie protokołu TCP do warunków panujących w tego typu transmisji, cechującej się zauważalnie większymi opóźnieniami. Wtedy to zaczęto wprowadzać modyfikacje wykrywające zbędne retransmisje. Elastyczność protokołu TCP umożliwiła jego efektywne usprawnianie oraz dostosowanie do pojawiających się

na przestrzeni lat nowych wyzwań. Co istotne, implementacja nowych mechanizmów w tym przypadku wymaga zmian jedynie po stronie nadajnika, co znacząco upraszcza ich wprowadzanie do powszechnego zastosowania.

Jednostka transportowa protokołu TCP umożliwia współpracę z warstwą Internetu, odpowiadającą za wymianę pakietów pomiędzy nadawcą a odbiorcą. W skład wspomnianej warstwy wchodzi również protokół IP, formuujący strumienie danych w odpowiedniej wielkości porcje, zwane datagramami [32]. Protokół IP sam w sobie nie zawiera jednak mechanizmów mających na celu zapewnienie niezawodności wymiany danych. W związku z tym wszelkie czynności związane z retransmisją danych w przypadku, gdy przekroczone zostają limity oczekiwania na odpowiedź z odbiornika, wynikają już całkowicie z własności TCP. Zasada funkcjonowania tego mechanizmu opiera się o regułę pozytywnych potwierdzeń. Host nadający dane (w przypadku połączeń TCP określanych mianem segmentów) nie inkrementuje samodzielnie numeru sekwencyjnego. Następny oczekiwany numer sekwencyjny wysyłany jest przez host docelowy, wraz z segmentem potwierdzającym. Tak więc jeżeli wiadomość ta nie dotrze w określonym limicie czasowym, host nadający ponowie wysyła ten sam zestaw danych. Warto wspomnieć, że powyższa cecha, wskazująca na możliwość jednoczesnego przesyłania danych przez hosty w obu kierunkach - sprawia, że połączenia te nazywane bywają również połączeniami pełno dupleksowymi [32].

Przedstawione cechy i własności protokołu TCP odpowiadają tylko i wyłącznie za niezawodność transmisji oraz gwarancję poprawnego przesłania danych. Jednak wymagania stawiane przez dzisiejszych użytkowników sieci Internet nieustannie ulegają zwiększeniu. Postulują oni, by transfer danych był szybki i efektywny, ale też dostosowany do współczesnych zaawansowanych systemów komputerowych oraz do transmisji treści multimedialnych. Ma być również odpowiedni i dostosowany do podłączonych do sieci urządzeń IoT (ang. Internet of Things). Warto zwrócić uwagę, że liczba aktywnych urządzeń już w 2020 roku oscylowała wokół 50 miliardów [33]. Fakty te sprawiły, że prace mające na celu optymalizację oraz zwiększenie wydajności i prędkości transmisji w sieci Internet stały się zadaniem jeszcze bardziej priorytetowym i nabrały jeszcze większego znaczenia niż kiedykolwiek wcześniej.

2.2 Sterowanie przeciążeniami w protokole TCP

Z uwagi na bardzo dużą liczbę stworzonych odmian i modyfikacji algorytmów TCP, w niniejszym podrozdziale przedstawione zostaną wyłącznie te rozwiązania,

które w momencie ich publikowania oraz implementowania doprowadziły do największych zmian oraz w największym stopniu przyczyniły się do rozwoju sieci Internet.

Pierwszym praktycznym i opisanym rozwiązaniem pozwalającym na sterowanie przeciążeniami w oknie TCP był mechanizm TCP Tahoe, autorstwa Van Jacobson'a w 1988 roku [2]. Była to tak naprawdę odpowiedź na sytuację, która zaistniała 2 lata wcześniej. Wtedy to brak algorytmów zapobiegania przeciążeniom spowodował całkowitą blokadę sieci Internet [34]. Rozwiązaniem udoskonalającym pierwsze zasady zarządzania przeciążeniami był algorytm "TCP Reno" [35]. Wzbogacono go o mechanizm szybkiego odtwarzania pakietów (ang. Fast Recovery).

Rozwiązanie to jednak nie było doskonałe. Problematyczne stało się zagadnienie wielokrotnych strat, gdzie w fazie szybkiego odzyskiwania przesyłany ponownie jest wyłącznie jeden pakiet. Efektem tego wielkość okna przeciążenia zmniejszana jest w tym przypadku do tak małej wartości, która uniemożliwia dalsze utrzymanie transmisji danych. Z tego powodu opracowany został algorytm "New-Reno" [36]. Udoskonalony w nim mechanizm szybkiego odzyskiwania powodował, że retransmisja utraconego wcześniej pakietu oraz uzyskanie częściowego potwierdzenia nie przerywa algorytmu, lecz powoduje kontynuację odzyskiwania następnego segmentu danych. Natomiast uzyskanie pierwszego częściowego potwierdzenia skutkuje zerowaniem wartości zegara retransmisji oraz zmniejszeniem wartości okna przeciążenia. Cykl działania algorytmu ma miejsce wtedy, gdy zostanie uzyskane potwierdzenie dla segmentu o największym wysłanym numerze przed rozpoczęciem fazy szybkiej retransmisji lub minie ustalony czas zegara retransmisji [36].

Wraz z rozwojem sieci bezprzewodowych pojawił się jeszcze jeden istotny czynnik związany ze sterowaniem wielkością okna przeciążenia. W przeciwieństwie do sieci przewodowych, w sieciach bezprzewodowych utrata pakietów występuje bardzo często również na skutek błędów w warstwie fizycznej, nie wynika jedynie z przeciążeń sieci. W takim przypadku oznacza to, że zmniejszenie prędkości transmisji jest zupełnie bezpodstawne i wpływa jedynie negatywnie na wydajność sieci. Algorytm "TCP Westwood" i jego późniejsza modyfikacja o nazwie "Westwood+", stanowią jedną z odpowiedzi na rozwiązanie tego problemu. Poprawiają one wydajność w sieciach heterogenicznych, czyli takich, które łączą w sobie zarówno transmisję przewodową, jak również radiową. Stanowią również dobry wybór dla sieci homogenicznych, w których występuje duża liczba błędów w warstwie fizycznej. Opierają się one na zasadzie ciągłego estymowania szerokości pasma. W czasie gdy potwierdzona zostaje strata, rozmiar okna przeciążenia

jest z kolei zmniejszany o wartość, która ustalana jest na podstawie przepustowości zanotowanej jeszcze przed zaistniałą sytuacją [37]. Najistotniejsze zmiany zasady działania algorytmu, względem wcześniejszych rozwiązań, polegają na rozróżnieniu podejmowanych akcji i reakcji, w zależności od zaobserwowanej przyczyny zanotowanych strat [5].

Twórcy algorytmu TCP Vegas zdecydowali się przyjąć zgoła odmienną koncepcję sterowania przeciążeniami, niż twórcy innych rozwiązań. Zgodnie z tą ideą, nadajnik wysyłał jak największą możliwą ilość danych, czego skutkiem było duże zapełnienie kolejki, a w efekcie dłuższy czasu obsługi. Nowa koncepcja TCP Vegas przyjęła założenie, że odrzucenie pakietu samo w sobie jest zjawiskiem negatywnym. W związku z tym twórcy starali się zapobiec tej sytuacji poprzez zmniejszenie poziomu zajętości kolejki. Efektem tego jest zwiększenie przepustowości sieci oraz skrócenie czasu oczekiwania pakietu w buforze [38].

Koncepcja algorytmu BI-TCP [39] zakłada, że straty pakietów to zjawisko negatywne, podobnie jak założenia mechanizmu TCP Vegas. Jednak zaimplementowana metoda wyszukiwania binarnego wykorzystuje tę informację, w celu poszukiwań optymalnej wielkości okna przeciążenia. Jest to mechanizm niewątpliwie bardziej agresywny, pozwalający na szybkie określenie możliwości łącza. Polecany jest w szczególności do zarządzania przeciążeniami w szybkich sieciach.

Wszystkie wyżej przytoczone mechanizmy, a także różnorodność przyjętych koncepcji sterowania przeciążeniami, wskazują na wysoką elastyczność protokołu TCP. Umożliwia to efektywne usprawnianie protokołu oraz dostosowywanie go do pojawiających się ciągle nowych wyzwań i wymagań stawianych sieci Internet. Co warte podkreślenia, mechanizmy te nie wymagają implementacji po stronie odbiorcy, ani też węzłów pośredniczących. Wystarczające jest dokonanie zmian i wdrożeń jedynie po stronie nadajnika, co w tym przypadku znacząco upraszcza całą procedurę.

2.3 Model aproksymacji Fluid-Flow

W przypadku prac mających na celu zwiększenie efektywności transmisji w sieci Internet poprzez modyfikację już istniejących mechanizmów aktywnego zarządzania kolejką oraz proponowaniem zupełnie nowych rozwiązań, konieczna staje się ewaluacja ich współpracy z protokołem TCP. W pracy [31] przedstawiono podejście, w którym do tego celu zastosowano aproksymację Fluid-Flow. Natomiast artykuł [40] przedstawia model Fluid-Flow, który zastosowano do modelowania wielu strumieni TCP/UDP. Z kolei w pracy [30] pierwszy raz wykorzystano

model TCP NewReno do oceny dynamiki protokołu TCP. Model ten ignoruje zależności czasowe (TCP Timeout) i może służyć do wyrażenia dynamiki protokołu TCP poprzez określenie średnich wartości parametrów transmisji, takich jak czas oczekiwania pakietu w kolejce, czy zajętość bufora.

$$\frac{dW_i(t)}{dt} = \frac{1}{R_i(t)} - \frac{W_i(t)}{2}\frac{W_i(t - R(t))}{R_i(t - R_i(t))}p(t - R_i(t)) \tag{2.1}$$

Wzór ten opisuje zmianę rozmiaru okna przeciążenia. Z kolei poniższe równanie przedstawia ewolucję przeciążonej kolejki routera:

$$\frac{dq(t)}{dt} = \sum_{i=1}^{N}\frac{W_i(t)}{R_i(t)} - C, \tag{2.2}$$

gdzie:

W_i - oczekiwany rozmiar okna TCP, wyrażony za pomocą liczby pakietów. Definiuje liczbę pakietów, jaka może zostać wysłana bez otrzymania potwierdzenia odbioru,

R_i - round-trip time, $R_i = q/C + T_p$, suma $\sum \frac{W_i}{R_i}$ oznacza całkowity przepływ wejściowy do routera,

q - oczekiwana liczba pakietów w buforze,

C - przepustowość łącza (liczba pakietów / sek.),

T_p - czas propagacji sygnału (sek.),

N - liczba strumieni TCP,

p - prawdopodobieństwo odrzucenia pakietów.

Maksymalne wartości W i q, czyli rozmiaru okna przeciążenia oraz zajętości kolejki zależne są od obciążenia bufora oraz maksymalnego rozmiaru okna [31]. Natomiast wartość prawdopodobieństwa odrzucenia pakietów p obliczana jest za pomocą wybranego algorytmu zarządzania kolejką. Szczegółowy opis powyższej metody zawarty jest w literaturze [41].

2.4 Samopodobieństwo i mechanizmy generowania samopodobnego ruchu sieciowego

Prace związane z modelowaniem ruchu sieciowego, umożliwiające późniejszą estymację wydajności transmisji w sieci Internet, wymagają bardzo dokładnego odzwierciedlenia statystycznych własności przesyłu danych. Wcześniejsze prace badawcze wykazały, że wszelkie próby pominięcia charakterystyk rzeczywistego ruchu sieciowego skutkowały niepoprawnym oszacowaniem poziomu zajętości kolejki, a przez to błędną oceną wydajności transmisji [25].

Tematyka dotycząca cech samopodobieństwa ruchu stała się popularna w literaturze, odkąd zostało potwierdzone, że zależności te są ściśle skorelowane z występującymi opóźnieniami, ich zmiennością (ang. jitter), a także liczbą pakietów odrzuconych [42]. Wobec tego można przyjąć, że w celu stworzenia wysoko wydajnych mechanizmów sieciowych, które zapewnią użytkownikowi odpowiedni poziom jakości usług QoS (ang. Quality of Service), analiza charakterystyk ruchu sieciowego staje się niejako jednym z obowiązkowych narzędzi tzw. inżynierii Internetu [43]. Pierwszym krokiem, jaki należy wykonać, by poznać to narzędzie, jest właściwe rozróżnienie pojawiającej się tutaj terminologii, a także poprawne zrozumienie występujących między nimi różnic. Problem ten nie jest jednak trywialny, o czym najlepiej świadczy fakt, że terminy samopodobieństwa oraz zależności długoterminowych LRD (ang. Long-range dependence) bywają czasem mylone i błędnie stosowane nawet w literaturze przedmiotowej [44]. Zależności długoterminowe LRD związane są ze statystycznymi korelacjami, występującymi w przypadku dużych skalach czasowych. Z drugiej strony termin samopodobieństwa stosowany jest do przedstawiania zjawisk, niezależenie od przyjętych ram i skali czasowych [43].

Tak więc ciągły szereg czasu $Y(t)$ jest samopodobny wtedy i tylko wtedy, gdy spełniony jest poniższy warunek:

$$Y(t) \stackrel{d}{=} a^{-H} Y(at), \tag{2.3}$$

dla $t \geqslant 0, a \ \geqslant 0$ i $0 < H < 1$. Parametr Hursta, za pomocą którego wyraża się stopień samopodobieństwa, przyjmować może wartości z zakresu $(0; 1)$, gdzie:

- $H \in (0; 0.5)$: negatywna korelacja - brak zależności długoterminowych, występują zależności krótkoterminowe.
- $H = 0.5$: brak korelacji.

- $H \in (0.5; 1)$: pozytywna korelacja - istnieją zależności długoterminowe.

W literaturze dostępny jest szereg prac i sposobów, za pomocą których wykryć można zależności długoterminowe ruchu. Do najpopularniejszych z nich zaliczyć należy metodę zagregowanej wariancji [45], czy też jeden z najstarszych znanych sposobów, czyli metodę R/S, opartą o centralne twierdzenie graniczne [46]. Istnieją też metody oparte na periodogramach [47], w których to jednak problem pojawia się w przypadku nieskończonej wariancji, czy też modyfikacje oparte na periodogramach półparametrycznego estymatora parametru Hursta - czyli na metodzie lokalnego estymatora Whittle'a [48]. Kolejna i zarazem jedna z najczęściej spotykanych metod oparta jest na teorii falkowej [49]. W przeciwieństwie do pozostałych metod, jest o wiele bardziej odporna na proste trendy występujące w ruchu [43]. Z drugiej strony, rezultaty jednej z naszych ostatnich prac badawczych dowiodły, że do wykrywania zależności w ruchu sieciowym można z powodzeniem zastosować konwolucyjne sieci neuronowe [50].

Istnieje szereg prac, które związane były z określeniem stopnia samopodobieństwa ruchu za pomocą parametru Hursta [51]. W Instytucie Informatyki Teoretycznej i Stosowanej Polskiej Akademii Nauk przeprowadzono rozszerzone badania z wykorzystaniem danych pochodzących z lokalnego ruchu sieciowego [52].

Najstarszą metodą umożliwiającą modelowanie strumienia danych, zawierających strukturę samopodobną była metoda Poissona. Podejście to nie charakteryzowało się jednak występowaniem zależności dalekosiężnych [53]. Skutkiem tego zaistniała potrzeba, by rozpocząć prace mające na celu stworzenie nowych modeli strumieniowania danych. Jednym z rozwiązań okazało się zastosowanie strumieni Poissona z wykorzystaniem modulacji procesem Markowa (MMPP) [54], a także modeli z zastosowaniem superpozycji wielu źródeł ON-OFF oraz rozkładu Pareto [20]. W późniejszych pracach zaczęto wykorzystywać stacjonarny proces gaussowski FGN (ang. Fractional Gaussian Noise). Metoda ta stała się powszechnie stosowana przy zagadnieniach modelowania ruchu w sieci Internet [17]. Oczywiście istnieje szereg rozwiązań alternatywnych, do których zaliczyć można ułamkowy model autoregresyjny FARIMA (ang. Fractional Auto-regressive Integrated Moving Average). Jednak głównym problemem w tym przypadku staje się zbyt duża złożoność obliczeniowa [55].

2.5 State-Of-The-Art: Adaptacyjne mechanizmy aktywnego zarządzania kolejką

Prace nad nowymi mechanizmami aktywnego zarządzania kolejką AQM trwają nieprzerwanie od wielu lat. Mechanizmy te zestawiane są z dotychczas istniejącymi rozwiązaniami i porównywane pod kątem takich parametrów jak sumaryczna liczba odrzuconych pakietów, średnia długość kolejki, czy opóźnienia w transmisji. Modyfikacje algorytmów RED oraz ARED w większości różnią się sposobem estymacji wartości jego parametrów [56]. Swoje zastosowanie w grupie algorytmów AQM znalazł również mechanizm niecałkowitego rzędu PI [57]. Kierunek ten wciąż zyskuje na popularności, a mechanizm kontrolera PID został również poddany analizie wpływu stopnia samopodobieństwa ruchu i zależności długoterminowych na efektywność transmisji [58]. Zaobserwować można również podejście, w którym autorzy [59] zastosowali algorytm metaheurystyczny GWO (ang. Grey Wolf Optimizer), w celu dostrojenia parametrów regulatora PI. Miało to za zadanie zapewnienie większej stabilności transmisji, zwłaszcza w przypadku większej liczby połączeń TCP.

Do osobnej grupy można zaklasyfikować badania, w których wykorzystano metody sztucznej inteligencji przy tworzeniu nowych mechanizmów aktywnego zarządzania kolejką. W literaturze przedmiotu znaleźć można wiele prac, w których dzięki wykorzystaniu sieci neuronowych znacząco udało zwiększyć się efektywność transmisji w sieciach komputerowych. Autorzy [60] przedstawili nowy mechanizm AQM, nazwany Q-learning RED, oparty o metody uczenia maszynowego ze wzmocnieniem. Z kolei w pracy [61] zaproponowano algorytm ANB-AQM z mechanizmem propagacji wstecznej w sieciach neuronowych, wykorzystujących funkcję aktywacji. Rezultaty przeprowadzonych badań eksperymentalnych dowiodły, że podejście to pozwala zwiększyć adekwatność podejmowanych decyzji o przyjęciu lub też odrzuceniu pakietów, a tym samym pozwala zwiększyć efektywność transmisji. Publikacja [62] przedstawia nowy model Fuzzy Neuron REM (FNREM). W pracy tej dzięki zastosowaniu sieci neuronowych zmodyfikowano wartość całki proporcjonalnej klasycznego algorytmu REM. Poprzez zaimplementowanie sztucznych sieci neuronowych zaproponowano również nowy algorytm Adaptive Neuron Proportional Integral Differential (ANPID) [63]. Mechanizm ten wykorzystuje pojedynczy neuron do doboru współczynników kontrolera PID. Publikacje [64, 65] stanowią kolejny przykład badań mających na celu wykorzystanie pojedynczego neuronu oraz uczenia przez wzmocnienie do stworzenia nowych adaptacyjnych mechanizmów AQM. Celem tych prac było w głównej

mierze zmniejszenie opóźnień występujących w transmisji. Autorzy [66] zaproponowali w swojej pracy adaptacyjny mechanizm GRPID, który oparty jest na klasycznym regulatorze PID, sieciach neuronowych RBF o radialnych funkcjach bazowych oraz algorytmie genetycznym. W rozwiązaniu tym algorytm genetyczny pomaga dostosować wagi sieci neuronowej oraz przyspieszyć prędkość uczenia, a sieć neuronowa RBF wykorzystana jest natomiast do doboru parametrów mechanizmu AQM. W związku z koniecznością stworzenia nowych mechanizmów, które pozwolą skutecznie zapobiegać przeciążeniom w transmisji, autorzy w poszukiwaniu nowych rozwiązań nie ograniczają się tylko do jednego podejścia. W jednej z takich prac [67], wykorzystano metodę optymalizacji stochastycznej, opartą o algorytm Cuckoo Search, do wyszukiwania współczynników rozmytego kontrolera PID (Fuzzy-PID). Natomiast w publikacji [68] badania oparto o szczególny rodzaj rekurencyjnej sieci neuronowej - Long short-term memory (LSTM), mającej pomóc określić prawdopodobną zajętość bufora w następnym kroku, a przez to zwiększyć wydajność kontrolera PID, zastosowanego jako mechanizm AQM.

Z kolei badania przedstawione w artykule [69] stanowią przykład zastosowania uczenia nienadzorowanego, za pomocą którego zaproponowano adaptacyjny algorytm PHAQM. Mechanizm ten, oparty o regułę uczenia Hebbowskiego pozwolił autorom zaobserwować ustabilizowanie zajętości bufora oraz zmniejszenie oscylacji względem klasycznych rozwiązań.

Literatura przedmiotu z tematyki związanej z mechanizmami aktywnego zarządzania kolejką oraz z transmisją danych w sieci Internet zawiera bardzo bogatą bibliotekę prac naukowych. Autorzy obierali różne kierunki w prowadzonych badaniach naukowych, jednak w czasie pisania niniejszej rozprawy, według mojej najlepszej wiedzy, nie istnieją inne adaptacyjne mechanizmy, które skalują się zarówno do natężenia ruchu jak i do zależności długoterminowych oraz cech samopodobieństwa ruchu w sieci Internet.

Rozdział 3

Implementacja oraz badania adaptacyjnych modeli mechanizmów AQM

W ramach niniejszego rozdziału omówiono zaproponowane adaptacyjne mechanizmy aktywnego zarządzania kolejką. Modele te następnie zaimplementowano, a rezultaty przeprowadzonych badań eksperymentalnych opublikowano w czasopismach naukowych oraz wygłoszono w ramach międzynarodowych konferencji naukowych. Publikacje te, wraz z dokładnym omówieniem wkładu autorskiego przytoczono w poniższych podrozdziałach.

3.1 Publikacja 1: Model mechanizmu aktywnego zarządzania kolejką wykorzystujący odpowiedź z kilku kontrolerów PI^{α}

W publikacji "AQM mechanism with the dropping packet function based on the answer of several PI controllers" przedstawiono nową koncepcję mechanizmu aktywnego zarządzania kolejką, która do wyznaczania wartości prawdopodobieństwa odrzucenia pakietu wykorzystuje odpowiedź z trzech różnych kontrolerów PI^{α}:

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

gdzie:
p_1 - odpowiedź pierwszego kontrolera,
p_2 - odpowiedź drugiego kontrolera,
p_3 - odpowiedź trzeciego kontrolera.
Uwzględnienie odpowiedzi z kolejnych kontrolerów przy wyliczaniu wartości funkcji prawdopodobieństwa odrzucenia pakietu, pozwala na jego adaptacyjne modyfikowanie, w sytuacji gdy zanotowana zajętość bufora przekracza kluczowe progi. W zaproponowanej formule wykorzystywana jest odpowiedź wyłącznie z jednego kontrolera, w sytuacji gdy zajętość kolejki mieści się w granicach pierwszego segmentu. Gdy wartość ta znajduje się w drugim segmencie, wtedy funkcja prawdopodobieństwa odrzucenia pakietu jest sumą odpowiedzi pierwszego i drugiego kontrolera. Z kolei w ostatnim - trzecim segmencie, sumowana jest odpowiedź z wszystkich trzech kontrolerów. Wartości stanowiące granicę poszczególnych segmentów wyznaczono empirycznie, na drodze przeprowadzonych eksperymentów badawczych. Wyniki zaproponowanego rozwiązania, bazującego na powyższych regułach, zestawiono z rezultatami uzyskanymi z wykorzystaniem standardowego mechanizmu AQM, który wykorzystuje pojedynczy kontroler PI^{α} do zarządzania transmisją.

Przeprowadzenie analizy uzyskanych wyników pozwoliło na sformułowanie kilku wniosków potwierdzających słuszność ścieżki obranej do badań. Po pierwsze, nawet w przypadku bardzo mocno obciążonej sieci, nie odnotowano odrzucania pakietów w wyniku przepełnienia bufora, niezależnie od stopnia samopodobieństwa ruchu, co jest kluczowe z punktu widzenia płynnej transmisji oraz zapobiegania problemom związanych z globalną synchronizacją w sieci Internet. Po drugie, zajętości kolejki bufora osiągają wszystkie wyznaczone segmenty, co z kolei wyraźnie wpływa na prędkość transmisji. Dzieję się tak, pomimo, że to dwa pierwsze kontrolery odpowiadają za odrzucanie większości pakietów. Z drugiej strony w przypadku bardzo zmiennego ruchu, warto zwrócić uwagę na niewielką reakcję pierwszego i drugiego kontrolera PI^{α}. W przypadku ruchu cechującego się bardzo dużym stopniem zależności długoterminowych ($H = 0.9$), najwięcej pakietów odrzucanych jest przez funkcję opartą o sumę wszystkich trzech kontrolerów PI^{α}. Zjawisko to spowodowane jest dużą zmiennością zajętości kolejki, która zwiększa się wraz ze wzrostem parametru Hursta.

Otrzymane rezultaty dowodzą tezie, stanowiącej że zastosowanie adaptacyjnego modelu mechanizmu AQM, reagującego na zmiany zajętości kolejki bufora, które wynikają ze zmiennej intensywności ruchu oraz cech jego samopodobieństwa, pozwala zwiększyć wydajność transmisji w sieci Internet. Badania potwierdziły istotny wpływ stopnia samopodobieństwa ruchu, wyrażonego za pomocą

parametru Hursta, na otrzymywane wartości średniej zajętości kolejki oraz liczbę odrzuconych pakietów.

Podczas gdy w rozpatrywanych przypadkach natężenie ruchu sieciowego było bardzo duże, większość pakietów odrzucana była z powodu przekroczenia maksymalnego rozmiaru kolejki. Natomiast wykorzystanie wyłącznie trzeciego kontrolera powodowało zbyt gwałtowną reakcję, która skutkowała odrzucaniem zbyt dużej liczby pakietów. Podobnie zresztą, jak w przypadku średniego oraz niskiego obciążenia sieci, które to przypadki również charakteryzowały się stosunkowo dużymi stratami oraz słabym wykorzystaniem pasma transmisyjnego. Z drugiej strony adaptacyjny charakter zaprezentowanego modelu pozwolił dostosować sposób zarządzania buforem transmisyjnym w zależności od aktualnie napotkanych warunków w sieci Internet, umożliwił wyeliminowanie strat wynikających z przepełnienia kolejki oraz stworzył możliwość optymalnego wykorzystania dostępnego pasma transmisyjnego.

Wkład autorski [60%] polegał na:

- współudziale w przeglądzie literatury oraz analizie stanu wiedzy,
- zaproponowaniu mechanizmu aktywnego zarządzania kolejką z funkcją prawdopodobieństwa odrzucenia pakietu w oparciu o odpowiedź z kilku kontrolerów PI^{α},
- implementacji modelu wykorzystanego w badaniach,
- przeprowadzeniu eksperymentów badawczych,
- analizie uzyskanych wyników oraz sformułowaniu wniosków,
- pełnieniu funkcji autora korespondującego.

3.2 Publikacja 2: Dobór parametrów mechanizmu AQM w oparciu o techniki uczenia maszynowego oraz uczenia przez wzmocnienie

Celem badań przedstawionych w publikacji ”AQM mechanism with neuron tuning parameters” było zaproponowanie rozwiązania decyzyjnego opartego

o mechanizmy aktywnego zarządzania kolejką, mającego za zadanie kontrolowanie przeciążeń w sieci Internet. Zaplanowane badania miały wykazać, czy monitorowanie natężenia transmisji danych wraz z zastosowaniem aktywnego adaptacyjnego doboru parametrów wybranej modyfikacji algorytmu aktywnego zarządzania kolejką, pozwoli wydatnie zwiększyć wpływy pakietów do węzła, a co za tym idzie poprawić przepustowość transmisji. W niniejszej pracy zaprezentowano algorytm modyfikujący współczynniki regulatora niecałkowitego rzędu PI^{α}. Zaimplementowane rozwiązanie zostało wykorzystane do stworzenia adaptacyjnego mechanizmu aktywnego zarządzania pakietami w węźle komunikacyjnym. Odpowiedź p regulatora PI^{α} wykorzystywana do obliczania prawdopodobieństwa prewencyjnego odrzucenia pakietu, wyznaczana jest za pomocą poniższej formuły:

$$p_i = max\{0, -(K_P E_k + K_I \Delta^{\alpha} E_k)\} \tag{3.1}$$

gdzie: K_P oraz K_I to odpowiednio proporcjonalny i całkujący człon kontrolera, E_k to uchyb regulatora, czyli różnica pomiędzy aktualnym rozmiarem kolejki Q_k, a wartością oczekiwaną Q.
Zaproponowany mechanizm aktywnego zarządzania pakietami wykorzystuje neuron do doboru parametrów kontrolera, w zależności od aktualnych warunków ruchu sieciowego oraz jego natężenia. Sposób uczenia neuronu i wyznaczania wartości parametrów kontrolera przebiega w sposób następujący [70]:

$$K_P(t) = k_1 \frac{w_1(t) w_4(t)}{\sum_{i+1}^{n} w_i(t)} \tag{3.2}$$

$$K_I(t) = k_2 \frac{w_2(t) w_5(t)}{\sum_{i+1}^{n} w_i(t)} \tag{3.3}$$

gdzie: k_1 i k_2 są stałymi proporcjonalnymi współczynnikami.
W części eksperymentalnej przeprowadzono badania z wykorzystaniem różnych wartości parametru α. Intensywność źródła pakietów w symulacji przyjęła wartość stałą ($\lambda = 0.5$). Natomiast współczynnik czasu obsługi pakietu w kolejce ustalono kolejno dla μ: 0.25 - w celu uzyskania systemu mocno obciążonego, 0.5 - systemu średnio obciążonego oraz 0.75 - systemu słabo obciążonego. Badania przeprowadzano z uwzględnieniem czterech wartości parametru Hursta: $H = 0.5; 0.7; 0.8; 0.9$. Brano więc pod uwagę zarówno ruch niesamopodobny (gdy $H = 0.5$), jak również ruch cechujący się bardzo dużym stopniem zależności długoterminowych ($H = 0.9$). Ponadto zaproponowany mechanizm porównano z algorytmem Adaptive Neuron AQM, by określić jego zdolności adaptacyjne do

dostosowania się do aktualnych warunków w transmisji. Wcześniejsze prace pokazały, że mechanizm kontrolera PI^{α} bardzo dobrze nadaje się jako regulator średniej zajętości kolejki. Dość dużym problemem, podobnie jak w innych tego typu algorytmach jest prawidłowy dobór parametrów. Decyzja ta w znacznym stopniu może zależeć od właściwości ruchu sieciowego, takich jak intensywność ruchu czy zależności długoterminowe. Wyniki przeprowadzonych eksperymentów badawczych, w tym wpływ stopnia samopodobieństwa ruchu sieciowego na średnią długość kolejki, średni czas oczekiwania pakietu w buforze, liczbę pakietów odrzuconych przez zastosowany kontroler PI^{α}, a także zmianę wartości parametrów K_P oraz K_I przez zaimplementowany neuron przedstawiono za pomocą tabel i wykresów. Analiza uzyskanych rezultatów dowodzi, że możliwy jest adaptacyjny dobór parametrów kontrolera z wykorzystaniem technik uczenia maszynowego, sieci neuronowych oraz uczenia przez wzmocnienie. Zaproponowana metoda w odpowiedni sposób reguluje średnią zajętość kolejki, która w przeprowadzonych eksperymentach zbliżona była do wartości oczekiwanej. Z drugiej strony badania wykazały istotny wpływ stopnia samopodobieństwa ruchu, który wyrażono za pomocą parametru Hursta, na liczbę odrzuconych pakietów oraz średnią długość kolejki. Uzyskane wyniki są więc ściśle związane ze stopniem samopodobieństwa. W przypadku ruchu cechującego się dużym stopniem zależności długoterminowych (gdy $H = 0.9$), liczba odrzuconych pakietów znacznie wzrasta. To z kolei sprawia, że odnotowana wartość średniej długości kolejki jest poniżej zakładanego poziomu, czyli zmniejsza się wykorzystanie dostępnego łącza transmisji w sieci Internet. Rezultaty prac badawczych zawarte w niniejszej publikacji nagrodzone zostały wyróżnieniem: ”BEST ICxS’2020 PAPER AWARD” podczas międzynarodowej konferencji naukowej Intelligent Information and Database Systems (ACIIDS 2020, 23-26 marzec 2020 Phuket, Tajlandia).

Wkład autorski [65%] związany był z:

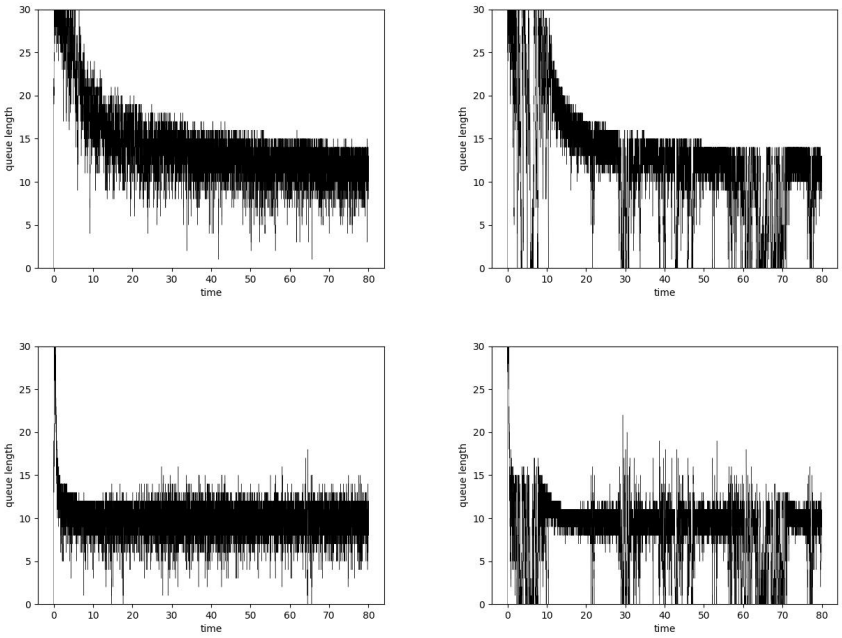
- współudziałem w analizie stanu wiedzy i wykonaniem przeglądu literatury,
- zaproponowaniem mechanizmu aktywnego zarządzania kolejką z wykorzystaniem technik uczenia maszynowego, w tym uczenia przez wzmocnienie,
- implementacją modelu wykorzystanego w eksperymentach symulacyjnych,
- współudziałem w przeprowadzonych badaniach modelu aktywnego zarządzania kolejką,
- analizą rezultatów, opracowaniem wniosków oraz zadaniami autora korespondującego.

3.3 Publikacja 3: Adaptacyjny mechanizm AQM dostosowujący się do intensywności ruchu oraz stopnia samopodobieństwa

W ramach prac przedstawionych w artykule "Adaptive Hurst-Sensitive Active Queue Management" zaproponowałem modyfikację podstawowego algorytmu Adaptive RED oraz algorytmów opartych o sieci neuronowe, które dostosowują się do intensywności oraz dodatkowo do stopnia samopodobieństwa strumienia sieciowego, wyrażonego za pomocą parametru Hursta. Metodologia badań zawarta w niniejszej publikacji stanowi rozwinięcie podejścia przedstawionego w pracy "AQM mechanism with neuron tuning parameters". W związku z tym, że operacje związane z obliczeniem wartości parametru Hursta są złożone obliczeniowo, a najpopularniejsze metody obliczania tego parametru są zbyt wolne aby można je było zastosować w rzeczywistym routerze, niniejszy artykuł przedstawia również modyfikację metody zagregowanej wariancji. Wykorzystuje ona pewne uproszczenia matematyczne, które pozwalają na wykonanie części obliczeń w tle. Wiadomości o każdym przychodzącym pakiecie gromadzone są w specjalnej strukturze, która to przechowuje informacje o liczbie pakietów w różnych skalach czasowych. Taki sposób wstępnego przetworzenia danych znacznie przyspiesza proces obliczania wartości parametru Hursta.

Zgodnie z założeniami przedstawionymi we wstępie niniejszej dysertacji, do oceny współpracy mechanizmów aktywnego zarządzania kolejką z protokołem TCP wykorzystano aproksymację Fluid-Flow (model pętli zamkniętej, ang. closed loop). Z drugiej strony dzięki zastosowaniu modelu pętli otwartej (ang. open loop), bazującego na metodach symulacyjnych, możliwe stało się dokładne przeanalizowanie specyfiki zachowania kolejki routera w przypadku ruchu o różnym natężeniu i stopniu samopodobieństwa. Do fazy badań eksperymentalnych wybrano następujące mechanizmy aktywnego zarządzania kolejką: ARED, ANRED, a także algorytmy niecałkowitego rzędu PI^{α} oraz $PI^{\alpha}D^{\beta}$. Dla wszystkich powyżej przytoczonych kontrolerów, zaproponowano nowe - analogiczne modele, które za pomocą struktur opartych o sieci neuronowe automatycznie dostosowują parametry mechanizmu aktywnego zarządzania kolejką, w zależności od intensywności ruchu oraz stopnia jego samopodobieństwa. Po zaimplementowaniu zaproponowanych modeli oraz przeprowadzeniu eksperymentów badawczych, uzyskano znaczącą poprawę parametrów transmisji, w porównaniu do analogicznych eksperymentów przeprowadzanych dla klasycznych rozwiązań. Wspomniana optymalizacja wyrażona była poprzez zmniejszenie wartości opóźnień transferu oraz

redukcji średniej zajętości kolejki, a także poprzez spadek sumarycznej liczby odrzuconych pakietów. Szczegółowa analiza uzyskanych wyników numerycznych pozwoliła zaobserwować zmniejszenie wartości opóźnień między 11.8% a 18.7%, w przypadku ruchu niesamopodobnego, bądź też ruchu o niskim stopniu zależności długoterminowych (odpowiednio dla wartości parametru Hursta = 0.5 oraz 0.6). Z drugiej strony dla ruchu cechującego się wysokim stopniem samopodobieństwa (dla parametru Hursta = 0.9) zanotowano spadek opóźnień między 14% a 16.1%. Podobnie do zanotowanych opóźnień zmieniała się również średnia zajętość kolejki. Wartości te w przypadku nowo zaproponowanego podejścia ulegały zmniejszeniu między 2.7% a 29%, gdy wartość parametru Hursta = 0.5 oraz między 11.8% a 18.7%, w przypadku gdy parametr Hursta = 0.9.



Rysunek 3.1: Wartości zajętości kolejki, μ = 0.25 - w przypadku zastosowania klasycznego kontrolera $PI^{\alpha}D^{\beta}$, $\alpha = 0.5$, H = 0.5 (**lewy**, **górny**), H = 0.9 (**prawy**, **górny**) oraz uzyskane rezultaty dla zaproponowanego modelu kontrolera $PI^{\alpha}D^{\beta}$ dostosowującego się do intensywności oraz stopnia samopodobieństwa ruchu, $\alpha = 0.5$, H = 0.5 (**lewy**, **dolny**), H = 0.9 (**prawy**, **dolny**).

Najbardziej znaczące zyski optymalizacyjne zaobserwowano jednak w przypadku redukcji liczby retransmitowanych pakietów. Dla kontrolerów niecałkowi-

tego rzędu PI^{α} oraz $PI^{\alpha}D^{\beta}$ sumaryczna liczba odrzuconych pakietów na skutek przepełnienia kolejki zmniejszała się między 86% a 92% w przypadku ruchu bez zależności długoterminowych. Natomiast dla ruchu cechującego się dużym stopniem zależności długoterminowych uzyskana optymalizacja mieściła się w zakresie od 81.5% do 85.5%. Na rysunku 3.1 przedstawiono szczegółowe porównanie uzyskanych rezultatów zajętości kolejki w przypadku zastosowania klasycznego kontrolera $PI^{\alpha}D^{\beta}$ oraz dla zaproponowanego modelu kontrolera $PI^{\alpha}D^{\beta}$, dostosowującego się do intensywności oraz stopnia samopodobieństwa ruchu. Z drugiej strony warto zwrócić uwagę, że w przypadku mechanizmu ANRED wdrożenie modyfikacji pozwalającej na dostosowanie się do stopnia samopodobieństwa ruchu sieciowego nie wpłynęło znacząco na uzyskane wyniki. Mechanizm ten bowiem w każdym z rozpatrywanych przypadków cechował się zbyt dużą intensywnością reagowania na napotkane zmiany. Rezultaty niniejszej publikacji dowodzą, że stworzony mechanizm dostosowujący się do zmiennego stopnia samopodobieństwa ruchu pozwala znacznie poprawić efektywność transmisji sieciowej.

Wkład autorski [50%] w powstanie tego artykułu polegał na:

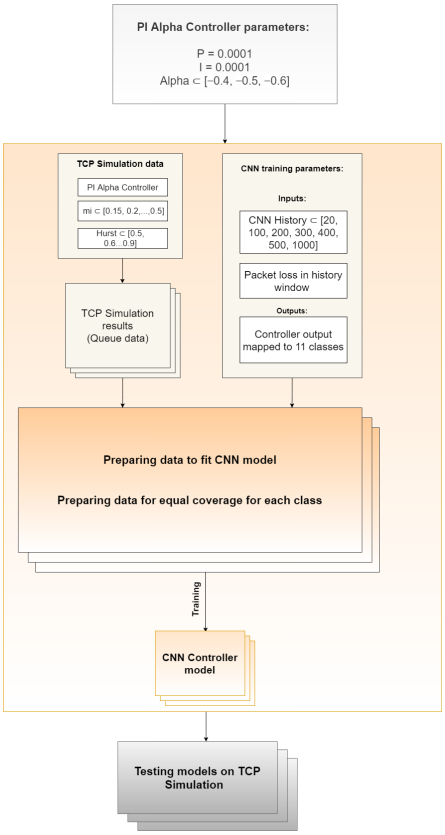
- współudziale w analizie stanu wiedzy i wykonaniu przeglądu literatury,
- zaproponowaniu i zaimplementowaniu modelu AQM, który dostosowuje się do stopnia samopodobieństwa oraz do intensywności ruchu,
- współudziale w przeprowadzaniu eksperymentów badawczych i analizie otrzymanych wyników,
- wykorzystaniu aproksymacji Fluid-Flow do oceny współpracy zaproponowanych mechanizmów AQM z protokołem TCP,
- współudziale w sformułowaniu wniosków i pełnieniu roli autora korespondującego.

3.4 Publikacja 4: Adaptacyjny mechanizm aktywnego zarządzania kolejką oparty o metody uczenia nadzorowanego

Publikacja "Supervised Learning of Neural Networks for Active Queue Management in the Internet" przedstawia model mechanizmu aktywnego zarządzania kolejką, który stworzono z wykorzystaniem sieci neuronowych oraz metod uczenia nadzorowanego. W celu przeprowadzenia procesu nauki wybranych struktur sieci konwolucyjnych wygenerowano dane uzyskane poprzez wykonanie eksperymentów symulacyjnych. Współczynnik natężenia ruchu wejściowego ustalono jako stałą wartość $\lambda = 0.5$. Źródło pakietów symulacji posiada więc zawsze stałą intensywność. Natomiast wartość parametru μ reprezentującego prawdopodobieństwo pobrania pakietu z kolejki wielokrotnie zmieniano. Przyjmował on wartości z zakresu od $\mu = 0.5$, dla systemu średnio obciążonego, do $\mu = 0.15$, czyli systemu bardzo mocno obciążonego. Dla każdego z wyżej wymienionych przypadków uwzględniono również zmienny stopień samopodobieństwa ruchu. Wartości te zmieniano od $H = 0.5$, co reprezentowało ruch niesamopodobny, do $H = 0.9$, czyli ruch cechujący się bardzo dużym stopniem zależności długoterminowych. Opisana w ten sposób charakterystyka ruchu sieciowego z wykorzystaniem metod symulacyjnych pozwoliła odwzorować wszystkie cechy wybranego mechanizmu aktywnego zarządzania kolejką. W tym przypadku zdecydowano się z kolei na kontroler niecałkowitego rzędu PI^{α}. Wartości parametrów kontrolera niecałkowitego rzędu PI^{α} przedstawiono w tabeli (3.1). Wartości te ustalono na podstawie wcześniejszych prac [71]. Jak wykazały wyniki wcześniej przeprowadzonych badań, dobór parametrów kontrolera znacząco wpływa na właściwości regulowania długości kolejki. Parametry kontrolerów dobrano w ten sposób, że kontroler $PI^{\alpha}1$ jest kontrolerem najsłabszym, a kontroler $PI^{\alpha}3$ jest mechanizmem najmocniejszym, co skutkuje dużą liczbę odrzuconych pakietów.

	K_P	K_I	α
$PI^{\alpha}1$	0.0001	0.0004	-0.4
$PI^{\alpha}2$	0.0001	0.0004	-0.5
$PI^{\alpha}3$	0.0001	0.0004	-0.6

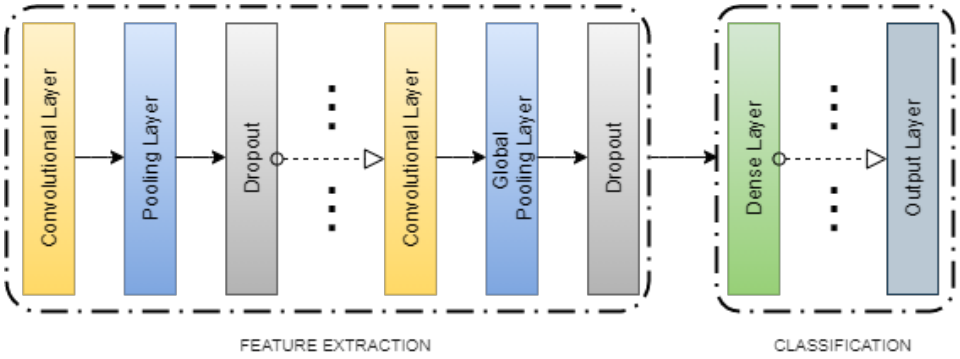
Tablica 3.1: Wartości parametrów kontrolera niecałkowitego rzędu PI^{α}



Rysunek 3.2: Schemat przedstawiający sposób przetwarzania danych w celu przygotowania zbioru uczącego.

Dane uczące sieci neuronowej składają się z:

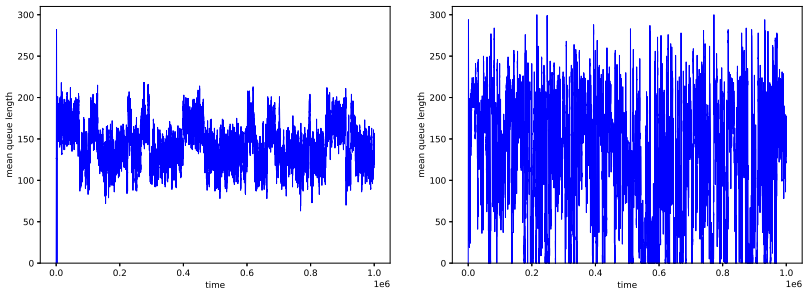
1. Danych wejściowych
   (a) Ostatnich n elementów z historii zajętości kolejki
   (b) Historii odrzucania pakietów w n ostatnich stanach kolejki, gdzie $n \in [20; 100; 200; 300; 400; 500; 1000]$
2. Danych wyjściowych
   (a) 11 etykiet mapujących prawdopodobieństwo odrzucenia pakietu, w zależności od aktualnie występujących warunków w transmisji



Rysunek 3.3: Koncepcja struktury modelu konwolucyjnej sieci neuronowej.

Dane uczące modelu konwolucyjnych sieci neuronowych odzwierciedlają zachowanie mechanizmu AQM. W przypadku każdego przedziału prawdopodobieństwa, przygotowano milion jednowymiarowych rekordów z danymi uczącymi. Cały zbiór uczący składa się więc z 11 milionów rekordów. Zawarte są w nich informacje dotyczące transmisji, takie jak długość kolejki w każdym kolejnym slocie czasowym, liczba pakietów odrzuconych oraz wartość funkcji prawdopodobieństwa odrzucenia pakietu przez mechanizm aktywnego zarządzania kolejką. Metody wykorzystane w celu przetworzenia danych i przygotowania zbioru uczącego przedstawiono na rysunku (3.2). W czasie badań rozpatrywano sieci neuronowe z różną liczbą warstw konwolucyjnych, a także stosowano różne optymalizatory oraz funkcje kosztu, w celu optymalizacji tworzonego modelu AQM. Na podjęte w niniejszej pracy decyzje wpłynęły też wcześniejsze doświadczenia z prac nad klasyfikowaniem ruchu sieciowego [50], które to również potwierdziły stosowność wykorzystania konwolucyjnych sieci neuronowych w rozpatrywanej w tej pracy tematyce. Ostatecznie po przeprowadzeniu badań empirycznych zdecydowano na oparcie modelu o cztery warstwy konwolucyjne oraz dwie warstwy gęste, w których to zastosowano funkcję Relu jako funkcję aktywacji. Jest ona stosowana do obliczania wartości wyjścia z neuronu. Z drugiej strony jako funkcję aktywacji ostatniej warstwy wybrano Sigmoid, w przypadku której uzyskiwano największą wartość wskaźnika precyzji uczenia. Jest to wybór o tyle interesujący, gdyż pomimo faktu, że w literaturze [72] częściej spotykana jest funkcja Softmax w ostatniej warstwie sieci neuronowej, to w rozpatrywanym przypadku lepiej sprawdza się właśnie funkcja Sigmoid. W najgorszym przypadku, gdy sieć która uzyskiwała precyzję uczenia na poziomie 32,31%, po zmianie funkcji akty-

wacji na Sigmoid wartość ta zwiększała się do 65,65%. Natomiast w najlepszym uzyskanym przypadku wskaźnik precyzji uczenia wzrastał z 58,90% do 86,65%. Funkcja kosztu zaprojektowanej sieci neuronowej to categorical cross-entropy. Natomiast w celu wytrenowania modelu i zminimalizowania wpływu funkcji kosztu, wykorzystano optymalizator Adaptive Moment Estimation (Adam). Rysunek (3.3) ilustruje strukturę sieci neuronowej zaproponowanej jako model mechanizmu aktywnego zarządzania kolejką.
Proces uczenia przygotowanych modeli trwał 10 epok. Była to wartość zupełnie wystarczająca, gdyż wskaźniki wartości precyzji uczenia zaczynały stabilizować się już po 5-6 epokach. W trakcie badań szczególną uwagę zwrócono na wpływ stopnia samopodobieństwa ruchu oraz zależności długoterminowych na wydajność zaproponowanego mechanizmu. Bardzo ważnym aspektem było również wyselekcjonowanie liczby ostatnich n elementów historii zajętości kolejki, które stosowane są jako dane wejściowe zaproponowanego modelu. Zagadnienie to jest szczególnie istotne z punktu widzenia wydajności routera, które jest urządzeniem o niewielkich zasobach. Z tego powodu każda możliwość zmniejszenia rozpatrywanej liczby elementów historii jest sytuacją bardzo korzystną. Po przygotowaniu danych uczących, stworzeniu struktury sieci neuronowej oraz wytrenowaniu modelu, przystąpiono do fazy eksperymentów badawczych mających na celu wykorzystanie zaproponowanego adaptacyjnego modelu mechanizmu aktywnego zarządzania kolejką w warunkach symulacyjnych, a następnie porównania otrzymanych rezultatów z wynikami uzyskiwanymi w przypadku klasycznego kontrolera. W czasie trwania wspomnianego etapu ewaluacji oceniano cztery odrębne modele sieci neuronowych. Pierwsze trzy modele sieci neuronowych wyuczono zgodnie z zachowaniem kontrolerów $PI^{\alpha}1$, $PI^{\alpha}2$ i $PI^{\alpha}3$. Czwarty model wytrenowano danymi reprezentującymi zachowanie trzech kontrolerów równocześnie. Szczegółowa analiza wyników z przeprowadzonych eksperymentów badawczych pozwala stwierdzić, że dla wszystkich trzech modeli aktywnego zarządzania kolejką opartych o sieci neuronowe, uzyskano lepsze rezultaty, a co za tym idzie bardziej efektywny mechanizm od standardowych rozwiązań. Nawet w przypadku modelu, który osiągnął najmniejszą wartość precyzji uczenia się - uzyskana w eksperymentach średnia długość kolejki jest większa niż dla klasycznego mechanizmu $PI^{\alpha}3$. Sytuacja taka zachodzi zarówno dla ruchu niesamopodobnego (gdy wartość parametru $H = 0.5$), jak również w przypadku ruchu cechującego się dużym stopniem zależności długoterminowych. Jednak jako najbardziej efektywny mechanizm aktywnego zarządzania kolejką wybrano model sieci konwolucyjnych, wyuczony zachowaniem wszystkich trzech kontrolerów PI^{α}. Kolejnym istotnym wnioskiem otrzymanym po analizie uzyskanych wyników jest również fakt, że



Rysunek 3.4: Rezultaty zajętości kolejki uzyskane dla modelu wyuczonego z wykorzystaniem danych z trzech kontrolerów PI^{α}, dla $n = 500$ elementów historii zajętości kolejki, $H = 0.5$ (lewy), $H = 0.9$ (prawy).

wraz ze wzrostem stosowanej liczby elementów historii zajętości kolejki (n), zwiększa się również zdolność adaptacji mechanizmu do aktualnych warunków transmisji w sieci Internet. Najlepsze wyniki uzyskiwano gdy wartość n wynosiła 500 (rysunek 3.4). Z drugiej strony, minimalna akceptowalna liczba zastosowanych elementów historii zajętości kolejki, dla której uzyskano dopuszczalne rezultaty, to $n = 100$. Wyniki przeprowadzonych badań potwierdziły, że model oparty o konwolucyjne sieci neuronowe potrafi skutecznie odwzorować wyniki klasycznego algorytmu AQM oraz efektywnie zarządzać transmisją danych. Utrzymuje on założoną średnią liczbę pakietów w kolejce oraz zmniejsza całkowitą liczbę pakietów odrzuconych, niezależenie od stopnia samopodobieństwa ruchu.

Wkład autorski [65%] w powstanie tego artykułu związany był z:

- współudziałem w przeglądzie literatury,
- zaproponowaniem modelu mechanizmu aktywnego zarządzania kolejką w oparciu o wykorzystanie technik uczenia nadzorowanego i sieci neuronowych,
- zaimplementowaniem modelu wykorzystanego w badaniach,
- współudziałem w procesie przygotowania zbioru uczącego,
- przeprowadzeniem eksperymentów badawczych,
- analizą rezultatów i sformułowaniem wniosków,
- pełnieniem funkcji autora korespondującego.

3.5 Publikacja 5: Model mechanizmu AQM dla priorytetowych danych ruchu urządzeń Internetu Rzeczy

Publikacja "The IoT gateway with active queue management" przedstawia adaptacyjny mechanizm aktywnego zarządzania kolejką, który pozwala wyeliminować opóźnienia w przypadku transmisji priorytetowych pakietów ruchu urządzeń Internetu Rzeczy (ang. Internet of Things). Model ten umożliwia transfer zarówno priorytetowych, jak również wszystkich pozostałych pakietów z wykorzystaniem jednego bufora transmisyjnego. Ma więc stanowić ekonomiczne i wartościowe rozwiązanie, zwłaszcza z punktu widzenia niezwykle dynamicznie rozwijającego się rynku urządzeń Internetu Rzeczy [74, 75]. Stworzony mechanizm dedykowany jest dla bramy urządzeń Internetu Rzeczy, gdzie poprzez zastosowanie odpowiednich metod umożliwiających oznaczenie pakietów priorytetowych (odpowiednio pole Type of Service w nagłówku IPv4 oraz Traffic Class w IPv6), a następnie za pomocą technik klasyfikacji danych, umożliwić ma spełnienie rygorystycznych wymogów jakości usług transmisji w sieci Internet (ang. Quality of Service). By lepiej zilustrować zasady zaproponowanej koncepcji, można przyjąć, że zadaniem stworzonego mechanizmu jest zarządzanie pakietami o standardowym poziomie uprzywilejowania w taki sposób, by pasmo transmisyjne zawsze zapewniało możliwość przesyłu pakietów priorytetowych zgodnie z regułami FIFO. Kryterium to zapobiega ryzyku retransmisji krytycznych danych oraz wystąpienia opóźnień w transmisji, a przez to pozwala spełnić umowy o gwarantowanym poziomie świadczenia usług między usługodawcą a klientem. W czasie przeprowadzania eksperymentów badawczych rozpatrywano trzy różnie zestrojone mechanizmy aktywnego zarządzania kolejką, oparte o kontroler niecałkowitego rzędu PI^{α}. Jego parametry do badań dobierano w taki sposób, by mechanizm cechował się kolejno bardzo szybką, szybką oraz wolną reakcją na zmiany zachodzące w buforze. Intensywność źródła pakietów przyjęła wartość stałą ($\lambda = 0.5$). Prawdopodobieństwo pobrania pakietu z kolejki, czyli czas obsługi danych, wyrażony za pomocą parametru μ, przyjmował wartości od 0.2 do 0.8, co pozwoliło odzwierciedlić zarówno bardzo mocno, jak również bardzo słabo obciążone pasmo transmisyjne. W badaniach szczególną uwagę zwrócono na wpływ stopnia samopodobieństwa ruchu na końcowe rezultaty. Stąd wielokrotnie zmieniane wartości parametru Hursta (0.5 - 0.9) reprezentowały zarówno ruch niesamopodobny, jak również ruch cechujący się bardzo dużym stopniem zależności długoterminowych. Ponadto we wszystkich powyżej rozpatrywanych przypadkach zmieniano pro-

centowy udział pakietów priorytetowych w paśmie transmisyjnym, chcąc określić ich maksymalną dopuszczalną wartość. W tym przypadku głównym kryterium był brak strat pakietów priorytetowych. Natomiast do oceny wpływu zaproponowanego mechanizmu na pojedynczy strumień TCP wykorzystano aproksymację Fluid-Flow. Analiza uzyskanych rezultatów dowiodła, że w zależności od doboru zestawów parametrów kontrolera PI^{α}, przedstawione rozwiązanie można elastycznie dostosowywać do zmiennego udziału pakietów priorytetowych. Jednak ich maksymalna wartość wyraźnie maleje wraz ze wzrostem stopnia zależności długoterminowych ruchu. Co jednak istotne z punktu widzenia praktyczności zaproponowanego modelu, sprawdza się on nawet w przypadku bardzo mocno obciążonej sieci. Brak utraty pakietów priorytetowych, przy bardzo intensywnym odrzucaniu pozostałych danych, pozwala wysnuć stwierdzenie, że przedstawiony mechanizm może być również bardzo wartościowy w przypadku prób odpierania ataków DDoS (ang. Distributed Denial of Service), polegających na zajęciu wszystkich wolnych zasobów (a tym samym zablokowaniu dostępu do łączy), poprzez atak przeprowadzany jednocześnie z wielu komputerów [76].

Wkład autorski [60 %] polegał na:

- współudziale w analizie stanu wiedzy,
- przedstawieniu koncepcji modelu aktywnego zarządzania kolejką, który pozwala oznaczyć pakiety priorytetowe oraz zapewnić ich transmisję w krytycznym czasie,
- implementacji modelu badawczego,
- przeprowadzeniu eksperymentów badawczych,
- wykorzystaniu aproksymacji Fluid-Flow do oceny wpływu zaproponowanego modelu na pojedynczy strumień TCP,
- współudziale w analizie wyników i sformułowaniu wniosków,
- pełnieniu roli autora korespondującego.

Rozdział 4

AQM mechanism with the dropping packet function based on the answer of several PI controllers

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.

AQ1

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

P. Gaj et al. (Eds.): CN 2019, CCIS 1039, pp. 1–13, 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). AQ2

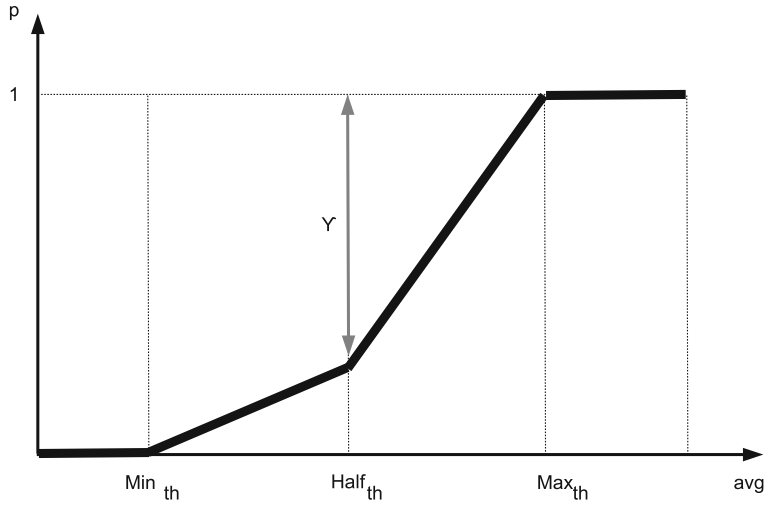


Fig. 1. The probability function of rejection the packet for the DSRED mechanism [4]

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}.

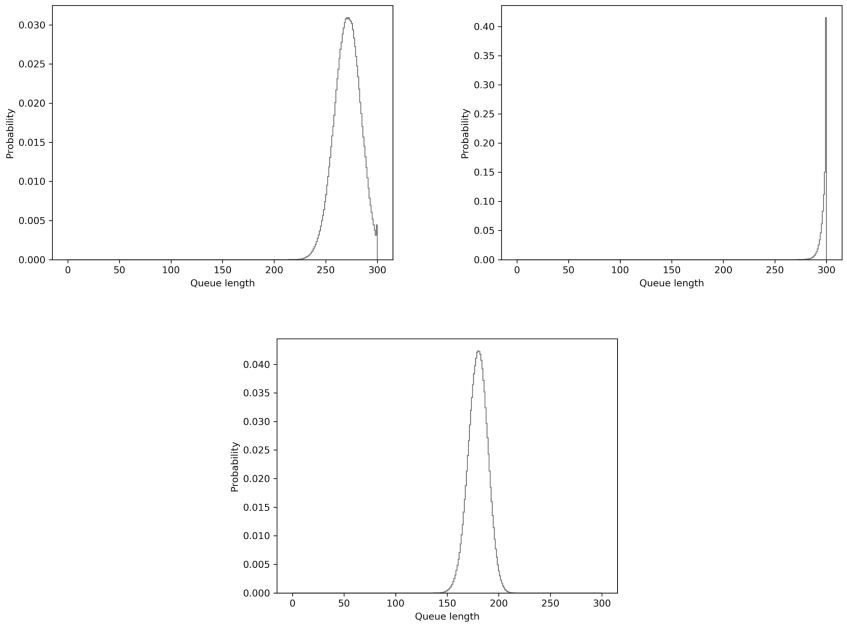


Fig. 2. Distribution of queue length for high traffic load ($\mu = 0.25$) and H = 0.5, left: $K_p = 0.0001$, $K_i = 0.00015$, $\alpha = -0.5$, right: $K_p = 0.0001$, $K_i = 0.0004$, $\alpha = -0.4$, center: $K_p = 0.0001$, $K_i = 0.00035$, $\alpha = -0.6$

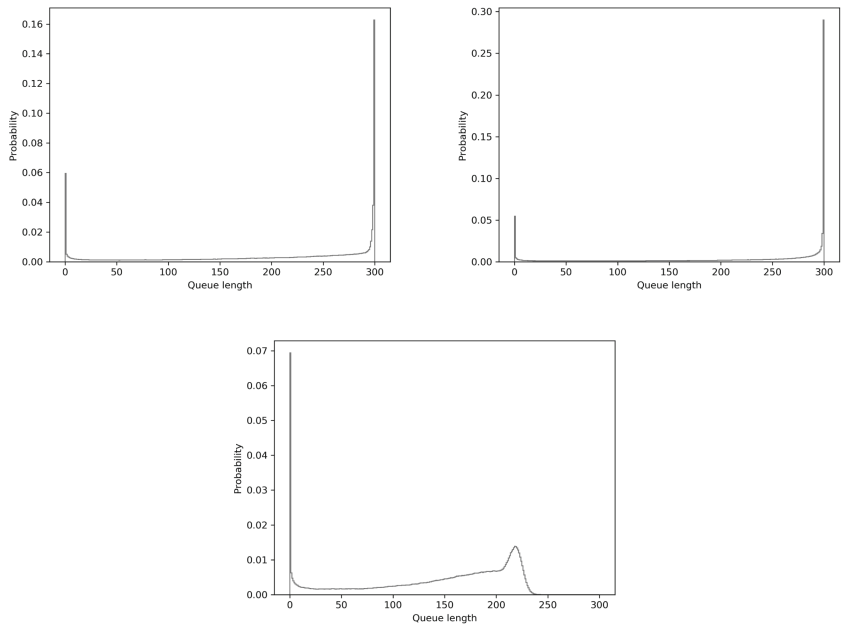


Fig. 3. Distribution of queue length for high traffic load ($\mu = 0.25$) and H = 0.9, left: $K_p = 0.0001$, $K_i = 0.00015$, $\alpha = -0.5$, right: $K_p = 0.0001$, $K_i = 0.0004$, $\alpha = -0.4$, center: $K_p = 0.0001$, $K_i = 0.00035$, $\alpha = -0.6$

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.

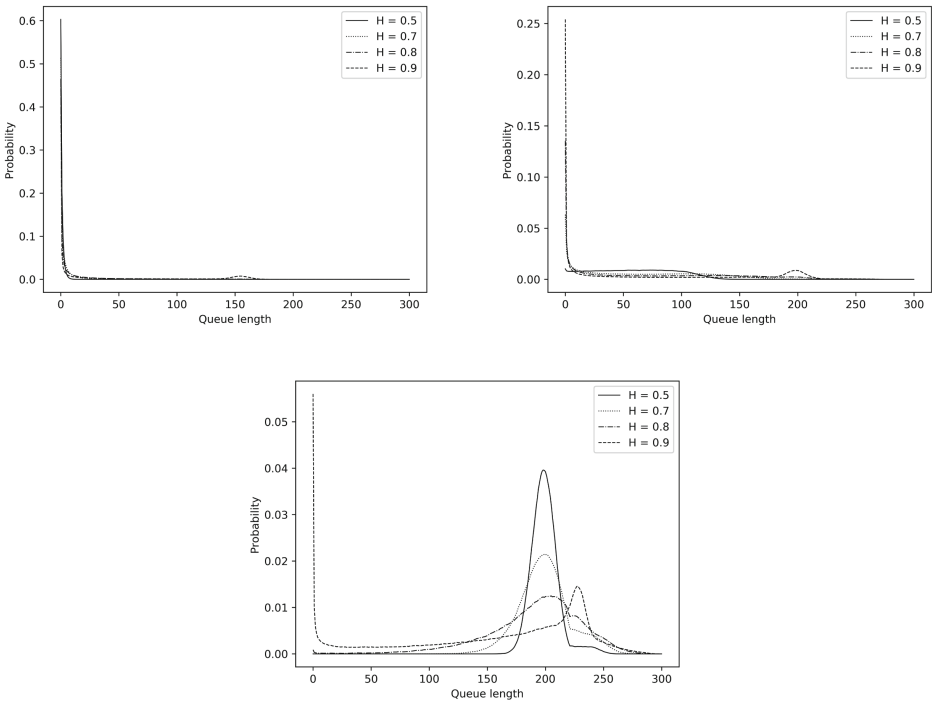


Fig. 4. The influence of degree of traffic self-similarity on queue distribution, three PI^{α} controllers, $\mu = 0.75$ (left), $\mu = 0.5$ (right), $\mu = 0.25$ (bottom)

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.

Acknowledgements. This research was partially financed by National Science Center project no. 2017/27/B/ST6/00145

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

Rozdział 5

AQM mechanism with neuron tuning parameters

AQM Mechanism with Neuron Tuning Parameters

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

[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. The congestion control is one of the most important questions in modern computer network performance. This article investigates the problem of adaptive neuron based choice of the Active Queue Mechanisms parameters. We evaluate the performance of the AQM mechanism in the presence of self-similar traffic based on the automatic selection of their parameters using the adaptive neuron. We have also proposed an AQM mechanism based on non-integer order PI^{α} controller with neuron tuning parameters and compared it with Adaptive Neuron AQM. The numerical results are obtained using the discrete event simulation model.

Keywords: AQM · Congestion control · PI^{α} controller · Neuron

1 Introduction

The rapid growth of Internet traffic due to the high demand for voice, video and data services has made the problem of congestion hard to be solved. This has led to the recommendation of an active queue management mechanism (AQM) by the Internet Engineering Task Force (IETF) [4]. This mechanism is based on early detection of the possibility of congestion and randomly drop off some packets even though the queue is not yet full. This mechanism prevents the tail dropping of packets when the buffer becomes full, which can drastically degrade the quality of service (QoS).

The AQM mechanisms, when used with the TCP congestion window mechanism in control, the traffic flows in the network enhance the efficiency of network transmission [4,27].

One of the first introduced active queue management algorithms was Random Early Detection (RED), which was proposed in 1993 by Sally Floyd and Van Jacobson [18]. This mechanism is based on the estimation of packet dropping probability, which is a function of the queue length seen by arriving packets. The argument of this function is a weighted mean queue seen by arriving packets. The function is non-decreasing between two thresholds and has the value 0 at the

N. T. Nguyen et al. (Eds.): ACIIDS 2020, LNAI 12034, pp. 299–311, 2020.
https://doi.org/10.1007/978-3-030-42058-1_25

lower threshold (and below) and P_{max} at the upper one. For higher arguments, its value is 1, i.e., all packets are rejected, The other parameter w_q is the weight of the current queue when the moving average is defined [8,18].

The efficiency of the RED mechanism and other AQM algorithms depends on the choice of the parameters P_{max} and w_q. If w_q is too small, the packets may be dropped even if the queue is still small. Also, if w_q is too large, there are high fluctuations of queue size. The articles [18,21] recommended $w_q = 0.001$ or $w_q = 0.002$ and [29] showed the efficiency of the mechanism for $w_q = 0.05$ and $w_q = 0.07$. It was recommended in [3] that the queue size could be analyzed seeing the influence of w_q on waiting time fluctuations, obviously the larger w_q, the higher fluctuations. The value of P_{max} has also a significant influence on the performance of the RED mechanism: if it is too large, the overall throughput is unnecessarily choked and if it's too small the danger of synchronization arises; [17] recommends $P_{max} = 0.1$. The problem of parameter selection was discussed in [5,22].

The numerous propositions of basic algorithms improvements appear [2,9, 20], their comparison may be found e.g. in [19]. Our previous works [6,10], also presented a study of the influence of RED modifications on its performance.

In this paper, we evaluate the performance of AQM mechanisms based on automatic selection of their parameters using the adaptive neuron in the presence of self-similar traffic. The self-similarity of a process means that a change of time scales does not influence the statistical characteristics of the process. The level of self-similarity is characterized by the Hurst parameter H, $H \in (0.5, 1)$ [1,11], the higher H, the higher degree of self-similarity. It results in long-distance autocorrelation and makes possible the occurrence of very long periods of high (or low) traffic intensity. These features have a high impact on network performance [7]. We have also proposed an AQM mechanism based on non-integer order PI^{α} controller with neuron tuning parameters and compared it with Adaptive Neuron AQM [28].

The remainder of the paper is organized as follows: Sect. 2 gives basic notions on AQM mechanisms with neuron tuning parameters and presents briefly theoretical basis for non-integer PI^{α} controller. Section 3 discusses numerical results. Some conclusions are given in Sect. 4.

2 AQM Mechanisms with Neuron Tuning Parameters

This section presents artificial intelligence algorithms used to select the proper AQM parameters. In the presented methods, the neuron's input data are mean queue length and network traffic parameters. The task of the mechanism is a selection of AQM parameters to keep the assumed mean queue length.

2.1 Adaptive Neuron AQM

The article [28] proposes a novel neuron-based AQM scheme. This solution is named Adaptive Neuron AQM (AN-AQM) and uses the single neuron to calcu-

late the probability p of packet dropping. The probability is calculated for each incoming packets and can be presented as follows:

$$p(k) = p(k-1) + \Delta p(k) \tag{1}$$

where $\Delta p(k)$ reflects changes in packet dropping probability and depends on the state of the neuron. The neuron value can be described by the following equation:

$$\Delta p(k) = K \sum_{i=1}^{n} w_i(k) x_i(k) \tag{2}$$

where K is the neuron proportional coefficient and takes positive values, $x_i(k)$ for $i = 1, 2, 3, \ldots n)$ are the neuron inputs and $w_i(k)$ is a connection weight of $x_i(k)$. The weights depend on the learning rule. For each incoming packets the algorithms calculates the error $e(k)$. The error is the difference between actual queue length $q(k)$ and the desired queue length Q. Thus:

$$e(k) = q(k) - Q. \tag{3}$$

The algorithm also defines normalized rate error $\gamma(k)$:

$$\gamma(k) = \frac{r(k)}{C} - 1, \tag{4}$$

where $r(k)$ is the input rate of the buffer at the bottleneck link, and C is the capacity of the bottleneck link.

The inputs of the neuron are: $x_1(k) = e(k) - e(k-1)$, $x_2(k) = e(k)$, $x_3(k) = e(k) - 2e(k-1) + e(k-2)$, $x_4(k) = \gamma(k) - \gamma(k-1)$, $x_5(k) = \gamma(k)$ and $x_6(k) = \gamma(k) - 2\gamma(k-1) + \gamma(k-2)$.

The learning rule of a neuron presents formula [25]:

$$w_i(k+1) = w_i(k) + d_i y_i(k) \tag{5}$$

where $d_i > 0$ is the learning rate, and $y_i(k)$ is the learning strategy. The article [25] suggests the following learning strategy:

$$y_i(k) = e(k) p(k) x_i(k). \tag{6}$$

where $e(k)$ is a teacher signal.

Such a strategy implies that an adaptive neuron self-organizes depending on signals $e(k)$ and $\gamma(k)$.

2.2 Non-integer PI^{α} Controller with Neuron Tuning Parameters

Fractional Order Derivatives and Integrals (FOD/FOI) are a natural extension of the well-known integrals and derivatives. Differintegrals of non-integer orders enable better and more precise control of physical processes. A proportional-integral controller (PI^{α} controller) is a traditional mechanism used in feedback

control systems. Earlier works show that the non-integer order controllers have better behavior than classic controllers [26].

The articles [12,14,16], describe how to use the response of the PI^{α} (non-integer integral order) to determine the response of the AQM mechanism. This response is based on the probability function and is given by the formula:

$$p_i = max\{0, -(K_P e_k + K_I \Delta^{\alpha} e_k)\} \tag{7}$$

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.

Thus, the dropping probability depends on three parameters: the coefficients for the proportional and integral terms K_p, K_i and integral order α.

In the active queue management, packet drop probabilities are determined at discrete moments of packet arrivals, so the queue model should be considered as a case of discrete systems. There is only one definition of the discreet 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 is analogous to a generalization used in Grunwald-Letnikov (GrLET) formula.

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

$$\triangle^{\alpha} f_k = \sum_{j=0}^{k} (-1)^j \binom{\alpha}{j} f_{k-j} \tag{8}$$

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

$$\binom{\alpha}{j} = \begin{cases} 1 & \text{for } j = 0 \\ \dfrac{\alpha(\alpha-1)(\alpha-2)..(\alpha-j+1)}{j!} & \text{for } j = 1, 2, \ldots \end{cases} \tag{9}$$

The neural mechanism of the selection PI^{α} controller parameters for multi-model plants was presented in the articles [23,25]. It presents an adaptation of a proposed solution to the problem of active queue management. The construction of the neuron and the teaching strategy is similar to the AN-AQM algorithm (Sect. 2.1) and the mapping of the neuron response to PI^{α} parameters is as follows [25]:

$$K_P(t) = k_1 \frac{w_1(t) w_4(t)}{\sum_{i+1}^{n} w_i(t)} \tag{10}$$

$$K_I(t) = k_2 \frac{w_2(t) w_5(t)}{\sum_{i+1}^{n} w_i(t)} \tag{11}$$

where k_1 and k_2 are the constant proportional coefficients.

3 Obtained Results

This section presents the influence of the presented mechanism on queue performance. We study the impact of the degree of self-similarity on the examined

AQM mechanisms. The tests analyzed the following parameters of the transmission with AQM: the mean length of the queue, queue waiting times and the number of rejected packets.

All presented results in this article were obtained using the simulation model. The simulations were done using the Simpy Python packet. To accelerate the calculations, a PI^{α} module was written in C language. The input traffic intensity $\lambda = 0.5$ was considered independent 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 Fractional Gaussian noise process, first introduced in [24]. We considered different queue loads. The high load was studied for parameter $\mu = 0.25$, where μ is the service intensity (mean number of packets served per time unit). The average load we obtained for $\mu = 0.5$. Small load was considered for parameter $\mu = 0.75$.

The AQM parameters were as follows:

- desired queue length $Q = 100$,
- the neuron proportional coefficient $K = 0,01$,
- the learning rate d_i oscillated between 0.00001 *and* 0.0001,
- PI constant proportional coefficient $K_P, K_I = 0.0001$,
- the non-integer integral order α, depending on the experiment, took the following values: $-0.5, -1.0, -1.8$,
- maximum queue size $Q_{max} = 300$.

The normalized rate error $\gamma(k)$ was calculated as the proportion of input traffic intensity λ and service time μ. In packet-switched networks, service time is the time required to transmit information, and:

$$\gamma(k) = \frac{\lambda(k)}{\mu(k)} - 1, \tag{12}$$

Tables 1, 2 and 3 present the obtained results for Adaptive Neuron AQM (AN-AQM). The results correspond to three types of network traffic. The Table 1 reflects the values for an overloaded network (the packet arrival speed is higher than the queue capacity). The AQM mechanism, independently of the degree of traffic self-similarity (Hurst parameter), maintains the average queue length at the assumed level. The changes in queue size over time are shown in Fig. 1. The queue length, except the initial phase, does not exceed the assumed level. The queue length oscillation increases with the Hurst parameter. Table 2 and Fig. 2 present the results in the case of an average traffic load. The average queue length does not exceed 50 packets (independently of the traffic self-similarity). However, the detailed results suggest that temporarily, the queue exceeds the desired length and the mechanism drops packets. In the case of light traffic (Table 3), the average queue lengths are small, but the number of discarded packets shows that for the higher degree of self-similarity, the queue could exceed the desired size.

In the case of AN-AQM, the neuron controls the packet dropping probability. For considered here PI^{α} controller, the neuron determines the values of the

Table 1. AN-AQM, $\mu = 0.25$, setpoint = 100

Hurst	Avg. queue length	Avg. waiting times	Packet drop by	
			AN-AQM	Queue
0.50	99.2656	39.8558	24945	0
0.70	98.3449	39.1545	24920	0
0.80	96.6901	38.2377	23727	0
0.90	68.5395	29.8168	17410	0

Table 2. AN-AQM, $\mu = 0.50$, setpoint = 100

Hurst	Avg. queue length	Avg. waiting times	Packet drop by	
			AN-AQM	Queue
0.50	47.8359	9.5532	272	0
0.70	48.3206	10.0806	2543	0
0.80	44.8137	9.7713	3585	0
0.90	33.4571	9.3997	5101	0

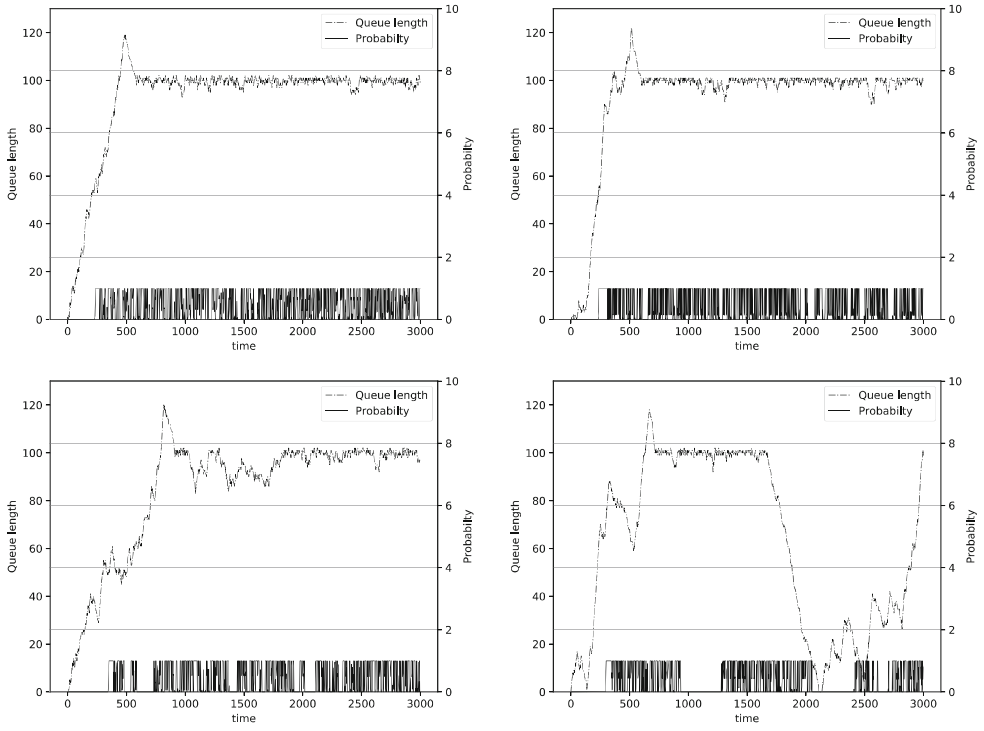


Fig. 1. Queue length and packet dropping probability, AN-AQM, $\mu = 0.25$, top: $H = 0.5$ and 0.7, bottom: $H = 0.8$ and 0.9

Table 3. AN-AQM, $\mu = 0.75$, setpoint = 100

Hurst	Avg. queue length	Avg. waiting times	Packet drop by	
			AN-AQM	Queue
0.50	1.2833	0.1575	0	0
0.70	2.9008	0.4799	0	0
0.80	5.7419	1.0730	35	0
0.90	12.7406	3.0883	360	0

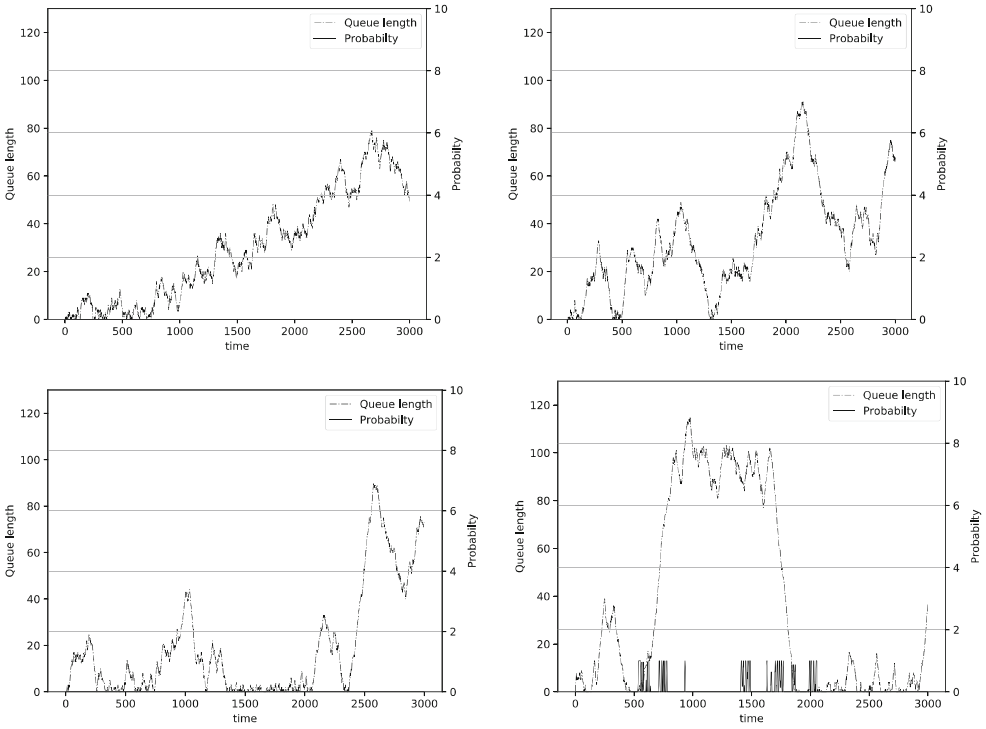


Fig. 2. Queue length and packet dropping probability, AN-AQM, $\mu = 0.50$, top: $H = 0.5$ and 0.7, bottom: $H = 0.8$ and 0.9

controller parameters, and packet dropping probability is obtained as the controller's feedback signal. The results below consider the AQM mechanism when the neuron sets only the K_P and K_I parameters. The value of integral order α is constant. In experiments, we considered different values of α parameter. The results below consider only the situation of an overloaded network ($\mu = 0.25$).

The Table 4 presents the results for standard non fractional PI^{α} controller ($\alpha = -1$). As noticed, the average queue size remains close to the desired level. However, Fig. 3 shows that the queue length varies greatly. This variability increases with the increase of the integration order (see Fig. 4). On the other hand, the oscillations decrease when the integral order decreases (Fig. 4). The comparison of the Tables 4, 5, and 6 allows us to conclude that while the integra-

tion order influences the queue evolution, it does not affect the obtained average values (Fig. 5).

The Figs. 6 and 7 display the evolution of the parameters K_I and K_P. It may be seen that regardless of the integral order α, the obtained by the algorithm K_P parameter, is much larger than the K_I. On the other hand, evolution is very similar. As the integration order decreases, the parameter changes become smoother.

Table 4. $\alpha = -1.0$, $\mu = 0.25$, setpoint = 100

Hurst	Avg. queue length	Avg. waiting times	Packet drop by	
			PI^{α}	Queue
0.50	100.8211	39.9768	49603	0
0.70	99.5037	39.6021	49838	0
0.80	97.8870	39.0255	49891	0
0.90	76.4802	32.6089	53131	0

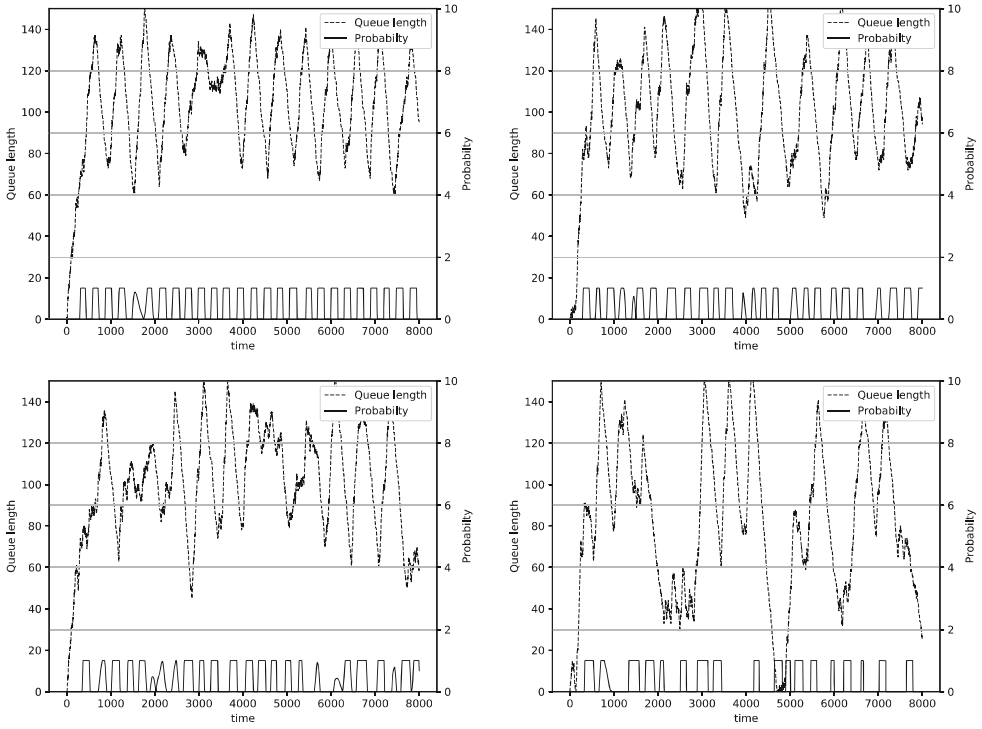


Fig. 3. Queue length and packet dropping probability, $\alpha = -1.0$, $\mu = 0.25$, top: $H = 0.5$ and 0.7, bottom: $H = 0.8$ and 0.9

Table 5. $\alpha = -1.8$, $\mu = 0.25$, setpoint = 100

Hurst	Avg. queue length	Avg. waiting times	Packet drop by	
			PI^{α}	Queue
0.50	100.1562	39.8435	49830	0
0.70	98.2007	39.3388	50188	0
0.80	95.7958	38.0802	49762	0
0.90	78.9514	34.0898	53799	0

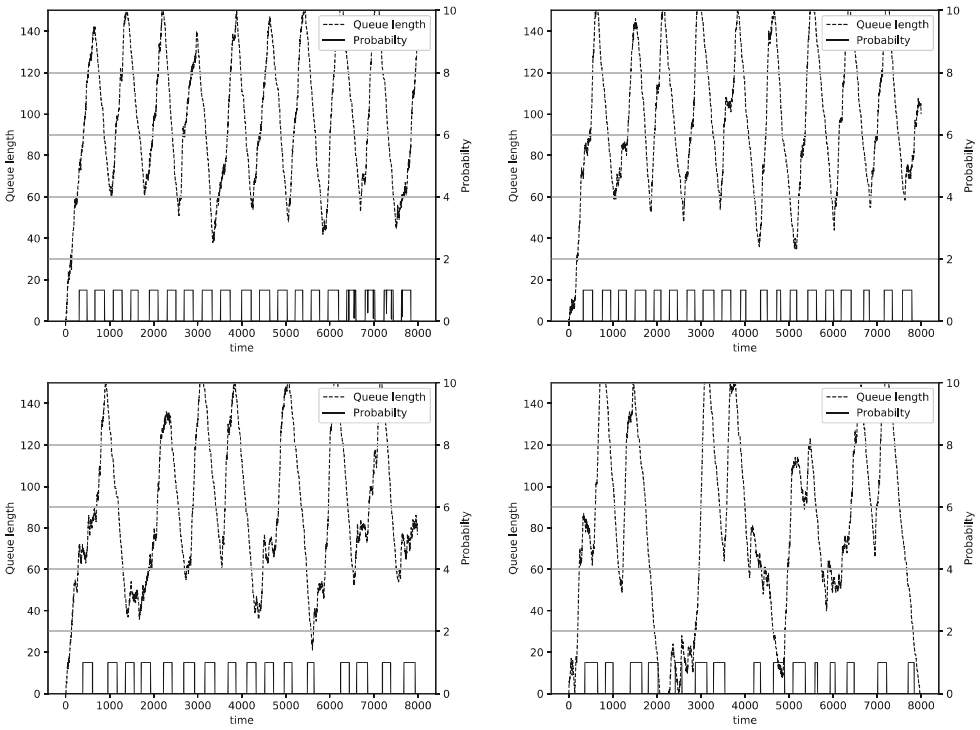


Fig. 4. Queue length and packet dropping probability, $\alpha = -1.8$, $\mu = 0.25$, top: $H = 0.5$ and 0.7, bottom: $H = 0.8$ and 0.9

Table 6. $\alpha = -0.5$, $\mu = 0.25$, setpoint = 100

Hurst	Avg. queue length	Avg. waiting times	Packet drop by	
			PI^{α}	Queue
0.50	102.8591	40.8138	49681	0
0.70	102.0629	40.6503	49855	0
0.80	101.2295	40.5846	50184	0
0.90	83.4922	35.14951	52589	0

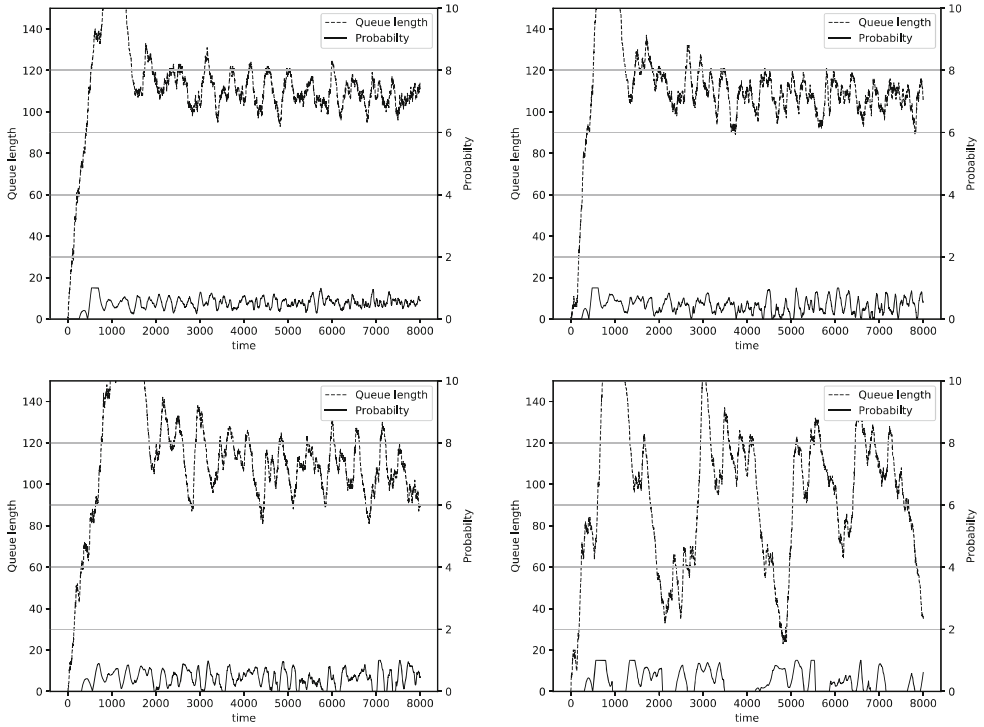


Fig. 5. Queue length and packet dropping probability, $\alpha = -0.5$, $\mu = 0.25$, top: $H = 0.5$ and 0.7, bottom: $H = 0.8$ and 0.9

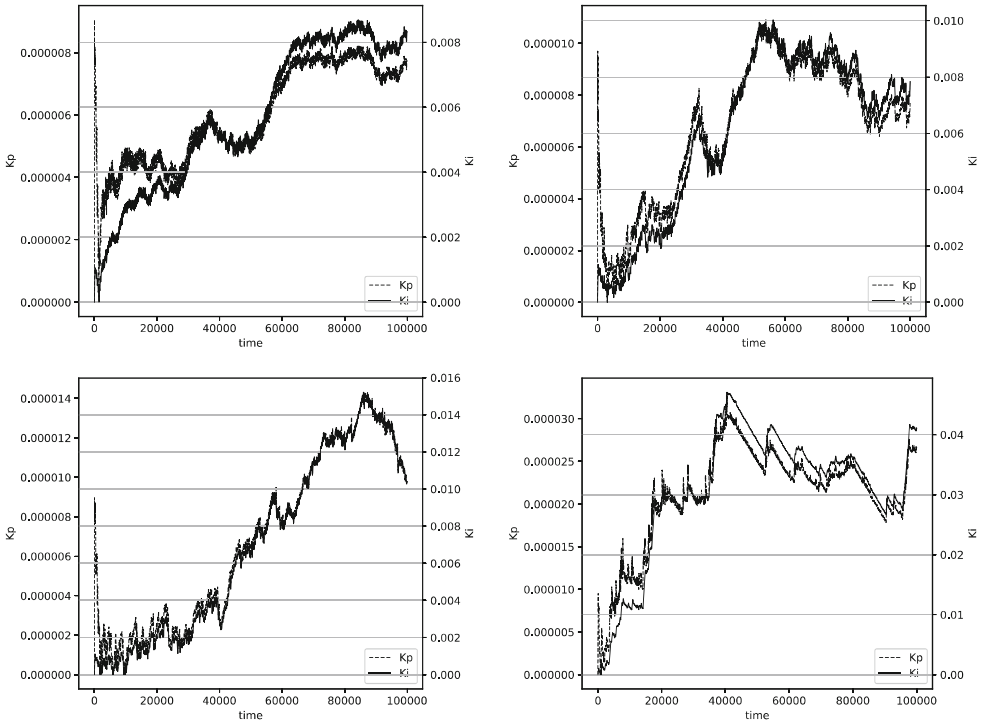


Fig. 6. The parameters K_P and K_I evolution, $\alpha = -1.0$, $\mu = 0.25$, top: $H = 0.5$ and 0.7, bottom: $H = 0.8$ and 0.9

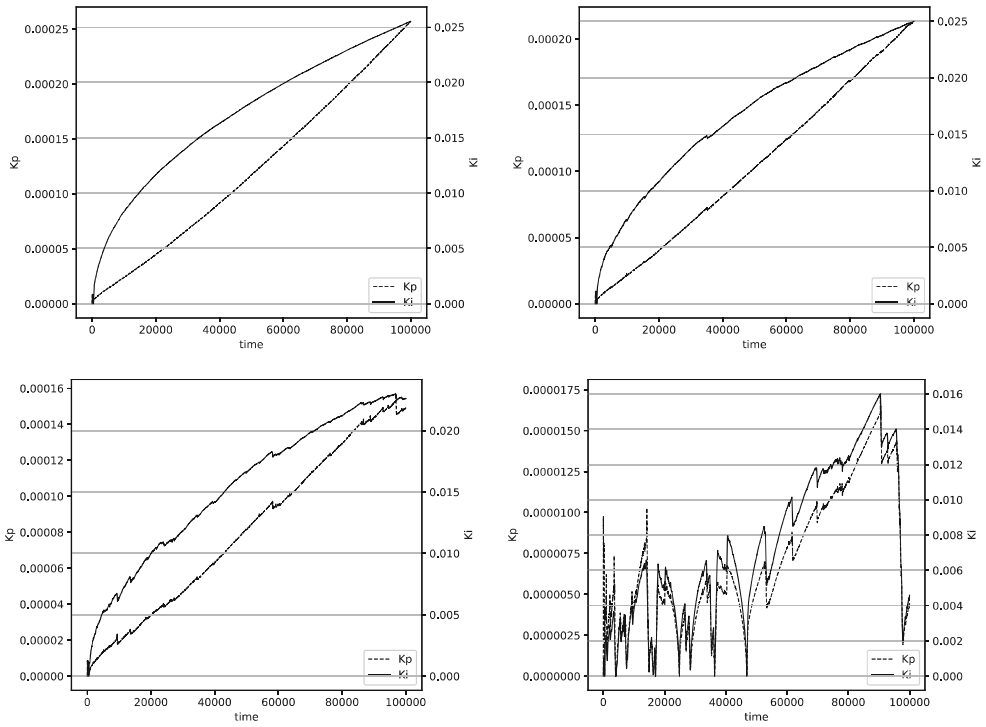


Fig. 7. The parameters K_P and K_I evolution, $\alpha = -0.5$, $\mu = 0.25$, top: $H = 0.5$ and 0.7, bottom: $H = 0.8$ and 0.9

4 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 the PI^{α} controller. Our previous work has shown the advantages of this solution [12,13].

The efficient operation of the AQM algorithms depends on the proper selection of its parameters. Additionally, the optimal parameters may depend on traffic intensity and degree of traffic self-similarity (expressed in Hurst parameter) [15].

This paper introduces a way of a neuro-intelligent tuning of the AQM controller parameters. In the article, the authors presented two algorithms: Adaptive Neuron AQM and PI^{α} controller with neuron tuning parameters. Both algorithms worked correctly. The AN-AQM maintains the queue size slightly below the assumed level. In the case of PI^{α} the obtained average queue lengths have corresponded to our expectations. However, we observe a large variability of the queue occupancy. These fluctuations can be controlled by proper setting the integral order parameter α.

Our article also presents 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 ($H = 0.5, 0.7, 0.8, 0.9$). As can be seen, when the degree of self-similarity increases, the average queue occupancy is below the assumed level.

Acknowledgements. This research was partially financed by National Science Center project no. 2017/27/B/ST6/00145.

This research was partially financed by 02/020/BKM19/0183.

References

1. Abry, P., Veitch, D.: Wavelet analysis of long-range-dependent traffic. IEEE Trans. Inform. Theory **44**(1), 2–15 (1998)
2. Bhatnagar, S., Patro, R.: A proof of convergence of the B-RED and P-RED algorithms for random early detection. IEEE Commun. Lett. **13**, 809–811 (2009)
3. Bonald, T., May, M., Bolot, J.C.: Analytic evaluation of RED performance. In: Proceedings of INFOCOM (2000)
4. Braden, B., et al.: Recommendations on queue management and congestion avoidance in the internet. RFC 2309, IETF (1998)
5. Feng, W.C., Kandlur, D., Saha, D.: Adaptive packet marking for maintaining end to end throughput in a differentiated service internet. IEEE/ACM Trans. Netw. **7**(5), 685–697 (1999)
6. Domańska, J., Augustyn, D.R., Domański, A.: The choice of optimal 3-rd order polynomial packet dropping function for NLRED in the presence of self-similar traffic. Bull. Polish Acad. Sci.: Tech. Sci. **60**(4), 779–786 (2012)
7. Domańska, J., Domański, A.: The influence of traffic self-similarity on QoS mechanisms. In: Proceedings of the International Symposium on Applications and the Internet, Saint, Trento, Italy, pp. 300–303 (2005)
8. Domańska, J., Domański, A., Augustyn, D.R., Klamka, J.: A RED modified weighted moving average for soft real-time application. Int. J. Appl. Math. Comput. Sci. **24**(3), 697–707 (2014)
9. Domańska, J., Domański, A., Czachórski, T.: Fluid flow analysis of RED algorithm with modified weighted moving average. In: Dudin, A., Klimenok, V., Tsarenkov, G., Dudin, S. (eds.) BWWQT 2013. Communications in Computer and Information Science, vol. 356, pp. 50–58. Springer, Heidelberg (2013). https://doi.org/10.1007/978-3-642-35980-4_7
10. Domańska, J., Domańska, A., Czachórski, T.: A few investigations of long-range dependence in network traffic. In: Czachórski, T., Gelenbe, E., Lent, R. (eds.) Information Sciences and Systems 2014, pp. 137–144. Springer, Cham (2014). https://doi.org/10.1007/978-3-319-09465-6_15
11. 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
12. 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)

13. 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
14. 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
15. 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
16. 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
17. Floyd, S.: Discussions of setting parameters (1997)
18. Floyd, S., Jacobson, V.: Random early detection gateways for congestion avoidance. IEEE/ACM Trans. Netw. **1**(4), 397–413 (1993)
19. Hassan, M., Jain, R.: High Performance TCP/IP Networking - Concepts, Issues and Solutions. Pearson Education Inc., London (2004)
20. Ho, H.-J., Lin, W.-M.: AURED - autonomous random early detection for TCP congestion control. In: 3rd International Conference on Systems and Networks Communications Malta (2008)
21. May, M., Bonald, T., Bolot, J.: Analytic evaluation of RED performance. In: Proceedings of the IEEE Infocom, Tel-Aviv, Izrael (2000)
22. May, M., Diot, C., Lyles, B., Bolot, J.: Influence of active queue management parameters on aggregate traffic performance. Technical report, Institut de Recherche en Informatique et en Automatique (2000)
23. Ning, W., Shuqing, W.: Neuro-intelligent coordination control for a unit power plant. In: IEEE International Conference on Intelligent Processing Systems (Cat. No. 97TH8335), vol. 1, pp. 750–753 (1997)
24. 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)
25. Ping, Y.D., Wang, N.: A PID controller with neuron tuning parameters for multi-model plants. In: Proceedings of 2004 International Conference on Machine Learning and Cybernetics (IEEE Cat. No. 04EX826), vol. 6, pp. 3408–3411 (2004)
26. Podlubny, I.: Fractional order systems and $PI^{\lambda}D^{\mu}$ controllers. IEEE Trans. Autom. Control **44**(1), 208–214 (1999)
27. Sawicki, M., Kwiecień, A.: Unexpected anomalies of isochronous communication over USB 3.1 Gen 1. Comput. Stand. Interfaces. **49**, 67–70 (2017)
28. Sun, J., Zukerman, M.: An adaptive neuron AQM for a stable internet. In: Akyildiz, I.F., Sivakumar, R., Ekici, E., Oliveira, J.C., McNair, J. (eds.) NETWORKING 2007. LNCS, vol. 4479, pp. 844–854. Springer, Heidelberg (2007). https://doi.org/10.1007/978-3-540-72606-7_72
29. Zheng, B., Atiquzzaman, M.: A framework to determine the optimal weight parameter of RED in next generation internet routers. Technical report, The University of Dayton, Department of Electrical and Computer Engineering (2000)

Rozdział 6

Adaptive Hurst-Sensitive Active Queue Management

entropy

MDPI

Article

Adaptive Hurst-Sensitive Active Queue Management

Dariusz Marek [1], Jakub Szyguła [1,*], Adam Domański [1], Joanna Domańska [2], Katarzyna Filus [2] and Marta Szczygieł [1]

[1] Faculty of Automatic Control, Electronics and Computer Science, Department of Distributed Systems and Informatic Devices, Silesian University of Technology, Akademicka 16, 44-100 Gliwice, Poland; dariusz.marek@polsl.pl (D.M.); adam.domanski@polsl.pl (A.D.); martszc484@student.polsl.pl (M.S.)

[2] Institute of Theoretical and Applied Informatics, Polish Academy of Sciences, Bałtycka 5, 44-100 Gliwice, Poland; joanna@iitis.pl (J.D.); kfilus@iitis.pl (K.F.)

* Correspondence: jakub.szygula@polsl.pl

Abstract: An Active Queue Management (AQM) mechanism, recommended by the Internet Engineering Task Force (IETF), increases the efficiency of network transmission. An example of this type of algorithm can be the Random Early Detection (RED) algorithm. The behavior of the RED algorithm strictly depends on the correct selection of its parameters. This selection may be performed automatically depending on the network conditions. The mechanisms that adjust their parameters to the network conditions are called the adaptive ones. The example can be the Adaptive RED (ARED) mechanism, which adjusts its parameters taking into consideration the traffic intensity. In our paper, we propose to use an additional traffic parameter to adjust the AQM parameters—degree of self-similarity—expressed using the Hurst parameter. In our study, we propose the modifications of the well-known AQM algorithms: ARED and fractional order $PI^{\alpha}D^{\beta}$ and the algorithms based on neural networks that are used to automatically adjust the AQM parameters using the traffic intensity and its degree of self-similarity. We use the Fluid Flow approximation and the discrete event simulation to evaluate the behavior of queues controlled by the proposed adaptive AQM mechanisms and compare the results with those obtained with their basic counterparts. In our experiments, we analyzed the average queue occupancies and packet delays in the communication node. The obtained results show that considering the degree of self-similarity of network traffic in the process of AQM parameters determination enabled us to decrease the average queue occupancy and the number of rejected packets, as well as to reduce the transmission latency.

Keywords: neural networks; adaptive AQM; self similarity; PID; reinforcement learning

Citation: Marek, D.; Szyguła, J.; Domański, A.; Domańska, J.; Filus, K.; Szczygieł, M. Adaptive Hurst-Sensitive Active Queue Management. *Entropy* **2022**, *24*, 418. https://doi.org/10.3390/e24030418

Received: 25 December 2021
Accepted: 11 March 2022
Published: 17 March 2022

1. Introduction

To properly evaluate the performance of computer networks, it is necessary to develop appropriate models of network mechanisms and a realistic model of packet traffic. Models used for computer network evaluation can be analytical or, as an alternative, they can use discrete event simulation. In the case of computer network modeling, analytical models based on queueing theory are often found in the literature [1,2]. The obtained results are then used in the design phase of network mechanisms to evaluate and compare the created mechanisms with existing solutions, as well as in the operation phase to adjust the configuration of network devices and parameters of network protocols to the required objectives [3–6].

There are two basic principles for managing queue occupancy in the Internet transmission. The first one—a traditional approach—assumes that packets arriving in the buffer are dropped only when the buffer is completely full. Active Queue Management (AQM) approaches are based on the idea of preemptively dropping packets even if there is still space to store incoming packets. These packets are dropped randomly according to a calculated probability function, which allows for increasing the network throughput and providing

fair access to the link. It also eliminates the problem of global synchronization. The performance of the TCP protocol is closely related to the AQM algorithm implemented in the router. The first and still the most popular [7,8] AQM algorithm is Random Early Detection (RED), proposed in 1993 by Sally Floyd and Van Jacobson [9]. A great number of works exist in which the effect of changing its parameters has been studied, and which modifications of this mechanism have been presented to improve transmission performance.

The RED mechanism maintains a reasonable queue length and acceptable transmission latency. Nevertheless, it is necessary to choose its parameters properly [10]. Otherwise, the TCP/RED system becomes unstable [11]. Research related to the attempts to increase the performance of the RED mechanism has been presented in [12]. Our earlier work [13] showed that a more detailed study of the previous queue occupancy and a change in the implementation of the weighted average queue length can also improve the transmission performance.

Most of the RED type algorithms are based on preventive packet dropping when the queue occupancy is between certain predetermined thresholds (Min_{th}, Max_{th}). Its idea is based on a dropping function yielding a probability of packet rejection. The existing versions of the RED algorithm mostly differ in the way of defining the packet dropping probability function [14–16]. Proper selection of the parameters of this function is extremely important and should depend on network conditions. For the RED algorithm, the average queue length oscillates around the minimum threshold Min_{th} for a small load or when high values of parameter P_{max} (the maximum value of packet dropping probability function) are used. For high load or low values of P_{max}, the average queue length is close to or even exceeds the threshold Max_{th}.

In an operating network, the traffic intensity is highly variable. Thus, the AQM parameters should also change. Algorithms whose parameters change during operation are called the adaptive ones. The first algorithm of this type was ARED (Adaptive RED). For the ARED algorithm parameter, P_{max} varies during the router operation, so that the queue occupancy is maintained between values Min_{th} and Max_{th}. Such approach reduces the variability of the queue delays and minimizes the amount of rejected packets [17,18].

Unfortunately, in the first ARED algorithm, adaptation of P_{max} is time-consuming; therefore, matching queue parameters also takes a lot of time [19]. The existing types of the ARED algorithm mostly differ in the method of parameter values estimation [20]. Ref. [21] discusses the problem of real-time video transmission and its self-similar nature. The work shows that such traffic characteristics cause large delays. They postulate the necessity of creating new AQM mechanisms because traditional algorithms (such as RED or ARED) are not recommended in this case.

Parameters of Adaptive AQM algorithms are set based on intensity of network traffic. In our paper, we propose to set them not only based on intensity but also to incorporate the degree of traffic self-similarity to the selection process. Many studies have shown that the network traffic exhibits self-similarity (defined in Section 3), which has a large impact on a network performance: enlarges the queue occupancy and increases the number of the dropped packets in the nodes [22]. Unfortunately, the algorithms for calculating the degree of traffic self-similarity are computationally complex. Long computation time makes them unsuitable for this type of application. The paper proposes modifications of the Hurst estimation method, in which some of the computation procedures (collecting traffic information) are performed in a continuous manner, regardless of the Hurst estimation process. The proposed modifications make it possible to use it in queue scheduling in the router. In our paper, we examine how incorporation of Self-Similarity degree sensing into different AQMs affects the queue behavior. In our experiments, we modify two families of AQM mechanisms: ARED and non-integer order $PI^{\alpha}D^{\beta}$ controller and compare the performance with their basic equivalents. We apply artificial neural networks to tune the AQM mechanisms' parameters.

The remainder of the paper is organized as follows: In Section 2, we describe the related works. Section 3 provides the background regarding LRD, self-similarity and

Hurst parameter calculation. Section 4 and 5 describes different AQM mechanisms, neural network tuning of their parameters and theoretical basis for non-integer $PI^{\alpha}D^{\beta}$ controller. Section 6 presents experiments and discusses numerical results. Conclusions can be found in Section 7.

2. State of the Art

The original RED algorithm and its later modifications, such as Nonlinear RED (NLRED) [23] or Double Slope RED (DSRED) [14]), tend to be very sensitive to the network traffic properties (such as intensity or degree of self-similarity). When the network nodes are overloaded [24], these mechanisms cannot be used to maintain the intended queue length and frequently the maximum queue size is exceeded [13,23,25]. For this reason, they are not suitable for the proposed solution.

To analyze the performance and dynamics of Internet connections, the control theory methods can be used. They can contribute to the improvement of network stability and reduction of the reaction time. Some feedback control mechanisms have been proposed in the literature. In work [26], a dynamic Fluid Flow TCP/RED network model based on stochastic differential equations has been presented. This work contributed to the creation of several AQM algorithms based on different control theory approaches. In ref. [27], a mechanism based on a Proportional-Integral (PI) controller was proposed. In ref. [28], an adaptive Proportional (P) and Proportional-Integral (PI) controller were created. The conclusion was that the PI controller can easily adapt to the Internet traffic fluctuation. In ref. [29], a new variant of the RED mechanism, Proportional-Derivative-RED (PD-RED), was proposed. It was proven that the presented mechanism performed better than the Adaptive RED. In ref. [30], a Proportional-Integral-Differential (PID) controller was presented. The aim was to accelerate the responsiveness of the system. In the domain of control theory-based AQMs, the PI controllers are frequently used due to their implementation and computation simplicity [11]. In ref. [31], a self-tuning compensated PID controller was proposed, and the authors put the emphasis on the simplicity of the method. In ref. [32], the authors have proven that the key advantage of the Fractional–Order PID controller is its insensitivity to the parameters of the systems. As a result, these methods can ensure a stable performance.

Ref. [33] compares AQM mechanisms based on a PID controller and RBF neural networks. Less fluctuation in queue occupancy and faster steady-state time were observed for the neural network approach.

The advantages of using new concepts to create AQM mechanisms based on the reinforcement learning for network resource management have been described in [34]. This paper highlights that such mechanisms automatically adapt to changing network conditions without using additional tuning parameters.

The issues of TCP/AQM congestion control along with the occurring UDP streams have been addressed in [35]. The authors proposed a modification of the PID mechanism by implementing the disturbance and the time delay compensation in an integrated manner.

In addition, the increased interest in the use of AQM mechanisms is due to their use in 5G networks. Ref. [36] presents the problem of packet dropping and queuing delay for mobile 5G networks. In this paper, the authors present a new CoDel-based AQM mechanism that does not require information about the current network state.

3. Hurst Estimation Methods

Many studies, both theoretical and empirical, have shown that one of the important problems that should be taken into account when network solutions are analyzed are the traffic self-similarity and long-range dependence [37–41]. The occurrence of this phenomena in network traffic increases the queue lengths and the number of dropped packets in the routers [22]. Ignoring them may cause an underestimation of performance measures [42,43]. Our previous work has shown how the traffic self-similarity affects the behavior of the AQM queues [44,45]. In addition, the selection of the optimal AQM parameters depends on the degree of self-similarity [46].

The term "self-similar" was first introduced by Benoit Mandelbrot in 1967 [47]. Self-similarity means that a continuous stochastic process and the rescaled one have the same distribution [48]. The condition that a continuous stochastic process $Y(t)$ is self-similar can be written as follows [48]:

$$Y(t) \stackrel{d}{=} a^{-H} Y(at), \quad for \quad t \geq 0, \quad a \geq 0 \quad and \quad 0 < H < 1, \tag{1}$$

where H is the Hurst parameter—a measure used to estimate the degree of self-similarity and a is any positive stretching factor. In the case of the network traffic, we usually represent the data in a time series form and not a continuous process [49]. We measure the traffic in specified time slots. Such an obtained discrete-time stochastic process $X_1, X_2, \ldots X_k, \ldots$ is self-similar when for the aggregated (the original series X is averaged over non-overlapping blocks of size m) sequence $X_k^{(m)}$ [50]:

$$X_k^{(m)} = \frac{1}{m}(X_{km-m+1} + \ldots + X_{km}), \text{ where } m > 1 \text{ and } k \geq 1 \tag{2}$$

and the variance equals [50]:

$$Var[X^{(m)}] = \frac{Var[X]}{m^{\beta}}, \quad \text{where} \quad 0 < \beta < 1, \quad H = 1 - \beta/2 \tag{3}$$

or

$$\log \text{Var}\,(X_k^{(m)}) \approx \log \text{Var}\,(X) - \beta \log m \tag{4}$$

In the literature, the notions of self-similarity and long-range dependence (LRD) are often used as equivalents. Its not true [51]. When the process exhibits LRD, it is an asymptotically second-order self-similar process. The occurrence of LRD means that the temporal similarities can be observed in data. The self-similar intensity of traffic in computer networks is affected in periods of intensive traffic. During such periods, queue occupancy increases. Therefore, we can observe increased waiting times and massive losses of packets. The classical approach to the LRD analysis is based on statistical methods. They are utilised to estimate the value of the Hurst exponent, denoted as H:

- $H \in (0; 0.5)$—the process is negatively correlated, which means that the Long-Range Dependence does not occur;
- $H = 0.5$—the process is uncorrelated;
- $H \in (0.5; 1)$—the process is positively correlated, which means that the LRD occurs.

The traditional estimation methods are, among others, aggregated variance, R/S plot, the periodogram-based method, detrended fluctuation analysis or local Whittle's estimator. These methods use different principles to estimate the Hurst parameter value, thus the obtained values can significantly differ [37,41,52,53]. The big disadvantage of all mentioned methods is their complexity. Due to time-consuming calculations, they cannot be used to manage network traffic in real time. In this paper, we propose some modifications to one of the algorithms that allows us to use it for an Adaptive AQM mechanism.

One of the most popular algorithms of Hurst estimation is the Aggregate Variance method. This method is based on the formulas presented below. A stationary time series of length N, which shows long-term dependencies, is characterized by an average variance of samples of the order N^{2H-2} [48]. Hence, the following algorithm for determining the Hurst parameter value can be used:

Step 1: Divide the time series into blocks of length m (where m takes the values between 2 and $\frac{N}{2}$), and then compute the mean value for each k-th block [48]:

$$\overline{X}^{(m)}(k) = \frac{1}{m} \sum_{t=(k-1)m+1}^{km} X(t) \tag{5}$$

for $k = 1, 2, \ldots, \frac{N}{m}$

Step 2: Compute the variance of the averaged process $\overline{X}^{(m)}(k)$ (for every m) [48]:

$$\sigma_m^2 = \frac{1}{\frac{N}{m} - 1} \sum_{k=1}^{\frac{N}{m}} (\overline{X}^{(m)}(k) - \mu)^2 \tag{6}$$

Step 3: Using the least squares method, we determine the approximation line for the values of logarithm of σ_m^2 as a function of the logarithm of m.

Step 4: We determine the Hurst parameter value from the expression below:

$$H = 1 - \frac{\zeta}{2}, \tag{7}$$

where ζ is the slope of the approximated straight line.

The input data of the described algorithm are the intervals between arrival times of successive packets. As an output, we obtain the Hurst parameter value.

In this paper, we propose some modifications to this Hurst estimation method. Our goal is to carry out some calculations in the background. To achieve this objective, we changed the computation procedure in the first two steps of the above algorithm.

In the first step, instead of mean values, the sum for each k-th block is determined:

$$X^{(m)}(k) = \sum_{t=(k-1)m+1}^{km} X(t), \quad for \quad k = 1, 2, \ldots, \frac{N}{m} \tag{8}$$

This simple trick allows us to modify all k-blocks with each new packet arrival; see Figure 1. We collect information about the number of packets that came in a single time slot. Then, at the end of each time slot, a slot with information about the number of packets from that time slot is added to the first block (2^k for k = 1). If two new slots with packets appear in the k-th block, a new slot is created in the k+1 block with the sum of the values from these two new slots from the k-th block. With this modification, when more packets arrive in the pessimistic case, k + 1 summations must be performed.

Additionally, we modify the formula of variance calculation:

$$\sigma_m^2 = \frac{1}{\frac{N}{m} - 1} \sum_{k=1}^{\frac{N}{m}} (X^{(m)}(k)^2) - \mu^2, \tag{9}$$

where μ is a mean value. As can be observed, the first part of the formula can also be calculated with block modification. Since most of the data are computed all the time (in background) and the number of blocks is small, the rest of the calculations (calculation of the mean value and approximation of obtained variances) are less time-consuming.

Table 1 compares the Hurst estimation results obtained using a standard Aggregate Variance method and our proposition. The results are identical, which confirms that calculating the sum values instead of mean values does not affect the estimation of the degree of self-similarity. Table 2 presents times of Hurst estimations depending on the length of the sample. The first column (Method 1) presents times for the standard Aggregate Variance method. Column 2 (method 2 (ver. 1)) presents results for our method. Presented times are slightly larger (despite the profit which should be gained by resigning from calculating average values in blocks). The increased time is caused by building the structures needed to store information in blocks. The advantage of our solution is that modifications of blocks and partial computation of variances can take place in the background. Column 3 (method 2 (ver. 2)) shows the computation times without operations possible in the background. As can be seen, the presented results are small enough to use the proposed Hurst estimation in the queuing mechanisms.

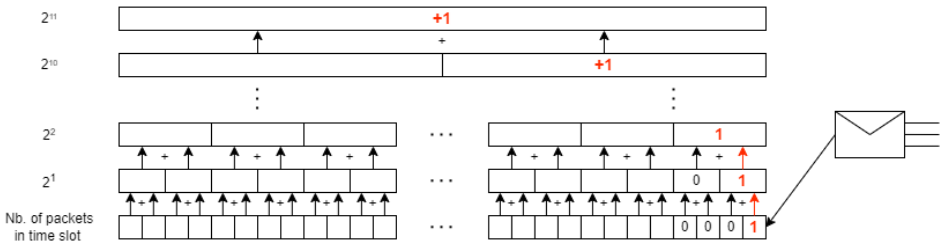


Figure 1. Hurst calculation algorithm.

Table 1. Assumed and estimated Hurst parameter. Length of the sample 2^{18}.

H	Method 1	Method 2
0.5	0.4975	0.4975
0.6	0.5918	0.5918
0.7	0.7124	0.7124
0.8	0.8098	0.8098
0.9	0.9108	0.9108

Table 2. Time of calculating Hurst estimation.

n	Method 1	Method 2 (ver. 1)	Method 2 (ver. 2)
2^{10}	0.000992	0.000995	0.000009
2^{12}	0.003968	0.004454	0.000010
2^{14}	0.015376	0.018848	0.000011
2^{16}	0.062462	0.076383	0.000013
2^{18}	0.262380	0.311451	0.000014

4. Adaptive AQM

In the case of the RED algorithm, the queue is divided into three areas. According to this rule, Min_{th} and Max_{th} values are the assumed queue size threshold values necessary for the proper operation of the RED algorithm [17], whereas Avg is an average queue occupancy.

The dropping probability P is growing linearly from 0 to P_{max}:

$$P = \begin{cases} 0 & \text{for } Avg < Min_{th} \\ \frac{avg - Min_{th}}{Max_{th} - Min_{th}} P_{max} & \text{for } Min_{th} <= Avg <= Max_{th} \\ 1 & \text{for } Avg > Max_{th} \end{cases} \quad (10)$$

The argument Avg is a weighted moving average queue length estimated based on current and past queue lengths. Its value is calculated at the arrival of each packet. The recommended value of P_{max} is 0.1 [54].

The fixed setting of the RED algorithm parameters in the case of variable network traffic may cause its instability (alternately empty and full queue). For the ARED algorithm, the parameter P_{max} changes adaptively (ranging from 0 to 0.5) according to the measured traffic [55]. There are plenty of papers regarding the modification of RED that shows the impact of changes in the determination of the packet rejection probability function on the efficiency of these mechanisms and perform a comparison of efficiency of different algorithms. Such a comparison can be found in [25].

In the algorithm, two parameters are defined: α and β. The first one defines how much P_{max} increases and the second one—how much P_{max} decreases ($P_{max} = P_{max} + \alpha$ or

$P_{max} = P_{max} - \beta$). The decision about a possible P_{max} increase or decrease depends on the *Target* parameter, where:

$$Target(t) \in [Min_{th} + 0.4 \cdot (Max_{th} - Min_{th}), Min_{th} + 0.6 \cdot (Max_{th} - Min_{th})] \quad (11)$$

If the average queue length exceeds the target value and P_{max} is less or equal to 0.5, the parameter P_{max} is increased by a factor α defined as the lower value of 0.01 and $P_{max}/4$; otherwise, the P_{max} is reduced by a factor β (authors of the ARED algorithm proposed 0.9). The P_{max} parameter changes between 0.01 and 0.5, which causes an increase in the packet rejection rate in the case of the growing traffic intensity (when compared to traditional RED). The disadvantage of the algorithm is a relatively slow correction of P_{max}. The algorithm needs 10 to 20 s to stabilise the parameter values. In the case of large variability in traffic, the algorithm may have difficulty obtaining optimal performance.

Our modification of the ARED algorithm incorporates adjusting the changes of P_{max} parameter in accordance with the degree of self-similarity of the examined traffic. We propose to change the *Target* parameter depending on the Hurst parameter value:

$$\begin{aligned} Target(t) \in [&t_{min} + (0.4 - (Hurst(k) - 0.5)) \cdot (t_{max} - t_{min}), \\ &t_{min} + (0.6 - (Hurst(k) - 0.5)) \cdot (t_{max} - t_{min})] \end{aligned} \quad (12)$$

The second AQM we present in this paper is based on the Fractional Order $PI^{\alpha}D^{\beta}$ controller. Fractional Order Derivatives and Integrals (FOD/FOI) are extensions of the well-known integrals and derivatives. A proportional-integral-derivative controller (PID controller) is a traditional mechanism used in many feedback control systems. The non-integer order controllers can have better behavior than the classic controllers [56]. Refs. [57–60] show the advantages of such a mechanism used for queue control. They also describe how to use the $PI^{\alpha}D^{\beta}$(non-integer integral order) as an AQM mechanism. In our solution, we use the controller response as the dropping packet probability function.

The probability of a packet loss is given by the following formula:

$$P = max\{0, -(K_P e_k + K_I \Delta^{\alpha} e_k + K_D \Delta^{\beta} e_k)\} \quad (13)$$

where K_P, K_I and K_D are the tuning parameters (they correspond to the proportional, integral and derivative parameters, respectively), e_k is the error in a current slot $e_k = Q_k - Q$, i.e., the difference between current queue Q_k and desired queue Q.

The dropping probability function depends on five parameters: the coefficients for the proportional and integral terms (K_P, K_I, K_D) and the integral and derivative orders (α, β).

In adaptive approaches, these parameters should change regardless of network intensity and the value of the Hurst parameter. The computation of the PID parameters and the packet loss probability is performed in the discrete moments (at the arrival of a new packet). Such models can be considered as a discrete system. The most popular method of the calculations of discrete differ-integrals of non-integer order is a solution based on the generalization used in the Grünwald–Letnikov (GrLET) formula [61,62].

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

$$\triangle^q f_k = \sum_{j=0}^{k} (-1)^j \binom{q}{j} f_{k-j} \quad (14)$$

where $q \in R$ is a non-integer fractional order, f_k is a differentiated discrete function and $\binom{q}{j}$ is a generalized Newton (for real numbers) symbol defined in the following manner:

$$\binom{q}{j} = \begin{cases} 1 & \text{for } j = 0 \\ \dfrac{q(q-1)(q-2)...(q-j+1)}{j!} & \text{for } j = 1, 2, \ldots \end{cases} \quad (15)$$

5. Selection of the AQM Parameters with the Use of Neural Networks

This section presents the artificial intelligence algorithms used to select the proper AQM parameters. In the presented methods, the neuron's input data are queue and network traffic parameters. The target of the mechanism is to select such AQM parameter values in order to keep the assumed queue length.

Adaptive Neuron AQM

In ref. [63], a method to adjust the AQM parameters was proposed. This solution is named Adaptive Neuron AQM (AN-AQM) and uses the single neuron to calculate the probability of packet dropping. Based on this method, we propose the method of setting the AQM parameters.

The new value of parameter A is calculated for each incoming packet and can be obtained as follows:

$$A(k) = A(k-1) + \Delta A(k) \tag{16}$$

where $\Delta A(k)$ reflects changes in parameter A. The value of A depends on state of neuron, which can be described as:

$$\Delta A(k) = K \sum_{i=a}^{b} w_i(k) x_i(k) \tag{17}$$

where K is the proportional coefficient of the neuron. K has to take values greater than zero. $x_i(k)$ for $i = a, a+1, \ldots b)$ is the neuron's input. Parameters a and b define the subset of neuron inputs, which affects the parameter A. Weight $w_i(k)$ is a connection weight of $x_i(k)$. The weights are set according to the learning rule.

With the arrival of each packet, the algorithms calculate the error $e(k)$, which can be presented as a difference between actual queue occupancy $q(k)$ and the desired queue length Q:

$$e(k) = q(k) - Q \tag{18}$$

Paper presents two different types of Adaptive AQM mechanisms. The first one makes the parameters dependent only on the intensity of the network traffic intensity. The second one additionally takes into account the degree of self-similarity (expressed using the Hurst parameter).

For the first type, the inputs of the neuron, we set the following input values: $x_1(k) = e(k) - e(k-1)$, $x_2(k) = e(k) - e(k-2)$, $x_3(k) = e(k-1) - e(k-2)$, $x_4(k) = e(k)$, $x_5(k) = e(k) - 2e(k-1) + e(k-2)$, $x_6(k) = \gamma(k)$, $x_7(k) = \gamma(k-1)$ and $x_8(k) = \gamma(k-2)$.

We use the following input values for the neuron in the case of the Hurst-depended algorithm: $x_1(k) = e(k) - e(k-1)$, $x_2(k) = e(k) - e(k-2)$, $x_3(k) = e(k-1) - e(k-2)$, $x_4(k) = e(k)$, $x_5(k) = e(k) - 2e(k-1) + e(k-2)$, $x_6(k) = \gamma(k)$, $x_7(k) = \gamma(k-1)$, $x_8(k) = \gamma(k-2)$ and $x_9(k) = Hurst(k)$,

where: $\gamma(k)$ is a normalized error rate:

$$\gamma(k) = \frac{r(k)}{C} - 1 \tag{19}$$

where $r(k)$ is the input rate of the buffer at the bottleneck link, and C is the capacity of the bottleneck link.

The learning rule of a neuron can be presented using the following formula [64]:

$$w_i(k+1) = w_i(k) + d_i y_i(k) \tag{20}$$

where $d_i > 0$ is the learning rate, and $y_i(k)$ is the learning strategy. Ref. [64] recommends to use the following learning strategy:

$$y_i(k) = e(k) p(k) x_i(k). \tag{21}$$

where $e(k)$ is a teacher signal.

Such strategy implies that an adaptive neuron self-organizes regardless of $e(k)$ and $\gamma(k)$.

We propose two methods of mapping of the neuron response to the ARED P_{max} parameter. The first method does not consider self-similarity:

$$P_{max}(k) = max(0, min(\sum_{i=a}^{b} w_i(k)x_i(k), 0.5)), \tag{22}$$

and the second one is sensitive to Hurst parameter values:

$$P_{max}(k) = max(0, min(\sum_{i=a}^{b} w_i(k)x_i(k), 0.5)) * (0.5 + Hurst(k)) \tag{23}$$

The neural mechanism of choosing the PI controller parameters for multi-plant models has been presented in refs. [64,65]. Ref. [66] presents the adaptation of the previously proposed solution to the problem of Active Queue Management.

Mapping of the neuron response to $PI^{\alpha}D^{\beta}$ is similar to the Adaptive ARED solution. The formulas below (24)–(33) show how to determine the values of the coefficients for the proportional and integral terms (K_P, K_I, K_D) and the integral and derivative orders (α, β). As can be observed, these values are determined by the neuron weights selected for a given parameter.

The solution for a mechanism that does not consider self-similarity of traffic can be defined as follows:

$$K_P(t) = k_1 \frac{w_1(t)w_6(t)}{\sum_{i+1}^{n} w_i(t)} \tag{24}$$

$$K_I(t) = k_2 \frac{w_4(t)w_7(t)}{\sum_{i+1}^{n} w_i(t)} \tag{25}$$

$$K_D(t) = k_3 \frac{w_5(t)w_4(t)}{\sum_{i+1}^{n} w_i(t)} \tag{26}$$

$$\lambda(t) = k_4 \frac{w_2(t)w_5(t)w_8(t)}{\sum_{i+1}^{n} w_i(t)} \tag{27}$$

$$\beta(t) = k_5 \frac{w_3(t)w_4(t)w_6(t)}{\sum_{i+1}^{n} w_i(t)} \tag{28}$$

where $k_1 \ldots k_5$ are the constant proportional coefficients and $w_i(k)$ for $i = 1 \ldots 8$ are connection weights that depend on corresponding neuron inputs and the learning rule.

For the second Hurst-sensitive solution, the terms and the derivative orders are calculated as follows:

$$K_P(t) = k_1 \frac{w_9(t)w_1(t)w_6(t)}{\sum_{i+1}^{n} w_i(t)} \tag{29}$$

$$K_I(t) = k_2 \frac{w_9(t)w_4(t)w_7(t)}{\sum_{i+1}^{n} w_i(t)} \tag{30}$$

$$K_D(t) = k_3 \frac{w_9(t)w_5(t)w_4(t)}{\sum_{i+1}^{n} w_i(t)} \tag{31}$$

$$\lambda(t) = k_4 \frac{w_9(t)w_2(t)w_5(t)w_8(t)}{\sum_{i+1}^{n} w_i(t)} \tag{32}$$

$$\beta(t) = k_5 \frac{w_9(t)w_3(t)w_4(t)w_6(t)}{\sum_{i+1}^{n} w_i(t)}, \tag{33}$$

where $k_1 \dots k_5$ are the constant proportional coefficients and $w_i(k)$ for $i = 1 \dots 9$ are connection weights. Weight w_9 is associated with an input to which the self-similarity degree of the network stream is specified.

6. Results

Paper presents the results for two different base AQM models. The simulation models of ARED, PI^{α} and $PI^{\alpha}D^{\beta}$ AQM mechanisms allowed us to show the influence of traffic self-similarity on the behavior of queue. The Fluid Flow approximation models allowed us to show the cooperation of AQM with TCP transport protocol. We investigate the impact of Adaptive AQM mechanisms on the transmission performance. We study how the degree of self-similarity affects the queue behavior. In addition, we aim to show that adjusting AQM parameters to the degree of self-similarity can improve the queue characteristics. In addition, we want to show that adjusting AQM parameters to the degree of self-similarity can improve network transmission.

In the simulation method, a self-similar source approximates a large number of TCP sources. For the Fluid Flow approximation, the number of TCP/UDP streams was specified. During the experiments, different AQM mechanisms implemented in the node were used. In the simulation case, this source is equivalent to the TCP streams, for which we also changed the value of the Hurst parameter. In the Fluid Flow analysis, we changed the number of TCP/UDP senders.

6.1. Fluid Flow Analysis

A diagram of the Fluid Flow analytical model has been shown in Figure 2. In ref. [67], we presented a Fluid Flow model that can be used to model multiple TCP/UDP streams. The model created for the purpose of the current study considers a packet stream that can consist of a single TCP stream. As shown in Figure 2, packet losses affect the TCP sender and reduce its transmission intensity.

The fluid flow model [26] can be used to demonstrate the dynamics of the TCP protocol. It ignores the TCP timeout mechanisms. The TCP NewReno model is based on the nonlinear differential equation presented below [68]:

$$\frac{dW_i(t)}{dt} = \frac{1}{R_i(t)} - \frac{W_i(t)}{2} \frac{W_i(t - R(t))}{R_i(t - R_i(t))} p(t - R_i(t)) \tag{34}$$

The equation describes the evolution of the congestion window size. The next equation is related to the queue evolution of the congested router:

$$\frac{dq(t)}{dt} = \sum_{i=1}^{N} \frac{W_i(t)}{R_i(t)} - C, \tag{35}$$

where:

W_i is the expected TCP congestion window size (in packets) for the i-th flow. It defines a number of packets that may be sent without waiting for the acknowledgements of the reception of previous packets;
R_i is the round-trip time, $R_i = q/C + T_p$, the sum $\sum \frac{W_i}{R_i}$ denotes the total input flow to the congestion router;
q is queue length (in packets);
C is link capacity (packets/time unit), the constant output flow of the router;
T_p is propagation delay;
N is the number of TCP sessions passing through the router;
p is the packet drop probability.

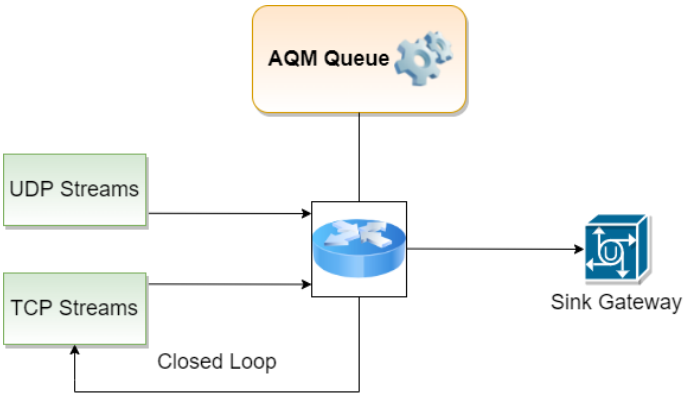


Figure 2. TCP/UDP streams in the adopted Fluid Flow approximation.

For numerical Fluid Flow computations, the software written in Python was used. The detailed description of the methods can be found in [69]. The examined model considers the independent TCP/UDP connections (such models were described in [70]. In experiments, the following TCP/UDP connection parameters were considered:

- transmission capacity of AQM router: $C = 0.075$;
- propagation delay for i-th flow: $T_{p_i} = 2$;
- starting time for i-th flow (TCP and UDP);
- the number of packets sent by i-th flow (TCP and UDP).

We used the following ARED parameters:

- $Min_{th} = 10$;
- $Max_{th} = 15$;
- buffer size (measured in packets) $= 20$;
- $P_{max} = 0.1$;
- eight parameter $w = 0.007$.

The $PI^{\alpha}D\beta$ setpoint equals $= 10$.

In our analysis, the TCP stream starts at time t = 0 and finishes at time t = 80.

Figure 3 presents the TCP and UDP intensity and queue lengths in the case of queue controlled by ARED and ANRED algorithm. The figures on the left present the version of algorithm which does not consider the value of the Hurst parameter. The figures on the right show the results for the mechanism considering the degree of self-similarity. The positioning of the figures described below is the same for all Fluid Flow results.

As can be observed, the ARED Hurst-sensitive algorithm version decreases the queue occupancy. The obtained average queue length for this algorithm is 13.8. In the case of the insensitive algorithm, the average queue size grows to the level of 17.6. Decreasing the average queue size results in a decrease in packet delays.

The Fluid Flow approximation results for the ANRED algorithm controlled by a single neuron are presented at the bottom of Figure 3. The desired queue length is set to 10 packets. This algorithm is robust. Switching on the UDP streams causes changes in the node load, resulting in the TCP congestion mechanism modifying the intensity of its stream. In the figures, it can be observed as fluctuations in the queue occupancy. For both types of algorithms (Hurst-sensitive and insensitive), the obtained average queue lengths are about 10. Nevertheless, it can be noticed that, for the Hurst-sensitive algorithm, stabilization of the queue (reaching the desired queue size) is a little bit faster.

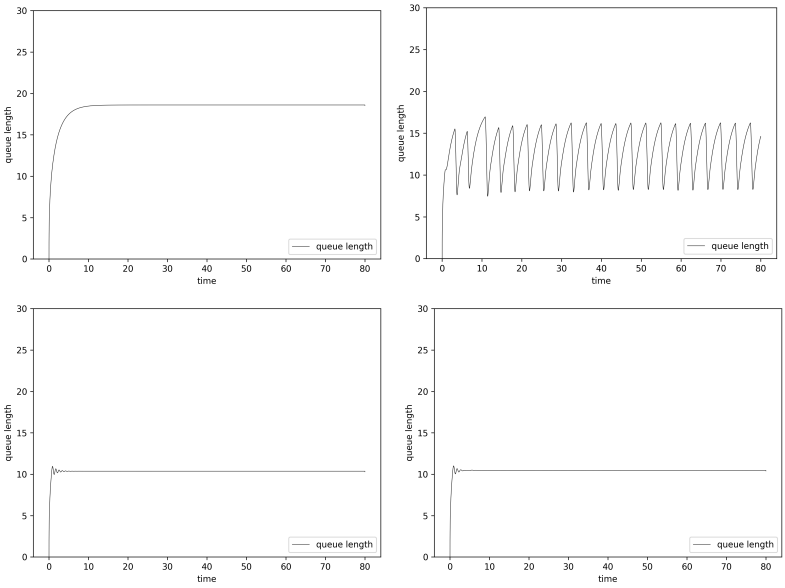


Figure 3. Router average queue length values, Fluid Flow approximation, ARED Hurst-insensitive 1 TCP stream (**left**, **top**), ARED Hurst-sensitive 1 TCP stream (**right**, **top**) and ANRED Hurst-insensitive 1 TCP stream (**left**, **bottom**), ANRED with Hurst-sensitive 1 TCP stream (**right**, **bottom**).

Figure 4 presents the results obtained for AQM mechanisms based on fractional order PI^{α} and $PI^{\alpha}D^{\beta}$ controllers. In the case of PI^{α}, three parameters have been changed during the operation of the mechanism (K_P, K_I and the fractional order α). In the case of $PI^{\alpha}D^{\beta}$, the neuron sets two additional parameters (K_D and the derivative order β). The queue behavior for both controllers is quite similar (barely visible). However, a careful analysis of the results shows that, in the case of a controller with the derivative term, the queue reaches its final length a bit faster. For both types of controllers, their Hurst-sensitive versions allowed us to reach a stable state faster and to obtain smaller queue occupancy.

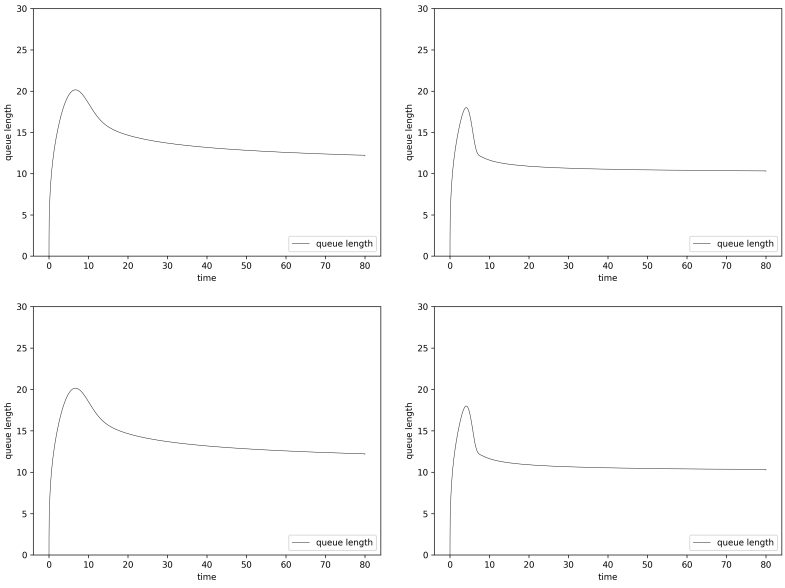


Figure 4. Router average queue length values, Fluid Flow approximation, $ANPI^{\alpha}$ Hurst-insensitive 1 TCP stream (**left**, **top**), $ANPI^{\alpha}$ Hurst-sensitive 1 TCP stream (**right**, **top**) and $ANPI^{\alpha}D^{\beta}$ Hurst-insensitive 1 TCP stream (**left**, **bottom**), $ANPI^{\alpha}D^{\beta}$ Hurst-sensitive 1 TCP stream (**right**, **bottom**).

6.2. Simulation

The simulation model used for the purpose of the current study has been implemented in Python. The Python module SimPy is based on Python generators and allows us to prepare process-based discrete-event simulations [71]. SimPy is released under the MIT License and is frequently used in the area of network simulation [72,73].

Figure 5 presents the simulation model used in the study. Using such a model, the behavior of a single node connected to a large network can be analyzed. A source of packets with a given intensity and Hurst parameter replicates the Internet traffic corresponding to the sum of multiple TCP and UDP streams.

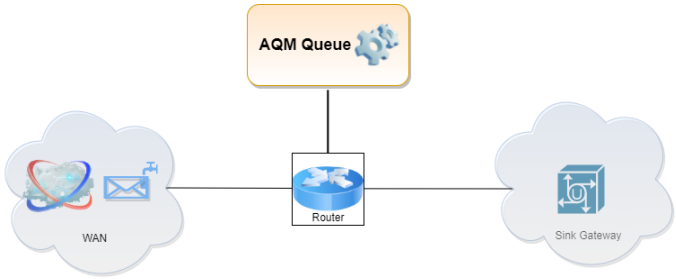


Figure 5. Network node topology in the adopted simulation method.

We analyse the following parameters of a transmission with AQM: the length of the queue and the number of rejected packets. The following parameters of simulations have been used: input traffic intensities, service time and Hurst parameter of input traffic. Input traffic intensity is $\lambda = 0.5$. We have been changing the degree of self-similarity. We used the following values of the Hurst parameter: 0.5, 0.7, 0.8 and 0.9.

The distribution of service time is geometric. We consider three different values of its parameter. We obtain a large node load for $\mu = 0.25$ and medium for $\mu = 0.25$.

The traffic is considered small when $\mu = 0.75$. To improve the readability of the paper, we present the results only for the largest network load case. The parameters μ and λ reflect the load and the parameters of the input and output link. The case in which $\lambda = 0.5$ and $\mu = 0.25$ means that the output bandwidth is two times smaller. The parameters of queues and AQM mechanisms are identical to those used in the Fluid Flow approximation. In the simulation experiments, we analyze the following queue parameters: queue average occupancy, queue average delay and minimum and maximum packet delays.

The top part of Figure 6 presents the queue behavior in the case of the standard ARED algorithm and overloaded buffer. An increase in the Hurst parameter value significantly changes the queue behavior. More detailed results have been presented in Table 3. Regardless of the load, the number of dropped packets increases with the Hurst parameter. In the case of a heavily loaded system, the number of dropped packets may exceed 50%.

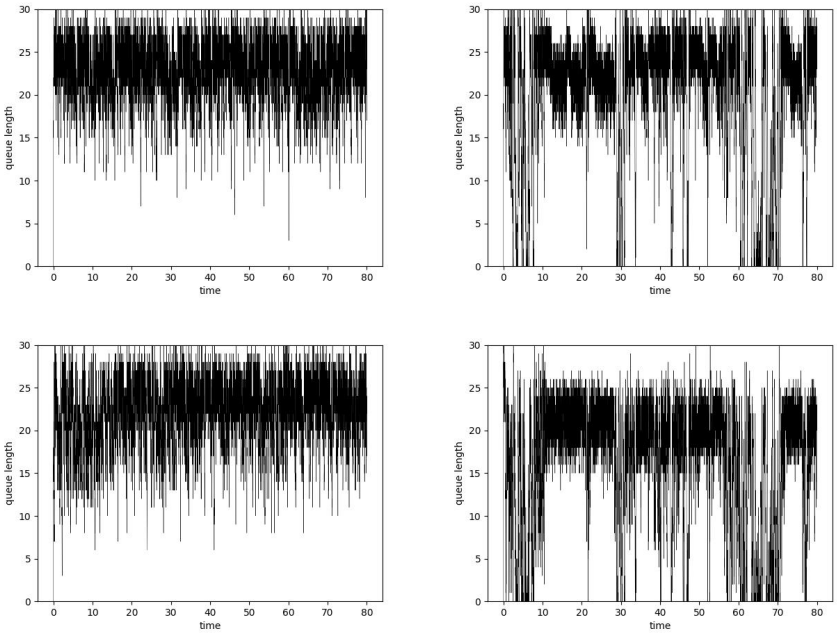


Figure 6. Router queue length values, $\mu = 0.25$, ARED Hurst-insensitive algorithm, $\alpha = 0.5$, H = 0.5 (**left**, **top**), H = 0.9 (**right**, **top**) and ARED Hurst-sensitive algorithm, $\alpha = 0.5$, H = 0.5 (**left**, **bottom**), H = 0.9 (**right**, **bottom**).

Table 3. ARED Hurst-insensitive queue, $\mu = 0.25$.

Hurst	Mean Queue Length	Lost	No. of Dropped Packets		Delay	
			AQM	Queue	Average	Min–Max
0.5	22.93	0.49%	19,261	266	0.092	$2.04 \cdot 10^{-2}$–0.18
0.6	23.05	0.49%	19,270	341	0.093	$3.12 \cdot 10^{-3}$–0.19
0.7	23.55	0.54%	22,950	605	0.089	$3.14 \cdot 10^{-4}$–0.18
0.8	23.31	0.57%	25,943	919	0.086	$1.80 \cdot 10^{-4}$–0.20
0.9	22.56	0.65%	34,040	717	0.081	$1.46 \cdot 10^{-6}$–0.18

Figure 6 shows that queue occupancy decreases. It is especially visible for the traffic with a high degree of self-similarity. Even more interesting behavior has been presented in Table 4. Regardless of the buffer load, the average queue lengths obtained are smaller than those obtained for the standard ARED algorithm. These differences between standard ARED and the Hurst-sensitive ARED become even more significant when the degree of traffic self-similarity increases.

Table 4. Hurst-sensitive ARED queue, $\mu = 0.25$.

Hurst	Mean Queue Length	Lost	No. of Dropped Packets		Delay	
			AQM	Queue	Average	Min–Max
0.5	22.30	0.49%	19,391	290	0.091	$6.94 \cdot 10^{-3}$–0.18
0.6	20.99	0.50%	19,306	359	0.082	$5.49 \cdot 10^{-3}$–0.18
0.7	19.92	0.54%	23,275	364	0.073	$3.24 \cdot 10^{-4}$–0.18
0.8	19.17	0.58%	26,873	268	0.072	$2.21 \cdot 10^{-5}$– 0.21
0.9	19.51	0.65%	34,873	47	0.068	$2.52 \cdot 10^{-6}$–0.16

Figure 7 presents the queue behavior for the ANRED algorithm. The parameter P_{max} for this solution is set by a single neuron. In the case of a heavily loaded queue, this parameter (regardless of the Hurst parameter value) quickly reaches its maximum value. The detailed results (presented in Table 5) confirm the aggressive behavior of the proposed mechanism.

In the case of a neuron-controlled Hurst-sensitive mechanism, the obtained mean queue lengths are even more similar regardless of the degree of self-similarity of the traffic. This dependence is the most visible for the heavily loaded system (bottom part of Figure 7 and Table 6). Contrary to the previous Hurst-insensitive method, this process is more time-consuming and differs depending on the traffic self-similarity degree. The importance of the additional Hurst-sensitive neuron input decreases the load of the system.

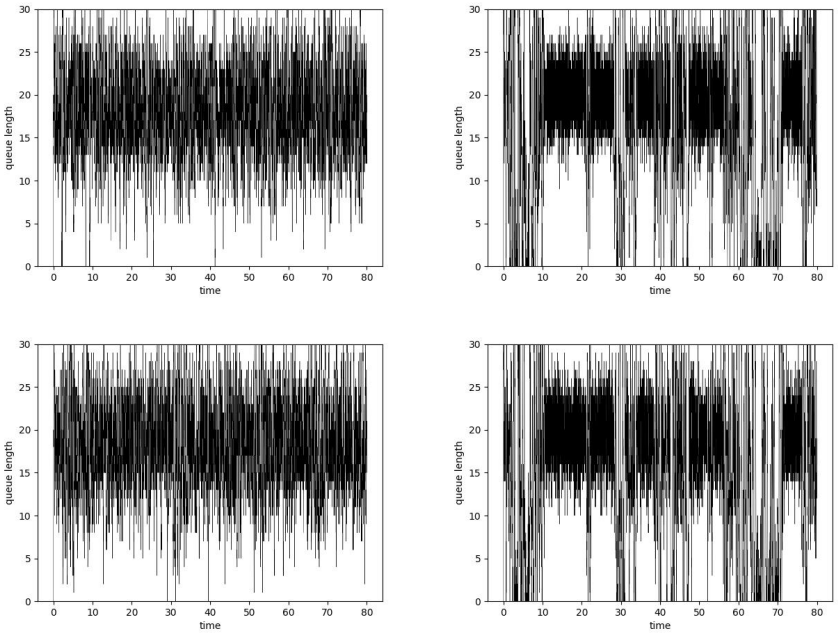


Figure 7. Router queue length values, μ = 0.25, ANRED Hurst-insensitive algorithm, $\alpha = 0.5$, H = 0.5 (**left**, **top**), H = 0.9 (**right**, **top**) and ANRED Hurst-sensitive algorithm, $\alpha = 0.5$, μ = 0.25, H = 0.5 (**left**, **bottom**), H = 0.9 (**right**, **bottom**).

Table 5. ANRED Hurst-insensitive, $\mu = 0.25$.

Hurst	Mean Queue Length	Lost	No. of Dropped Packets		Delay	
			AQM	Queue	Average	Min–Max
0.5	18.24	0.50%	19,498	297	0.073	$7.47 \cdot 10^{-3}$–0.16
0.6	18.14	0.50%	19,179	590	0.073	$9.25 \cdot 10^{-3}$–0.18
0.7	18.55	0.55%	22,865	944	0.070	$1.33 \cdot 10^{-4}$–0.18
0.8	18.64	0.58%	25,591	1409	0.068	$8.04 \cdot 10^{-5}$–0.16
0.9	19.19	0.65%	33,786	1272	0.067	$2.05 \cdot 10^{-5}$–0.18

Table 6. ANRED Hurst-sensitive, $\mu = 0.25$.

Hurst	Mean Queue Length	Lost	No. of Dropped Packets		Delay	
			AQM	Queue	Average	Min–Max
0.5	18.37	0.50%	19,290	476	0.073	$5.39 \cdot 10^{-2}$–0.17
0.6	18.06	0.49%	18,790	586	0.073	$1.27 \cdot 10^{-2}$–0.18
0.7	18.45	0.55%	22,964	916	0.070	$2.08 \cdot 10^{-4}$–0.18
0.8	18.09	0.58%	26,124	1247	0.069	$1.46 \cdot 10^{-5}$–0.19
0.9	18.49	0.65%	34,195	934	0.069	$6.22 \cdot 10^{-5}$–0.19

The previously discussed Adaptive AQM mechanisms based on the RED mechanism modify a single parameter. In the case of AQM based on a PID controller, the number of parameters increases. When we consider a PI^{α} controller, we can modify three parameters. The $PI^{\alpha}D^{\beta}$ controller allows us to modify five parameters. The same as in the previous part of the paper, we compare the Hurst-sensitive selection of the PID parameters results with the non-sensitive ones. Additionally, the next part of our paper presents the impact of the degree of the traffic self-similarity on the evolution of controller parameters.

The impact of the Hurst parameter value on the queue lengths is presented in Figure 8. The figure presents the situation of an overloaded router. For all cases, the queue after a certain period of instability is set to the desired level. The obtained average queue lengths are similar regardless of the Hurst parameter value. By comparing the PI controller with the ARED algorithm, it can be concluded that it leads to a smaller queue length with a similar number of losses. Figure 9 presents changes in parameters K_P, K_I and λ. As can be seen, the bursty nature of traffic causes greater variability in parameters. The detailed results have been presented in Table 7.

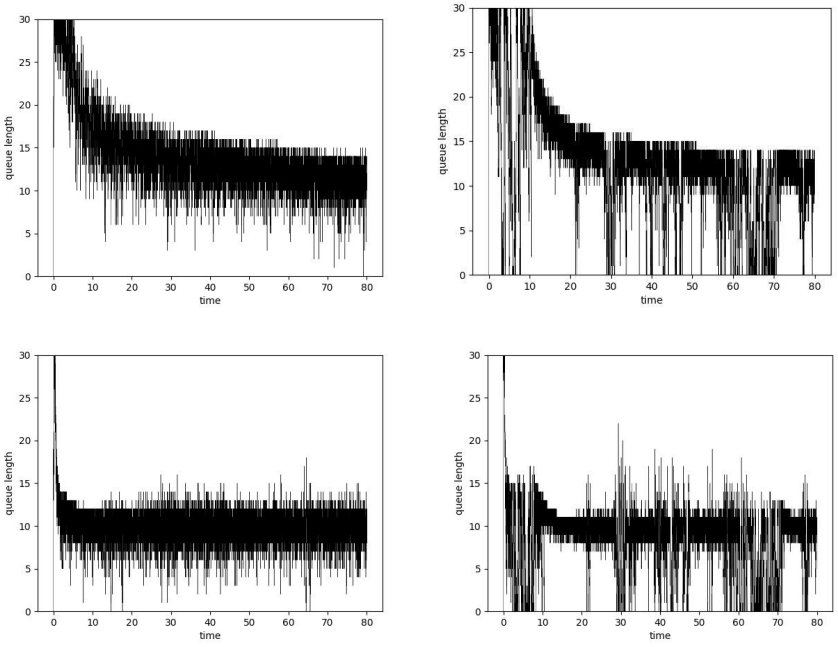


Figure 8. Queue lengths, $\mu = 0.25$, PI^{α} Hurst-insensitive algorithm, $\alpha = 0.5$, H = 0.5 (**left**, **top**), H = 0.9 (**right**, **top**) and PI^{α} Hurst-sensitive algorithm, $\alpha = 0.5$, H = 0.5 (**left**, **bottom**), H = 0.9 (**right**, **bottom**).

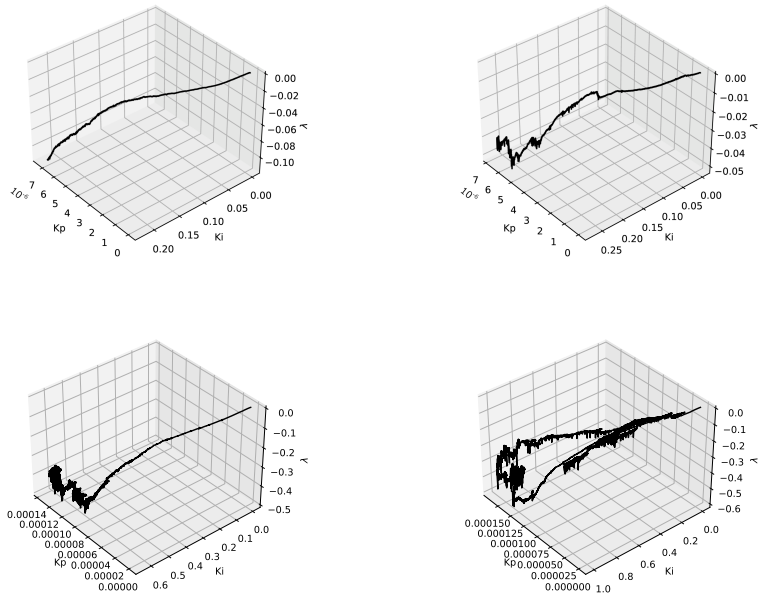


Figure 9. Parameters evolution, μ = 0.25, PI^{α} Hurst-insensitive algorithm, $\alpha = 0.5$, H = 0.5 (**left**, **top**), H=0.9 (**right**, **top**) and PI^{α} Hurst-sensitive algorithm, $\alpha = 0.5$, H = 0.5 (**left**, **bottom**), H = 0.9 (**right**, **bottom**).

Table 7. PI^{α} Hurst-insensitive algorithm, $\mu = 0.25$.

Hurst	Mean Queue Length	Lost	No. of Dropped Packets		Delay	
			AQM	Queue	Average	Min–Max
0.5	14.40	0.49%	18942	655	0.048	$3.77 \cdot 10^{-3}$–0.12
0.6	14.45	0.49%	18655	705	0.048	$4.38 \cdot 10^{-4}$–0.11
0.7	14.57	0.54%	22628	1051	0.047	$9.87 \cdot 10^{-6}$–0.12
0.8	14.55	0.58%	25867	1147	0.044	$1.33 \cdot 10^{-5}$–0.10
0.9	14.41	0.66%	34215	1027	0.044	$1.62 \cdot 10^{-6}$–0.13

The bottom part of Figure 8 presents queue lengths in the case of the Hurst-sensitive PI^{α} controller. This controller needs less time to reach optimal queue occupancy. This is achieved due to the high variability of the controller parameters (see the bottom part in Figure 9). This variability is greater for the larger Hurst parameter values. By comparing with the previous mechanism, it can be seen that the average queue lengths are smaller with a similar rate of packet rejection (see Table 8).

Table 8. PI^{α} Hurst-sensitive algorithm, $\mu = 0.25$.

Hurst	Mean Queue Length	Lost	No. of Dropped Packets		Delay	
			AQM	Queue	Average	Min–Max
0.5	10.26	0.50%	19622	89	0.039	$1.68 \cdot 10^{-4}$–0.10
0.6	10.27	0.49%	19417	81	0.039	$1.26 \cdot 10^{-4}$–0.10
0.7	10.24	0.54%	23562	52	0.038	$4.94 \cdot 10^{-5}$–0.10
0.8	10.31	0.59%	27245	316	0.037	$1.10 \cdot 10^{-5}$–0.10
0.9	10.27	0.66%	35145	189	0.037	$4.76 \cdot 10^{-6}$–0.10

The last results obtained show the behavior of the fractional order $PI^{\alpha}D^{\beta}$ controller (see Figure 10).

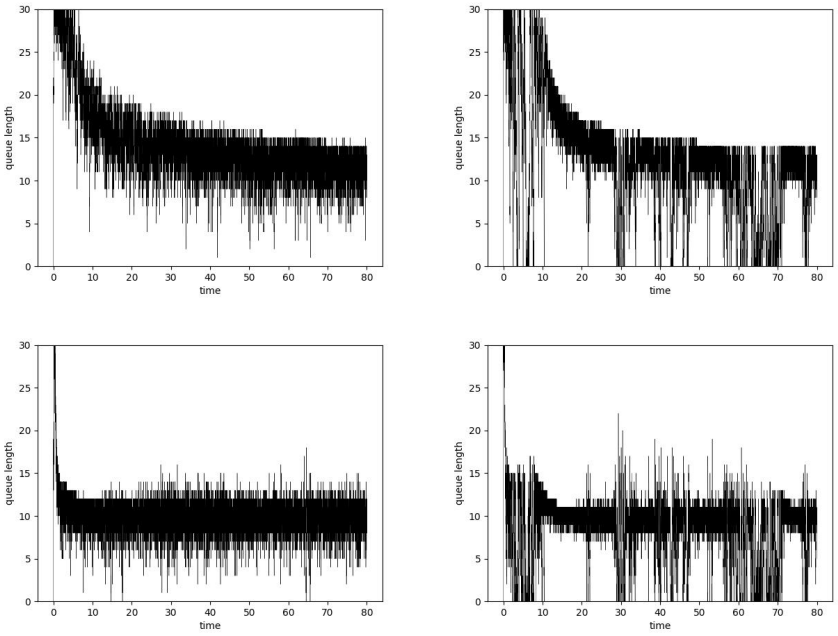


Figure 10. Queue length values, $\mu = 0.25$, $PI^{\alpha}D^{\beta}$ Hurst-insensitive algorithm, $\alpha = 0.5$, H = 0.5 (**left**, **top**), H = 0.9 (**right**, **top**) and $PI^{\alpha}D^{\beta}$ Hurst-sensitive algorithm, $\alpha = 0.5$, H = 0.5 (**left**, **bottom**), H = 0.9 (**right**, **bottom**).

In the case of this controller, we have been changing two additional parameters related to the derivative term. Figure 10 compares the obtained queue length values in the case of Hurst-insensitive $PI^{\alpha}D^{\beta}$ and Hurst-sensitive $PI^{\alpha}D^{\beta}$ controllers. For both versions of the controller, the obtained differences (compared to the PI^{α} controller) are not significant. However, the detailed results presented in Tables 9 and 10 show advantages of the $PI^{\alpha}D^{\beta}$ controller. In the case of the controller with the derivative term, with the same number of rejected packets, the number of packets dropped due to queue overflow has been reduced.

Table 9. $PI^{\alpha}D^{\beta}$ Hurst-insensitive algorithm, $\mu = 0.25$.

Hurst	Mean Queue Length	Lost	No. of Dropped Packets		Delay	
			AQM	Queue	Average	Min–Max
0.5	14.48	0.50%	19,163	772	0.049	$5.29 \cdot 10^{-3}$–0.13
0.6	14.52	0.50%	18,935	755	0.049	$1.30 \cdot 10^{-4}$–0.11
0.7	14.59	0.54%	22,590	1041	0.048	$4.76 \cdot 10^{-5}$–0.15
0.8	14.57	0.58%	25,870	1196	0.045	$1.11 \cdot 10^{-5}$–0.11
0.9	14.32	0.65%	34,000	867	0.043	$2.51 \cdot 10^{-5}$–0.13

Table 10. $PI^{\alpha}D^{\beta}$ Hurst-sensitive algorithm, $\mu = 0.25$.

Hurst	Mean Queue Length	Lost	No. of Dropped Packets		Delay	
			AQM	Queue	Average	Min–Max
0.5	10.28	0.50%	19,806	62	0.040	$1.19 \cdot 10^{-5}$–0.12
0.6	10.27	0.51%	20,096	55	0.039	$1.97 \cdot 10^{-4}$–0.10
0.7	10.25	0.54%	23,688	70	0.038	$2.23 \cdot 10^{-4}$–0.10
0.8	10.33	0.59%	27,212	355	0.037	$2.04 \cdot 10^{-5}$–0.10
0.9	10.23	0.65%	35,078	125	0.037	$6.33 \cdot 10^{-6}$–0.12

7. Conclusions

The performance of the AQM mechanism depends on the selection of its parameters. This selection may be difficult. Proper parameters depend on traffic intensity and degree of traffic self-similarity (expressed in Hurst parameter) [46]. The problem of parameter value search can be solved by an adaptive selection during the operation of the router. This paper proposes Adaptive AQM mechanisms in which the parameter selection process depends on the degree of self-similarity of the network traffic (expressed using the Hurst parameter). The authors have proposed the modifications of the well-known AQM algorithms: standard ARED, ARED with neuron tuning parameters and fractional order $PI^{\alpha}D^{\beta}$ with neuron tuning parameters and built into them an analysis of the degree of self-similarity of network traffic.

The performance of the examined AQM mechanisms has been investigated with the use of two methods: Fluid Flow approximation (closed-loop control) and simulation (open loop scenario). The Fluid Flow approximation has allowed us to test the cooperation of the TCP NewReno protocol with AQM mechanisms. The simulation has been used to verify the operation of AQM mechanisms in the case of traffic of varying intensity and degree of self-similarity. The experiments have been carried out for the four degrees of traffic self-similarity and three different levels of router load.

The analytical results presented in this paper demonstrate how the AQM queues evolve. It can be clearly seen that, for AQM mechanisms that adapt their parameters also to the characteristics of the network traffic, the queues reach a certain steady state faster.

For the simulation results, the proposed model allows the evaluation of a router used in the transmission of a large number of TCP and UDP streams. Experiments have shown the advantages of Hurst-sensitive AQM mechanism. For all described algorithms, Hurst-sensitive modifications led to a decrease in the average queue lengths and reduction of the differences in queue sizes in the case of different levels of Hurst parameter of the network traffic.

Depending on the chosen AQM solution (ARED, PI^{α} lub $PI^{\alpha}D^{\beta}$) and the use of Hurst-sensitive AQM, a reduction in transmission latency values between 11.8% and 18.7% has been observed for traffic without LRD and for traffic characterized by a low degree of LRD (for parameter values H = 0.5 and H = 0.6, respectively). On the other hand, for traffic characterized by a high degree of LRD (H = 0.9), a decrease in delays between 14% and 16.1% was recorded. Similarly to the observed delays, the average queue occupancy has also changed. The decreases between 2.7% and 29% for traffic without LRD (H = 0.5) and between 13.5% and 28.7% for traffic with high LRD (H = 0.9) have been observed. In the case of the PI^{α} and $PI^{\alpha}D^{\beta}$, a significant reduction in the number of dropped packets can also be observed. This number decreased for traffic without LRD (H = 0.5) by about 86% for the first controller and by 92% for the second controller. For traffic characterized by a high degree of LRD (H = 0.9), these decreases were 81.6% and 85.6%, respectively. The only case, in which the use of the Hurst-sensitive mechanism did not significantly affect the results, was the ANRED mechanism. This mechanism in all of the examined cases exhibited a high severity of performance, resulting in a significant number of rejected packets.

The Hurst parameter calculating is computationally complex. The well-known methods of calculating this parameter are too slow to be used in actual routers. The authors of the paper propose a modification of the aggregated variance method. We propose some mathematical simplifications that allow us to perform a large part of the calculations in the background. Information about each incoming packet is stored in a special structure which stores information about the number of packets at different timescales. Such preliminary data preparation significantly speeds up the Hurst parameter value calculation process. Despite the simplifications made to limit the number of computationally-demanding operations, the AQM algorithms used in this paper are still undoubtedly more computationally intensive than the simplest algorithms from the RED family, but at the same time offer better queue management. We believe that, with further development of the technology

and introduction of more powerful routers, it will be possible to fully use such solutions in the near future.

Author Contributions: Conceptualization, D.M. and J.S.; Methodology, J.S. and A.D.; Software, D.M. and J.S.; Investigation, D.M., J.S. and A.D.; Validation, J.D. and D.M.; Project administration, D.M.; Formal analysis, J.D., J.S. and K.F.; Data duration, J.S. and D.M.; Writing—original draft preparation, J.S., J.D., A.D., K.F., D.M. and M.S.; Writing—review and editing, J.S. and J.D.; Visualization, J.S. and D.M.; Supervision, A.D. All authors have read and agreed to the published version of the manuscript.

Funding: This publication was supported by the Excellence Initiative–Research University programme implemented at the Silesian University of Technology, in year 2020/2021, Grant No.: 02/110/SDU/10-22-01.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Not applicable.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Larionov, A.; Vishnevsky, V.; Semenova, O.; Dudin, A. A multiphase queueing model for performance analysis of a multihop IEEE 802.11 wireless network with DCF channel access. In Proceedings of the International Conference on Information Technologies and Mathematical Modelling, Saratov, Russia, 26–30 June 2019; Springer: Berlin, Germany, 2019; pp. 162–176.
2. Chisci, G.; ElSawy, H.; Conti, A.; Alouini, M.S.; Win, M.Z. Uncoordinated massive wireless networks: Spatiotemporal models and multiaccess strategies. *IEEE/ACM Trans. Netw.* **2019**, *27*, 918–931. [CrossRef]
3. Swami, N.; Bairwa, A.; Choudhary, M. A Literature Survey of Network Simulation Tools. In *IJCRT International Conference Proceeding ICCCT*; Association for Computing Machinery: New York, NY, USA, 2017; Volume 5, pp. 206–208.
4. Borboruah, G.; Nandi, G. A Study on Large Scale Network Simulators. *Int. J. Comput. Sci. Inf. Technol.* **2014**, *5*, 7318–7322.
5. Dou, Y.; Liu, H.; Wei, L.; Chen, S. Design and simulation of self-organizing network routing algorithm based on Q-learning. In Proceedings of the 21st Asia-Pacific Network Operations and Management Symposium (APNOMS), Daegu, Korea, 22–25 September 2020; pp. 357–360. [CrossRef]
6. Guo, X.; Guo, B.; Li, K.; Fan, C.; Yang, H.; Huang, S. A SDN-enabled Integrated Space-Ground Information Network Simulation Platform. In Proceedings of the 18th International Conference on Optical Communications and Networks (ICOCN), Huangshan, China, 5–8 August 2019; pp. 1–3. [CrossRef]
7. Adamu, A.; Shorgin, V.; Melnikov, S.; Gaidamaka, Y. Flexible Random Early Detection Algorithm for Queue Management in Routers. In Proceedings of the International Conference on Distributed Computer and Communication Networks, Moscow, Russia, 14–18 September 2020; Springer: Berlin, Germany, 2020; pp. 196–208.
8. Bisoy, S.K.; Pattnaik, P.K. A neuron-based active queue management scheme for internet congestion control. *Int. J. Reason.-Based Intell. Syst.* **2020**, *12*, 238–247.
9. Floyd, S.; Jacobson, V. Random Early Detection gateways for congestion avoidance. *IEEE/ACM Trans. Netw.* **1993**, *1*, 397–413. [CrossRef]
10. Tan, L.; Zhang, W.; Peng, G.; Chen, G. Stability of TCP/RED systems in AQM routers. *IEEE Trans. Autom. Control* **2006**, *51*, 1393–1398. [CrossRef]
11. Unal, H.; Melchor-Aguilar, D.; Ustebay, D.; Niculescu, S.I.; Ozbay, H. Comparison of PI controllers designed for the delay model of TCP/AQM. *Comput. Commun.* **2013**, *36*, 1225–1234. [CrossRef]
12. Hassan, M.; Jain, R. *High Performance TCP/IP Networking—Concepts, Issues and Solutions*; Pearson Education Inc.: Boston, MA, USA, 2004.
13. 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.* **2014**, *24*, 697–707. [CrossRef]
14. Zheng, B.; Atiquzzaman, M. DSRED: An active queue management scheme for next generation networks. In Proceedings of the 25th Annual IEEE Conference on Local Computer Networks, Tampa, FL, USA, 8–10 November 2000.
15. Athuraliya, S.; Low, S.; Li, V.; Yin, Q. REM: Active queue management. *IEEE Netw.* **2001**, *15*, 48–53. 65.923940. [CrossRef]
16. Zhou, K.; Yeung, K.; Li, V. Nonlinear RED: A simple yet efficient Active Queue Management scheme. *Comput. Netw. Int. J. Comput. Telecommun. Netw.* **2006**, *50*, 3784–3794. [CrossRef]
17. Floyd, S.; Gummadi, R.; Shenker, S. Adaptive RED: An Algorithm for Increasing the Robustness of RED's Active Queue Management. 2001. Available online: www.icir.org/floyd/papers/adaptiveRed.pdf (accessed on 25 February 2022).
18. Verma, R.; Iyer, A.; Karandikar, A. Towards an Adaptive RED Algorithm for Archiving Dale-Loss Performance. Available online: https://www.ee.iitb.ac.in/~karandi/assets/attachment/verma_iyer_karandikar_IEEproc03.pdf (accessed on 25 February 2022).

19. Lin, D.; Morris, R. Dynamics of Random Early Detection. In Proceedings of the ACM SIGCOMM '97 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communication. Association for Computing Machinery, Cannes, France, 14–18 September 1997; pp. 127–137. [CrossRef]
20. Abdel-jaber, H.; Mahafzah, M.; Thabtah, F.; Woodward, M. Fuzzy logic controller of Random Early Detection based on average queue length and packet loss rate. In Proceedings of the 2008 International Symposium on Performance Evaluation of Computer and Telecommunication Systems, Edinburgh, UK, 16–18 June 2008; pp. 428–432.
21. de Morais, W.; Santos, C.; Pedroso, C. Application of active queue management for real-time adaptive video streaming. *Telecommun. Syst.* **2021**, *79*, 260–270. [CrossRef] [PubMed]
22. Stallings, W. *High-Speed Networks: TCP/IP and ATM Design Principles*; Prentice-Hall: New York, NY, USA, 1998.
23. Domańska, J.; Augustyn, D.; Domański, A. The choice of optimal 3-rd order polynomial packet dropping function for NLRED in the presence of self-similar traffic. *Bull. Pol. Acad. Sci. Tech. Sci.* **2012**, *60*, 779–786. [CrossRef]
24. Chydzinski, A. On the structure of data losses induced by an overflowed buffer. *Appl. Math. Comput.* **2022**, *415*, 126724. [CrossRef]
25. Domański, A.; Domańska, J.; Czachórski, T. The Impact of the Degree of Self-Similarity on the NLREDwM Mechanism with Drop from Front Strategy. In Proceedings of the CN: International Conference on Computer Networks, Brunów, Poland, 14–17 June 2016; pp. 192–203. [CrossRef]
26. Misra, V.; Gong, W.; Towsley, D. Fluid-based analysis of network of AQM routers supporting TCP flows with an application to RED. *Comput. Commun. Rev.* **2000**, *30*, 151–160. [CrossRef]
27. Hollot, C.V.; Misra, V.; Towsley, D. Analysis and design of controllers for AQM routers supporting TCP flows. *IEEE Trans. Autom. Control.* **2002**, *47*, 945–959. [CrossRef]
28. Hong, Y.; Yang, O.W.W. Adaptive AQM controllers for IP routers with a heuristic monitor on TCP flows. *Int. J. Commun. Syst.* **2006**, *19*, 17–38. [CrossRef]
29. Sun, J.; Ko, K.-T.; Chen, G.; Chan, S.; Zukerman, M. PD-RED: To improve the performance of RED. *IEEE Commun. Lett.* **2003**, *7*, 406–408.
30. Fan, Y.; Ren, F.; Lin, C. Design a PID controller for Active Queue Management. In Proceedings of the Eighth IEEE Symposium on Computers and Communications (ISCC 2003), Kemer-Antalya, Turkey, 3 July 2003; Volume 2, pp. 985–990.
31. Kahe, G.; Jahangir, A.H. A self-tuning controller for queuing delay regulation in TCP/AQM networks. *Telecommun. Syst.* **2019**, *71*, 215–229. [CrossRef]
32. Bingi, K.; Ibrahim, R.; Karsiti, M.; Hassan, S. Frequency Response Based Curve Fitting Approximation of Fractional–Order PID Controllers. *Int. J. Appl. Math. Comput. Sci.* **2019**, *29*, 311–326. [CrossRef]
33. Zhang, W.; Jing, Y. Active Queue Management Algorithm Based on RBF Neural Network Controller. In Proceedings of the 2020 Chinese Control And Decision Conference (CCDC), Hefei, China, 22–24 August 2020; pp. 2289–2293. [CrossRef]
34. AlWahab, D.A.; Gombos, G.; Laki, S. On a Deep Q-Network-based Approach for Active Queue Management. In Proceedings of the 2021 Joint European Conference on Networks and Communications 6G Summit (EuCNC/6G Summit), Porto, Portugal, 8–11 June 2021; pp. 371–376. [CrossRef]
35. Hotchi, R.; Chibana, H.; Iwai, T.; Kubo, R. Active Queue Management Supporting TCP Flows Using Disturbance Observer and Smith Predictor. *IEEE Access.* **2020**, *8*, 173401–173413. [CrossRef]
36. Jung, S.; Kim, J.; Kim, J.H. Intelligent Active Queue Management for Stabilized QoS Guarantees in 5G Mobile Networks. *IEEE Syst. J.* **2021**, *15*, 4293–4302. [CrossRef]
37. Kaur, G.; Saxena, V.; Gupta, J. Detection of TCP targeted high bandwidth attacks using self-similarity. *J. King Saud Univ.-Comput. Inf. Sci.* **2020**, *32*, 35–49. [CrossRef]
38. Deka, R.K.; Bhattacharyya, D.K. Self-similarity based DDoS attack detection using Hurst parameter. *Secur. Commun. Netw.* **2016**, *9*, 4468–4481. [CrossRef]
39. Park, C.; Hernández-Campos, F.; Le, L.; Marron, J.S.; Park, J.; Pipiras, V.; Smith, F.D.; Smith, R.L.; Trovero, M.; Zhu, Z. Long-range dependence analysis of Internet traffic. *J. Appl. Stat.* **2011**, *38*, 1407–1433. [CrossRef]
40. Pramanik, S.; Datta, R. Self-Similarity of Data Traffic in a Delay Tolerant Network. In *2017 Wireless Days*; IEEE: Porto, Portugal, 2017.
41. Xu, Y.; Li, Q.; Meng, S. Self-similarity Analysis and Application of Network Traffic. In Proceedings of the International Conference on Mobile Computing, Applications, and Services, Hangzhou, China, 14–15 June 2019; pp. 112–125. [CrossRef]
42. Kim, Y.; Min, P. On the prediction of average queueing delay with self-similar traffic. In Proceedings of the IEEE Global Telecommunications Conference GLOBECOM '03, San Francisco, CA, USA, 1–5 December 2003; Volume 5, pp. 2987–2991. [CrossRef]
43. Gorrasi, A.; Restaino, R. Experimental comparison of some scheduling disciplines fed by self-similar traffic. In Proceedings of the IEEE International Conference on Communications (ICC '03), Anchorage, AK, USA, 11–15 May 2003; Volume 1, pp. 163–167. [CrossRef]
44. Domański, A.; Domańska, J.; Czachórski, T.; Klamka, J.; Szyguła, J.; Marek, D. The IoT gateway with active queue management. *Int. J. Appl. Math. Comput. Sci.* **2021**, *31*, 165–178. [CrossRef]
45. Szyguła, J.; Domański, A.; Domańska, J.; Marek, D.; Filus, K.; Mendla, S. Supervised Learning of Neural Networks for Active Queue Management in the Internet. *Sensors* **2021**, *21*, 4979. [CrossRef] [PubMed]

46. 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 Proceedings of the 32nd International Symposium, ISCIS 2018, Poznan, Poland, 20–21 September 2018; Volume 935, pp. 76–83. [CrossRef]
47. Mandelbrot, B. How Long Is the Coast of Britain? Statistical Self-Similarity and Fractional Dimension. *Science* **1967**, *156*, 636–638. [CrossRef] [PubMed]
48. Beran, J. *Statistics for Long-Memory Processes*, 1st ed.; Chapman Hall/Routledge: Boston, MA, USA, 1994, Volume 61. [CrossRef]
49. Czachórski, T.; Domańska, J.; Pagano, M. On stochastic models of internet traffic. In Proceedings of the International Conference on Information Technologies and Mathematical Modelling, Anzhero-Sudzhensk, Russia, 18–22 November 2015; pp. 289–303.
50. Cox, D. Long-range dependance: A review. In *Statistics: An Appraisal*; Iowa State University Press: Ames, IA, USA, 1984; pp. 55–74.
51. Domańska, J.; Domański, A.; Czachórski, T. Estimating the Intensity of Long-Range Dependence in Real and Synthetic Traffic Traces. *Commun. Comput. Inf. Sci.* **2015**, *522*, 11–22.
52. Li, Q.; Wang, S.; Liu, Y.; Long, H.; Jiang, J. Traffic self-similarity analysis and application of industrial internet. *Wirel. Netw.* **2020**, 1–15. [CrossRef]
53. Barsukov, I.S.; Bobreshov, A.M.; Riapolov, M.P. Fractal Analysis based Detection of DoS/LDoS Network Attacks. In Proceedings of the 2019 International Russian Automation Conference (RusAutoCon), Sochi, Russia, 8–14 September 2019; pp. 1–5.
54. Floyd, S. Discussions of Setting Parameters. 1997. Available online: http://www.icir.org/floyd/REDparameters.txt (accessed on 20 December 2021).
55. Xu, Y.D.; Wang, Z.Y.; Wang, H. ARED: A novel adaptive congestion controller. In Proceedings of the 2005 International Conference on Machine Learning and Cybernetics, Guangzhou, China, 18–21 August 2005; Volume 2, pp. 708–714. [CrossRef]
56. Podlubny, I. Fractional order systems and $PI^{\lambda}D^{\mu}$ controllers. *IEEE Trans. Autom. Control.* **1999**, *44*, 208–214. [CrossRef]
57. Domańska, J.; Domański, A.; Czachórski, T.; Klamka, J. Self-similarity Traffic and AQM Mechanism Based on Non-integer Order $PI^{\alpha}D^{\beta}$ Controller. *Commun. Comput. Inf. Sci.* **2017**, *718*, 336–350. [CrossRef]
58. 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.* **2016**, *26*, 777–789. [CrossRef]
59. Domańska, J.; Domański, A.; Czachórski, T.; Klamka, J.; Szyguła, J. The AQM Dropping Packet Probability Function Based on Non-integer Order $PI^{\alpha}D^{\beta}$ Controller. In *Non-Integer Order Calculus and Its Applications*; Lecture Notes in Electrical Engineering; Springer International Publishing: Cham, Switzerland, 2019; Volume 496, pp. 36–48._4. [CrossRef]
60. 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 *Computer Networks Information Science*; Springer: Berlin, Germany, 2018; Volume 860, pp. 286–299. [CrossRef]
61. Podlubny, I. *Fractional Differential Equations*; Academic Press: San Diego, CA, USA, 1999; Volume 198.
62. Ciesielski, M.; Leszczynski, J. A Numerical Method for Solution of Ordinary Differential Equations of Fractional Order. In Proceedings of the Parallel Process. Appl. Mathematics; Springer: Berlin/Heidelberg, Germany, 2002; Volume 2328, pp. 695–702. [CrossRef]
63. Sun, J.; Zukerman, M. An Adaptive Neuron AQM for a Stable Internet. In Proceedings of the NETWORKING. Ad Hoc and Sensor Networks, Wireless Networks, Next Generation Internet, 6th International IFIP-TC6 Networking Conference, Atlanta, GA, USA, 14–18 May 2007; Springer: Berlin/Heidelberg, Germany, 2007; Volume 4479, pp. 844–854. [CrossRef]
64. Ping, Y.; Wang, N. A PID controller with neuron tuning parameters for multi-model plants. In Proceedings of the 2004 International Conference on Machine Learning and Cybernetics (IEEE Cat. No.04EX826), Shanghai, China, 26–29 August 2004; Volume 6, pp. 3408–3411. [CrossRef]
65. Ning, W.; Shuqing, W. Neuro-intelligent coordination control for a unit power plant. In Proceedings of the IEEE International Conference on Intelligent Processing Systems (Cat. No.97TH8335), Beijing, China, 28–31 October 1997; Volume 1, pp. 750–753. [CrossRef]
66. Szyguła, J.; Domański, A.; Domańska, J.; Czachórski, T.; Marek, D.; Klamka, J. AQM Mechanism with Neuron Tuning Parameters. In *Intelligent Information and Database Systems*; Springer International Publishing: Cham, Switzerland, 2020; pp. 299–311. [CrossRef]
67. Domańska, J.; Domański, A.; Czachórski, T.; Klamka, J. Fluid flow approximation of time-limited TCP/UDP/XCP streams. *Bull. Pol. Acad. Sci. Tech. Sci.* **2014**, *62*, 217–225. [CrossRef]
68. Hollot, C.; Misra, V.; Towsley, D. A control theoretic analysis of RED. In Proceedings of the IEEE/INFOCOM 2001, Anchorage, AK, USA, 22–26 April 2001; pp. 1510–1519.
69. Domańska, J.; Domański, A.; Czachórski, T. Comparison of AQM Control Systems with the Use of Fluid Flow Approximation. *Commun. Comput. Inf. Sci.* **2012**, *291*, 82–90. [CrossRef]
70. Domański, A.; Domańska, J.; Czachórski, T.; Klamka, J.; Szyguła, J.; Marek, D. Diffusion Approximation Model of TCP NewReno Congestion Control Mechanism. *Springer Nat. Comput. Sci.* **2020**, *1*, 43. [CrossRef]
71. SimPy Documentation. Available online: https://simpy.readthedocs.io/en/latest/ (accessed on 23 December 2021).
72. Tinini, R.I.; dos Santos, M.R.P.; Figueiredo, G.B.; Batista, D.M. 5GPy: A SimPy-based simulator for performance evaluations in 5G hybrid Cloud-Fog RAN architectures. *Simul. Model. Pract. Theory* **2020**, *101*, 102030. [CrossRef]

73. Karanjkar, N.; Tejasvi, P.C.; Amrutur, B. A simpy-based simulation testbed for smart-city IoT applications. In Proceedings of the International Conference on Internet of Things Design and Implementation, Montreal, QC, Canada, 15–18 April 2019; pp. 273–274.

Rozdział 7

Supervised Learning of Neural Networks for Active Queue Management in the Internet

sensors

MDPI

Article

Supervised Learning of Neural Networks for Active Queue Management in the Internet

Jakub Szyguła [1,*], Adam Domański [1], Joanna Domańska [2], Dariusz Marek [1], Katarzyna Filus [2] and Szymon Mendla [1]

[1] Faculty of Automatic Control, Electronics and Computer Science, Department of Distributed Systems and Informatic Devices, Silesian University of Technology, Akademicka 16, 44-100 Gliwice, Poland; adam.domanski@polsl.pl (A.D.); dariusz.marek@polsl.pl (D.M.); szymmen835@student.polsl.pl (S.M.)

[2] Institute of Theoretical and Applied Informatics Polish Academy of Sciences, Bałtycka 5, 44-100 Gliwice, Poland; joanna@iitis.pl (J.D.); kfilus@iitis.pl (K.F.)

* Correspondence: jakub.szygula@polsl.pl

Abstract: The paper examines the AQM mechanism based on neural networks. The active queue management allows packets to be dropped from the router's queue before the buffer is full. The aim of the work is to use machine learning to create a model that copies the behavior of the AQM PI^{α} mechanism. We create training samples taking into account the self-similarity of network traffic. The model uses fractional Gaussian noise as a source. The quantitative analysis is based on simulation. During the tests, we analyzed the length of the queue, the number of rejected packets and waiting times in the queues. The proposed mechanism shows the usefulness of the Active Queue Management mechanism based on Neural Networks.

Keywords: neural networks; Hurst exponent; self-similarity; internet traffic; congestion control; dropping packets; active queue management; PI^{α} controller

Citation: Szyguła, J.; Domańsk, A.; Domańska, J.; Marek, D.; Filus, K.; Mendla, S. Supervised Learning of Neural Networks for Active Queue Management in the Internet. *Sensors* **2021**, *21*, 4979. https://doi.org/10.3390/s21154979

Academic Editor: Klaus Stefan Drese

Received: 12 June 2021
Accepted: 15 July 2021
Published: 22 July 2021

1. Introduction

Cisco predicts that by 2022, the Internet traffic will increase to 77 exabytes per month due to the rapid development of mobile technologies. The mobile data transfer will increase sevenfold compared to 2017, with an average annual growth of 46% [1]. The rapid increase in the number of Internet users as well as the transmission of multimedia content of increasing quality force the continuous development of data transmission mechanisms.

Wide area networks have their origins in the 1970s and were created for the American army. Thus, the most important aspect of the network based on a distributed architecture was to deliver reliable transmission of data and low connection costs. Unfortunately, the design assumptions proposed at the beginning turned out to be insufficient over the years.

Initially, IP routers handled packets according to the FIFO (First In First Out) rule (the first incoming packet in the queue is the first one to be served) [2]. For such scheduling, packets are dropped when the queue length exceeds the maximum length which results in the retransmission of a large number of packets in a short period of time. For such a network model, it is very difficult to control transmission throughput, delay and packet dropping [3].

To solve this problem, the Internet Engineering Task Force (IETF) proposed Active Queue Management (AQM) mechanisms [4]. These mechanisms preemptively drop packets before queue overflow occurs. In addition, the rejection of a packet should force the sender to reduce the transmission speed, which is provided by TCP congestion window mechanism [5]. The AQM algorithms used with TCP can enhance the efficiency of network transmission [4].

One of the first active queue management algorithm—Random Early Detection (RED) [6]—was proposed in 1993 by Sally Floyd and Van Jacobson. This mechanism estimates the packet dropping probability, which depends on the queue length. Despite the advantages of the RED algorithm, it also has some limitations. One of them is the problem of adjusting parameters to varying network traffic. Furthermore, the efficiency of the RED mechanism is closely related to the current network conditions [7]. There are many improvements and modifications of the classic RED algorithm [8–13] but none of them fully solves these problems. Performance of all RED family algorithms depends on coefficients of the dropping packet probability function. These coefficients should differ depending on the parameters of traffic such as intensity, burstiness or long-term dependence [14]. Article [15] presents the algorithm of finding the optimal parameters using the Hooke-Jeeves optimizing method. One of the newest solutions combines AQM mechanisms with a well-known method adopted from the theory of Automatic Control-PI controller. In this context, the information obtained from a classic PI controller is used as a packet dropping function [16–18]. The article [19] highlights the advantages of the PIE (Proportional Integral Enhanced Controller) algorithm. The authors state that mechanism easily adapts to varying transmission conditions and turned out to be a compromise between the degree of queue utilization and transmission delays.

The literature states that non-integer order controllers may have better performance than classic integer order ones. The first implementation of the fractional order PI controller used in queue management was presented in [20]. Our previous articles [21] investigate the performance of a fractional order PI controller (PI^{α}) utilized as an Internet traffic controller.

Increase in popularity of machine learning methods may enable the creation of a more efficient AQM mechanism. Artificial Neural Networks (ANNs) are a powerful tool with high ability to recognize patterns, even in the case of incomplete and partially distorted training data [22]. One of their applications is time series processing and analysis, which is applied in many different fields. To process time-series data with Artificial Neural Networks, different types of network layers can be used, namely Recurrent layers (including Long-Short Term Memory (LSTM) layers and Gated Recurrent Unit (GRU) Layers) and 1D Convolutional Neural Networks (CNNs). Here, CNNs can be used as a fast alternative to recurrent layers [22]. Paper [23] proposes the CNN model for processing data from time series and forecasting prices in financial markets. In a different work, [24], CNNs were used to discover the network attacks, namely Distributed Denial of Service (DDoS) attacks. Additionally, our previous work [25] uses ANNs to examine the self-similar properties of the network traffic expressed by the Hurst parameter H. This approach also uses Convolutional Neural Networks. The promising results obtained in this work prompted us to create an efficient adaptive algorithm of Active Queue Management based on Convolutional Neural Networks.

Mechanisms that select AQM parameters based on the decisions of neural networks have been proposed in the literature [26,27]. Nevertheless, these methods are based on reinforcement learning. This paradigm relies on trial-and-error to make a specific decision in each iteration of the algorithm. The neural network receives feedback (i.e., queue length) after each step, which is then used to evaluate the previously made decision. Based on this feedback, the ANN changes its weights to optimize the accuracy of the decision-making process [28]. Thus, the configuration of the neural network varies depending on the current queue occupancy.

Our contribution. The aim of the work is to propose an algorithm for Active Queue Management based on supervised learning paradigm. We use a previously trained Convolutional Network to manage the queue. The ANN is trained based on the data obtained in simulations. We observe the impact on the behavior of the AQM mechanism based on the PI^{α} controller. In experiments we change the intensity and degree of self-similarity of network sources and observe behavior of the controller. The samples contain the sequence of incoming packets and the probability of packet dropping. The model trained

this way is used as a new AQM mechanism. This paper presents its influence on the Internet transmission.

The remainder of the paper is organized as follows: Section 2 describes the current state of the art in this field. Section 3 presents the theoretical background. Section 4 is a description of the structure of the Artificial Neural Network, the data and the experimental methods used to obtain the results for this research. In Section 5 there is a description of the results of the conducted experiments. Section 6 concludes our research.

2. Related Works

There are many works regarding new AQM algorithms. These mechanisms are compared with existing solutions in terms of transmission parameters such as total number of dropped packets, average queue length, or transmission delays. In the article [7] passive and active queue management mechanisms were compared. Other works focus only on the comparison of the AQM mechanisms [6,29]. The topics of research in network and computer system performance evaluation also include works considering the impact of self-similarity of network traffic on transmission efficiency [10].

Additionally, the fractional order PI controller [30] is used for the Active Queue Management. This research is still under development, and its mechanisms have also been subjected to an analysis of the effect of the degree of self-similarity and long-term dependence of the traffic [31].

A separate group includes studies that have used neural networks to improve the queue management mechanism in TCP networks. The article [32] proposed the AQM mechanism based on reinforcement learning—Q-learning RED. The authors of [33] proposed an ANB-AQM mechanism, in which a back-propagation algorithm was used to train the neural networks to make decisions about accepting or rejecting packets. Article [34] proposes a neural network model, which modifies the REM algorithm, called the Fuzzy Neuron REM (FNREM) mechanism. This mechanism modifies the value of the proportional integral of the REM algorithm, by using the value of the proportional-integral derivative neuron as an indicator of overload.

ANNs were also used to create a new algorithm—Adaptive Neuron Proportional Integral Differential (ANPID) [35]. This mechanism used a single neuron to tune the PID controller coefficients. The authors of [36] presented the results based on the simulation and the real tests in the Linux Kernel, which resulted in the presentation of another adaptive modification of the PID controller using neural networks—the GRPID mechanism.

In article [37] authors presented an improved PID AQM/TCP system based on the network built using the Long Short-Term Memory (LSTM) layers (a specific type of a recurrent layer). It allows to predict queue length in the next step. They used Root Mean Square Error (RMSE) as a loss function. LSTM layers were also used to predict the occurrence of transmission overloads [38].

The research presented in XuIeee is an example of an attempt to use unsupervised learning to create a more efficient AQM mechanism. For that purpose, the Hebbian Learning rule is used and a new adaptive PHAQM algorithm is presented.

Bisoy and Pattnaik [39] used feed-forward neural network to create an AQM mechanism, namely FFNN-AQM. The network consisted of two input neurons, three neurons in a single hidden layer and the single output neuron.

Zhou et al. [40] also presented an adaptive AQM mechanism based on a single neuron whose weights were selected using reinforcement learning rules. The application of reinforcement learning was also used in [41] to build a mechanism to reduce transmission delays.

There are many works on the topic of AQMs based on neural networks. However, in these works, in contrast to our approach, the neural networks were mainly created using reinforcement learning. In addition, the research results did not consider the analysis of the effect of traffic self-similarity and long-term dependence on transmission efficiency.

3. Theoretical Background

Self-similarity is widely observed in nature, but the term itself was introduced by Mandelbrot in 1960s and it generally means that the portion of the whole object can be considered an image of the whole in a reduced scale. The object is self-similar, when it exhibits the same statistical properties independently of the scale. Mandelbrot described it on the example of the scaled coastlines, which also exhibited self-similarity. This property can also be used in the case of time-series analysis. The degree of self-similarity in this case determines whether Long-Range Dependence (LRD) and Short-Range Dependence (SRD) occur in data. These relationships were observed as early as the middle of the twentieth century, when Sir H. E. Hurst described the occurrence of long-range dependence based on the value of water level fluctuations in the Nile River. Although the terms of self-similarity and LRD are sometimes used interchangeably, they are not the same [42].

A continuous-time series $Y(t)$ is exactly self-similar when the following condition is satisfied:

$$Y(t) \stackrel{d}{=} a^{-H} Y(at), \tag{1}$$

for $t \geq 0, a \geq 0$ and $0 < H < 1$. It results in the statistical invariability in different time scales. H is usually used to denote the Hurst exponent/parameter, which expresses the degree of self-similarity. The parameter can take values from range $(0;1)$, and specific values represent:

- $H \in (0;0.5)$: negative correlation—the LRD does not occur (the SRD occurs).
- $H = 0.5$: no correlation.
- $H \in (0.5;1)$: positive correlation—the LRD occurs.

It was first proven in [43] that actual network traffic exhibits self-similarity. This work provided the motivation for numerous studies that demonstrated the significant impact of self-similarity on TCP transmissions [44], or to confirm its occurrence in Wide Area Networks (WANs) [45]. Self-similarity results in performance degradations, such as mean queue length enlargement and the increase in packet loss probability [42]. The topic of self-similarity is still relevant in the literature and found its application in e.g., DoS attack detection (e.g., [46]). Our previous works were also related to this topic. They regarded determining the degree of traffic self-similarity expressed by the Hurst parameter and also using data obtained from the IITiS data traffic traces to examine self-similar properties [25]. Self-similarity significantly impacts queue occupancy and transmission performance [47]. For that reason, the samples generated for the purpose of this article are characterized by different degrees of self-similarity.

Artificial Neural Networks have found application in many different domains, e.g., image classification, natural language processing, signal processing etc. Additionally, Deep Learning approaches have become a solution to many problems due to their better ability to extract patterns than shallow learning [48]. The versatility of neural networks has resulted in them also being frequently used in the network traffic domain for tasks including attack detection [49,50], traffic generation [51] and classification of the traffic type [52].

Network traffic and its features are often represented as a time series. To process time-series data with Artificial Neural Networks, different types of networks (e.g., Autoencoders) and layers can be used, namely Recurrent layers (including Long-Short Term Memory (LSTM) layers and Gated Recurrent Unit (GRU) Layers) and 1D Convolutional Neural Networks.

Autoencoders can be built using different types of layers, e.g., Dense Layers or Convolutional layers. The goal of this type of network is to compress input data and then reconstruct it on output [53]. It can be used for the purpose of data denoising, but also anomaly detection. When the neural network is not able to reconstruct the input data well, it suggests that the sample can be anomalous [53].

LSTM layers are often used for the purpose of time-series data processing. Single LSTM units solve the gradient vanishing and exploding issues typical for simple Recurrent Layers and are able to propagate gradients over a long period of time [54]. The key

characteristic of this type of layer is that they store the internal state, which enables them to 'remember' the past information [52]. Due to that their internal 'memory' is longer than in the traditional recurrent units.

The alternative for LSTM layers is a GRU layer. It is very similar to LSTM layer, also stores the Long-Time memory of the past information, which is vital for time-series processing. Nevertheless, it is simpler to implement and compute than the LSTM layer, thus more efficient [55].

Additionally, convolutional layers can be used to process time-series data. In this case, time has to be treated as a spatial dimension [22]. In fact, it is an efficient alternative to recurrent layers. In a Convolutional Neural Network, transformed time-series data are processed in turns using convolutional and pooling layers. As a result deep, more abstract representations are generated on the basis of raw data. Processing ends with a classifier part (Multi-Layer Perceptron), which consists of dense layers.

4. Data Preparation and Neural Networks Training Process

In this paper, we used artificial neural network models to develop an active queue management mechanism. The neural networks were trained to mimic the operation of the AQM based on the fractional order PI^{α} controller mechanism. The training data were generated based on simulation data, and a detailed description of the learning model is given in this section.

The neural network model was based on four convolutional layers and two dense layers. After each convolutional layer, the data were normalized and the results were averaged. Additionally, a dropout layer was placed to prevent over-fitting to the learning data. Python and Keras libraries were used to implement the model. The conceptual structure of the model used in this paper is presented in Figure 1. To design this model structure we relied on the experience of our earlier work [25], where the degree of self-similarity of network traffic was classified using Convolutional Neural Networks expressed by Hurst parameter.

In order to prepare the training set for the proposed neural network model, network simulations were performed, reflecting the queueing behavior of a fractional order controller PI^{α}. The values of the fractional order PI^{α} controller parameters have been presented in the Table 1. These values were determined based on our previous work [26]. The results of these articles have shown that the choice of controller parameters significantly affects the queue length control properties. The process of choosing proper AQM/PI controller parameters is non-trivial. It has a significant impact on the packet dropping function (i.e., for an integral order α it can strengthen and accelerate the response of a controller). Properly selected AQM parameters should allow us to obtain adaptation to the changing transmission conditions and desired queue behavior. We discussed the influence of these parameters on queue behavior in papers [15]. The controller parameters were chosen in such a manner that controller $PI^{\alpha}1$ was the weakest controller, and controller $PI^{\alpha}3$ was the strongest one, which implies a large number of packet rejections and ease of maintaining the desired queue length.

Table 1. The PI^{α} controller parameters.

	K_P	K_I	α
$PI^{\alpha}1$	0.0001	0.0004	−0.4
$PI^{\alpha}2$	0.0001	0.0004	−0.5
$PI^{\alpha}3$	0.0001	0.0004	−0.6

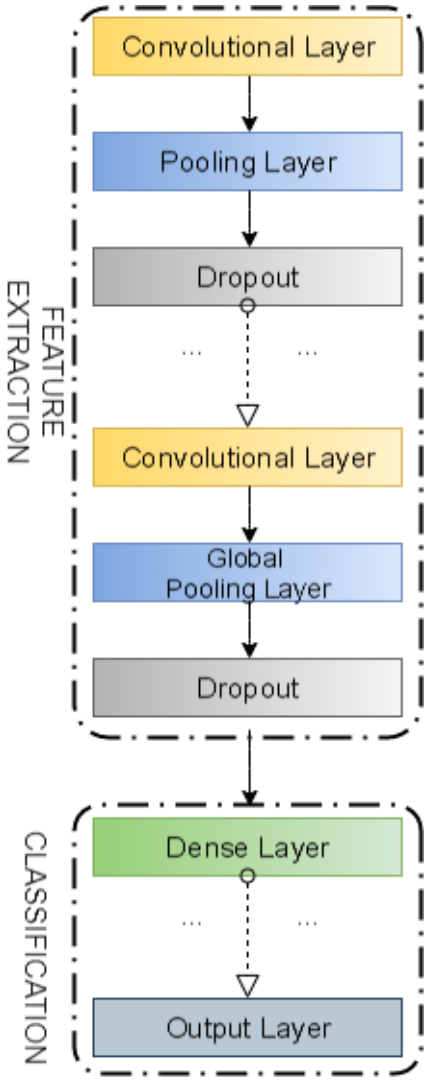


Figure 1. The conceptual structure of a Convolutional Neural Network based classifier used to model an Active Queue Management mechanism.

To obtain training data for an AQM model based on Convolutional Networks, network simulations were performed using the AQM mechanism. For this purpose, the discrete event simulator SimPy (written in Python) was used. This software is available under the MIT License and has been used in our previous works regarding the evaluation of AQMs [21,26].

Our simulation model was a discrete model of a G/M/1/N queue. The simulation time was divided into discrete time intervals of length dt. Arrival of a packet was generated (or not) in a given time slot by a traffic source. The source of traffic was self-similar and based on Fractional Gaussian Noise (FGN) process. The advantages of such a source have been described previously in the articles [10,15,25].

All experiments considered different degrees of traffic self-similarity expressed using Hurst parameter. In experiments the Hurst parameter changed between $H = 0.5$ (no correlation) and $H = 0.9$ (high degree of LRD).

The input intensity coefficient was set to a constant value $\lambda = 0.5$. Thus, the simulation packet source always had a constant intensity. Parameter μ represents the time of packet processing and dispatching (probability of taking a packet from the queue). Different values

of this coefficient were used in the experiments. The parameter μ took values between $\mu = 0.5$ (moderately stressed system) to $\mu = 0.15$ (highly stressed system). This choice of simulation parameters allowed us to observe all properties of the AQM mechanism.

In our experiments, we considered different numbers of items from queue occupancy history taken into consideration in the samples used to train Convolutional Networks. For simplicity, we refer to this number of samples as 'CNN History'. This length corresponded to the number of time slots in the simulation model that were used as training data for the network. For example $CNN = 200$ refers to $200 * dt$ time intervals taken into consideration. Throughout this time, we observed the behavior of the AQM queue.
Thus, the training data consisted of:

1. Learning features:
   (a) The last n items from the queue's occupancy history (CNN History).
   (b) History of packet rejections in n last queue states
   where $n \in [20; 100; 200; 300; 400; 500; 1000]$.
2. Classes:
   (a) 11 labels that mapped the probability of packet rejection to the current transmission conditions, according to the principle shown in Table 2.

Table 2. Decision class labels representing ranges of probabilities of packet being dropped.

Decision Class	Probability Interval [%]
1	[0;5)
2	[5;15)
3	[15;25)
4	[25;35)
5	[35;45)
6	[45;55)
7	[55;65)
8	[65;75)
9	[75;85)
10	[85;95)
11	[95;100]

Therefore, we considered different lengths of queue occupancy history, because from the perspective of the router, which is a low resource device, minimizing the length of the history would be beneficial. In our study, we tried to determine the minimum acceptable length of n last items of the queue's occupancy history.

For each probability interval, one million one-dimensional learning records were prepared. Therefore, the training set consisted of 11 million records. They contained transmission information such as the length of the queue in each consecutive time slot, the number of dropped packets, and the value of the PI^{α} controller's packet rejection probability function. We present the process of data preparation in Figure 2. This amount of data seemed to be sufficient in comparison with the cardinality of data reported in the literature [56].

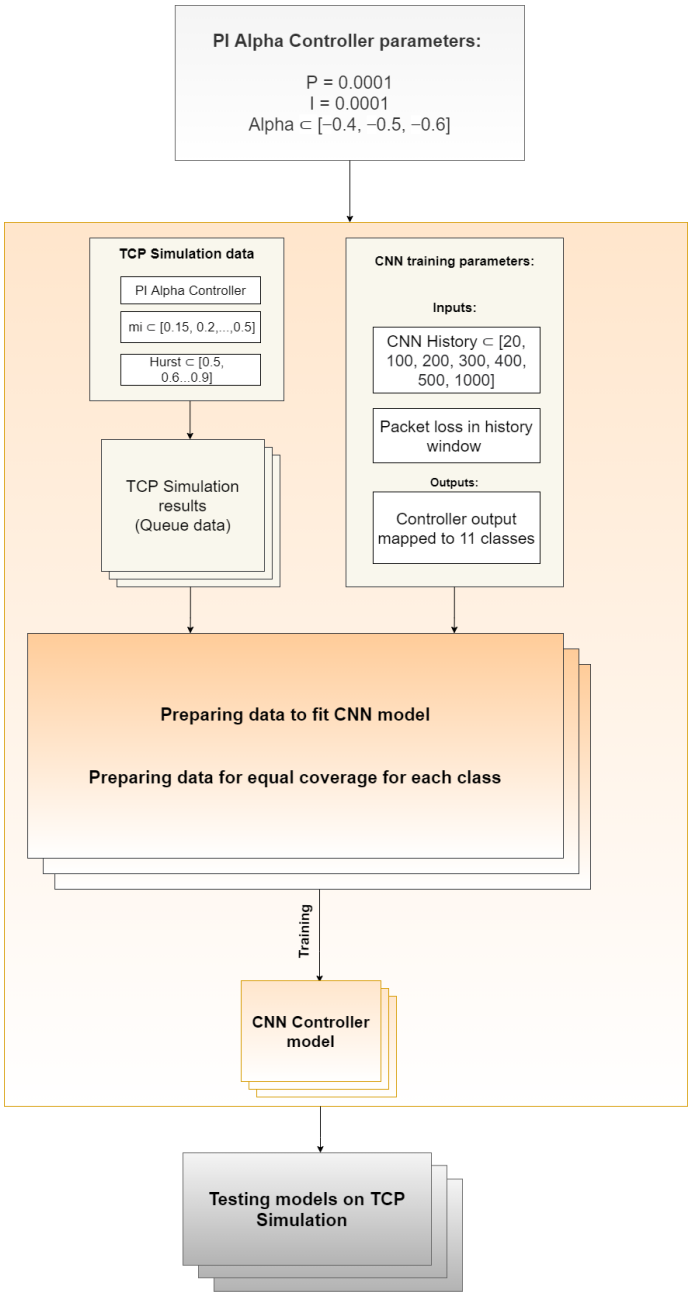


Figure 2. Training data preprocessing process.

Input data prepared in such a manner were used in the process of supervised learning of the neural network models. In order to train the model and minimize the cost function, the optimizer Adaptive Moment Estimation (Adam) was used with the following parameters:

$$\eta = 10^{-3}, \beta_1 = 0.9, \beta_2 = 0.999 \tag{2}$$

where: η is the learning rate, β_1 is the exponential decay rate for the first moment estimates and β_2 is the exponential decay rate for the second moment estimates. The Adam optimizer is expressed by the equation [57]:

$$\begin{aligned} v_t &= \beta_1 v_{t-1} + (1 - \beta_1) g_t \\ s_t &= \beta_2 s_{t-1} + (1 - \beta_2) g_t \end{aligned} \tag{3}$$

where v is the first moment, which resembles momentum that records the past normalized gradient, s is the second moment and g denotes the gradient descent.

In both the four convolutional layers and the two dense layers, ReLU was used as the activation function and Sigmoid/Softmax functions were used to determine the activation of the output layer. Categorical cross-entropy was used as a cost function. Figure 1 shows the conceptual structure of a neural network model used for the purpose of active queue management mechanism.

We limited the training process to 10 epochs. This value was sufficient, since the values start to stabilize after only 5–6 epochs, as confirmed by the results in Tables 3–5. We also compare the accuracy of the model, when Softmax activation function (Table 3) and Sigmoid activation function (Table 4) were used in the output layer. Higher results were obtained for the Sigmoid function.

In the case of Softmax function (Table 3), the minimum accuracy was 32.3%, and the maximum 58.9%. For the models in which we applied the Sigmoid activation function for the last layer the minimum accuracy was 48.77% (for the network trained on the data from the $PI^{\alpha}3$ controller, where the CNN History $= 20$), and the maximum 89.46% (for the network trained on the data from the $PI^{\alpha}3$ controller, where the CNN History $= 1000$). Taking all the results into consideration, the best results were obtained for the CNN History ≥ 500, and the worst for the CNN History < 100 (Table 4).

In the case of the model trained on data representing the behavior of three controllers simultaneously and the use of the Sigmoid activation function of the output layer, the maximum accuracy was 72.1% for the CNN History ≥ 500 (see Table 5).

Table 3. The accuracy measurements for testing the CNN model trained on data regarding three $PI^{\alpha}1$, $PI^{\alpha}2$ and $PI^{\alpha}3$ controllers, n last items in queue occupancy history taken into consideration (we used Softmax function as an activation function of the last layer).

		Softmax						
		***n* History Length**						
		20	**100**	**200**	**300**	**400**	**500**	**1000**
CNN by behavior $PI^{\alpha}1$	5 epochs	52.27	54.48	54.50	56.67	58.40	58.81	51.59
	6 epochs	52.36	54.88	54.68	56.70	58.42	58.84	51.72
	10 epochs	52.40	55.50	54.83	56.74	58.47	58.90	51.82
CNN by behavior $PI^{\alpha}2$	5 epochs	48.37	47.23	40.29	41.99	43.68	44.06	43.94
	6 epochs	48.45	47.61	40.32	42.06	43.74	44.06	44.00
	10 epochs	48.54	48.42	40.31	42.30	43.78	44.24	44.04

Table 3. *Cont.*

		Softmax						
		***n* History Length**						
		20	**100**	**200**	**300**	**400**	**500**	**1000**
CNN by behavior $PI^{\alpha}3$	5 epochs	47.07	41.83	33.80	32.30	32.92	33.79	38.45
	6 epochs	47.18	42.09	33.81	32.31	32.93	33.82	38.44
	10 epochs	47.41	42.62	34.18	32.32	32.92	33.83	38.50

Table 4. The accuracy measurements for test data for a neural network model trained with data representing the behavior of $PI^{\alpha}1$, $PI^{\alpha}2$ and $PI^{\alpha}3$ controllers, n last items in queue occupancy history taken into consideration (we used Sigmoid function as an activation function of the last layer).

		Sigmoid						
		***n* Last Items in Queue Occupancy History**						
		20	**100**	**200**	**300**	**400**	**500**	**1000**
CNN by behavior $PI^{\alpha}1$	5 epochs	55.98	73.95	80.76	83.67	85.22	86.15	88.46
	6 epochs	56.04	74.06	80.88	83.80	85.39	86.31	88.74
	10 epochs	56.16	74.38	81.19	84.13	85.69	86.64	89.46
CNN by behavior $PI^{\alpha}2$	5 epochs	50.34	70.70	75.94	76.92	77.03	77.20	83.71
	6 epochs	50.38	70.83	76.08	77.06	77.18	77.32	84.04
	10 epochs	50.53	71.13	76.40	77.36	77.57	77.79	84.81
CNN by behavior $PI^{\alpha}3$	5 epochs	48.77	67.16	67.54	65.65	64.39	64.04	77.46
	6 epochs	48.82	67.29	67.70	65.83	64.57	64.24	77.76
	10 epochs	49.06	67.60	68.03	66.23	64.99	64.68	78.68

Table 5. The accuracy measurements for test data for a neural network model trained with data representing the behavior of three controllers simultaneously, given n recent queue occupancy history elements.

		Sigmoid						
		***n* Last Items in Queue Occupancy History**						
		20	**100**	**200**	**300**	**400**	**500**	**1000**
CNN by behavior 3 PI^{α}	5 epochs	49.52	67.03	68.55	68.48	68.40	68.42	70.98
	6 epochs	49.58	67.12	67.70	68.65	68.60	68.61	71.30
	10 epochs	49.74	67.34	68.99	69.03	69.15	69.21	72.10

5. Evaluation of the Neural Network-Based AQM

This section presents the behavior of the trained neural network (as assumed in Section 4 and evaluates its effectiveness as an AQM mechanism. This evaluation was performed using previously described simulation mechanisms. During the study, we evaluated the number of packets dropped from the queue and the average queue occupancy. We compared the effectiveness of the neural network-based AQM mechanism with the results of the PI^{α} controller-based AQM mechanism. We used the network traffic with different degrees of self-similarity during the experiments.

To increase the readability of the paper, we present only two extreme cases - the results obtained for a non-self-similar traffic ($H = 0.5$) and for a traffic with high degree of LRD ($H = 0.9$).

The intensity of the packet source in the simulation was assumed to be ($lambda = 0.5$). On the other hand, the packet service time in a system was set to a constant value ($\mu = 0.25$) in order to obtain a heavily loaded system.

In our experiments, we evaluated four separate neural network models. The first three neural networks were trained with the data obtained from controllers $PI^{\alpha}1$, $PI^{\alpha}2$, and $PI^{\alpha}3$. The fourth model was trained with data regarding all of these controllers. In the first phase of the experiment, we considered two neural network models (see Figure 1): the first one with Softmax, and the second one with Sigmoid activation function of the last layer.

A comparison of Tables 3 and 4 shows that although Softmax function is more commonly used in the literature as an activation function of the output layer of the neural network for multiclass classification, Sigmoid function performs better in our case. In the worst case, in which the network obtained accuracy of 32.31%, changing the activation function to Sigmoid resulted in significant accuracy increase (65.65%). Additionally, in the best obtained case accuracy changed from 58.90% to 86.65%. Figures 3 and 4 show average queue lengths for AQM mechanism based on neural network. Detailed results are compared on Tables 6 and 7 for Sigmoid function and on Tables 8 and 9 for Softmax function. Both presented networks imitate the behavior of the first controller—PI^{α} (see Table 1). Comparing the number of discarded packets and the average queue sizes, we find that they are similar regardless of the chosen network activation function in the last layer. As Hurst increases, the number of dropped packets decreases slightly in the case of Sigmoid function (<1%).

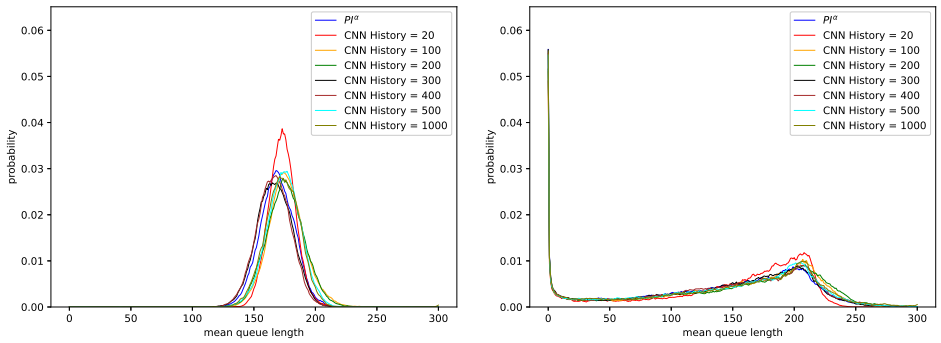


Figure 3. Distribution of queue length obtained for CNN model with the last layer activation function Sigmoid, trained using data regarding $PI^{\alpha}1$ controller and parameters: $K_P = 0.0001$, $K_I = 0.0004$, $\alpha = -0.4$, $H = 0.5$ (**left**), $H = 0.9$ (**right**).

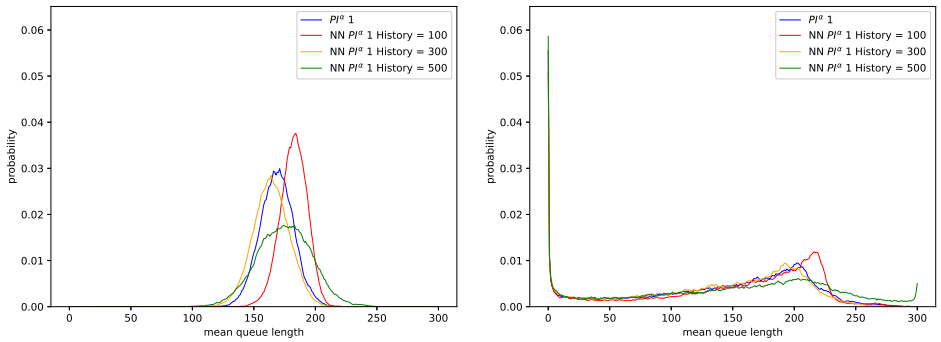


Figure 4. Distribution of queue length obtained for CNN model with the last layer activation function Softmax, trained using data regarding $PI^{\alpha}1$ controller and parameters: $K_P = 0.0001$, $K_I = 0.0004$, $\alpha = -0.4$, $H = 0.5$ (**left**), $H = 0.9$ (**right**).

Taking into consideration higher accuracy obtained using Sigmoid function, we chose this function to be used in further experiments.

Figure 3 compares the behavior of two AQM mechanisms: $PI^{\alpha}1$ controller and the CNN-based AQM trained on the data reflecting the behavior of this controller.

For the CNN model, different lengths of the last n elements of the queue occupancy history (input to the neural network) were considered. Regardless of the value of n, the resulting queue length distributions are similar to the queue length distribution of the $PI^{\alpha}1$ controller. For Poisson traffic (non-self-similar traffic, $H = 0.5$), the average queue length oscillates between 166 and 176 packets (see Table 6). For highly self-similar traffic (parameter $H = 0.9$), the average queue length was between 139 and 147 (see Table 7). In this case, all the Convolutional Neural Network models (with different numbers of CNN History) obtained larger values of the average queue length, with fewer packets dropped, than the $PI^{\alpha}1$ mechanism.

Figure 5 presents the results for stronger AQM mechanism $PI^{\alpha}2$. The detailed results of dropped packets numbers and queue lengths are presented in Tables 10 and 11. Because of the fact that the $PI^{\alpha}2$ controller was stronger than the one presented above, the obtained average queue lengths were smaller.

Table 6. Detailed results of queue occupancy obtained for CNN model with the last layer activation function Sigmoid, trained using data regarding $PI^{\alpha}1$ controller and parameters: $K_P = 0.0001$, $K_I = 0.0004$, $\alpha = -0.4$ and $H = 0.5$.

AQM	Packet Dropped	Average Queue Length
$PI^{\alpha}1$	249,878	168.98
CNN History = 20	251,198	172.97
CNN History = 100	248,936	176.45
CNN History = 200	250,063	175.69
CNN History = 300	249,510	166.87
CNN History = 400	250,104	166.23
CNN History = 500	250,800	174.31
CNN History = 1000	249,561	173.33

Table 7. Detailed results of queue occupancy obtained for CNN model with the last layer activation function Sigmoid, trained using data regarding $PI^{\alpha}1$ controller and parameters: $K_P = 0.0001$, $K_I = 0.0004$, $\alpha = -0.4$ and $H = 0.9$.

AQM	Packet Dropped	Average Queue Length
$PI^{\alpha}1$	263,387	139.16
CNN History = 20	261,678	145.66
CNN History = 100	262,304	145.81
CNN History = 200	262,518	147.22
CNN History = 300	262,935	139.28
CNN History = 400	263,872	140.89
CNN History = 500	263,440	142.22
CNN History = 1000	263,654	143.14

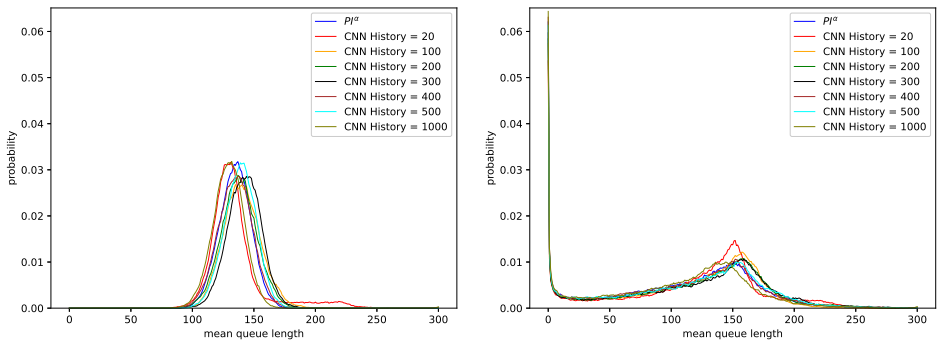


Figure 5. Distribution of queue length obtained for CNN model with the last layer activation function Sigmoid, trained using data regarding $PI^{\alpha}2$ controller and parameters: $K_P = 0.0001$, $K_I = 0.0004$, $\alpha = -0.5$, $H = 0.5$ (**left**), $H = 0.9$ (**right**).

Table 8. Detailed results of queue occupancy obtained for CNN model with the last layer activation function Softmax, trained using data regarding $PI^{\alpha}1$ controller and parameters: $K_P = 0.0001$, $K_I = 0.0004$, $\alpha = -0.4$ and $H = 0.5$.

AQM	Packet Dropped	Average Queue Length
$PI^{\alpha}1$	249,878	168.98
CNN History = 100	250,038	182.21
CNN History = 300	250,455	164.06
CNN History = 500	250,017	174.73

Table 9. Detailed results of queue occupancy obtained for CNN model with the last layer activation function Softmax, trained using data regarding $PI^{\alpha}1$ controller and parameters: $K_P = 0.0001$, $K_I = 0.0004$, $\alpha = -0.4$ and $H = 0.9$.

AQM	Packet Dropped	Average Queue Length
$PI^{\alpha}1$	263,387	139.16
CNN History = 100	261,271	148.89
CNN History = 300	263,609	134.64
CNN History = 500	264,569	145.64

Table 10. Detailed results of queue occupancy obtained for CNN model with the last layer activation function Sigmoid, trained using data regarding $PI^{\alpha}2$ controller parameters: $K_P = 0.0001$, $K_I = 0.0004$, $\alpha = -0.5$ and $H = 0.5$.

AQM	Packet Dropped	Average Queue Length
$PI^{\alpha}2$	250,314	134.72
CNN History = 20	249,610	135.27
CNN History = 100	250,657	140.37
CNN History = 200	249,633	137.95
CNN History = 300	249,752	142.29
CNN History = 400	248,852	134.86
CNN History = 500	249,960	138.85
CNN History = 1000	249,744	129.60

Figure 6 compares the last pair of controllers: controller $PI^{\alpha}3$, with the corresponding models based on Convolutional Neural Networks. The results prove that this controller is the strongest one. The AQM mechanism increased considerably the number of dropped packets and decreased the obtained queue lengths. In the case of traffic without LRD (see Table 12, for parameter $H = 0.5$) the average queue occupancy oscillateds between 116 and 139 packets, and in the case of traffic characterized by a high degree of LRD (see Table 13, for parameter $H = 0.9$) between 94 and 121 packets.

It should be noted that for all three CNN-based AQM models, a more efficient AQM model was obtained compared to the controllers that were used to create the test data. Even for the model that obtained the smallest accuracy during the learning process (48.77%, see Table 4), based on non-integer controller data of order $PI^{\alpha}3$, for CNN History < 100, the obtained average queue length was larger than for the base mechanism $PI^{\alpha}3$. This situation occurred both for traffic without LRD (see Table 12) and for traffic characterized by a high degree of LRD (see Table 13).

Table 11. Detailed results of queue occupancy obtained for CNN model with the last layer activation function Sigmoid, trained using data regarding $PI^{\alpha}2$ controller parameters: $K_P = 0.0001$, $K_I = 0.0004$, $\alpha = -0.5$ and $H = 0.9$.

AQM	Packet Dropped	Average Queue Length
$PI^{\alpha}2$	264,819	109.37
CNN History = 20	263,014	117.57
CNN History = 100	264,135	112.98
CNN History = 200	264,217	115.69
CNN History = 300	264,668	116.24
CNN History = 400	265,538	110.05
CNN History = 500	264,533	112.47
CNN History = 1000	265,839	105.25

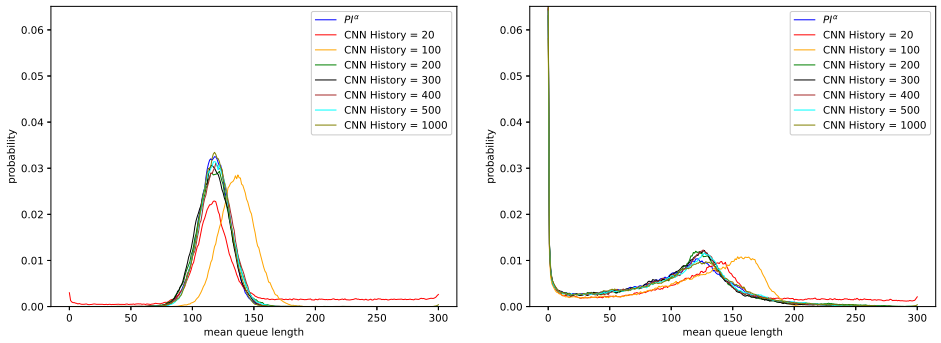


Figure 6. Distribution of queue length obtained for CNN model with the last layer activation function Sigmoid, trained using data regarding $PI^{\alpha}3$ controller and parameters: $K_P = 0.0001$, $K_I = 0.0004$, $\alpha = -0.6$, $H = 0.5$ (**left**), $H = 0.9$ (**right**).

Table 12. Detailed results of queue occupancy obtained for CNN model with the last layer activation function Sigmoid, trained using data regarding $PI^{\alpha}3$ controller and parameters: $K_P = 0.0001$, $K_I = 0.0004$, $\alpha = -0.6$ and $H = 0.5$.

AQM	Packet Dropped	Average Queue Length
$PI^{\alpha}3$	250,840	117.53
CNN History = 20	251,892	139.26
CNN History = 100	250,362	136.35
CNN History = 200	248,878	117.67
CNN History = 300	250,533	116.40
CNN History = 400	250,011	118.85
CNN History = 500	250,166	118.67
CNN History = 1000	249,801	118.20

Table 13. Detailed results of queue occupancy obtained for CNN model with the last layer activation function Sigmoid, trained using data regarding $PI^{\alpha}3$ controller and parameters: $K_P = 0.0001$, $K_I = 0.0004$, $\alpha = -0.6$ and $H = 0.9$.

AQM	Packet Dropped	Average Queue Length
$PI^{\alpha}3$	265,707	95.13
CNN History = 20	265,737	121.19
CNN History = 100	263,952	110.79
CNN History = 200	265,383	97.28
CNN History = 300	266,295	94.09
CNN History = 400	266,366	95.79
CNN History = 500	265,592	97.31
CNN History = 1000	266,184	96.90

In the next simulation step, we evaluated the AQM-CNN mechanism whose learning data were generated from the behavior of all three PI^{α} controllers. Figure 7 shows the queue distribution, and Figure 8 shows the changes in queue occupancy over time. Details of the number of packets dropped and the resulting average queue occupancy are presented in Table 14 for the traffic without LRD and in Table 15, for traffic with a high degree of LRD.

The results show that when the number of last n elements of queue occupancy history taken as a CNN input is too small (CNN History < 100), then, independent of the degree of self-similarity of the traffic, the number of dropped packets, and the average queue length, approximates the results obtained using the sets of controllers $PI^{\alpha}2$ and $PI^{\alpha}3$ (see Tables 10–13).

On the other hand, when the considered number of last n queue occupancy history elements is larger (CNN History ≥ 100), the obtained average queue length increases by 46 packets for traffic without LRD (Table 14, for $H = 0.5$), or by 32 packets, for traffic characterized by a high degree of LRD (see Table 15, for $H = 0.9$). This means that the resulting queue distribution matches the one of the original and the most efficient controller $PI^{\alpha}1$ (Figure 3).

This feature indicates that for the AQM model based on Convolutional Networks, as the number of story elements used increases, the ability of the mechanism to adapt to current Internet transmission conditions also improves.

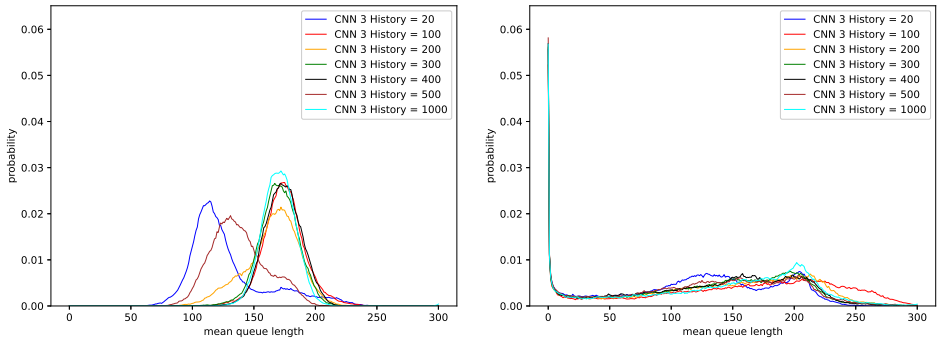


Figure 7. Distribution of queue length obtained for CNN controller trained using data regarding three PI^{α} controllers, $H = 0.5$ (**left**), $H = 0.9$ (**right**).

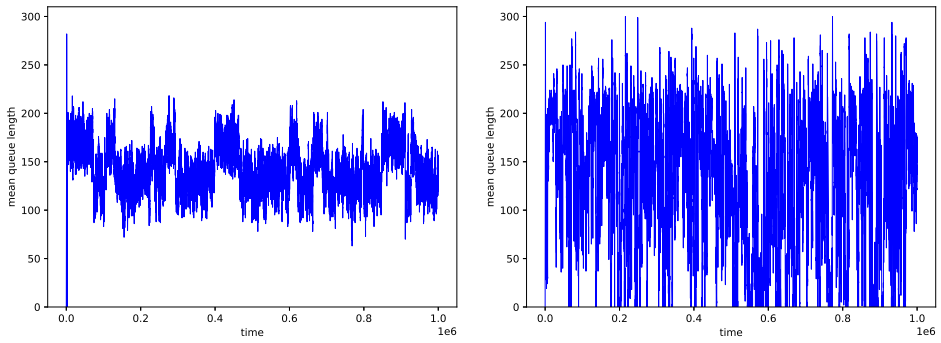


Figure 8. Queue occupancy obtained for CNN controller trained using data regarding three PI^{α} controllers, with CNN History $= 500$, $H = 0.5$ (**left**), $H = 0.9$ (**right**).

Table 14. Detailed results of queue occupancy results obtained for CNN model trained using data regarding three PI^{α} controllers and $H = 0.5$.

AQM	Packet Dropped	Average Queue Length
CNN 3 History = 20	249,727	128.75
CNN 3 History = 100	249,988	174.64
CNN 3 History = 200	250,583	164.87
CNN 3 History = 300	249,907	169.98
CNN 3 History = 400	249,593	173.41
CNN 3 History = 500	250,157	138.08
CNN 3 History = 1000	249,334	170.46

Table 15. Detailed results of queue occupancy results obtained for CNN model trained using data regarding three PI^{α} controllers and $H = 0.9$.

AQM	Packet Dropped	Average Queue Length
CNN 3 History = 20	262,841	120.94
CNN 3 History = 100	263,859	152.15
CNN 3 History = 200	262,298	137.21
CNN 3 History = 300	263,205	131.49
CNN 3 History = 400	263,818	129.81
CNN 3 History = 500	263,872	127.37
CNN 3 History = 1000	263,554	138.32

6. Conclusions

The paper presents a new Active Queue Management mechanism based on Convolutional Neural Networks and supervised learning.

To train the Convolutional Networks used in the experiments, data obtained through simulation have been used. The training data of the CNN model reflect the behavior of the AQM mechanism, based on a fractional order controller PI^{α}.

In our experiments, we took into account the effect of the degree of traffic self-similarity and long-term dependence on the performance of the proposed mechanism.

We also considered the effect of the number of last n elements of the queue occupancy history, used as input of the neural network, on the efficiency of the proposed mechanism.

The best results were obtained for CNN History = 500. The minimum length of CNN History for which results are still acceptable is 100.

In the experiments, neural networks with different number of convolutional layers and different optimizers and cost functions were considered to build the AQM model. After comparing the results obtained with different activation functions, the results have shown that the most efficient model used Sigmoid activation function in the output layer, therefore we chose this function for further experiments. The decisions made in this work were also influenced by our previous work regarding traffic classification in terms of the degree of self-similarity [25].

The most efficient AQM obtained in our study was based on the Convolutional Neural Network model, trained using the data reflecting the behavior of all three PI^{α} controllers jointly.

The results confirmed that the model based on Convolutional Neural Networks can effectively reproduce the results of the classical AQM algorithm and effectively manage the data transmission. Such a model maintains the assumed average number of packets in the queue and reduces the total number of dropped packets, independent of the degree of traffic self-similarity.

It seems that the proposed mechanism exhibits some advantages over previously proposed mechanisms encountered in the literature. Our previous study [26] demonstrated that the reinforcement learning methods are well suited for maintaining the assumed queue size. However, in computer networks, the process of controlling packet traffic is more complex. The objective is to maximize the transmission efficiency. This efficiency is characterized by: throughput, delay, and possible retransmissions. Efficiency of AQM mechanisms is influenced by self-similarity of network traffic. The higher the Hurst parameter value is, the greater problems with correct packet management occur. The proposed solution addresses this problem much more effectively. The biggest disadvantage of this solution is greater computational and memory complexity of solutions based on Convolutional Neural Networks. This complexity may affect the difficulty of implementing this solution in real routers.

In our previous study [58], we used a Linux-based computer as a router. In that study, we used a special router implementation based on a special forwarding mechanism (based on the iptables mechanism), which delivered all packets to the user program implementing AQM. This solution greatly simplifies the research model. Unfortunately, the tests have shown that forwarding packets from kernel space to userspace requires a significant amount of time and is not optimal. In the target solutions the whole implementation should be realized in the kernel of the system. The implementation may be a great challenge on routers with low hardware resources. In such solutions instead of multiplication operations bit shifting is used, which causes calculation errors. For CNN calculations these errors may be too high. However, it seems that the computational power of routers will increase in the future. We want to devote a separate article to the problems of implementing AQMs in real routers.

Author Contributions: Conceptualization, J.S. and A.D.; methodology, J.S.; software, J.S.; investigation, A.D., J.S. and S.M.; validation, J.D., D.M.; project administration, J.S.; formal analysis, J.D. and K.F.; data curation, J.S. and D.M.; funding acquisition, J.S.; writing—original draft preparation, J.S., J.D., A.D. and K.F.; writing—review and editing, J.S., J.D.; visualization, J.S., D.M. and S.M.; supervision, A.D. All authors have read and agreed to the published version of the manuscript.

Funding: Publication supported by Own Schoolarship Fund of the Silesian University of Technology in year 2019/2020, grant number: 22/FSW18/0003-03/2019.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Not applicable.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Index, C.V.N. *Global Mobile Data Traffic Forecast Update, 2017–2022*; White Paper; Cisco Press: Indianapolis, IN, USA, 2019.
2. Lakshman, T.V.; Madhow, U. The performance of TCP/IP for networks with high bandwidth-delay products and random loss. *IEEE/ACM Trans. Netw.* **1997**, *5*, 336–350. [CrossRef]
3. Hema, R.M.; Murugesan, G.; Jude, M.J.A.; Diniesh, V.C.; Sree Arthi, D.; Malini, S. Active queue versus passive queue—An experimental analysis on multi-hop wireless networks. In Proceedings of the International Conference on Computer Communication and Informatics (ICCCI), Coimbatore, India, 5–7 January 2017; pp. 1–5. [CrossRef]
4. Braden, B.; Clark, D.; Crowcroft, J.; Davie, B.; Deering, S.; Estrin, D.; Floyd, S.; Jacobson, V.; Minshall, G.; Partridge, C.; et al. Recommendations on Queue Management and Congestion Avoidance in the Internet. *RFC* **1998**, *2309*, 1–17.
5. Wu, H.; Feng, Z.; Guo, C.; Zhang, Y. ICTCP: Incast Congestion Control for TCP in Data-Center Networks. *IEEE/ACM Trans. Netw.* **2013**, *21*, 345–358. [CrossRef]
6. Floyd, S.; Jacobson, V. Random Early Detection gateways for congestion avoidance. *IEEE/ACM Trans. Netw.* **1993**, *1*, 397–413. [CrossRef]
7. Xue, L. Simulation of Network Congestion Control Based on RED Technology. In Proceedings of the 2013 International Conference on Computational and Information Sciences, Shiyan, Hubai, China, 21–23 June 2013; pp. 1497–1500.
8. Bhatnagar, S.; Patro, R. A proof of convergence of the B-RED and P-RED algorithms for random early detection. *IEEE Commun. Lett.* **2009**, *13*, 809–811. [CrossRef]
9. Ho, H.J.; Lin, W.M. AURED—Autonomous Random Early Detection for TCP Congestion Control. In Proceedings of the 3rd International Conference on Systems and Networks Communications Malta, Sliema, Malta, 26–31 October 2008.
10. Domańska, J.; Augustyn, D.; Domański, A. The choice of optimal 3-rd order polynomial packet dropping function for NLRED in the presence of self-similar traffic. *Bull. Pol. Acad. Sci. Tech. Sci.* **2012**, *60*, 779–786. [CrossRef]
11. Kachhad, K.; Lathigara, A. ModRED : Modified RED an Efficient Congestion Control Algorithm for Wireless Network. *Int. Res. J. Eng. Technol. (IRJET)* **2018**, *5*, 1879–1884.
12. Hamdi, M.M.; Rashid, S.A.; Ismail, M.; Altahrawi, M.A.; Mansor, M.F.; AbuFoul, M.K. Performance Evaluation of Active Queue Management Algorithms in Large Network. In Proceedings of the 2018 IEEE 4th International Symposium on Telecommunication Technologies (ISTT), Selangor, Malaysia, 26–28 November 2018; pp. 1–6. [CrossRef]
13. Liu, Z.; Sun, J.; Hu, S.; Hu, X. An Adaptive AQM Algorithm Based on a Novel Information Compression Model. *IEEE Access* **2018**, *6*, 31180–31190. [CrossRef]
14. Tan, L.; Zhang, W.; Peng, G.; Chen, G. Stability of TCP/RED systems in AQM routers. *IEEE Trans. Autom. Control* **2006**, *51*, 1393–1398. [CrossRef]
15. Domańska, J.; Domański, A.; 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 *Communications in Computer and Information Science*; Springer International Publishing: Berlin/Heidelberg, Germany, 2018; Volume 935, pp. 76–83. [CrossRef]
16. Melchor-Aquilar, D.; Castillo-Tores, V. Stability Analysis of Proportional-Integral AQM Controllers Supporting TCP Flows. *Comput. Sist.* **2007**, *10*, 401–414. [CrossRef]
17. Ustebay, D.; Ozbay, H. Switching Resilient PI Controllers for Active Queue Management of TCP Flows. In Proceedings of the 2007 IEEE International Conference on Networking, Sensing and Control, London, UK, 15–17 April 2007; pp. 574–578. [CrossRef]
18. Melchor-Aquilar, D.; Niculescu, S. Computing non-fragile PI controllers for delay models of TCP/AQM networks. *Int. J. Control.* **2009**, *82*, 2249–2259. [CrossRef]
19. Grazia, C.A.; Patriciello, N.; Klapez, M.; Casoni, M. Which AQM fits IoT better? In Proceedings of the IEEE 3rd International Forum on Research and Technologies for Society and Industry (RTSI), Modena, Italy, 11–13 September 2017; pp. 1–6. [CrossRef]
20. Krajewski, W.; Viaro, U. On robust fractional order PI controller for TCP packet flow. In Proceedings of the BOS Conference: Systems and Operational Research, Warsaw, Poland, 24–26 September 2014.
21. Marek, D.; Domański, A.; Domańska, J.; Czachórski, T.; Klamka, J.; Szyguła, J. Combined diffusion approximation–simulation model of AQM's transient behavior. *Comput. Commun.* **2021**, *166*, 40–48. [CrossRef]
22. Chollet, F. *Deep Learning with Python*; Manning: Berkeley, CA, USA, 2017.
23. Chen, J.; Chen, W.; Huang, C.; Huang, S.; Chen, A. Financial Time-Series Data Analysis Using Deep Convolutional Neural Networks. In Proceedings of the 7th International Conference on Cloud Computing and Big Data (CCBD), Macau, China, 16–18 November 2016; pp. 87–92. [CrossRef]
24. Jia, W.; Liu, Y.; Liu, Y.; Wang, J. Detection Mechanism Against DDoS Attacks based on Convolutional Neural Network in SINET. In Proceedings of the IEEE 4th Information Technology,Networking, Electronic and Automation Control Conference (ITNEC), Chongqing China, 12–14 June 2020; Volume 1, pp. 1144–1148. [CrossRef]
25. Filus, K.; Domański, A.; Domańska, J.; Marek, D.; Szyguła, J. Long-Range Dependent Traffic Classification with Convolutional Neural Networks Based on Hurst Exponent Analysis. *Entropy* **2020**, *22*, 1159. [CrossRef] [PubMed]
26. Szyguła, J.; Domański, A.; Domańska, J.; Czachórski, T.; Marek, D.; Klamka, J. AQM Mechanism with Neuron Tuning Parameters. In *Intelligent Information and Database Systems*; Springer International Publishing: Berlin/Heidelberg, Germany, 2020; pp. 299–311.
27. Sun, J.; Zukerman, M. An Adaptive Neuron AQM for a Stable Internet. In *NETWORKING. Ad Hoc and Sensor Networks, Wireless Networks, Next Generation Internet*; Springer: Berlin/Heidelberg, Germany, 2007; Volume 4479, pp. 844–854. [CrossRef]

28. Busoniu, L.; Babuska, R.; De Schutter, B. A Comprehensive Survey of Multiagent Reinforcement Learning. *IEEE Trans. Syst. Man Cybern. Part C Appl. Rev.* **2008**, *38*, 156–172. [CrossRef]
29. Hamadneh, N.; Obiedat, M.; Qawasmeh, A.; Bsoul, M. HRED, An Active Queue Management Approach For The NS2 Simulator. *Recent Patents Comput. Sci.* **2018**, *12*. [CrossRef]
30. Krajewski, W.; Viaro, U. Fractional order PI controllers for TCP packet flow ensuring given modulus margins. *Control Cybern.* **2014**, *43*, 493–505.
31. Domanski, A.; Domanska, J.; 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.* **2016**, *26*, 777–789. [CrossRef]
32. Su, Y.; Huang, L.; Feng, C. QRED: A Q-Learning-based active queue management scheme. *J. Internet Technol.* **2018**, *19*, 1169–1178. [CrossRef]
33. Bisoy, S.K.; Pandey, P.K.; Pati, B. Design of an active queue management technique based on neural networks for congestion control. In Proceedings of the IEEE International Conference on Advanced Networks and Telecommunications Systems (ANTS), Bhubaneswar, Odisha, India, 17–20 December 2017; pp. 1–6.
34. Wang, H.; Chen, J.; Liao, C.; Tian, Z. An Artificial Intelligence Approach to Price Design for Improving AQM Performance. In Proceedings of the 2011 IEEE Global Telecommunications Conference—GLOBECOM 2011, Houston, TX, USA, 5–9 December 2011; pp. 1–5.
35. Xiao, P.; Tian, Y. Design of a Robust Active Queue Management Algorithm Based on Adaptive Neuron Pid. In Proceedings of the 2006 International Conference on Machine Learning and Cybernetics, Dalian, China, 13–16 August 2006; pp. 308–313.
36. Meng, Z.; Qiao, J.; Zhang, L. Design and Implementation: Adaptive Active Queue Management Algorithm Based on Neural Network. In Proceedings of the 10th International Conference on Computational Intelligence and Security, Kunming, Yunnan, China, 15–16 November 2014; pp. 104–108. [CrossRef]
37. Hu, M.; Mukaidani, H. Nonlinear Model Predictive Congestion Control Based on LSTM for Active Queue Management in TCP Network. In Proceedings of the 12th Asian Control Conference (ASCC), Kitakyushu-shi, Japan, 9–12 June 2019; pp. 710–715.
38. Gomez, C.A.; Wang, X.; Shami, A. Intelligent Active Queue Management Using Explicit Congestion Notification. In Proceedings of the IEEE Global Communications Conference (GLOBECOM), Waikoloa, HI, USA, 9–13 December 2019; pp. 1–6. [CrossRef]
39. Bisoy, S.K.; Pattnaik, P.K. An AQM Controller Based on Feed-Forward Neural Networks for Stable Internet. *Arab. J. Sci. Eng.* **2018**, 3993–4004. [CrossRef]
40. Zhou, C.; Di, D.; Chen, Q.; Guo, J. An Adaptive AQM Algorithm Based on Neuron Reinforcement Learning. In Proceedings of the IEEE International Conference on Control and Automation, Christchurch, New Zealand, 9–11 December 2009; pp. 1342–1346.
41. Jin, W.; Gu, R.; Ji, Y.; Dong, T.; Yin, J.; Liu, Z. Dynamic traffic aware active queue management using deep reinforcement learning. *Electron. Lett.* **2019**, *55*. [CrossRef]
42. Domańska, J.; Domański, A.; Czachórski, T. Estimating the Intensity of Long-Range Dependence in Real and Synthetic Traffic Traces. In *Computer Networks*; Springer International Publishing: Berlin/Heidelberg, Germany, 2015; Volume 522, pp. 11–22. [CrossRef]
43. Willinger, W.; Leland, W.; Taqqu, M. On the self-similar nature of traffic. *IEEE/ACM Trans. Netw.* **1994**. [CrossRef]
44. Paxson, V.; Floyd, S. Wide area traffic: The failure of Poisson modeling. *IEEE/ACM Trans. Netw.* **1995**, *3*, 226–244. [CrossRef]
45. Feldmann, A.; Gilbert, A.C.; Willinger, W.; Kurtz, T.G. The Changing Nature of Network Traffic: Scaling Phenomena. *SIGCOMM Comput. Commun. Rev.* **1998**, *28*, 5–29. [CrossRef]
46. Li, Z.; Xing, W.; Khamaiseh, S.; Xu, D. Detecting saturation attacks based on self-similarity of OpenFlow traffic. *IEEE Trans. Netw. Serv. Manag.* **2019**, *17*, 607–621. [CrossRef]
47. Willinger, W.; Taqqu, M.S.; Wilson, D.V. Lessons from "On the Self-Similar Nature of Ethernet Traffic". *SIGCOMM Comput. Commun. Rev.* **2019**, *49*, 56–62. [CrossRef]
48. Yin, C.; Zhu, Y.; Fei, J.; He, X. A deep learning approach for intrusion detection using recurrent neural networks. *IEEE Access* **2017**, *5*, 21954–21961. [CrossRef]
49. Kravchik, M.; Shabtai, A. Efficient cyber attack detection in industrial control systems using lightweight neural networks and pca. *IEEE Trans. Dependable Secur. Comput.* **2021**. [CrossRef]
50. Jiang, F.; Fu, Y.; Gupta, B.B.; Liang, Y.; Rho, S.; Lou, F.; Meng, F.; Tian, Z. Deep learning based multi-channel intelligent attack detection for data security. *IEEE Trans. Sustain. Comput.* **2018**, *5*, 204–212. [CrossRef]
51. Ring, M.; Schlör, D.; Landes, D.; Hotho, A. Flow-based network traffic generation using generative adversarial networks. *Comput. Secur.* **2019**, *82*, 156–172. [CrossRef]
52. D'Angelo, G.; Palmieri, F. Network traffic classification using deep convolutional recurrent autoencoder neural networks for spatial–temporal features extraction. *J. Netw. Comput. Appl.* **2021**, *173*, 102890. [CrossRef]
53. Meidan, Y.; Bohadana, M.; Mathov, Y.; Mirsky, Y.; Shabtai, A.; Breitenbacher, D.; Elovici, Y. N-baiot—Network-based detection of iot botnet attacks using deep autoencoders. *IEEE Pervasive Comput.* **2018**, *17*, 12–22. [CrossRef]
54. Li, J.; Liu, C.; Gong, Y. Layer trajectory LSTM. *arXiv* **2018**, arXiv:1808.09522.
55. Fu, R.; Zhang, Z.; Li, L. Using LSTM and GRU neural network methods for traffic flow prediction. In Proceedings of the 2016 31st Youth Academic Annual Conference of Chinese Association of Automation (YAC), Wuhan, China, 11–13 November 2016; IEEE: Red Hook, NY, USA, 2016; pp. 324–328. [CrossRef]

56. Zhou, B.; Lapedriza, A.; Khosla, A.; Oliva, A.; Torralba, A. Places: A 10 Million Image Database for Scene Recognition. *IEEE Trans. Pattern Anal. Mach. Intell.* **2018**, *40*, 1452–1464. [CrossRef] [PubMed]
57. Optimisation Algorithm—Adaptive Moment Estimation(Adam). Available online: https://towardsdatascience.com/optimisation-algorithm-adaptive-moment-estimation-adam-92144d75e232 (accessed on 12 June 2021).
58. Domańska, J.; Domański, A. AQM in Linux based routers—Comparing with analytical and simulation results. In Proceedings of the 5th International Conference: Internet in the Information Society, Theoretical and Applied Informatics, Heraklion, Greece, 22–24 July 2008; Volume 20, pp. 277–292.

Rozdział 8

The IoT gateway with active queue management

sciendo

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

amcs

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

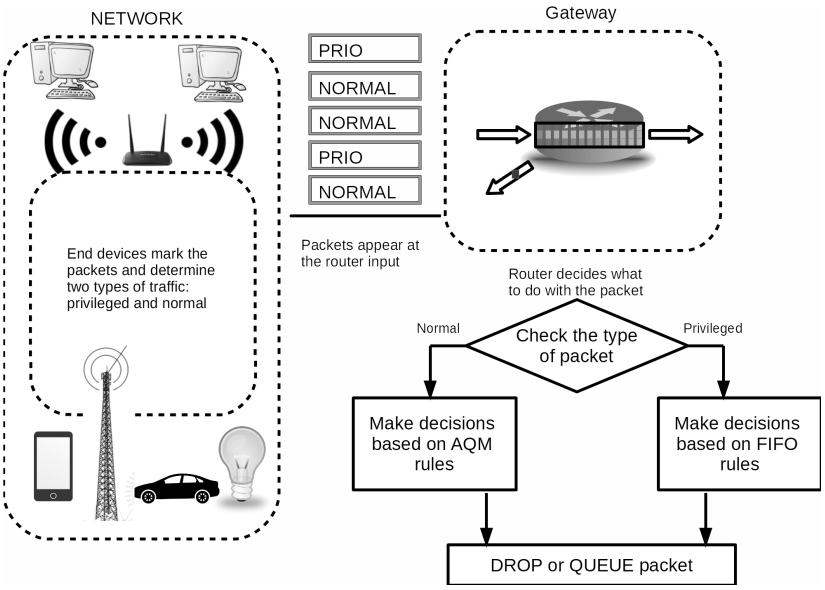


Fig. 1. IoT gateway: the node between the network of small devices and the Internet.

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)

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

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

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)

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

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

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

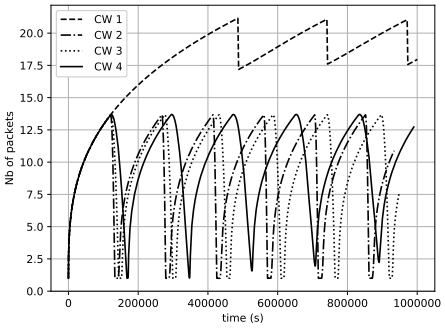


Fig. 2. TCP congestion window evolution.

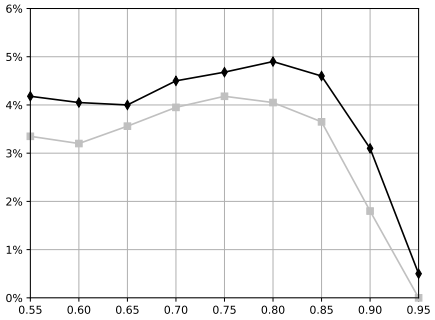
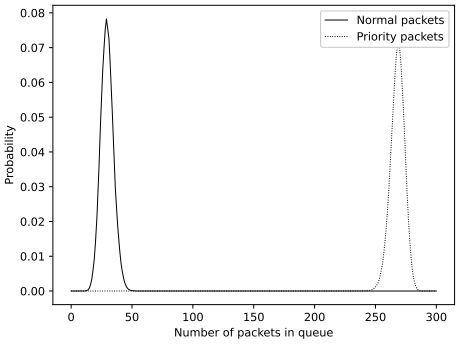


Fig. 3. Maximum and minimum differences between the assumed and estimated Hurst parameters.



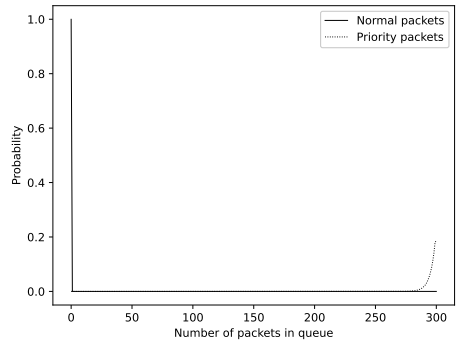


Fig. 4. Distribution of the queue length for high traffic load ($\mu = 0.2$) and $H = 0.5$, priority traffic 50%, $K_P = 0.0001$, $K_I = 0.0004$, $\alpha = -0.4$ (top), $K_P = 0.0001$, $K_I = 0.054$, $\alpha = -1.2$ (bottom).

(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

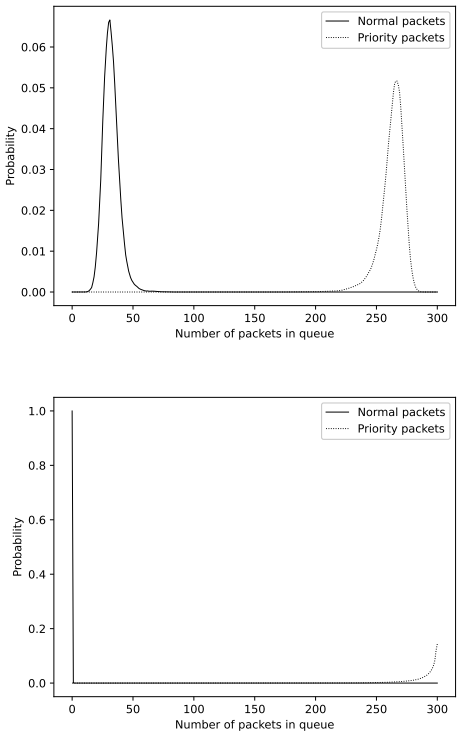


Fig. 5. Distribution of the queue length for high traffic load ($\mu = 0.2$) and $H = 0.7$, priority traffic 50%, $K_P = 0.0001$, $K_I = 0.0004$, $\alpha = -0.4$ (top), $K_P = 0.0001$, $K_I = 0.054$, $\alpha = -1.2$ (bottom).

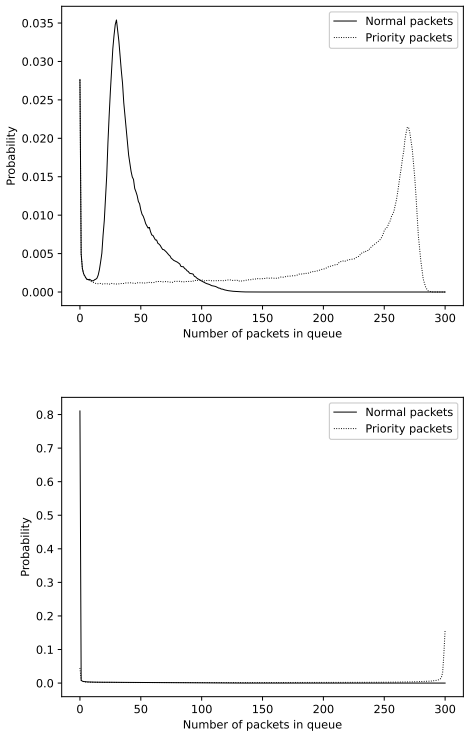


Fig. 6. Distribution of the queue length for high traffic load ($\mu = 0.2$) and $H = 0.9$, priority traffic 50%, $K_P = 0.0001$, $K_I = 0.0004$, $\alpha = -0.4$ (top), $K_P = 0.0001$, $K_I = 0.054$, $\alpha = -1.2$ (bottom).

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.

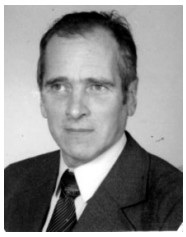
Willinger, W., Leland, W. and Taqqu, M. (1994). On the self-similar nature of traffic, *IEEE/ACM Transactions on Networking* **2**(1): 1–15.

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

Rozdział 9

Podsumowanie

Analiza uzyskanych wyników z przeprowadzonych eksperymentów badawczych, które w trakcie prac nad niniejszą dysertacją opublikowano w uznanych czasopismach naukowych oraz które przedstawiono na międzynarodowych konferencjach naukowych, dowodzi prawdziwości postawionej w tej pracy tezie, którą udowodniono poniżej:

Wniosek. *Stworzenie adaptacyjnych mechanizmów zarządzania pakietami, dostosowujących się nie tylko do natężenia ruchu sieciowego, ale również do zależności długoterminowych, może znacząco zwiększyć efektywność transmisji w sieciach komputerowych.*

Jednakże, pomimo zaproponowanych uproszczeń, wprowadzonych w celu ograniczenia liczby wykonywanych operacji, adaptacyjne mechanizmy zarządzania kolejką przedstawione w niniejszej rozprawie są bez wątpienia bardziej złożone obliczeniowo, niż najprostsze algorytmy z rodziny RED. Z drugiej strony oferują one znaczący poziom optymalizacji parametrów transmisji oraz zarządzania zajętością kolejki bufora. Jestem jednak przekonany, że wraz z dalszym rozwojem technologii, zwiększaniem dostępności wysoko wydajnego sprzętu komputerowego oraz dostępem do coraz mocniejszych i wydajniejszych obliczeniowo routerów, w niedalekiej przyszłości możliwe będzie pełne wykorzystanie rozwiązań zaproponowanych w niniejszej rozprawie.

Kierunki dalszych prac badawczych powstały w pewien sposób samodzielnie i stanowić będą odpowiedź na sugestie otrzymane od recenzentów, które pojawiły się przy okazji procesu publikowania artykułów przedstawionych w niniejszej rozprawie. Pierwszym możliwym kierunkiem będzie implementacja zaproponowanych mechanizmów aktywnego zarządzania kolejką w środowisku Linux,

a także w routerach kompatybilnych z systemem OpenWrt. Praca ta umożliwi przeprowadzenie badań z wykorzystaniem rzeczywistego ruchu pakietów z sieci Internet, a także będzie stanowiła pierwszy krok do wdrożenia przedstawionych mechanizmów aktywnego zarządzania kolejką, co też pozwoli na ich praktyczne wykorzystanie.

W przyszłych pracach dostrzegam również potrzebę dalszej optymalizacji zaproponowanych rozwiązań, tak aby dostosować je do wydajności i możliwości dzisiejszych urządzeń sieciowych. Z drugiej jednak strony zaproponowane mechanizmy już teraz są możliwe do praktycznego wykorzystania. Wbrew pozorom to właśnie sam proces uczenia modelu opartego o sieci neuronowe jest najbardziej złożony obliczeniowo i wymaga dostępu do najbardziej wydajnego sprzętu. Jednak samo jego wykorzystanie nie jest już tak kosztowne, co niweluje obawy związane z wydajnością współcześnie dostępnych i powszechnie stosowanych urządzeń sieciowych.

Bibliografia

[1] Pan, R., Prabhakar, B., Psounis, K.: CHOKe, A Stateless AQM scheme for Approximating fair Bandwidth Allocation. IEEE INFOCOM, s. 942-952, 2000.

[2] Jacobson V.: Congestion Avoidance and Control. ACM SIGCOMM, Vol. 18(4), 1988.

[3] Hellal, O. A., Altman, E.: Analysis of TCP Vegas and TCP Reno. Telecommunication Systems, 2000.

[4] Sarolahti, P., Kojo, M., Raatikainen, K.: F-RTO: An Enhanced Recovery Algorithm for TCP Retransmission Timeouts. ACM SIGCOMM Computer Communication Review, 2003.

[5] Gerla, M., Sanadidi, M. Y., Wang, R., Zanella, A.: TCP Westwood: Congestion Window Control Using Bandwidth Estimation. IEEE Global Telecommunications Conference, 2001.

[6] Floyd, S., Jacobson, V.: Random early detection gateways for congestion avoidance. IEEE/ACM Transactions on Networking, Vol. 1(4), 1993. doi:10.1109/90.251892.

[7] Hassan, M., Jain, R.: High Performance TCP/IP Networking. Pearson Education Inc., 2004.

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

[9] Tan, L., Zhang, W., Peng, G., Chen, G.: Stability of TCP/RED systems in AQM routers. IEEE Transactions on Automatic Control, Vol. 51(8), s. 1393-1398, 2006.

[10] Unal, H., Melchor-Aguilar, D., Ustebay, D., Niculescu, S.I., Ozbay, H.: Comparison of PI controllers designed for the delay model of TCP/AQM. Computer Communications, Vol. 36(10), s. 1225-1234, 2013.

[11] Xu, Y.-D., Wang, Z.-Y, Wang, H.: ARED: a novel adaptive congestion controller. International Conference on Machine Learning and Cybernetics, Vol. 2, s. 708 - 714, 2005. doi:10.1109/ICMLC.2005.1527036.

[12] Floyd, S., Gummadi, R., Shenker, S.: Adaptive RED: An Algorithm for Increasing the Robustness of RED's Active Queue Management, 2001.

[13] Lin, D., Morris, R.: Dynamics of Random Early Detection. Proceedings of the ACM SIGCOMM Conference on Applications, Technologies, Architectures, and Protocols for Computer Communication. Association for Computing Machinery, s. 127–137, 1997. doi:10.1145/263105.263154.

[14] Zacharewicz, A.: Metody Analizy Długozasięgowej. Hugo Steinhaus Center (1999-2002).

[15] Borys P.: Sztuczki karciane, wylewy Nilu i wykładnik Hursta. Wydział Chemiczny, Wydawnictwo Politechniki Śląskiej, 2011.

[16] Kiłyk, A., Wilimowska, Z.: Wykorzystanie wykładnika Hursta do prognozowania zmian cen na Giełdzie Papierów Wartościowych. Innowacje w zarządzaniu i inżynierii produkcji. Oficyna Wydawnicza Polskiego Towarzystwa Zarządzania Produkcją, Vol. 1, s. 455-463, 2016.

[17] Domańska, J., Domański, A., Czachórski, T.: Estimating the Intensity of Long-Range Dependence in Real and Synthetic Traffic Traces. International Conference on Computer Networks, Springer International Publishing, Vol. 522, s. 11-22, 2015. doi:10.1007/978-3-319-19419-6_2.

[18] Domańska, J., Domański, A., Czachórski, T., Klamka, J.: Self-similarity Trafic and AQM Mechanism Based on Non-integer Order $PI^{\alpha}D^{\beta}$ Controller. Communications in Computer and Information Science. Springer International Publishing, Vol. 718, s. 336-350, 2017. doi:10.1007/978-3-319-59767-6_27.

[19] Willinger, W., Taqqu, M. S., Wilson, D. V.: Lessons from ”On the Self-Similar Nature of Ethernet Traffic”. SIGCOMM Comput. Commun. Rev. Association for Computing Machinery, Vol. 49(5), s. 56-62, 2019, New York, NY, USA. doi:10.1145/3371934.3371955.

[20] Leland, W. E., Taqqu, M. S., Willinger, W., Wilson, D. V.: On the self-similar nature of Ethernet traffic. IEEE/ACM Transactions on Networking, Vol. 2 (1), s. 1-15, 1994. doi:10.1109/90.282603.

[21] Paxson, V., Floyd, S.: Wide area traffic: the failure of Poisson modeling. IEEE/ACM Transactions on Networking, Vol. 3(3), s. 226-244, 1995.

[22] Feldmann, A., Gilbert, A. C., Willinger, W., Kurtz, T. G.: The Changing Nature of Network Traffic: Scaling Phenomena. SIGCOMM Comput. Commun. Rev., Vol. 28(2), s. 5–29, 1998, New York, USA. doi:10.1145/279345.279346.

[23] Suchacka, G., Dembczak, A.: Verification of Web Traffic Burstiness and Self-similarity for Multiple Online Stores. Information Systems Architecture and Technology: Proceedings of 38th International Conference on Information Systems Architecture and Technology (ISAT), Springer International Publishing, s. 305-314, 2018. doi:10.1007/978-3-319-67220-5_28.

[24] CAIDA: Center for Applied Internet Data Analysis based at the University of California’s San Diego Supercomputer Center. https://www.caida.org/

[25] Kim, Y., Min, P.: On the prediction of average queueing delay with self-similar traffic. IEEE Global Telecommunications Conference, GLOBECOM ’03, Vol. 5, s. 2987 – 2991, 2004. doi:10.1109/GLOCOM.2003.1258782.

[26] Swami, N.; Bairwa, A.; Choudhary, M.: A LITERATURE SURVEY OF NETWORK SIMULATION TOOLS. International Conference On Communication & Computational Technologies by RIET, Proceeding ICCCT, Vol. 5, s. 206–208, 2017.

[27] Borboruah, G.; Nandi, G.: A Study on Large Scale Network Simulators. International Journal of Computer Science and Information Technologies, Vol. 5, s. 7318–7322, 2014.

[28] Dou, Y.; Liu, H.; Wei, L.; Chen, S.: Design and simulation of self-organizing network routing algorithm based on Q-learning. 21st Asia-Pacific Network Operations and Management Symposium (APNOMS), s. 357–360, 2020. doi:10.23919/APNOMS50412.2020.9237020.

[29] Guo, X.; Guo, B.; Li, K.; Fan, C.; Yang, H.; Huang, S. A.: SDN-enabled Integrated Space-Ground Information Network Simulation Platform. 18th International Conference on Optical Communications and Networks (ICOCN), s. 1–3. doi:10.1109/ICOCN.2019.8934091.

[30] Misra, V.; Gong, W.-B.; Towsley, D.: Fluid-based Analysis of a Network of AQM Routers Supporting TCP Flows with an Application to RED. ACM SIGCOMM, 2000. doi:10.1145/347059.347421.

[31] Domański, A.: Wpływ mechanizmów protokołu TCP oraz algorytmów kolejkowania na transmisję danych w sieci Internet. Studia Informatica, Vol. 38(1A), s. 3-245, 2017.

[32] Tanenbaum, A.S., Wetherall, D.J.: Sieci komputerowe. Wydanie V, Wyd. Helion, 2012. ISBN: 9788324630790.

[33] Miao, D., Liu, L., Xu, R., Panneerselvam, J., Wu, Y., Xu, W.: IEEE Transactions on Industrial Informatics, Vol. 14(10), s. 4487-4496, 2018. doi:10.1109/TII.2018.2799598.

[34] Hassan, M., Jain, R.: Wysoko wydajne Sieci TCP/IP. Wyd. Helion, 2004.

[35] Stevens, W.: TCP Slow Start, Congestion Avoidance, Fast Retransmit and Fast Recovery Algorithms. RFC 2001, 1997.

[36] Floyd, S., Henderson, T., Gurtov, A.: The NewReno Modification to TCP's Fast Recovery Algorithm. RFC 3782, 2004.

[37] Mascolo, S., Casetti, C., Gerla, M., Sanadidi, M. Y., Wang, R.: TCP westwood: Bandwidth estimation for enhanced transport over wireless links. MobiCom '01, Proceedings of the 7th annual international conference on Mobile computing and networking, s. 287-297, 2001. doi:10.1145/381677.381704.

[38] Brakmo, L. S., Peterson L. L.: TCP Vegas: End to end congestion avoidance on a global Internet. IEEE Journal on Selected Areas in Communication, Vol. 13(8), s. 1465-1480, 1995. doi:10.1109/49.464716.

[39] Xu, L., Harfoush, K., Rhee, I.: Binary Increase Congestion Control for Fast, Long Distance Networks. INFOCOM, 23th Conference of the IEEE Computer and Communications Societies, Vol. 4, s.2514-2524, 2004. doi:10.1109/INFCOM.2004.1354672.

[40] Domańska, J., Domański A., Czachórski T., Klamka, J.: Fluid-Flow approximation of time-limited TCP/UDP/XCP streams. Bulletin of the Polish Academy of Sciences: Technical Sciences, Vol. 62(2), s. 217-225, 2014. doi:10.2478/bpasts-2014-0021.

[41] Domańska, J., Domański, A., Czachórski, T.: Comparison of AQM Control Systems with the Use of Fluid-Flow Approximation. Communications in Computer and Information Science, Vo. 291, s. 82-90, 2012. doi:10.1007/978-3-642-31217-5_9.

[42] Fretwell, R.J., Kouvatsos, D.D.: Modelling LRD and SRD Traffic with Batch Renewal Process: Review of Results and Open Issues. Lecture Notes in Computer Science, Vol. 5233, s. 141-173, 2011. doi:10.1007/978-3-642-02742-0_7

[43] Domańska, J.: Markowowskie modele natężenia przesyłów internetowych. Instytut Informatyki Teoretycznej i Stosowanej Polskiej Akademii Nauk. ISBN: 978-83-62652-69-3.

[44] Gong, W.-B., Liu, Y., Misra, V., Towsley, D.: Self-Similarity and Long Range Dependence on the Internet: A Second Look at the Evidence, Origins and Implications. Elsevier Computer Networks, Vol. 48(3), s. 377-399, 2005. doi: 10.5555/1081434.1648606.

[45] Beran, J.: Statistics for Long-Memory Processes (1st ed.). Chapman & Hall. Routledge, Vol. 61, 1994. doi: 10.1201/9780203738481.

[46] Mandelbrot, B.B., Wallis, J.: Computer Experiments with Fractional Gaussian Noises. Water Resources Research, Vol. 5(1), s. 228-241, 1969. doi:10.1029/WR005i001p00260.

[47] Geweke, J., Porter-Hudak, S.: The Estimation and Application of Long Memory Time Series Models. Journal of Time Series Analysis, Vol. 4(4), s. 221-238, 1983. doi:10.1111/j.1467-9892.1983.tb00371.x.

[48] Kuensch, H.R.: Statistical aspects of self-similar processes. Proceedings of the First World Congress of the Bernoulli Society, VNU Science Press, Vol. 1, s. 67-74, 1987. doi:10.1515/9783112314227-005.

[49] Abry, P., Flandrin, P., Taqqu, M., Veitch, D.: Wavelets for the Analysis, Estimation and Synthesis of Scaling Data. In Self-Similar Network Traffic and Performance Evaluation, 1999. doi:10.1002/047120644X.ch2.

[50] Filus, K.; Domański, A.; Domańska, J.; Marek, D.; Szyguła, J. Long-Range Dependent Traffic Classification with Convolutional Neural Networks Based on Hurst Exponent Analysis. Entropy, vol. 22, 1159. doi:10.3390/e22101159.

[51] Czachórski, T., Domańska, J., Pagano, M.: On Stochastic Models of Internet Traffic. International Conference on Computer Networks, Communications in Computer and Information Science, Vol. 564, s. 289-303, 2015.

[52] Domańska, J., Domański, A., Czachórski, T.: Modeling Packet Traffic with the Use of Superpositions of Two-State MMPPs. Communications in Computer and Information Science. Springer International Publishing, Vol. 431, s. 24-36, 2014. doi:10.1007/978-3-319-07941-7_3.

[53] Wójcicki, R.: Nowe metody modelowania samopodobnego ruchu w sieciach w oparciu o procesy Poissona z markowską modulacją. Studia Informatica, Vol. 26(2), s. 23-39, 2005.

[54] Fischer, W., Meier-Hellstern, K.: The Markov-modulated Poisson process (MMPP) cookbook. Performance Evaluation, Vol. 18 (2), s. 149-171, 1993. doi:10.1016/0166-5316(93)90035-S.

[55] Mirzaei, M., Mizanian, K., Rezaeian, M.: Modeling of self-similar network traffic using artificial neural networks. 4th International Conference on Computer and Knowledge Engineering (ICCKE), s. 741-746, 2014.

[56] Abdel-Jaber, H., Mahafzah, M., Thabtah, F., Woodward, M.: Fuzzy logic controller of Random Early Detection based on average queue length and packet loss rate. International Symposium on Performance Evaluation of Computer and Telecommunication Systems, s. 428-432, 2008.

[57] Krajewski, W., Viaro, U.: Fractional order PI controllers for TCP packet flow ensuring given modulus margins. Control and cybernetics, Vol. 43(4), s. 493-505, 2014.

[58] Domańska, J., Domański, A., Czachórski, T., Klamka, J.: The use of a non-integer order PI controller with an Active Queue Management Mechanism. International Journal of Applied Mathematics and Computer Science, Vol. 26, s. 777-789, 2016. doi:10.1515/amcs-2016-0055.

[59] Sabry, S.S., Kaittan, N.M.: Grey wolf optimizer based fuzzy-PI active queue management design for network congestion avoidance. Indonesian Journal of Electrical Engineering and Computer Science, Vol. 18(1), s. 199-208, 2020. doi:10.11591/ijeecs.v18.i1.pp199-208.

[60] Su, Y., Huang, L., Feng, C.: QRED: A Q-Learning-based active queue management scheme. Journal of Internet Technology, Vol. 19, s. 1169-1178, 2018. doi:10.3966/160792642018081904019.

[61] Bisoy, S.K., Pandey, P. K., Pati, B.: Design of an active queue management technique based on neural networks for congestion control. IEEE International Conference on Advanced Networks and Telecommunications Systems (ANTS), s. 1-6, 2017.

[62] Wang, H., Chen, J., Liao, C., Tian, Z.: An Artificial Intelligence Approach to Price Design for Improving AQM Performance. IEEE Global Telecommunications Conference - GLOBECOM, s. 1-5, 2011.

[63] Xiao, P., Tian, Y.: Design of a Robust Active Queue Management Algorithm Based on Adaptive Neuron PID. International Conference on Machine Learning and Cybernetics, s. 308-313, 2006.

[64] Zhou, C., Di, D., Chen, Q., Guo, J.: An Adaptive AQM Algorithm Based on Neuron Reinforcement Learning. IEEE International Conference on Control and Automation, s. 1342-1346, 2009.

[65] Weiqi, J., Rentao, G., Yuefeng, J., Tao, D., Jie, Y., Zhihui, L.: Dynamic traffic aware active queue management using deep reinforcement learning. Electronics Letters, Vol. 55, 2019. doi:10.1049/el.2019.1146.

[66] Meng, Z., Qiao, J., Zhang, L.: Design and Implementation: Adaptive Active Queue Management Algorithm Based on Neural Network. 10th International Conference on Computational Intelligence and Security, s. 104-108, 2014. doi:10.1109/CIS.2014.104.

[67] Lin, L., Shi, Y., Chen, J., Ali, S.: A Novel Fuzzy PID Congestion Control Model Based on Cuckoo Search in WSNs. Sensors, 20(7), 1862, 2020. doi:10.3390/s20071862.

[68] Hu, M., Mukaidani, H.: Nonlinear Model Predictive Congestion Control Based on LSTM for Active Queue Management in TCP Network. 12th Asian Control Conference (ASCC), s. 710-715, 2019.

[69] Xu, Q., Ma, G., Ding, K., Xu, B.: An Adaptive Active Queue Management Based on Model Predictive Control. IEEE Access, Vol.8, s. 174489-174494, 2020. doi:10.1109/ACCESS.2020.3025377.

[70] Ping, Y.D., Wang, N.: A PID controller with neuron tuning parameters for multi-model plants. Proceedings of 2004 International Conference on Machine Learning and Cybernetics (IEEE Cat. No.04EX826), Vol. 6, s. 3408 - 3411, 2004. doi:10.1109/ICMLC.2004.1380375.

[71] Domański, A., Domańska, J., Czachórski, T., Klamka, J., Szyguła, J., Marek, D.: AQM mechanism with the dropping packet function based on the answer of several PI^{α} controllers. Communications in Computer and Information Science, Springer International Publishing, Vol. 1039, s. 400-412, 2019. doi:10.1007/978-3-030-21952-9_29.

[72] Mercioni, M. A., and Holban, S.: The Most Used Activation Functions: Classic Versus Current. International Conference on Development and Application Systems (DAS), s. 141-145, 2020. doi:10.1109/DAS49615.2020.9108942.

[73] Optimisation Algorithm — Adaptive Moment Estimation (Adam): https://towardsdatascience.com/optimisation-algorithm-adaptive-moment-estimation-adam-92144d75e232, [online: 04.04.2022].

[74] Stoica, I., Morris, R., Karger, D., Kaashoek, F. M., Balakrishnan, H.: Chord: A Scalable Peer-to-Peer Lookup Service for Internet Applications.

SIGCOMM '01 Proceedings of the 2001 conference on Applications, Technologies, Architectures and Protocols for Computer Communications, Vol. 31, s. 149-160, 2001. doi:10.1145/383059.383071

[75] Miao, D., Liu, L., Xu, R., Panneerselvam, J., Wu, Y., Xu, W.: An Efficient Indexing Model for the Fog Layer of Industrial Internet of Things. IEEE Transactions on Industrial Informatics, Vol. 14(10), s. 4487-4496, 2018. doi:10.1109/TII.2018.2799598.

[76] Geng, X., Whinston, A.B.: Defeating distributed denial of service attacks. IT Professional, Vol. 2(4), s. 36-42, 2000. doi:10.1109/6294.869381.

Oświadczenia współautorstwa

Oświadczenia współautorów publikacji naukowych, określające indywidualny wkład każdego z nich w ich powstanie.

Declarations of co-authorship, detailing each co-author's individual contribution to each publication.

dr hab. inż. Adam Domański, Prof. PŚ.
Katedra Systemów Rozproszonych i Urządzeń Informatyki
Politechnika Śląska
ul. Akademicka 16, 44-100 Gliwice

Gliwice, 06.06.2022

Oświadczenie współautorstwa

Oświadczam, że w pracy:

> Domański, A.; Domańska, J.; Czachórski, T.; Klamka, J.; Marek, D.; Szyguła, J.; AQM mechanism with the dropping packet function based on the answer of several Piα controllers. 26th International Conference on Computer Networks (CN 2019), Communications in Computer and Information Science, Springer International Publishing, vol. 1039, pp. 400-412, 2019, https://doi.org/10.1007/978-3-030-21952-9_29

mój udział [20 %] polegał na:

- współudziale w postawieniu problemu badawczego,
- współudziale w interpretacji uzyskanych wyników,
- współudziale w redakcji tekstu publikacji.

dr hab. inż. Adam Domański, Prof. PŚ.

dr hab. inż. Joanna Domańska
Instytut Informatyki Teoretycznej i Stosowanej
Polskiej Akademii Nauk
ul. Bałtycka 5, 44-100 Gliwice

Gliwice, 14.06.2022

Oświadczenie współautorstwa

Oświadczam, że w pracy:

Domański, A.; Domańska, J.; Czachórski, T.; Klamka, J.; Marek, D.; Szyguła, J.; AQM mechanism with the dropping packet function based on the answer of several Piα controllers. 26th International Conference on Computer Networks (CN 2019), Communications in Computer and Information Science, Springer International Publishing, vol. 1039, pp. 400-412, 2019. https://doi.org/10.1007/978-3-030-21952-9_29

mój udział [5%] polegał na:

- współudziale w interpretacji uzyskanych wyników,
- współudziale w redakcji tekstu publikacji.

dr hab. inż. Joanna Domańska

Gliwice, 23.02.2022

Prof. dr hab. inż. Tadeusz Czachórski
Instytut Informatyki Teoretycznej i Stosowanej
Polskiej Akademii Nauk
ul. Bałtycka 5, 44-100 Gliwice

Oświadczenie współautorstwa

Oświadczam, że w pracy:

Domański, A.; Domańska, J.; Czachórski, T.; Klamka, J.; Marek, D.; Szyguła, J.; AQM mechanism with the dropping packet function based on the answer of several Piα controllers. 26th International Conference on Computer Networks (CN 2019), Communications in Computer and Information Science, Springer International Publishing, vol. 1039, pp. 400-412, 2019, https://doi.org/10.1007/978-3-030-21952-9_29.

mój udział [5 %] polegał na:

- współudziale w interpretacji uzyskanych wyników,
- współudziale w redakcji tekstu publikacji.

.....

Prof. dr hab. inż. Tadeusz Czachórski

Gliwice, 23.02.2022

Prof. dr hab. inż. Jerzy Klamka
Instytut Informatyki Teoretycznej i Stosowanej
Polskiej Akademii Nauk
ul. Bałtycka 5, 44-100 Gliwice

Oświadczenie współautorstwa

Oświadczam, że w pracy:

> Domański, A.; Domańska, J.; Czachórski, T.; Klamka, J.; Marek, D.; Szyguła, J.; AQM mechanism with the dropping packet function based on the answer of several Piα controllers. 26th International Conference on Computer Networks (CN 2019), Communications in Computer and Information Science, Springer International Publishing, vol. 1039, pp. 400-412, 2019, https://doi.org/10.1007/978-3-030-21952-9_29.

mój udział [5 ..... %] polegał na:

- współudziale w merytorycznej ocenie poprawności utworzonego modelu,

- współudziale w redakcji tekstu publikacji.

Jerzy Klamka
.....
Prof. dr hab. inż. Jerzy Klamka

mgr inż. Dariusz Marek
Katedra Systemów Rozproszonych i Urządzeń Informatyki
Politechnika Śląska
ul. Akademicka 16, 44-100 Gliwice

Gliwice, 02.06.2022

Oświadczenie współautorstwa

Oświadczam, że w pracy:

> Domański, A.; Domańska, J.; Czachórski, T.; Klamka, J.; Marek, D.; Szyguła, J.; AQM mechanism with the dropping packet function based on the answer of several Piα controllers. 26th International Conference on Computer Networks (CN 2019), Communications in Computer and Information Science, Springer International Publishing, vol. 1039, pp. 400-412, 2019, https://doi.org/10.1007/978-3-030-21952-9_29

mój udział [5 %] polegał na:

- współudziale w redakcji oraz edycji tekstu publikacji.

Marek Dariusz

mgr inż. Dariusz Marek

dr hab. inż. Adam Domański, Prof. PŚ.
Katedra Systemów Rozproszonych i Urządzeń Informatyki
Politechnika Śląska
ul. Akademicka 16, 44-100 Gliwice

Gliwice, 06.06.2022

Oświadczenie współautorstwa

Oświadczam, że w pracy:

> Szyguła, J.; Domański, A.; Domańska, J.; Czachórski, T.; Marek, D.; Klamka, J.; AQM mechanism with neuron tuning parameters. 12th Asian Conference on Intelligent Information and Database Systems (ACIIDS), Lecture Notes in Computer Science, vol. 12034, Springer, Cham, 2020. https://doi.org/10.1007/978-3-030-42058-1_25

mój udział [15 %] polegał na:

- współudziale w postawieniu problemu badawczego: zaproponowaniu mechanizmu AQM w oparciu o techniki uczenia przez wzmocnienie,
- współudziale w interpretacji uzyskanych wyników,
- współudziale w redakcji tekstu publikacji.

dr hab. inż. Adam Domański, Prof. PŚ.

dr hab. inż. Joanna Domańska
Instytut Informatyki Teoretycznej i Stosowanej
Polskiej Akademii Nauk
ul. Bałtycka 5, 44-100 Gliwice

Gliwice, 14.06.2022

Oświadczenie współautorstwa

Oświadczam, że w pracy:

> Szyguła, J.; Domański, A.; Domańska, J.; Czachórski, T.; Marek, D.; Klamka, J.; AQM mechanism with neuron tuning parameters. 12th Asian Conference on Intelligent Information and Database Systems (ACIIDS), Lecture Notes in Computer Science, vol. 12034, Springer, Cham, 2020. https://doi.org/10.1007/978-3-030-42058-1_25

mój udział [5%] polegał na:

- współudziale w interpretacji uzyskanych wyników,
- współudziale w redakcji tekstu publikacji.

dr hab. inż. Joanna Domańska

Gliwice, 14.12.2021

Prof. dr hab. inż. Tadeusz Czachórski
Instytut Informatyki Teoretycznej i Stosowanej
Polskiej Akademii Nauk
ul. Bałtycka 5, 44-100 Gliwice

Oświadczenie współautorstwa

Oświadczam, że w pracy:

> Szyguła, J.; Domański, A.; Domańska, J.; Czachórski, T.; Marek, D.; Klamka, J.; AQM mechanism with neuron tuning parameters. 12th Asian Conference on Intelligent Information
> and Database Systems (ACIIDS), Lecture Notes in Computer Science, vol. 12034, Springer, Cham, 2020. https://doi.org/10.1007/978-3-030-42058-1_25

mój udział [..5.. %] polegał na:

- współudziale w interpretacji uzyskanych wyników,
- współudziale w redakcji tekstu publikacji.

T. Czachórski

.....

Prof. dr hab. inż. Tadeusz Czachórski

mgr inż. Dariusz Marek
Katedra Systemów Rozproszonych i Urządzeń Informatyki
Politechnika Śląska
ul. Akademicka 16, 44-100 Gliwice

Gliwice, 02.06.2022

Oświadczenie współautorstwa

Oświadczam, że w pracy:

> Szyguła, J.; Domański, A.; Domańska, J.; Czachórski, T.; Marek, D.; Klamka, J.; AQM mechanism with neuron tuning parameters. 12th Asian Conference on Intelligent Information and Database Systems (ACIIDS), Lecture Notes in Computer Science, vol. 12034, Springer, Cham, 2020. https://doi.org/10.1007/978-3-030-42058-1_25

mój udział [5 %] polegał na:

- współudziale w analizie uzyskanych wyników,

- współudziale w redakcji tekstu publikacji.

Marek Dariusz

mgr inż. Dariusz Marek

Gliwice, 14.12.2021

Prof. dr hab. inż. Jerzy Klamka
Instytut Informatyki Teoretycznej i Stosowanej
Polskiej Akademii Nauk
ul. Bałtycka 5, 44-100 Gliwice

Oświadczenie współautorstwa

Oświadczam, że w pracy:

> Szyguła, J.; Domański, A.; Domańska, J.; Czachórski, T.; Marek, D.; Klamka, J.; AQM mechanism with neuron tuning parameters. 12th Asian Conference on Intelligent Information and Database Systems (ACIIDS), Lecture Notes in Computer Science, vol. 12034, Springer, Cham, 2020. https://doi.org/10.1007/978-3-030-42058-1_25

mój udział [50 %] polegał na:

- współudziale w merytorycznej ocenie poprawności utworzonego modelu,
- współudziale w redakcji tekstu publikacji.

.....

Prof. dr hab. inż. Jerzy Klamka

mgr inż. Dariusz Marek
Katedra Systemów Rozproszonych i Urządzeń Informatyki
Politechnika Śląska
ul. Akademicka 16, 44-100 Gliwice

Gliwice, 02.06.2022

Oświadczenie współautorstwa

Oświadczam, że w pracy:

> Marek, D.; Szyguła, J; Domański, A.; Domańska, J.; Filus, K.; Szczygieł, M. Adaptive Hurst-Sensitive Active Queue Management. Entropy, vol. 24(3), 418, 2022. https://doi.org/10.3390/e24030418

mój wkład [25 %] polegał na:

- współudziale w wykonaniu przeglądu literatury,
- współudziale w procesie planowania i przeprowadzania eksperymentów badawczych,
- współudziale w interpretacji uzyskanych wyników,
- współudziale w redakcji tekstu publikacji.

Marek Dariusz

mgr inż. Dariusz Marek

dr hab. inż. Adam Domański, Prof. PŚ.
Katedra Systemów Rozproszonych i Urządzeń Informatyki
Politechnika Śląska
ul. Akademicka 16, 44-100 Gliwice

Gliwice, 06.06.2022

Oświadczenie współautorstwa

Oświadczam, że w pracy:

> Marek, D.; Szyguła, J; Domański, A.; Domańska, J.; Filus, K.; Szczygieł, M. Adaptive Hurst-Sensitive Active Queue Management. Entropy, vol. 24(3), 418, 2022. https://doi.org/10.3390/e24030418

mój wkład [10 %] polegał na:

- współudziale w postawieniu problemu badawczego: zbadanie struktur oraz możliwości zastosowania różnego rodzaju głębokich sieci neuronowych, umożliwiających ocenę charakterystyk ruchu sieciowego,

- współudziale w interpretacji uzyskanych wyników,

- współudziale w redakcji tekstu publikacji.

.....

dr hab. inż. Adam Domański, Prof. PŚ.

dr hab. inż. Joanna Domańska
Instytut Informatyki Teoretycznej i Stosowanej
Polskiej Akademii Nauk
ul. Bałtycka 5, 44-100 Gliwice

Gliwice, 14.06.2022

Oświadczenie współautorstwa

Oświadczam, że w pracy:

> Marek, D.; Szyguła, J; Domański, A.; Domańska, J.; Filus, K.; Szczygieł, M. Adaptive Hurst-Sensitive Active Queue Management. Entropy, vol. 24(3), 418, 2022. https://doi.org/10.3390/e24030418

mój wkład [5%] polegał na:

- współudziale w merytorycznej ocenie poprawności utworzonego modelu,
- współudziale w interpretacji uzyskanych wyników,
- współudziale w redakcji tekstu publikacji.

dr hab. inż. Joanna Domańska

mgr inż. Katarzyna Filus
Instytut Informatyki Teoretycznej i Stosowanej
Polskiej Akademii Nauk
ul. Bałtycka 5, 44-100 Gliwice

Gliwice, 09.06.2022

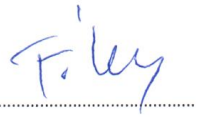
Oświadczenie współautorstwa

Oświadczam, że w pracy:

> Marek, D.; Szyguła, J; Domański, A.; Domańska, J.; Filus, K.; Szczygieł, M. Adaptive Hurst-Sensitive Active Queue Management. Entropy, vol. 24(3), 418, 2022. https://doi.org/10.3390/e24030418.

mój wkład [5 %] polegał na:

- współudziale w merytorycznej ocenie poprawności utworzonego modelu,
- współudziale w redakcji i edycji tekstu publikacji.



mgr inż. Katarzyna Filus

Gliwice, 09.05.2022

inż. Marta Szczygieł
Wydział Automatyki, Elektroniki i Informatyki
Politechnika Śląska
ul. Akademicka 16, 44-100 Gliwice

Oświadczenie współautorstwa

Oświadczam, że w pracy:

> Marek, D.; Szyguła, J; Domański, A.; Domańska, J.; Filus, K.; Szczygieł, M. Adaptive Hurst-Sensitive Active Queue Management. Entropy, vol. 24(3), 418, 2022. https://doi.org/10.3390/e24030418.

mój wkład [5 %] polegał na:

- współudziale w redakcji tekstu publikacji.

Marta Szczygieł

inż. Marta Szczygieł

dr hab. inż. Adam Domański, Prof. PŚ.
Katedra Systemów Rozproszonych i Urządzeń Informatyki
Politechnika Śląska
ul. Akademicka 16, 44-100 Gliwice

Gliwice, 06.06.2022

Oświadczenie współautorstwa

Oświadczam, że w pracy:

> Szyguła, J.; Domański, A.; Domańska, J.; Marek, D.; Filus, K.; Mendla, S. Supervised Learning of Neural Networks for Active Queue Management in the Internet. Sensors, vol. 21(15), 4979, 2021. https://doi.org/10.3390/s21154979

mój wkład [15 %] polegał na:

- współudziale w postawieniu problemu badawczego: zaproponowaniu mechanizmu aktywnego zarządzania kolejką opartego o sieci neuronowe i uczenie nadzorowane,
- współudziale w interpretacji uzyskanych wyników,
- współudziale w redakcji tekstu publikacji.

.....

dr hab. inż. Adam Domański, Prof. PŚ.

dr hab. inż. Joanna Domańska
Instytut Informatyki Teoretycznej i Stosowanej
Polskiej Akademii Nauk
ul. Bałtycka 5, 44-100 Gliwice

Gliwice, 14.06.2022

Oświadczenie współautorstwa

Oświadczam, że w pracy:

> Szyguła, J.; Domański, A.; Domańska, J.; Marek, D.; Filus, K.; Mendla, S. Supervised Learning of Neural Networks for Active Queue Management in the Internet. Sensors, vol. 21(15), 4979, 2021. https://doi.org/10.3390/s21154979

mój wkład [5%] polegał na:

- współudziale w merytorycznej ocenie poprawności utworzonego modelu,
- współudziale w interpretacji uzyskanych wyników,
- współudziale w redakcji tekstu publikacji.

.....

dr hab. inż. Joanna Domańska

mgr inż. Dariusz Marek
Katedra Systemów Rozproszonych i Urządzeń Informatyki
Politechnika Śląska
ul. Akademicka 16, 44-100 Gliwice

Gliwice, 02.06.2022

Oświadczenie współautorstwa

Oświadczam, że w pracy:

> Szyguła, J.; Domański, A.; Domańska, J.; Marek, D.; Filus, K.; Mendla, S. Supervised Learning of Neural Networks for Active Queue Management in the Internet. Sensors, vol. 21(15), 4979, 2021. https://doi.org/10.3390/s21154979

mój wkład [5 %] polegał na:

- współudziale w procesie przygotowania zbioru uczącego,
- współudziale w interpretacji uzyskanych wyników.

Marek Dariusz

mgr inż. Dariusz Marek

mgr inż. Katarzyna Filus
Instytut Informatyki Teoretycznej i Stosowanej
Polskiej Akademii Nauk
ul. Bałtycka 5, 44-100 Gliwice

Gliwice, 09.06.2022

Oświadczenie współautorstwa

Oświadczam, że w pracy:

> Szyguła, J.; Domański, A.; Domańska, J.; Marek, D.; Filus, K.; Mendla, S. Supervised Learning of Neural Networks for Active Queue Management in the Internet. Sensors, vol. 21(15), 4979, 2021. https://doi.org/10.3390/s21154979.

mój wkład [5 %] polegał na:

- współudziale w merytorycznej ocenie poprawności utworzonego modelu,
- współudziale w redakcji tekstu publikacji.

.....

mgr inż. Katarzyna Filus

Gliwice, 04.05.2022

mgr inż. Szymon Mendla
Wydział Automatyki, Elektroniki i Informatyki
Politechnika Śląska
ul. Akademicka 16, 44-100 Gliwice

Oświadczenie współautorstwa

Oświadczam, że w pracy:

> Szyguła, J.; Domański, A.; Domańska, J.; Marek, D.; Filus, K.; Mendla, S. Supervised Learning of Neural Networks for Active Queue Management in the Internet. Sensors, vol. 21(15):4979, 2021. https://doi.org/10.3390/s21154979

mój wkład [5 %] polegał na:

- współudziale w przeglądzie literatury i analizie stanu wiedzy,
- współudziale w redakcji tekstu publikacji.

Szymon Mendla

mgr inż. Szymon Mendla

dr hab. inż. Adam Domański, Prof. PŚ.
Katedra Systemów Rozproszonych i Urządzeń Informatyki
Politechnika Śląska
ul. Akademicka 16, 44-100 Gliwice

Gliwice, 06.06.2022

Oświadczenie współautorstwa

Oświadczam, że w pracy:

> Domański, A.; Domańska, J.; Czachórski, T.; Klamka, J.; Szyguła, J.; Marek, D. The IoT gateway with Active Queue Management. International Journal of Applied Mathematics and Computer Science, vol. 31(1), pp. 165-178, 2021. https://doi.org/10.34768/amcs-2021-0012

mój udział [20 %] polegał na:

- współudziale w postawieniu problemu badawczego i wyborze aproksymacji płynnej jako metody odpowiedniej do zastosowania w problemie badawczym,
- współudziale w interpretacji uzyskanych wyników,
- współudziale w redakcji tekstu publikacji.

.....

dr hab. inż. Adam Domański, Prof. PŚ.

dr hab. inż. Joanna Domańska
Instytut Informatyki Teoretycznej i Stosowanej
Polskiej Akademii Nauk
ul. Bałtycka 5, 44-100 Gliwice

Gliwice, 14.06.2022

Oświadczenie współautorstwa

Oświadczam, że w pracy:

> Domański, A.; Domańska, J.; Czachórski, T.; Klamka, J.; Szyguła, J.; Marek, D. The IoT gateway with Active Queue Management. International Journal of Applied Mathematics and Computer Science, vol. 31(1), pp. 165-178, 2021. https://doi.org/10.34768/amcs-2021-0012

mój udział [5%] polegał na:

- współudziale w merytorycznej ocenie poprawności utworzonego modelu,
- współudziale w redakcji tekstu publikacji.

dr hab. inż. Joanna Domańska

Gliwice, 14.12.2021

Prof. dr hab. inż. Tadeusz Czachórski
Instytut Informatyki Teoretycznej i Stosowanej
Polskiej Akademii Nauk
ul. Bałtycka 5, 44-100 Gliwice

Oświadczenie współautorstwa

Oświadczam, że w pracy:

> Domański, A.; Domańska, J.; Czachórski, T.; Klamka, J.; Szyguła, J.; Marek, D. The IoT gateway with Active Queue Management. International Journal of Applied Mathematics and Computer Science, vol. 31(1), pp. 165-178, 2021. https://doi.org/10.34768/amcs-2021-0012

mój udział [5.... %] polegał na:

- współudziale w weryfikacji poprawności zastosowanego modelu matematycznego,
- współudziale w redakcji tekstu publikacji.

T. Czachórski

.....

Prof. dr hab. inż. Tadeusz Czachórski

Gliwice, 14.12.2021

Prof. dr hab. inż. Jerzy Klamka
Instytut Informatyki Teoretycznej i Stosowanej
Polskiej Akademii Nauk
ul. Bałtycka 5, 44-100 Gliwice

Oświadczenie współautorstwa

Oświadczam, że w pracy:

> Domański, A.; Domańska, J.; Czachórski, T.; Klamka, J.; Szyguła, J.; Marek, D. The IoT gateway with Active Queue Management. International Journal of Applied Mathematics and Computer Science, vol. 31(1), pp. 165-178, 2021. https://doi.org/10.34768/amcs-2021-0012

mój udział [50 %] polegał na:

- współudziale w weryfikacji poprawności zastosowanego modelu matematycznego,
- współudziale w redakcji tekstu publikacji.

.....

Prof. dr hab. inż. Jerzy Klamka

mgr inż. Dariusz Marek
Katedra Systemów Rozproszonych i Urządzeń Informatyki
Politechnika Śląska
ul. Akademicka 16, 44-100 Gliwice

Gliwice, 02.06.2022

Oświadczenie współautorstwa

Oświadczam, że w pracy:

> Domański, A.; Domańska, J.; Czachórski, T.; Klamka, J.; Szyguła, J.; Marek, D. The IoT gateway with Active Queue Management. International Journal of Applied Mathematics and Computer Science, vol. 31(1), pp. 165-178, 2021. https://doi.org/10.34768/amcs-2021-0012

mój udział [5 %] polegał na:

- współudziale interpretacji uzyskanych wyników,

- współudziale w edycji tekstu publikacji.

Marek Dariusz

mgr inż. Dariusz Marek

Nagrody i wyróżnienia uzyskane podczas pracy nad przygotowaniem dysertacji

ACIIDS 2020

BEST ICxS'2020 PAPER AWARD

presented to

Jakub Szyguła

for the excellent paper entitled:

AQM mechanism with neuron tuning parameters

Dr. Maciej Huk
Session Chair

Prof. Keun Ho Ryu
Session Chair

Wroclaw University of Science and Technology

岩手県立大学
Iwate Prefectural University

ACIIDS 2020 - 12th Asian Conference on Intelligent Information and Database Systems (online), March 23-25, 2020

Politechnika Śląska

UCZELNIA BADAWCZA
INICJATYWA DOSKONAŁOŚCI

mgr inż.
Jakub Szyguła

otrzymuje
zespołową
NAGRODĘ REKTORA
stopnia III
za osiągnięcia naukowe

prof. dr hab. inż.
Arkadiusz Mężyk
Rektor Politechniki Śląskiej

Gliwice, 01.10.2021 r.

Certificate of Award

This is to certify that

MISCEA.PL ENGINEERING SP. Z O.O.

MISCEA.PL'S TEAM – LESZEK REMIORZ, ERYK REMIORZ,OLEG ANTEMIJCZUK, ADRIAN CZAJKOWSKI, JAROSŁAW PADUCH, GABRIEL DRABIK, JAKUB SZYGUŁA, DARIUSZ MAREK, SEBASTIAN PAWLAK, MARCIN PASZKUTA, GRZEGORZ BARON

has been awarded the

GOLD MEDAL

for the invention

A REVOLUTIONARY BREAKTHROUGH SHOWER SOLUTION

on the occasion of
2020 Kaohsiung International Invention & Design EXPO
10~12 December, 2020
Kaohsiung, Taiwan

Manli Hsieh
President of WIIPA

Neven Marković
President of the International Jury

World Invention Intellectual Property Associations

2020 Hong Kong International Invention and Design Competition

Silver Medal

For

A REVOLUTIONARY BREAKTHROUGH SHOWER SOLUTION

Leszek Remiorz, Eryk Remiorz, Oleg Antemijczuk, Adrian Czajkowski,

Jarosław Paduch, Gabriel Drabik, Jakub Szyguła,

Dariusz Marek, Sebastian Pawlak, Marcin Paszkuta, Grzegorz Baron

Jung Chuan Chou

Jung-Chuan Chou
Chairman of Jury member

K.C. Wu

Kou-Chen Wu
Chairman of IIDC

4 Dec 2020

No.202040