International Journal of Communication Networks and Information Security (IJCNIS) Vol. 12, No. 1, April 2020

Adaptive Active Queue Management based on Queue Ratio of Set-point Weighting

Misbahul Fajri[1] and Kalamullah Ramli[1]

[1]Department of Electrical Engineering, Faculty of Engineering, Universitas Indonesia, Kampus Baru UI, Depok 16424, Indonesia

Abstract: Presently, active queue management (AQM) is one of the important considerations in communication networks. The challenge is to make it simple and robust in bursty traffic and uncertain network conditions. This paper proposes a new AQM scheme, an adaptive ratio proportional integral (ARPI), for adaptively controlling network congestion in dynamic network traffic conditions. First, AQM was designed by adding a set-point weighting structure to a proportional integral (PI) controller to reduce the burstiness of network traffic. Second, an adaptive set-point weighting based on the ratio of instantaneous queue length to the set-point queue and the buffer size was proposed to improve the robustness of a non-linear network. The proposed design integrates the aforementioned expectations into one function and needs only one parameter change to adapt to fluctuating network condition. Hence, this scheme provides lightweight computation and simple software and hardware implementation. This approach was analyzed and compared with the PI AQM scheme. Evaluation results demonstrated that our proposed AQM can regulate queue length with a fast response, good stability under any traffic conditions, and small queuing delay.

Keywords: Network traffic, Congestion Control, Active Queue Management; Adaptive controller, Proportional Integral, Set-point weighting**.**

1. Introduction

For a comprehensive network, active queue management (AQM) is an important intermediate network solution for network traffic control that could not be obtained by former attempts. Buffer overflow is one of mainly caused in a node that makes congestion in communication problem affecting network performance [1]. AQM performs packet dropping in a buffer network router before congestion. Concerning in the buffer occupancy makes AQM to know traffic condition and decides how to treat the arrival packet, also buffer size awareness can shorten the delay experienced by packet [2]. The first AQM, random early detection (RED) [3] was proposed as a solution to support 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 ability to maintain the desired queue length. The drawback of RED is its sensitivity to parameter setting in dynamic network load. Many variants of RED have been released to tackle the limitation of RED in a heuristic approach. Hollot proposed proportional integral (PI) AQM [4] to overcome the challenges faced by RED and to provide PI tuning parameters using the control theory approach through the TCP flow model. Nonetheless, PI is still problematic for obtaining proper parameter values for good response and robustness in an uncertain traffic network. From the limitation of two well-known AQMs above arise enhanced research to introduce the adaptivity of AQM.
Adaptive AQM scheme based on queue length in heuristic method such as ARED [5] and improved in [6] which varies the maximum drop probability using two different constant factors based on the online average queue length. it is simple but in specific conditions, it has a bad response and the low-pass filter used for queue averaging leads to the sluggish and indolence behavior of this scheme [7]. Furthermore, Adaptive AQM based on queuing delay using PI controller is PIE [8] which to overcome bufferbloat on access links that much attention during the past few years. It achieved good performance in normal delay traffic conditions but worst as delay increase [9]. However, PIE is not scalable because it is effective on a medium-scale networks, such as on the access link of the user. Meanwhile, more advanced PI AQMs have been proposed using the optimization method. In [10,11] regulate queue length with adaptive PI with neuron based, leads to high computation and hard in implementation along with heuristic optimization algorithm.
Other research [12] was to obtain PID parameters using an optimization approach and to suggest to develop with different structure of PID controllers to implement it with an adaptive mechanism because many AQMs have problems with robustness, especially in changing traffic conditions and effectively handling bursty traffic. To overcome this, we propose an adaptive queue ratio proportional integral (ARPI) regulates queue length to the reference value. The addition of set-point weighting at PI AQM is expected to improve robustness and to reduce burst traffic for controlling dynamic network traffic. The main contributions of this paper are summarized as follows. First, a new weighted formula to estimate traffic load based on the ratio of occupancy queue in a buffer, and introduces PI Set-point weighting controller to reduce bursty traffic in AQM. Second, a low-cost self-tuning algorithm with only one parameter to adjust periodically adaptive controller, therefore it will be scalable and have a potentially easy implementation in software and hardware.
The paper is organized as follows. The related works of this research are reviewed in Section 2. A concept of TCP/AQM congestion control is given in Section 3. ARPI AQM design is defined in Section 4. Simulation analysis is given in Section 5. Section 6 concludes this paper.

2. Related Works

PI controller has been massively used in the industrial process until now. PI has been implemented in network congestion control protocols for the past decade because it is a simple mechanism, easy implementation, and good response. Through the control theory approach, AQM can be designed and analyzed practically such as PI [4], 2DoF PID [13], PI–PD [14] and PID AQM [15] solution to overcome lacks of heuristic approach. Likewise, another work [12] uses an optimal control method to tune parameters of PID AQM. It showed that the structure type of PID can be deployed in communication networks.

The first AQM was designed using control theory with a PI controller and a method for tuning its parameters [4]. However, the fixed-parameter PI controller is sluggish and hard to speed up a response [16]. Alternatively, an adaptive controller to overcome the fixed controller to tackle dynamic network conditions. Many proposed adaptive AQMs have been implemented proposed, but most of them were not the satisfaction of AQM goals, e.g.; the existing adaptive AQM schemes are less appropriate under dynamic conditions, their need for pre-specification and pre-tuning parameters [17], as below.

Zhang et al. [18] proposed a self-tuning structure AQM (STPI) which online estimates link capacity and traffic load, this approach is a complex algorithm and difficult to estimate network parameters. Chen et al. [19] Proposed adaptive PI which adjusts AQM parameters using a pole-placement method. But it was a complex tuning algorithm and needed unknown network parameters. Chang et al. [20] proposed an adaptive PI AQM (RPI), in which queue length error is increased cause it is larger than a certain threshold. Then they improved this method in terms of transient performance over a wide range of network parameters [21]. Hong et al. [22] introduced an adaptive PI AQM, that used gain and phase margins to adjust parameters. They also improved their method [23] in using gain margin specifications. However, both of their methods are complex and require network parameters, which are hard to get these values in a router.

Xu et al. proposed N-PI [24] based on hyperbolic secant function tuning to get robust stability and good dynamic result, but it was not simple computation to get controller parameters. Sun et al. proposed IAPI in [25] using three mechanisms for adaptive PI AQM to regulate queue length, therefore, these cannot be considered as an effective scheme in long-delay networks [7]. Wang et al. Proposed API [26] that adjusts the integral gain to get a faster response and dynamically the packet drop probability based on three conditions of queue length error.

In summary, some of those above AQM's are complex computations and difficult to estimate tuning parameters. The goal of the AQM scheme should scalable and simple algorithm for deployment in software and hardware. The AQM algorithms should not require tuning or reconfiguration of initial parameters [9], as mention above, they should not need pre-specification and pre-tuning parameters.

3. TCP Flow and AQM Congestion Control

The TCP flow model in non-linear differential equations has been formulated by Misra as a solution for further analysis of the network congestion control algorithm [27]:

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

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

Where W = Average TCP window size (packets), $R(t)$ = Round-trip-time = $Tp + (q(t)/C)$ (s), Tp = Propagation delay (s), $q(t)$ = Queue length (packets), C = Link capacity (packet/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 TCP flow model can assist in designing and determining the best AQM controller parameters because TCP control has a feedback process, i.e., ACK signal, that is generated by a receiver to inform the sender when a packet is received; then, it is used by the sender 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 analyzed using a control theory approach, e.g., the transfer form of the TCP flow model.

In [4,28], the stochastic equation above was transformed by classic control theory through linearization and by ignoring the time-out mechanism. 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) is derived by $\dot{W} = 0$ 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 TCP/AQM flow is given by:

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

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

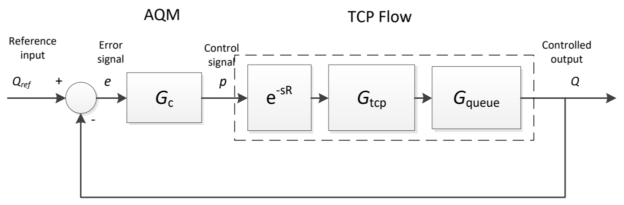


Figure 1. The Feedback of TCP/AQM system.

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

$$G_p(s) = G_{tcp}(s)G_{queue}(s)e^{-R_0 s} \qquad (4)$$

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

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

Where K_p is proportional gain and K_i is integral gain. To obtain the z-domain transfer function, equation (5) is converted from the s-domain using bilinear transform (Tustin's rule) to preserve stability [29]. Then the discrete PI controller is given by

$$p(n) = p(n-1) + a.e(n) + b.e(n-1) \qquad (6)$$

Where T_s is the sampling time and

$$a = K_P + \frac{K_I T_s}{2} \quad (7)$$
$$b = K_p - \frac{K_I T_s}{2}$$

4. Design of ARPI AQM

To achieving better robustness, the controller should adapt to changing network conditions by adjusting its parameters. Therefore, the challenge is to design and find the proper formula for the abovementioned purpose. Based on the TCP flow model, a non-linear factor is caused by three parameters: load network (*N*), round-trip time (*RTT*), and link capacity (*C*). These parameters are difficult to obtain in real-time processes. Simplifying the implementation is also challenging. We design ARPI AQM to overcome the above problems as below.

4.1 Proportional Integral Controller with Set-point Weighting

Firstly we used the PI set-point weighting controller to design our proposed AQM, it enhances from PI controller and its transfer function is.

$$G_c(s) = K_p(\beta r(s) - y(s)) + \frac{K_i}{s} e(s) \quad (8)$$

Where β is a set-point weighting. The function of set-point weighting reduces the overshoot response in a transient process [30]. In a case of network congestion, the function is to reduce bursty or bulk traffic. The dynamic network shows non-linear behavior. In a non-linear process, properly tuned parameters are not suitable for other circumstances, and constant values of the parameters will generate a conservative response [31].
Implementing set-point weighting is modified in the proportional term [32] that to include a parameter weighting factor β on the reference. Because proportional action is the present value of error [33], so the discrete of the proposed adaptive PI AQM (ARPI) as given.

$$p(n) = p(n-1) + a.[y(n) - \beta r(n)] + b.e(n-1) \quad (9)$$

4.2 Adaptive Queue Ratio for ARPI

Secondly, we present an ARPI controller to periodically adjust the parameter of set-point weighting in (9), based on the instantaneous queue occupancy over time in the router. The proposed design is shown in Figure 2.

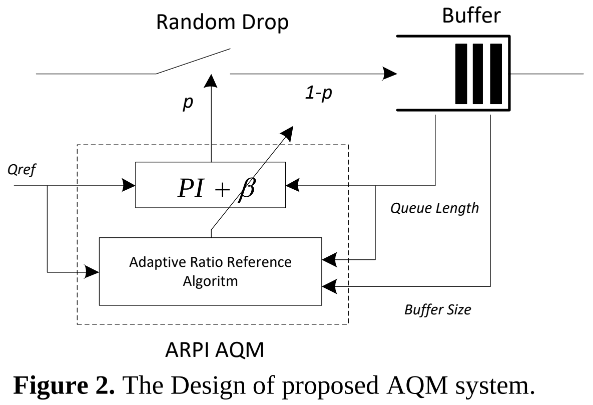
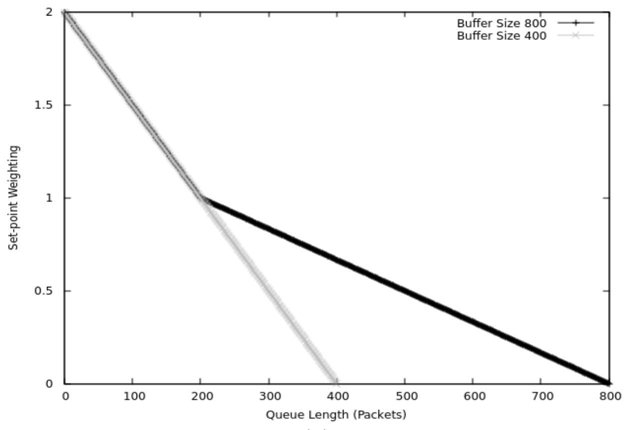


Figure 2. The Design of proposed AQM system.

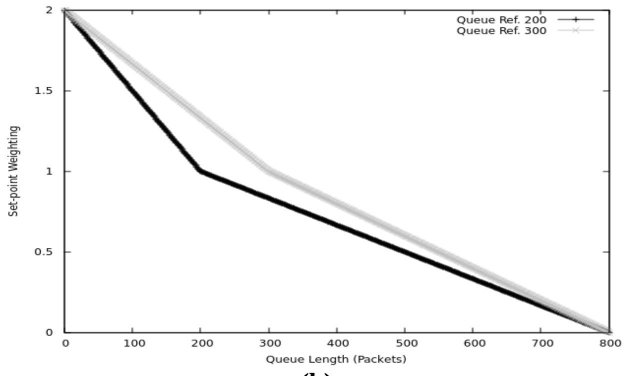
One of the literature presented online adaptive tuning set-point weighting using sigmoid function based on changing of variable process error [34], but still needs pre-tuning of the other parameter. Our proposed design determines set-point weighting by an analytical approach instead of a control theory method, to avoid high computation costs. Thus, β value is computed periodically in every arrival packet of the buffer based on the ratio of instantaneous queue length to reference queue length.
Queue ratio is divided into upper and lower ratios. The upper ratio is a condition when the error is more than zero and is seen as an overshoot response. This ratio is the error divided by buffer size minus the queue reference. A lower ratio indicates an error that is less than or equal to zero and is perceived as an overdamped response. This ratio is the error divided by queue length minus the queue reference. According to the above values, if error = 0, queue ratio = 0, then β = 1. Hence, this formula is one minus the queue ratio. The proposed formula for adaptive set-point weighting is given below:

$$\beta(t) = \begin{cases} 1 - \dfrac{q(t) - qref}{qsize - qref} & q(t) - qref > 0 \\ \\ 1 - \dfrac{q(t) - qref}{qref} & q(t) - qref \le 0 \end{cases} \quad (10)$$

Where, *q*(*t*)= Instantaneous queue length, *qref* = Queue reference, *qsize*= Buffer size of router, and $\beta(t)$= Set-point weighting. The characteristic of β against all of the instantaneous queue length in buffer size is depicted in Figure 3 (**a**) the black line represents buffer sizes of 800 packets, and the grey lines denote buffer sizes of 400 packets. The variance range of the queue length varies from empty to full, dynamically generating set-point weighting values between 0 and 2 for both overshoots and overdamped processes.



(a)



(b)

Figure 3. Value of β with: (**a**) Q_{ref} 200 for different buffer sizes; (**b**) buffer size of 800 for different Q_{ref}.

Our proposed method is simple because only one adaptive parameter (β), that does not yield complex formula is required. Thus, it makes lightweight computation and easy implementation in software and in hardware possible. ARPI pseudo-code algorithm for implementation is as below:

Algorithm 1. ARPI (*Called upon arrival a new packet*)

```
qer=qlen-qref;
 If (qer> 0) spw=1-(qer/brw);
 Else spw=1-(qer/qref);
 p=a(qlen-spw*qref)-b(qold-qref)+pold
 If (p<0) p=0
 Else (p>1) p =1
 Random = uniformRandom(0,1);
 If (Random > p) Enqueue the packet;
 Else Drop the packet;
```

Parameter:

qref	Reference of queue length
qlen	Queue length
qer	Queue error
brw	Buffer reference weighting (buffer limit–*qref*)
spw	Set-point weighting
qold	Old queue
pold	Old probability
p	Drop probability
a	Proportional gain
b	Integral gain

5. Simulation and Result

ARPI AQM protocol algorithm is implemented in an NS2 simulator [35] to evaluate its performance in six scenarios. Another AQM, i.e., PI, is also simulated for comparison. The PI parameters tuned by Hollot are still used in our proposed PI, i.e., $a = 1.822 \times 10^{-5}$, $b = 1.816 \times 10^{-5}$, and $F_s = 160$. This work focuses on the evolution of queue length, as one of the strategic keys for AQM performance. If an AQM scheme quickly regulates queue length to the desired value and maintains stability with less oscillation, then the desired network performance is expected to be achieved. Two of the desired network performance criteria predicted are queuing delay and high link utilization [36].

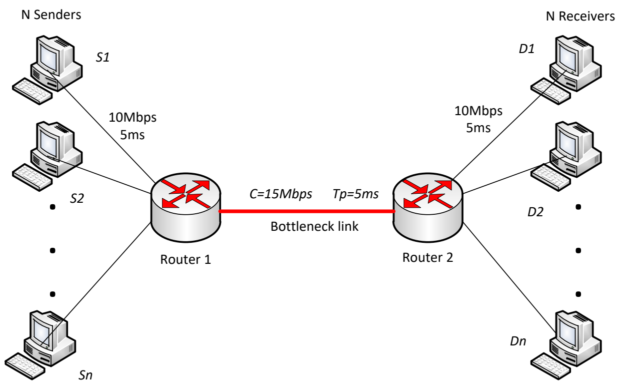


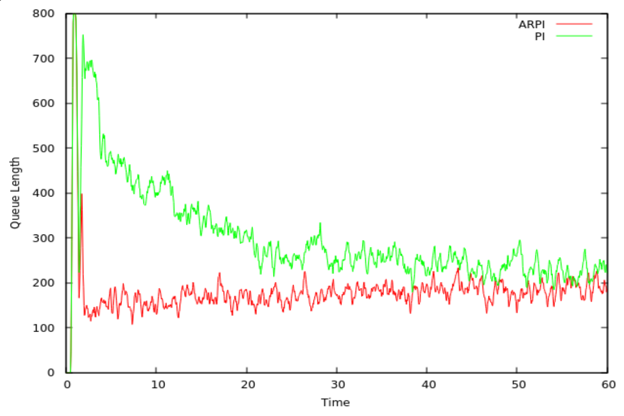
Figure 4. Network simulation topology.

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. 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, their capacities in 10 Mbps and propagation delay 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.

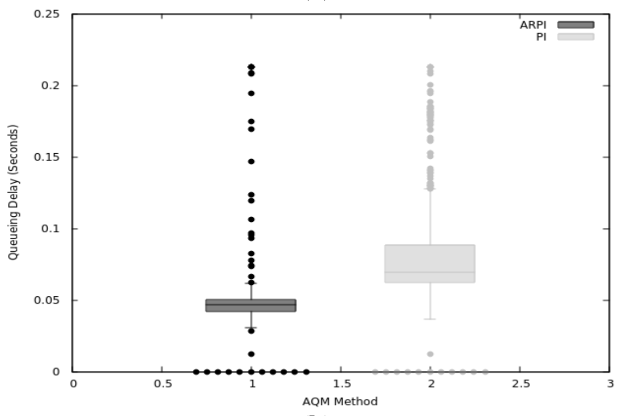
5.1 Performance in Normal Traffic

This simulation demonstrates the performance in normal traffic that is regular traffic condition, where there is a greedy 100 long-lived TCP flow, sharing bottleneck link with 15 Mbps capacity, and propagation delay T_p of 5 ms. Figure 5 (**a**) illustrates the evolution of queue length for both ARPI and PI. PI shows a sluggish response, which yields a big overshoot queue and long transient response time. However, ARPI reduces the big overshoot and quickly regulates the queue length to the target value.

Figure 5 (**b**) shows that ARPI has queuing delay under 50 seconds and small jitter, and both values are smaller than PI. A Jitter is approximated by the length from the lower to upper boundaries of the boxplot in the graph of queuing delay. Table 1. presents that both algorithms have the same throughput. However, ARPI has a smaller value of average queue length and shorter bottleneck link delays.



(a)



(b)

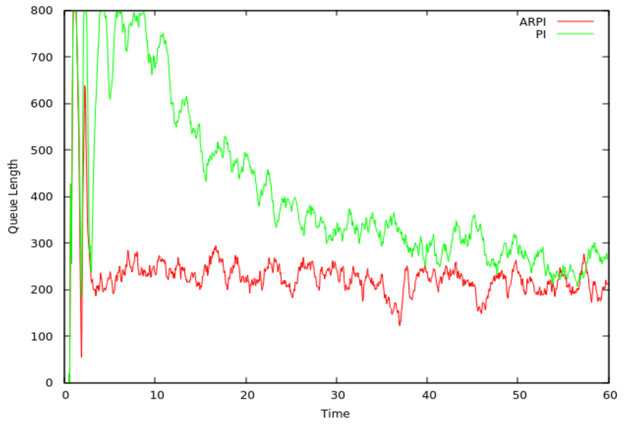
Figure 5. Performance in normal traffic based on **(a)** queue length and **(b)** queuing delay.

Table 1. AQM performance analysis at nominal traffic.

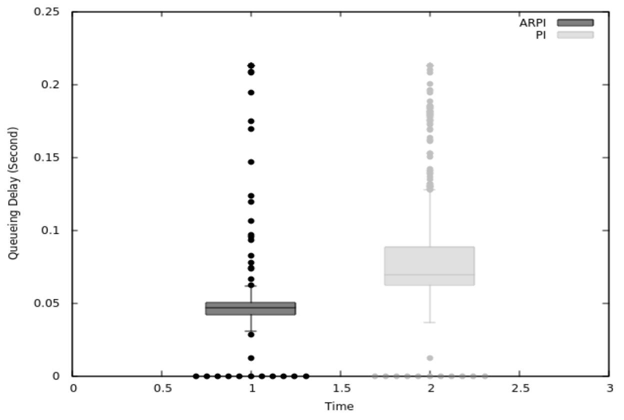
KPI	PI	ARPI
Average of queue (Packets)	301	179
Throughput (Kbps)	14993	14993
Bottlenect link delay (ms)	93	58

5.2 Performance in Heavy Traffic

The case of overload traffic with 300 greedy TCP connections and long delay propagation time, 50 ms. Figure 6 (**a**) shows that the PI cannot handle a big load initially. This leads to drop packets due to buffer overflow. PI also requires a long settling time. However, ARPI depicted by the red line can maintain a good response and a little overshoot in the overload packet. The queuing delay for Figure 6 (**b**) in this scenario points out that ARPI is having a smaller jitter than PI. The average queue length and the link delay for both AQMs are shown in Table 2. That ARPI is better than PI.



(a)



(b)

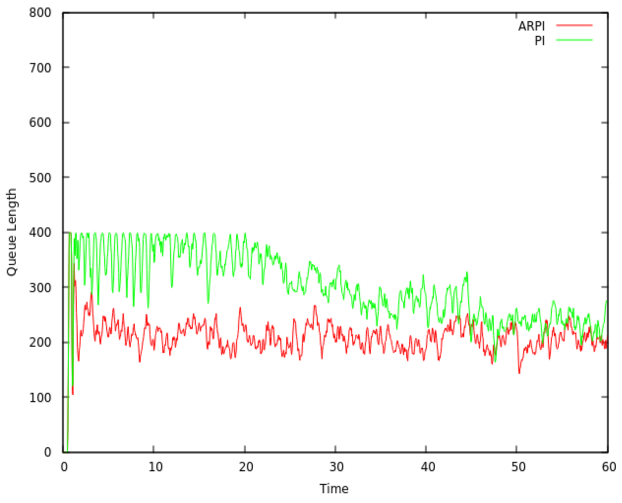
Figure 6. Performance in heavy traffic based on **(a)** queue length and **(b)** queuing delay.

Table 2. AQM performance analysis in heavy traffic.

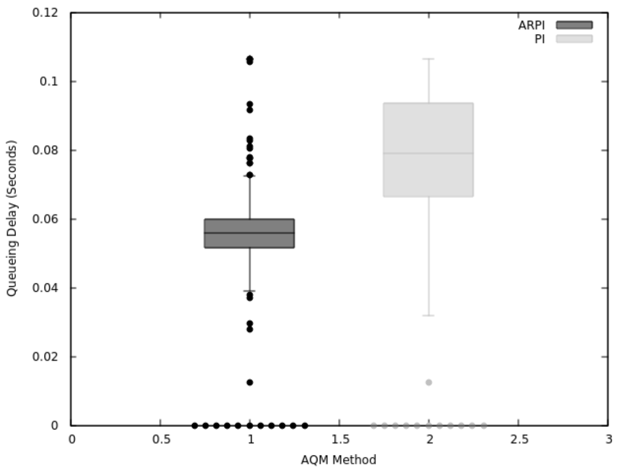
KPI	PI	ARPI
Average of queue (Packets)	407	231
Throughput (Kbps)	14971	14971
Bottlenect link delay (ms)	169	117

5.3 Performance for Small Buffer Size

Next, the proposed method is validated versus the well-known PI scheme by reducing the buffer size to 400 packets. In Figure 7 (**a**), the PI algorithm generates long buffer flow in 20 s, which means almost all of the arrived packets are dropped; this leads to a synchronization problem. Contrastingly, the ARPI scheme has a stable response by keeping the queue length around the desired target from the beginning of the process. Also, our proposed schema has better performance in regulating queuing delay as shown in Figure 7 (**b**) Even jitter ARPI is smaller than PI. Table 3 figures out that ARPI reduces the average of queue length and link delay up to 33 percent as compared to PI.



(a)



(b)

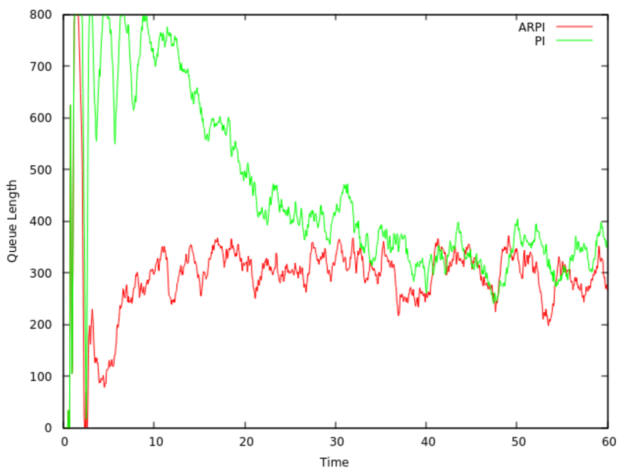
Figure 7. Performance for small buffer size based on **(a)** queue length and **(b)** queuing delay.

Table 3. AQM performance analysis in small buffer size.

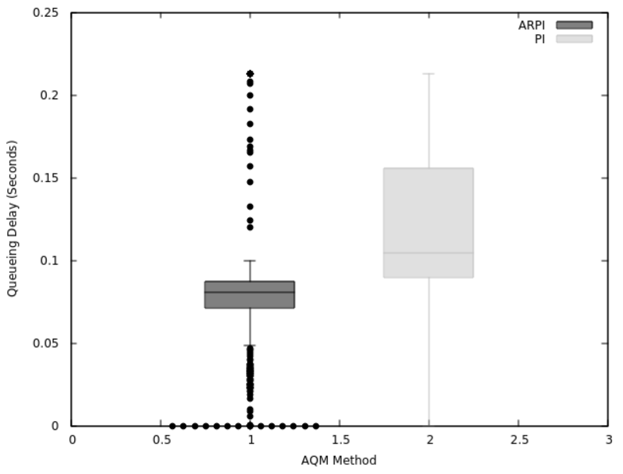
KPI	PI	ARPI
Average of queue (Packets)	299	209
Throughput (Kbps)	14993	14993
Bottlenect link delay (ms)	92	66

5.4 Performance for Changing of Set-Point

Different target values are tested by increasing the reference queue from 200 packets to 300 packets in a heavy traffic scenario. This is shown in Figure 8 (**a**) PI represented by the green line has a bad response with buffer overflow in the initial process, along with a long settling time of 33 s. ARPI represented by the red line is good enough to regulate the queue length around the set-point value and has a quick settling time of 8 s. Figure 8 (**b**) Depicts that ARPI has better performance in keeping shorter queuing delay and smaller jitter as compared with PI. ARPI can regulate the queue length close to the target value 300 and keep the small link delay as shown in Table 4.



(a)



(b)

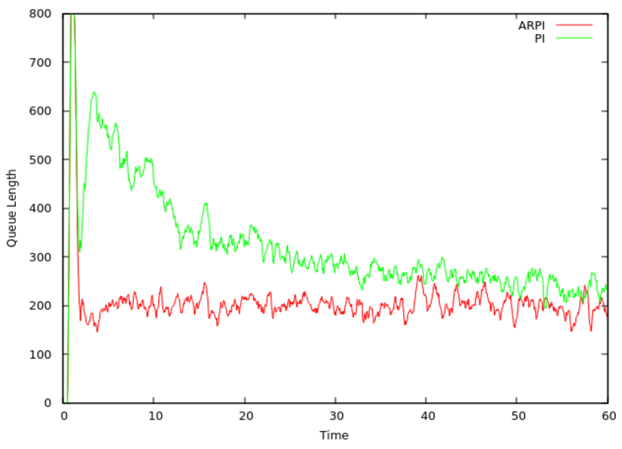
Figure 8. Performance for changing set-point based on **(a)** queue length and **(b)** queuing delay.

Table 4. AQM performance analysis in changing set-point.

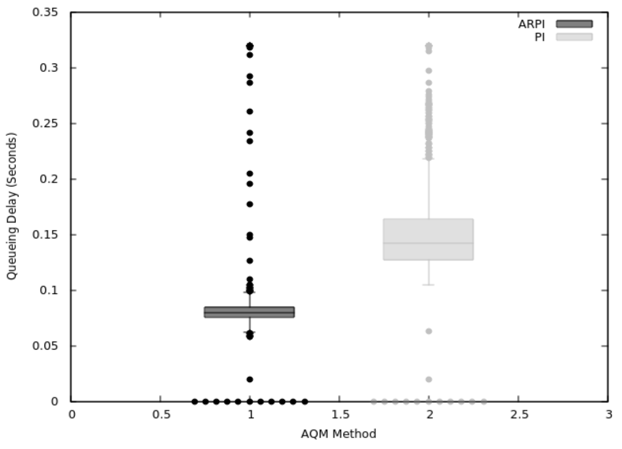
KPI	PI	ARPI
Average of queue (Packets)	454	295
Throughput (Kbps)	14945	14912
Bottlenect link delay (ms)	233	186

5.5 Performance for Different Link Capacities

The next scenario is changing the link capacity from 15 Mbps to 10 Mbps. This was presented in Figure 9 (**a**) PI (green line) experienced a long overshoot and failed to achieve the reference value of 200. Conversely, the proposed solution, i.e., ARPI (red line) has an excellent response, keeping the queue length to the desired reference value and offering a fast settling time of 3 s. ARPI queuing delay is kept small as illustrated in Figure 9 (**b**). Characteristics of PI and ARPI mechanisms are shown in Table 5 ARPI is more powerful than PI for average queue length and link delay, which is in 206 packets and 95 seconds, respectively.



(a)



(b)

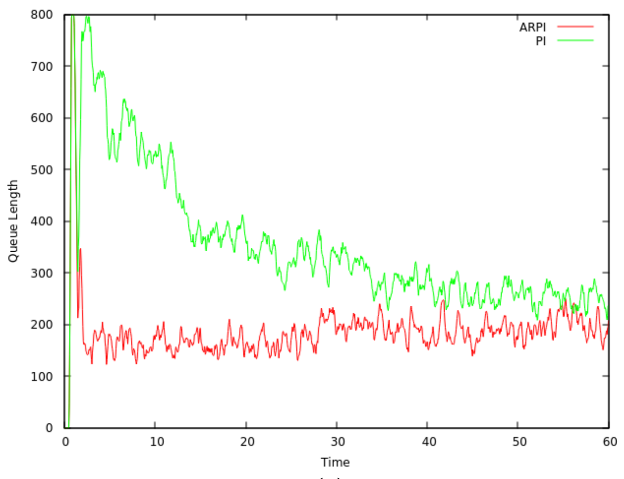
Figure 9. Performance for different link capacities based on **(a)** queue length and **(b)** Queuing delay.

Table 5. AQM performance analysis at different link capacities.

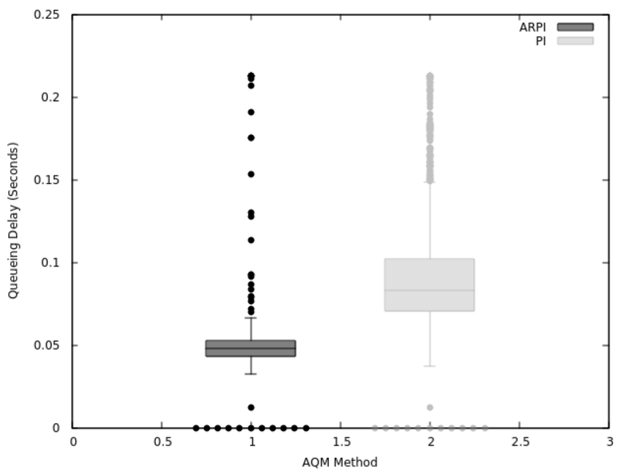
KPI	PI	ARPI
Average of queue (Packets)	321	206
Throughput (Kbps)	9995	9995
Bottlenect link delay (ms)	145	95

5.6 Mix Traffic TCP and UDP

The final examination is mixing 100 UDP flows to the first scenario conditions, these UDP traffics follow an exponential ON/OFF traffic model with the idle and burst times have means of 0.5 second and 1 second respectively. The packet size is set at 500 bytes with the sending rate during on-time being 64 Kbps. Figure 10 (**a**) presents the queue lengths for both AQMs. We can see again that ARPI reacts and converges to the target queue length faster than PI that can not regulate it under this scenario. Queuing delay is depicted in Figure 10 (**b**) that ARPI has under 50 ms and small jitter comparing with PI. Table 6 presents that the average of queue length ARPI is 189, which is closest to the target of 200, and the bottleneck link delay of ARPI is lower than PI.



(a)



(b)

Figure 10. Performance for mix TCP and UDP based on (**a**) queue length and (**b**) queuing delay

Table 6. AQM performance analysis at mix TCP and UDP.

KPI	PI	ARPI
Average of queue (Packets)	354	189
Throughput (Kbps)	14993	14993
Bottlenect link delay (ms)	106	58

6. Conclusions

This paper presented a new method of adaptive AQM protocol called ARPI. ARPI was analyzed for six network traffic scenarios. In addition to simple and low-cost computation, by adaptively setting the PI set-point weighting (β), ARPI offers a robust response in a dynamic network. In all simulated scenarios, ARPI is superior to PI AQM; i.e., it is robust and stable, has a small queuing delay, and it rapidly regulates queue to the desired values. Moreover, ARPI is adaptive not only to various network conditions but also to different router device specifications, i.e., buffer sizes. The new scheme is also potentially easier to implement in software and hardware.

7. Acknowledgement

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] Y. N. Reddy and P. Srinivas, "A Routing Delay Predication Based on Packet Loss and Explicit Delay Acknowledgement for Congestion Control in MANET," International Journal of Communication Networks and Information Security, vol. 10, pp. 447, 2018.

[2] O. A. Fdili, Y. Fakhri, and D. Aboutajdine, "Impact of queue buffer size awareness on single and multi service real-time routing protocols for WSNs," International Journal of Communication Networks and Information Security, vol. 4, pp. 104, 2012.

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

[4] 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.

[5] W. Feng, D. D. Kandlur, D. Saha, and K. G. Shin, "A self-configuring RED gateway," Conference on Computer Communications. Proceedings Eighteenth Annual Joint Conference of the IEEE Computer and Communications Societies, New York, USA, vol.3, pp. 1320-1328, 1999.

[6] S. Floyd, R. Gummadi, and S. Shenker, "Adaptive RED: An algorithm for increasing the robustness of RED's active queue management," ed: Technical report, ICSI, 2001.

[7] G. Kahe and A. H. Jahangir, "A self-tuning controller for queuing delay regulation in TCP/AQM networks," *Telecommunication systems*, vol. 71, pp. 215-229, 2019.

[8] R. Pan, P. Natarajan, C. Piglione, M. S. Prabhu, V. Subramanian, F. Baker, *et al.*, "PIE: A lightweight control scheme to address the bufferbloat problem," IEEE 14th International Conference on High Performance Switching and Routing (HPSR), Taipei, Taiwan, pp. 148-155, 2013.

[9] Z. Liu, J. Sun, S. Hu, and X. Hu, "An Adaptive AQM Algorithm Based on a Novel Information Compression Model," IEEE Access, vol. 6, pp. 31180-31190, 2018.

[10] W.-h. Dou, M. Liu, H.-y. Zhang, and Y.-x. Zheng, "A Framework for Designing Adaptive AQM Schemes," International Conference on Networking and Mobile Computing, Berlin, Heidelberg, pp. 789-799, 2005.

[11] L. Qing, Q. Zhu, and M. Wang, "Designing adaptive PI algorithm based on single neuron," International Conference on Networking and Mobile Computing, Berlin Heidelberg, pp. 800-807, 2005.

[12] M. Fajri and K. Ramli, "Optimizing PID TCP/AQM using nelder-mead simplex approach," Proceedings of the 3rd International Conference on Communication and Information Processing, Tokyo, Japan, pp. 292-295, 2017.

[13] R. Vilanova and V. M. Alfaro, "Robust 2-DoF PID control for Congestion control of TCP/IP Networks," International Journal of Computers Communications & Control, vol. 5, pp. 968-975, 2010.

[14] P. K. Padhy and R. K. Sundaram, "Analysis and design of improved PI-PD controller for TCP AQM routers," International Conference on Power, Control and Embedded Systems (ICPCES), Allahabad, India, pp. 1-5, 2010.

[15] Y. Fan, F. Ren, and C. Lin, "Design a PID controller for active queue management," Proceedings of the Eighth IEEE International Symposium on Computers and Communication (ISCC), Kemer-Antalya, Turkey, vol.2, pp. 985-990, 2003.

[16] Y. Chait, C. V. Hollot, V. Misra, S. Oldak, D. Towsley, and G. Wei-Bo, "Fixed and adaptive model-based controllers for active queue management," Proceedings of the American Control Conference, Arlington, VA, USA, vol.4, pp. 2981-2986, 2001.

[17] J. H. Novak and S. K. Kasera, "Auto-tuning active queue management," 9th International Conference on Communication Systems and Networks (COMSNETS), Bangalore, India, pp. 136-143, 2017.

[18] Z. Honggang, C. V. Hollot, D. Towsley, and V. Misra, "A self-tuning structure for adaptation in TCP/AQM networks," IEEE Global Telecommunications Conference (GLOBECOM), San Francisco, CA, USA, pp. 3641-3646, vol.7, 2003.
[19] C. Qiang and O. W. W. Yang, “On designing self-tuning controllers for AQM routers supporting TCP flows based on pple placement,” IEEE Journal on Selected Areas in Communications, vol. 22, pp. 1965-1974, 2004.
[20] C. XiaoLin, J. K. Muppala, and Y. Jen-te, "A robust nonlinear PI controller for improving AQM performance," IEEE International Conference on Communications, Paris, France Vol. 4, pp. 2272-2276, 2004.
[21] X. Chang and J. K. Muppala, "A stable queue-based adaptive controller for improving AQM performance," Computer Networks, vol. 50, pp. 2204-2224, 2006.
[22] H. Yang, O. W. W. Yang, and H. Changcheng, "Self-tuning PI TCP flow controller for AQM routers with interval gain and phase margin assignment," IEEE Global Telecommunications Conference (*GLOBECOM)*, Dallas, TX, USA, Vol. 3, pp. 1324-1328, 2004.
[23] Y. Hong and O. w. w. Yang, "Self-tuning tcp traffic controller using gain margin specification," IET Communications, vol. 1, pp. 27-33, 2007.
[24] X. Yue-Dong, Y. Jie, and D. Qing, "Nonlinear PI active queue management based on hyperbolic secant functions," International Conference on Machine Learning and Cybernetics, Guangzhou, China, Vol. 2, pp. 715-720, 2005.
[25] J. Sun, S. Chan, and M. Zukerman, "IAPI: An intelligent adaptive PI active queue management scheme," Computer Communications, vol. 35, pp. 2281-2293, 2012.
[26] H. Huang, G. Xue, Y. Wang, and H. Zhang, "An adaptive active queue management algorithm," International Conference on Consumer Electronics, Communications and Networks (CECNet), Xianning, China, pp. 72-75, 2013.
[27] 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.
[28] C. V. Hollot, V. Misra, D. Towsley, and G. Wei-Bo, "On designing improved controllers for AQM routers supporting TCP flows," Twentieth Annual Joint Conference of the IEEE Computer and Communications Societies, Achorage, AK, USA, pp. 1726-1734 vol.3, 2001.
[29] Y. Li, K. T. Ko, and G. R. Chen, "Transient behaviour of PI-controlled AQM," Electronics Letters, *v*ol. 42, pp. 494-495, 2006.
[30] R. K. Mudi and C. Dey, "Performance improvement of PI controllers through dynamic set-point weighting," ISA Transactions, vol. 50, pp. 220-230, 2011.
[31] R. J. Mantz, "A PI Controller with Dynamic Set-Point Weighting for Nonlinear Processes," Proceedings of IFAC Conference on Advances in PID Control, Brescia, Italia, vol. 45, pp. 512-517, 2012.
[32] C. C. Hang, K. J. Astrom, and W. K. Ho, "Refinements of the Ziegler-Nichols tuning formula," IEE Proceedings D - Control Theory and Applications, vol. 138, pp. 111-118, 1991.
[33] K. Astrom and R. Murray, "Analysis and design of feedback systems," Preprint, 2005.
[34] P. Mitra, C. Dey, and R. K. Mudi, "An Improved Dynamic Set Point Weighted PI Controller for Servo Position Control Application," 2nd International Conference on Computational Intelligence and Networks (CINE), Bhubaneswar, India, pp. 145-149, 2016.
[35] T. Issariyakul and E. Hossain, "Introduction to Network Simulator 2 (NS2)," ed: Springer, pp. 1-18, 2012.
[36] 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," IEEE Global Telecommunications Conference (GLOBECOM), Miami, FL, USA, pp. 1-5, 2010.