INTL JOURNAL OF ELECTRONICS AND TELECOMMUNICATIONS, 2020, VOL. 66, NO. 4, PP.715-721
Manuscript received April 20, 2020; revised October, 2020. DOI: 10.24425/ijet.2020.134032

Design of Network Traffic Congestion Controller with PI AQM Based on ITAE Index

Misbahul Fajri, and Kalamullah Ramli

***Abstract*—Establishing the proper values of controller parameters is the most important thing to design in active queue management (AQM) for achieving excellent performance in handling network congestion. For example, the first well known AQM, the random early detection (RED) method, has a lack of proper parameter values to perform under most the network conditions. This paper applies a Nelder-Mead simplex method based on the integral of time-weighted absolute error (ITAE) for a proportional integral (PI) controller using active queue management (AQM). A TCP flow and PI AQM system were analyzed with a control theory approach. A numerical optimization algorithm based on the ITAE index was run with Matlab/Simulink tools to find the controller parameters with PI tuned by Hollot (PI) as initial parameter input. Compared with PI and PI tuned by Ustebay (PIU) via experimental simulation in Network Simulator Version 2 (NS2) in five scenario network conditions, our proposed method was more robust. It provided stable performance to handle congestion in a dynamic network.**

***Keywords*— network congestion control, active queue management (AQM), Proportional integral (PI) controller, Nelder-Mead simplex method, Integral of time-weighted absolute error (ITAE)**

I. INTRODUCTION

THE Active queue management (AQM) is an essential intermediate network solution for network traffic to control congestion. AQM performs packet dropping in a buffer network router before congestion. The first AQM, random early detection (RED) [1], was proposed as a solution for network congestion to support the end-to-end transmission control protocol (TCP) congestion control, i.e., TCP RENO. RED reduces the synchronization problem because of its drop-tail mechanism and its ability to maintain the desired queue length. The drawback of RED is its sensitivity to parameter settings with a dynamic network load. Hollot proposed the proportional integral (PI) AQM [2] to overcome the challenges faced by RED and to provide PI tuning parameters using a control theory approach through the TCP flow model. This model converts the time function to the transfer function of *s*, so AQM can be designed and analysed in a practical manner. It is an alternative solution, especially for finding the best parameters for an AQM controller.

Through a control theory approach, AQM can be designed and analysed practically, this solution overcomes the drawbacks of the heuristic approach. The first AQM designed using control theory with a PI controller had a method for tuning its parameters [2]. Nonetheless, PI is still problematic for obtaining proper parameter values for good response and robustness in uncertain network traffic.

Other tuning approaches for PI AQM controllers have been described, as listed below. However, they also have problems, e.g., they require complex computations and still perform poorly. PIU [3] is another PI scheme that was examined in this research, and this paper found that PIU performed worse than Hollot's PI [2]. PIU proposed a resilient switching controller using the design method [4]. [5] introduced a parameter tuning algorithm for PI AQM based on the relationship between the controller parameter and the control damping ratio, resulting in an effective algorithm. Three sets of PI parameters were compared in [6], resulting in properly configured controller parameters that can stabilize and retain good performance in a dynamic network. [7] implemented a fractional order PI controller in AQM with three selected sets of parameters, resulting in more stable and higher queue occupancy than RED.

Reference [8] used a particle swarm optimization (PSO) algorithm to get the best parameter of a first-order controller for a TCP/AQM system, resulting in a proposed controller that is efficient in searching and better than PI AQM scheme. [9] formulated automatic tuning PID AQM parameters using quantitative feedback theory (QFT) with PSO optimization heuristic algorithms to reach robust stability with minimal cost effort. [10] implemented a genetic algorithm (GA) with a first delay TCP AQM model approximation to derive optimal PI controller parameters, resulting in a good performance in ITAE, shortest settling time, and lowest overshoot.

Other optimization methods, such as hybrid GA and bacterial foraging (BF) algorithms, were used to find optimal PID AQM parameter in [11], resulting in stable queue length, low packet loss, and high link utilization. Optimal PID AQM using ITAE criteria with a D-stable region approach was designed in [12] that resulted in high utilization, much faster and smaller oscillations than RED and PI AQM.

For better robustness in wide range network conditions, one should consider three parameters: network load (*N*), round-trip time (*RTT*), and link capacity (*C*). However, the parameter controller should also be properly tuned, so an AQM controller can quickly regulate queue length to the desired value and maintain stability with less oscillation. The proper parameter value can discard disturbance or error that has advantages for the system process [13]. Two criteria of the desired network performance that are expected to be achieved are short queuing delay and high link utilization [14]. Those will impact to the

Misbahul Fajri and Kalamullah Ramli are with Electrical Engineering Department, Engineering Faculty, Universitas Indonesia, Kampus Baru UI Depok, Indonesia (e-mail: misbahul.fajri61@ui.ac.id; k.ramli@eng.ui.ac.id).

application with better network QoS parameters, such as latency and throughput [15]. Therefore, the challenge is to design and find the proper formula for the abovementioned criteria.

The other research [16] aimed to obtain PID parameters using an optimization approach. That paper resulted that ITAE offered the best performance compared with ISE, IAE, and ITSE criteria. This paper studied and implemented ITAE with a PI controller in the NS2 network simulator [17] to understand its performance under real network conditions. This paper proposes a robust PI AQM, termed PITAE, that is tuned by an optimization method using a Nelder-Mead simplex algorithm with the ITAE index to get optimal parameter values that perform satisfactorily in a wide range of dynamic network traffic conditions. This method has been implemented in other areas except in AQM. So we interested to study and see this method in AQM to handle network traffic. Furthermore, this method is fast convergence and easy in implementation [18].

The paper is organized as follows. A theoretical background is given in Section 2. The proposed PITAE design is explained in Section 3. The simulation analysis is given in Section 4. Section 5 concludes this paper.

II. NETWORK CONGESTION CONTROL

A. TCP/AQM Flow Model

The TCP flow model in non-linear differential equations has been formulated by Misra that is a solution for further analysis of the network congestion control algorithm [19]:

$$\dot{W}(t) = \frac{1}{R(t)} - \frac{W(t)}{2}\frac{W(t-R(t))}{R(t-R(t))}p(t-R(t)) \tag{1}$$

$$\dot{q} = \begin{cases} -C + \dfrac{N(t)}{R(t)}W(t) & q > 0 \\ \max\left\{0, -C + \dfrac{N(t)}{R(t)}W(t)\right\}, & q = 0 \end{cases} \tag{2}$$

Where W is the average TCP window size (packets), $R(t)$ = round-trip time = T_p + ($q(t)/C$) (s), T_p = propagation delay (s), $q(t)$ = queue length (packets), C = link capacity (packets/s), P = packet drop probability, and N = number of TCP sessions. Herein, $\dot{W}$ denotes the time derivative of W. The initial part of the first formula is the window's additive increase ($1/R$), while the final part is the window's multiplicative decrease ($W/2$) in response to packet marking p. The second formula in (2) represents the bottleneck queue length.

The differential equation of the TCP flow model can assist in designing and determining the best AQM controller parameters because TCP control has a feedback process, the ACK signal, that is generated by a receiver to inform the sender when a packet has arrived; it is then used to decide whether the packet window size needs to be increased or decreased. The existence of a feedback process allows the TCP flow model to be analysed using a control theory approach, e.g., the transformed form of the TCP flow model.

In [2], the stochastic equation above was transformed by classic control theory through linearization and by ignoring the time-out mechanism. With assuming that both the TCP load and link capacity are constant, i.e., $N(t) \equiv N$ and $C(t) \equiv C$. W with q as the state and p as input and the operating points (w_o, q_o, and p_o) are derived by setting $\dot{W} = 0$ and $\dot{q} = 0$. Continuing to simplify this model while ignoring residual behavior means focusing on the nominal behavior of the window dynamic. The transfer function of the TCP/AQM flow is given by:

$$G_{tcp}(s) = \frac{\dfrac{R_0C^2}{2N^2}}{s + \dfrac{2N}{R_0^2C}}, \quad G_{queue} = \frac{\dfrac{N}{R_0}}{s + \dfrac{1}{R_0}} \tag{3}$$

Where G_{tcp} is TCP window control mechanism, R_o is round-trip time at the operating point, and G_{queue} is queue dynamic. The feedback of the TCP/AQM flow system is shown in Figure 1.

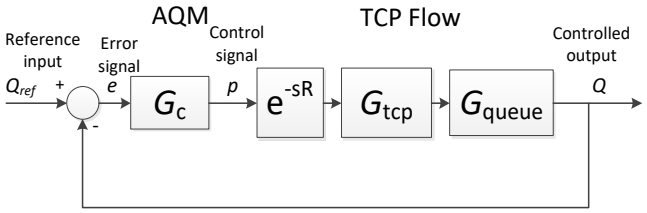


Fig. 1. Feedback of the TCP/AQM system.

According to (3) and Figure 1, the TCP/AQM dynamic can be expressed as

$$G_p(s) = G_{tcp}(s)G_{queue}(s)e^{-R_0s} \tag{4}$$

B. Proportional Integral Controller

The AQM is presented by the G_c function, which is applied with a PI controller. A transfer function of the PI controller is given by

$$G_c(s) = K_p e(s) + \frac{K_i}{s}e(s) \tag{5}$$

where K_p is a proportional gain, and K_i is an integral gain. To obtain the z-domain transfer function, (5) is converted from the s-domain using a bilinear transform (Tustin's rule) to preserve stability [20]. Then the discrete PI controller is given by

$$p(n) = p(n-1) + a.e(n) + b.e(n-1) \tag{6}$$

By using T_s as the sampling time, then

$$a = K_p + \frac{K_I T_s}{2}$$
$$b = K_p - \frac{K_I T_s}{2} \tag{7}$$

In network simulator NS2 has the module for the AQM in TCP/IP protocol, this simulator has build-in AQM scheme that can be used, and the default is the drop-tail mechanism. The default has to be changed to the appropriate AQM at the TCL script for running a simulator and adding a module for a new AQM algorithm.

C. Nelder-Mead Simplex Method

This method is based on some basic operations over a simplex. The initial simplex can be created using an initial guess [21]. A Nelder-Mead simplex does not need a constraint in computation to find a minimum of a function of n variables. The Nelder-Mead simplex has four methods [18], each with their

default coefficient. These are *reflection*=1, *expansion*=2, *contraction*=1/2, and *shrinkage*=1/2. The Nelder-Mead simplex method has a function

$$y = f(x) \tag{8}$$

Where x and y are a vector of independent variables and a real function value, respectively. The Nelder-Mead simplex is defined by n+1 vertices and x is defined as a centroid of all vertices excluding x_{n+1}. The vertices are ordered from lowest to the highest function value. In a triangle (M) centroid is calculated in all vertices except x_w. The steps to find the optimum value are

a. Reflection (R) which calculates new vertex $x_r=M+\alpha(M-x_w)$, where α=1
b. Expansion (E) which calculates a new vertex $x_e=M+\gamma(M-x_w)$, where γ=2
c. Contraction (C) which lays a new vertex $x_c=x_w+\beta(M-x_w)$, where β=0.5 or β=-0.5
d. Shrinkage/reduction (S) which replaces all vertices except the best one. $x_s=(x_s+x_b)/2$

The initial simplex is created from an initial guess x_1. The method compares function values at the three vertices of triangle which are considered as $f(x_g)$, $f(x_b)$ and $f(x_w)$ that are considered as good to worst point of triangle. The step Nelder-Mead method [22] are

1. Order the vertices $x_b<x_g<x_w$. Calculate centroid.
2. If $f(x_r)<f(x_b)$ the perform either reflection (R) or expansion (E) else perform contraction (C)
3. If $f(x_g)<f(x_r)$ then accept x_r and compute
4. If $f(x_r)<=f(x_g)$ then $R\rightarrow E$ and compute x_e
5. If $f(x_e)<=f(x_r)$ then accept xe and go to 1 else accept x_r
6. If $f(x_r)<f(x_w)$ then $x_w\rightarrow R$ and compute C with β=0.5
7. If $f(x_r)>=f(x_w)$ then C with β=-0.5
8. If $f(x_c)<f(x_w)$ then $x_w\rightarrow C$ else make S.
9. If stopping criterion is met, then stop, else go to 1.

The geometric algorithm with the figure is shown in Figure 2.

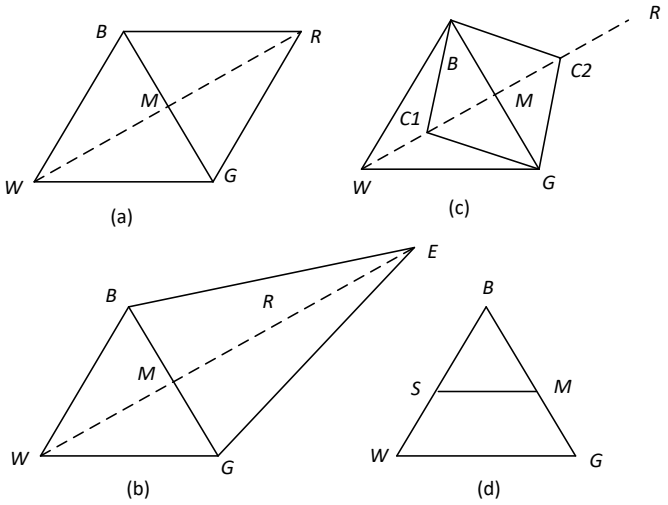


Fig. 2. Diagram of the Nelder Mead Simplex algorithm.

D. Cost Function of ITAE

The optimal algorithm needs an objective function as an initial value to start the process and usually uses performance criteria as a function of error. The best performance is obtained by selecting the correct performance criteria. There are several criteria that the parameters are dependent on, one of those is ITAE.

The ITAE offers the best performance because this criterion does not discriminate against the larger initial error in the first or transient response, but does penalize smaller error at a steady-state condition [23]. Moreover, the ITAE results in a conservative setting, and it is uncomplicated and more time-saving than ITSE. The ITAE objective function is given by (9) where t is time and e is an error.

$$J_0 = \int_0^\infty t\|e\|(t)dt \tag{9}$$

III. DESIGN OF OPTIMAL PITAE AQM

PI controllers have been used in industrial processes. They have been implemented in network congestion control protocols for the past decade because of their simple mechanism, easy implementation, and good response.

For the searching of optimum values, the Nelder-Mead simplex method was chosen because of its fast convergence and easy implementation. To use this, this method needs initial values and an objective function that will be described in the following subsection. The proposed scheme PITAE controller and its parameters K_p and K_i are tuned by optimization using the ITAE criteria as shown in Figure 3.

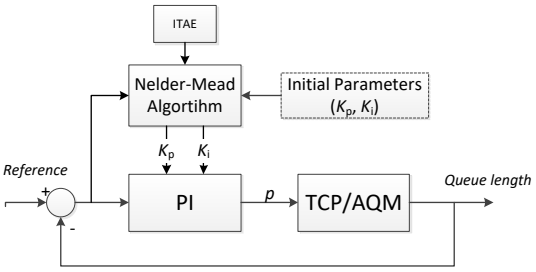


Fig. 3. Design of proposed PITAE AQM tuning.

The initial guess parameters used the parameters of PI, i.e., K_p=1.8182x10-5 and K_i=0.9612x10-5 that was tuned by Hollot with gain and phase margin approach. PIU was tuned by Ustebay with the small-gain theorem [24]. The result of our proposed design was calculated by the Matlab/Simulink toolbox that implements the ITAE objective function based on Hollot linearized network parameters [2]. The purpose of our proposed design was to avoid sensitive parameters by using an optimal searching method employing the Matlab *fminsearch* tool [23]. The resulting parameters of the PITAE are depicted in Table I.

TABLE I
AQM OPTIMIZATION RESULTS IN NORMAL TRAFFIC

Parameter	PI	PIU	PITAE
K_p	1.8182e-5	3.5232e-5	4.566e-5
K_i	0.9612e-5	0.8953e-5	2.317e-5

IV. SIMULATION AND ANALYSIS

The simulation topology with a bottleneck link is shown in Figure 4, where the dumbbell network with multiple TCP connections from $S_{1,2,\ldots N}$ to $D_{1,2,\ldots N}$, shares a bottleneck link between routers R_1 and R_2. The configuration of link capacity and propagation delay is also shown in Figure 4. The link capacity of C is set to 15 Mbps, and the propagation delay T_p is set to 5 ms in a normal traffic scenario. Other links have their capacities set to 10 Mbps and the propagation delays set to 5 ms. The maximum buffer size of each router is set to 800 packets, and the queue length target is set to 200 packets. The average packet size is 500 bytes and the simulation running time is 60 s.

The PI AQM algorithm was implemented in an NS2 simulator using the Hollot design with $a = 1.822 \times 10^{-5}$, $b = 1.816 \times 10^{-5}$, and $F_s = 160$. This was compared with the PIU scheme using the parameters of K_p=3.5243x10^{-5} and K_i=0.8953x10^{-5} [24]. This work focuses on the evolution of the queue length, as one of the strategic keys for AQM performance.

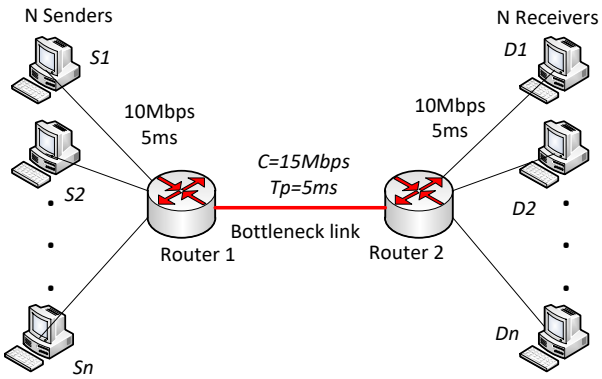


Fig. 4. Network simulation topology.

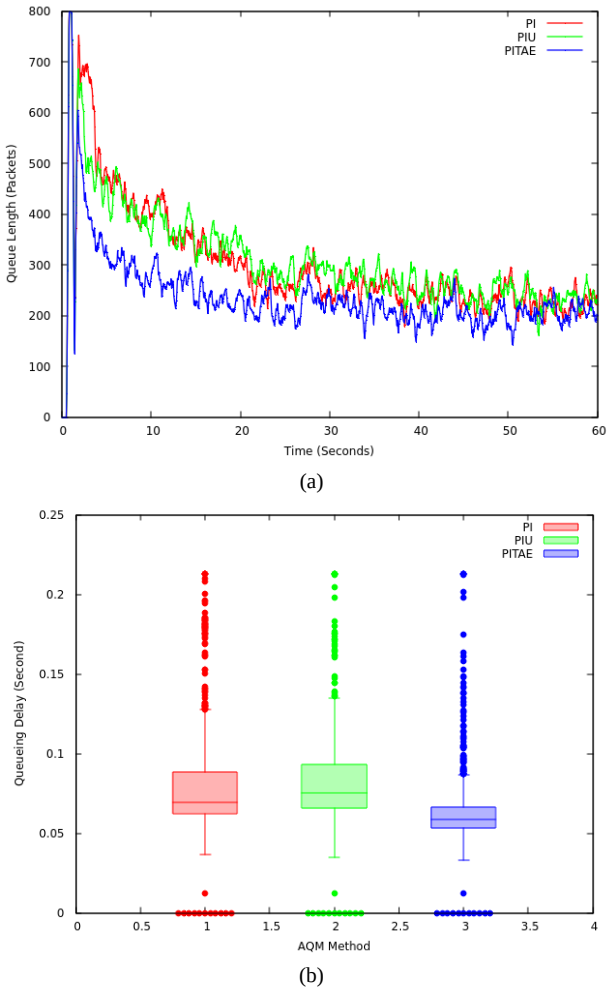


Fig.5. Performance in normal traffic (a) Queue length (b) Queuing delay

A. *Result in Normal Traffic Condition*

This simulation demonstrated the performance in normal traffic that is a usual traffic condition, where there was a greedy 100 long-live TCP flow, a shared bottleneck link with a 15 Mbps capacity, and a propagation delay T_p of 5 ms. Figure 5a illustrates the evolution of the queue length for PI, PIU, and PITAE. PI (the red line) and PIU (the green line) show that both have large overshoot in the queue and have long transient response times. Meanwhile, PITAE reduced the overshoot and quickly regulated the queue length to the target value, because the parameters of our proposed are tuned in minimizing error with the multiplication of time.

Figure 5b shows that PITAE had the shortest queuing delay and deviation or jitter from the others. The jitter was approximated by the length from the lower to upper boundaries of the boxplot in the graph of the queuing delay. Table II shows that all the methods had the same throughput. However, PITAE had the smallest average queue length and the shortest bottleneck link delay.

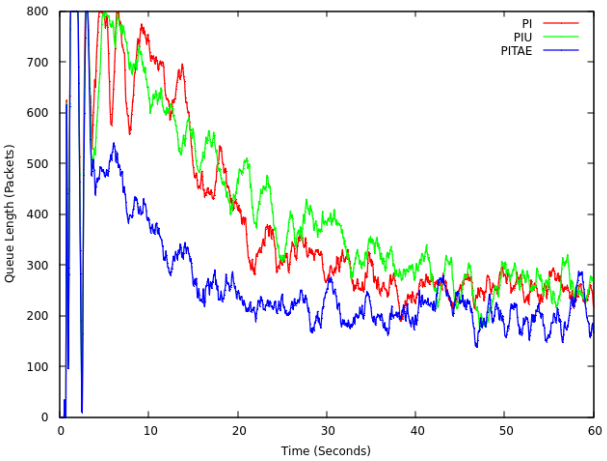
TABLE II
AQM OPTIMIZATION RESULTS IN NORMAL TRAFFIC

Parameter	PI	PIU	PITAE
Average of queue (packets)	301	307	237
Throughput (Kbps)	14993	14993	14993
Bottleneck link delay (ms)	93	94	74

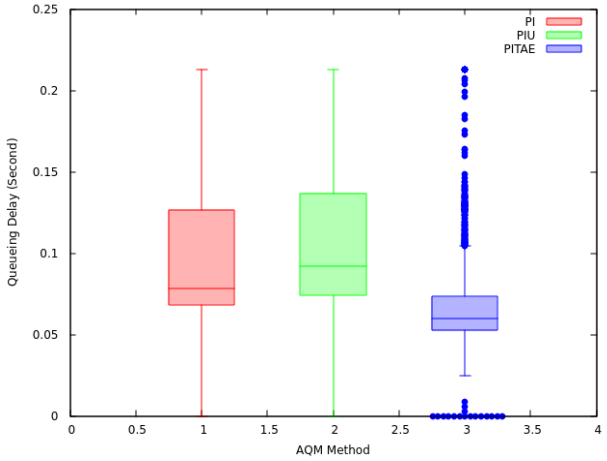
B. *Case in Heavy Traffic*

In the case of overload traffic with 400 greedy TCP connections and a long delay propagation time, 100 ms is depicted in Figure 6. It shows that PI and PIU could not handle a big load initially which is shown in Figure 6a. This led to dropped packets due to buffer overflow. Both PI and PIU also required a long settling time. However, PITAE could maintain the queue length with excellent response times and reduced overshoot in the overloaded case.

As presented in Figure 6b, the queuing delay in this scenario was shortest using PITAE. The average queue length and the link delay for those AQMs are presented in Table III. Again, PITAE performed the best.

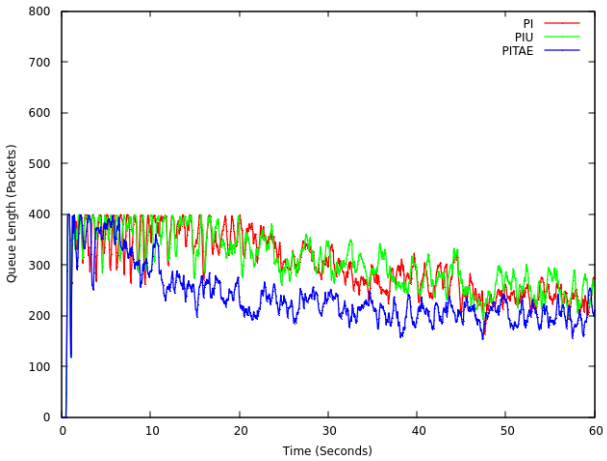


(a)

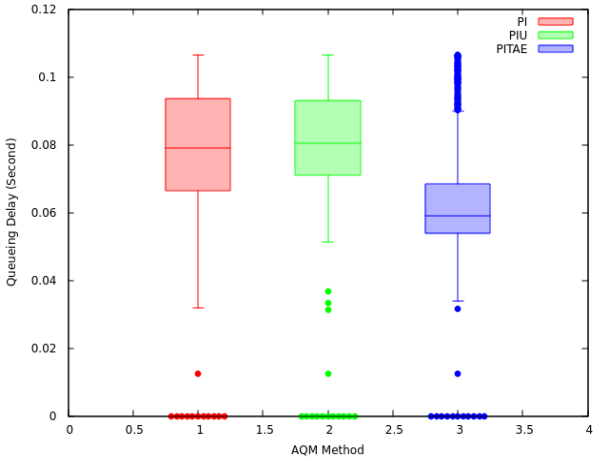


(b)

Fig.6. Performance in heavy traffic (a) Queue length (b) Queuing delay



(a)



(b)

Fig.7. Performance for small buffer size (a) Queue length (b) Queuing delay

TABLE III
AQM OPTIMIZATION RESULTS IN HEAVY TRAFFIC

Parameter	PI	PIU	PITAE
Average of queue (packets)	382	403	265
Throughput (Kbps)	14943	14942	14941
Bottleneck link delay (ms)	212	218	177

TABLE IV
AQM OPTIMIZATION RESULTS IN SMAL BUFFER SIZE

Parameter	PI	PIU	PITAE
Average of queue (packets)	299	305	237
Throughput (Kbps)	14993	14993	14993
Bottleneck link delay (ms)	92	94	74

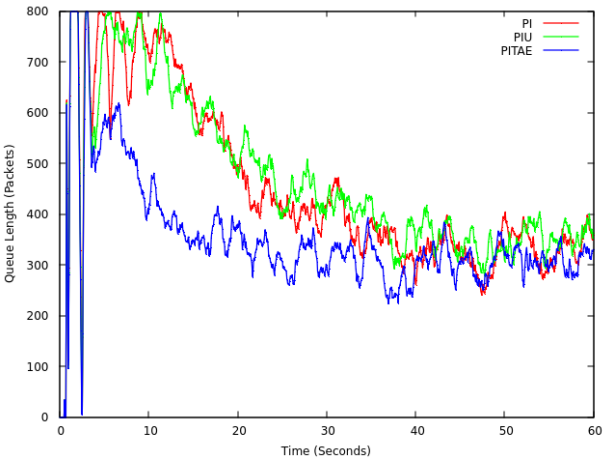
C. Test for Small Buffer Size

Next, the proposed method was validated versus the well-known PI and PIU schemes by reducing the buffer size to 400 packets. In Figure 7a, the PI and PIU algorithms generated long buffer overflows at 20 s, which means almost all of the arrived packets were dropped; this led to a synchronization problem. In contrast, the PITAE scheme had a stable response by keeping the queue length around the desired target from the beginning of the process.

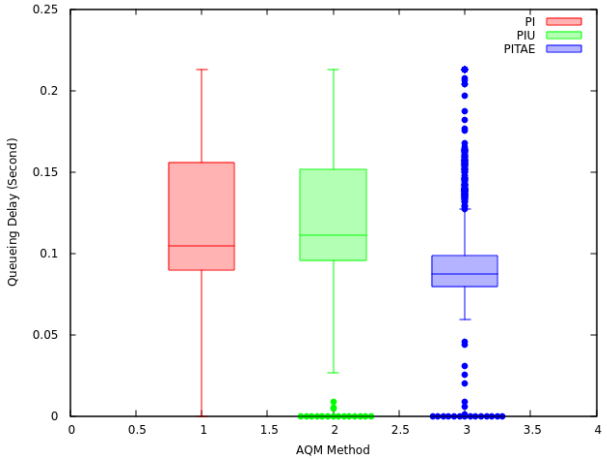
PITAE had the best performance in regulating the queuing delay as shown in Figure 7b. The further analysis shown in Table IV shows that our proposed scheme reduced the average queue length and link delay up to 25 percent when compared with the other methods.

D. Result on Changing the Set-Point

Different target values were tested by increasing the reference queue from 200 to 300 packets in a heavy traffic scenario. This is shown in Figure 8a, PI and PIU had a bad response with buffer overflow in the initial process, along with a long settling time at around 37 s. PITAE was good enough to regulate the queue length around the set-point value and had a quick settling time of 22 s. Figure 8b depicts that PITAE had the best performance in keeping the queuing delay shorter as compared with the others. PITAE could regulate the average queue length closes to the target value of 300 and keep a small link delay as shown in Table V.

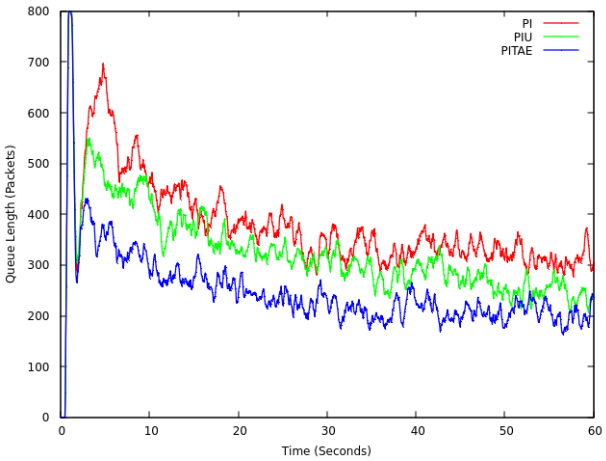


(a)

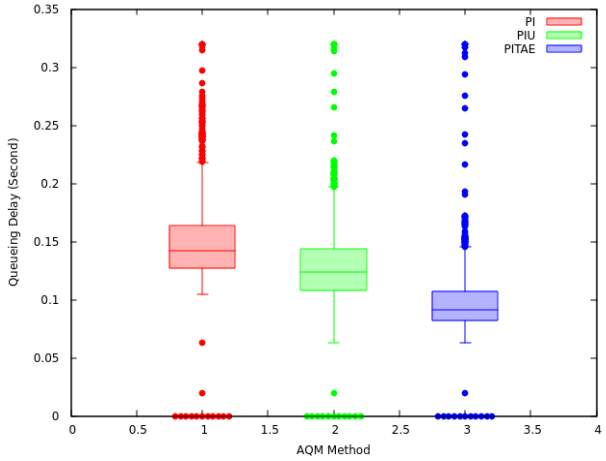


(b)

Fig.8. Performance on changing set-point (a) Queu length (b) Queuing delay



(a)



(b)

Fig.9. Performance in different link capacities (a) Queue length (b) Queuing delay

TABLE V
AQM OPTIMIZATION RESULTS ON CHANGING THE SET-POINT

Parameter	PI	PIU	PITAE
Average of queue (packets)	454	469	353
Throughput (Kbps)	14945	14942	14940
Bottleneck link delay (ms)	233	237	203

TABLE VI
AQM OPTIMIZATION RESULTS IN DIFFERENT LINK CAPACITIES

Parameter	PI	PIU	PITAE
Average of queue (packets)	321	326	245
Throughput (Kbps)	9995	9995	9995
Bottleneck link delay (ms)	145	147	113

E. Scenario of Different Link Capacities

This scenario changed the link capacity from 15 Mbps to 10 Mbps. This is presented in Figure 9a, PI (the red line) experienced a long overshoot and failed to achieve the reference value of 200, and PIU (the green line) reduced overshoot a little from PI. Meanwhile, the proposed solution, PITAE (the blue line), had the best response, keeping the queue length to the desired reference value. The PITAE queuing delay was the smallest as illustrated in Figure 9b. The characteristics of those three mechanisms are shown in Table VI. PITAE was more powerful than the other two for average queue length and bottleneck link delay, which was 245 packets and 113 seconds, respectively. Furthermore, based on Table I and performance in all scenarios above, it can be concluded that bigger K_p and K_i values keep the queue closed to the target value of 200 packets.

CONCLUSION

This paper has formulated a robust PI AQM based on the ITAE criterion for network congestion control. Tuning of the parameter controller was run by the Nelder-Mead simplex method to find the optimal value of PI parameters. This was implemented in a network simulator for further analysis and compared with PI (which was used as an initial parameter for optimization) and PIU. The results indicated that our proposed approach had the best performance in five network traffic scenarios in regulating the queue close to the target value and keeping the shortest queuing delays.

ACKNOWLEDGEMENTS

The authors would like to thank the Funding of Doctor Dissertation Program (PDD), Ministry of Higher Education and Technology Research of the Republic of Indonesia, for supporting this research.

REFERENCES

[1] S. Floyd and V. Jacobson, "Random early detection gateways for congestion avoidance," *IEEE/ACM Transactions on Networking*, vol. 1, pp. 397-413, 1993.

[2] C. V. Hollot, V. Misra, D. Towsley, and G. Weibo, "Analysis and design of controllers for AQM routers supporting TCP flows," *IEEE Transactions on Automatic Control*, vol. 47, pp. 945-959, 2002.

[3] D. Ustebay and H. Ozbay, "Switching Resilient PI Controllers for Active Queue Management of TCP Flows," in *2007 IEEE International Conference on Networking, Sensing and Control*, 2007, pp. 574-578.

[4] D. Üstebay, H. Özbay, and N. Gündes, "A new PI and PID control design method for integrating systems with time delays," in *Proceedings of the 6th WSEAS International Conference on Signal Processing, Robotics and Automation*, 2007, pp. 60-65.

[5] M. XiaoYan, L. HongGuangr, and C. District, "A novel parameter tuning algorithm for AQM-PI controllers," *Research Journal of Applied Sciences, Engineering and Technology*, vol. 4, pp. 75-78, 2012.

[6] P. Dash, S. Bisoy, N. Kumar Kamila, and M. Panda, *Parameter Setting and Stability of PI Controller for AQM Router*, 2016.

[7] A. Domański, J. Domańska, T. Czachórski, and J. Klamka, "The use of a non-integer order PI controller with an active queue management mechanism," in *International Journal of Applied Mathematics and Computer Science* vol. 26, ed, 2016, p. 777.

[8] S. Testouri, K. Saadaoui, and M. Benrejeb, "Evaluation of a PSO Approach for Optimum Design of a First-Order Controllers for TCP/AQM Systems," *Evaluation*, vol. 4569, p. 3122, 2013.

[9] P. N. Baldini, G. Calandrini, and P. Doñate, "PSO algorithm-based robust design of PID controller for variable time-delay systems: AQM application," *Journal of Computer Science & Technology*, vol. 15, 2015.

[10] S. Chebli, A. El Akkary, N. Sefiani, and N. Elalami, "PI Stabilization for Congestion Control of AQM Routers with Tuning Parameter Optimization," *IJIMAI*, vol. 4, pp. 52-55, 2016.

[11] A. E. Abharian and M. Alireza, "Hybrid GA-BF based intelligent PID active queue management control design for TCP network," in *Electronics Computer Technology (ICECT), 2011 3rd International Conference on*, 2011, pp. 227-232.

[12] X. Wang, Y. Wang, and H. Zhou, "Optimal Design of AQM Routers with D-Stable Regions Based on ITAE Performance," in *2006 First International Conference on Communications and Electronics*, 2006, pp. 78-83.

[13] M. Mohebbi and M. Hashemi, "Designing a 2-Degree of Freedom Model of an Unbalanced Engine and Reducing its Vibrations by Active Control," *International Journal of Technology*, vol. 8, pp. 858-866, 2017.

[14] H. Wang, W. Wei, Y. Li, C. Liao, Y. Qiao, and Z. Tian, "Two-Degree-of-Freedom Congestion Control Strategy against Time Delay and Disturbance," in *Global Telecommunications Conference (GLOBECOM 2010), 2010 IEEE*, 2010, pp. 1-5.

[15] P. Rukmani and R. Ganesan, "Enhanced Low Latency Queuing Algorithm for Real Time Applications in Wireless Networks," *International Journal of Technology*, vol. 7, pp. 663-672, 2016.

[16] M. Fajri and K. Ramli, "Optimizing PID TCP/AQM using nelder-mead simplex approach," presented at the Proceedings of the 3rd International Conference on Communication and Information Processing, Tokyo, Japan, 2017.

[17] T. Issariyakul and E. Hossain, *Introduction to network simulator NS2*: Springer Science & Business Media, 2011.

[18] J. Glos and P. Vaclavek, "Efficient control of automotive R744 heat pump using Nelder-Mead simplex method," in *2017 IEEE International Conference on Industrial Technology (ICIT)*, 2017, pp. 785-790.

[19] V. Misra, W.-B. Gong, and D. Towsley, "Fluid-based analysis of a network of AQM routers supporting TCP flows with an application to RED," *SIGCOMM Comput. Commun. Rev.*, vol. 30, pp. 151-160, 2000.

[20] Y. Li, K. T. Ko, and G. R. Chen, "Transient behaviour of PI-controlled AQM," *Electronics Letters*, vol. 42, pp. 494-495, 2006.

[21] R. Abraján Guerrero and L. Márquez Martınez, "Automatic pid tuning, a heuristic optimization approach," in *Congreso Nacional de Control Automático*, 2010, pp. 6-8.

[22] D. Mukherjee, P. K. Kundu, and A. Ghosh, "PID controller design for an interacting tank level process with time delay using MATLAB FOMCON toolbox," in *2016 2nd International Conference on Control, Instrumentation, Energy & Communication (CIEC)*, 2016, pp. 1-5.

[23] W. Xiuli, W. Yongji, Z. Hui, and H. Xiaoyong, "PSO-PID: a novel controller for AQM routers," in *2006 IFIP International Conference on Wireless and Optical Communications Networks*, 2006, pp. 5 pp.-5.

[24] H. U. Ünal, D. Melchor-Aguilar, D. Üstebay, S.-I. Niculescu, and H. Özbay, "Comparison of PI controllers designed for the delay model of TCP/AQM networks," *Computer Communications*, vol. 36, pp. 1225-1234, 2013.