Improvement and Comparative Study of Control Theory-based AQM Methods for Networks with Realistic Traffic

N. Bigdeli*, M. Haeri**, and M.R. Pakravan***

Advanced Control System Lab., Electrical Engineering Department,

Sharif University of Technology, Azadi Ave., Tehran, IRAN

*(Tel:+98-21-616-5971; E-mail: bigdeli@mehr.sharif.edu)

**(Tel:+98-21-616-5964; E-mail: haeri@sina.sharif.edu)

***(Tel:+98-21-616-5922; E-mail: pakravan@sharif.edu)

Abstract: In this paper the performance of some control theory-based Active Queue Management (AQM) methods are improved and investigated through simulation. The simulation scenario is considered to be closely realistic, in which the traffic is mainly due to http sessions. The results show the best performance of PID controller with respect to the others in good queue regulation as well as high link utilization and low delay metrics.

Keywords: Congestion Control, AQM, Control theory-based AQM, Realistic Traffic.

1. INTRODUCTION

By introduction of QoS requirements in recent years, control theory approaches have been employed widely for improving computer network management especially for resource sharing.

Internet congestion occurs when the aggregate demand for a resource (e.g., a link bandwidth) exceeds the available capacity of the resource and results in effects such as long delays in data delivery, wasted resources due to lost or dropped packets, and even congestion collapse, in which all communication in the entire network ceases. These effects must be limited through congestion control algorithms.

Active Queue Management (AQM) methods refer to the class of congestion control algorithms running on the routers and using a single FIFO queue, shared by all flows. A certain AQM algorithm manages the length of the packet queue by dropping packets when necessary or appropriate [1].

Up to now, more than 50 AQM methods have been proposed, which are based on heuristic, optimization, and control theory approaches. However, these methods are mostly evaluated through simulation scenarios mainly comprised of ftp flows [1-3], while according to [4], the Internet traffic is now mainly due to http web traffics.

In this paper the performance of the most famous control theory-based AQM methods are investigated through simulation. The simulation scenario is considered to be closely realistic. The AQM methods use P, PI, PD, and PID controllers in comparison with Random Early Detection (RED) and Adaptive RED (ARED) methods.

This paper is organized as follows: in Sec.2, the AQM is investigated from the control theory point of view. In Sec. 3, the PID controller and its variants are formulated for the AQM problem. Some metrics for evaluation of AQM methods in Sec.4 and the simulation results are represented in Sec. 5. Finally, the paper is concluded in Sec.6.

2. AQM FROM CONTROL THEORY POINT OF VIEW

Recently, some AQM algorithms have been proposed based on control-theory. In these approaches, the TCP/AQM flow dynamics are modeled and analyzed in terms of feedback control theory and then, AQM algorithms are designed to increase the speed of response (the short-term performance) and improve the stability and robustness (the long-term performance) of TCP/AQM congestion control. These goals can be achieved by regulating the queue length to agree with a desired value [2]. TCP congestion control dynamics with an AQM scheme can be modeled as a feedback control system including in (Fig. 1):

- A *plant,* representing subsystems such as TCP sources, routers, and TCP receivers that send, process, and receive TCP packets, respectively.
- An *AQM controller*, controlling the packet arrival rate to the router queue by generating a packet drop probability (P_d) as *control signal.*
- The *queue length*, Q, at a router as *controlled variable.*
- A *desired queue length,* Q_{ref}, at a router as the reference input.
- A *feedback signal*, the sampled system output (i.e., queue length) used to obtain the error term, $e(t) = Q_{ref} - Q$.

In [1], a nonlinear model for TCP/AQM dynamics is developed using fluid-flow analysis that ignores the TCP time-out mechanism and slow-start phase. Also, a simplified linearized TCP/AQM dynamic model is developed and analyzed, in which forward path transfer function of the plant, at the system equilibrium point is given by:

$$P(s) = P_{tcp}(s)P_{queue}(s)e^{-sR_0} = \left(\frac{\frac{C^2 R_0}{2N^2}}{s + \frac{2N}{R_0^2 C}} \right) \left(\frac{\frac{N}{R_0}}{s + \frac{1}{R_0}} \right) e^{-sR_0} \tag{1}$$

where N is the load factor (number of TCP connections), R_0 is the round trip time, C is the link capacity, and e^{-sR_0} is representative of the time delay. Based on this model, TCP/RED dynamics in terms of feedback control is analyzed and some methods for RED parameter tuning is proposed. Besides, some other linear controllers such as P, PI, PD, and PID controllers have been developed in [1-3]. So, in this paper, the performance of this family of controllers under realistic condition is investigated.

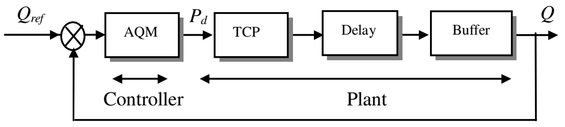

Fig. 1: Feedback control modeling of TCP congestion control with an AQM algorithm.

3. PID FAMILY OF AQM CONTROLLERS

A generic PID control equation is

$$u(t) = K_P e(t) + K_D \dot{e}(t) + K_I \int e(\tau)d\tau \qquad (2)$$

where $u(t)$ is control signal at time $t \geq 0$, K_P is a proportio nal gain, K_I is a integral gain, and K_D is a derivative time and $e(t)$ is the error signal. Other controllers such as P, PI, and PD can be derived from Eq. 2 setting unnecessary coefficient(s) zero and calculating the controller parameters using such methods as described in [1-3].

In almost all the designed controllers, the error signal is defined as the difference between desired and instantaneous queue size [1-2]. However, in the presence of large number of short-lived flows, it leads to a highly fluctuating behavior. So, here the error signal is considered to be:

$$e(t) = Q_{ref} - Q_{avg}(t) \qquad (3)$$

where $Q_{avg}(t)$ is exponentially weighted moving average queue length, that is,

$$Q_{avg}(t) = (1-\omega_Q)Q_{avg}(t-1) + \omega_Q Q(t) \qquad (4)$$

where, $0 \leq \omega_Q \leq 1$ is the averaging parameter (in this paper, t is used for both discrete and continuous time). This is the same approach employed in RED variants.

On the other hand, the routers should be able to store data equal to bandwidth-delay product. Therefore as mentioned in [4], it can be assumed that buffers grow linearly in transmission rate. Since, queue length fluctuations grow linearly with the bandwidth-delay product, thy also grow linearly with the buffer size. Therefore, we use the normalized error signal (*e(t)/Buffer size*) instead of the signal error signal.

PID controller and its variants (P, PI, and PD) can be implemented at a router by finding an equivalent discrete controller from a continuous model by *emulation*. Particularly once the sampling frequency ($f_s = 1/T_s$) is decided analytically or empirically, the digitized controller equation is obtained using *Tustin's method* for integration and the *backward* rectangle method for differentiation [5]. The packet drop pr obability at time t can be written as:

$$\begin{aligned} p(t) &= p(t-1) + \delta p(t) \\ &= p(t-1) + a_1\frac{e(t)}{B} - b_1\frac{e(t-1)}{B} + c_1\frac{e(t-2)}{B} \end{aligned} \qquad (5)$$

where, $a_1 = K_p + \frac{K_D}{T_s} + \frac{T_s K_I}{2}$, $b_1 = K_p \frac{2K_D}{T_s} - \frac{T_s K_I}{2}$, and $c_1 = \frac{K_D}{T_s}$. To maintain $p(t)$ between 0 and 1, if the calculated value becomes negative or greater than unity, it will be set to 0 or one, respectively. In addition, in order for maintaining high link utilization, an additional parameter Q_{thresh} is also introduced. That is, if $Q \leq Q_{thresh}$, no packet will be dropped.

So, the computations of $p(t)$ at time t can be summarized as follows:

1) Sample average queue length;
2) Compute current error signal by (3);
3) Compute current drop probability by (5) and modify if needed;
4) Use $p(t)$ if queue length is not larger than Q_{thresh};
5) Store $p(t)$ and $e(t)$ to be used at time.

4. EVALUATION METRICS OF AQM

To evaluate the AQM methods, several metrics are introduced in [6] that are used in this paper for comparison of different AQM methods, as well. In this Sec. we look over these metrics briefly. Beforehand, some parameters must be defined. Let T be the simulation time, C_N be the network capacity, and F be the number of active flows. Then, for each flow indexed by i, and $i \in [1, F]$, the following variables can be defined:

S_i: the total size of the data received.
S_0': the total size of the data sent.
N_i: the number of packets received.
μ_i: the average delay, weighted by packet size, i.e.,

$$\mu_i = \frac{\sum_{j=i}^{N_i} s_{i,j} d_{i,j}}{S_i} \qquad (6)$$

where $s_{i,j}$ and $d_{i,j}$ are respectively the size and delay related to the packet j of flow i.
v_i: the weighted variance of the delay, i.e.

$$v_i = \frac{N_i}{N_i - 1} \times \frac{\sum_{j=i}^{N_i} s_{i,j}(d_{i,j} - \mu_i)^2}{S_i} \qquad (7)$$

Based on this, the evaluation metrics are defined as:

- ***Utilization Metric:*** This is defined as the percentage of total network capacity utilized during a simulation run. For a given traffic mix and topology, the network capacity is defined as the maximum total flow in the network. To compute the final utilization metric, the sum of the aggregate good put of all flows is divided by the computed network capacity. That is,

$$Utilization\, metric = \frac{\sum_{i=1}^{F} S_i}{C_N T} \quad (8)$$

- ***Delay Metric:*** This is defined as the average end-to-end delay experienced by the packets. The average over all the packets in an individual flow is first computed and from these the average over all flows is calculated.

$$Delay\, metric = \frac{\sum_{i=1}^{F} S_i \mu_i}{\sum_{i=1}^{F} S_i} \quad (9)$$

- ***Jitter Metric:*** This is based on the variance of the overall delay of packets. First the coefficient of variation of the delay for each flow is computed and these values are then averaged over all flows.

$$Jitter\, metric = \frac{\sum_{i=1}^{F} \frac{S_i v_i}{\mu_i}}{\sum_{i=1}^{F} S_i} \quad (10)$$

- ***Drop Metric:*** This represents the percentage of packets dropped. It includes packets dropped due to the AQM policy (early drops) as well as those dropped due to buffer overflow. We do not differentiate between these as they have the same impact on the end-user experience.

$$Drop\, metric = \frac{\sum_{i=1}^{F} S_i}{\sum_{i=1}^{F} S'_i} \quad (11)$$

- ***Fairness Metric:*** The fair "share" of each flow is computed statically by reference to the capacity of the network using the fairness metric proposed by Jain et al. [7]. If each flow gets its fair share of the network capacity, the metric value is 1; this value decreases if resources are shared unevenly between the flows.

$$Fairness\ \ metric = \frac{(\sum_{i=1}^{F} S_i)^2}{F \sum_{i=1}^{F} S_i^2} \quad (12)$$

5. SIMULATION STUDY

5.1 Simulation scenario

In this paper, the performance of above-mentioned controllers in networks loaded mainly with http traffics, are investigated. The simulation scenario is a Dumbbell topology including 15 ftp sessions and 30 http sessions each generating 10web flows per second randomly with Petro-II distribution of shape parameter α = 1.2 and scale parameter β = 1000Bytes. Each node runs TCP/SACK to minimize the effect of timeout, not considered in the model of system. All sources and destinations use tail drop queue management with sufficient buffer capacity. Link delays are not equal, and ranging from 10msec to 110 msec with an average of 55msec. The bottleneck delay is set to 10msec with a link capacity of 15Mbps. So, the transport delay ranges from 60 to 460 msec. The buffer capacity is 150 packets of 1000byte, which is the average length of packets. For the designed traffic to be more realistic, low reverse path traffic is also supplied. For this network, the plant transfer function will be:

$$P(s) = \frac{\frac{C^2}{2N}}{\left(s + \frac{2N}{R_0^2 C}\right)\left(s + \frac{1}{R_0}\right)} = \frac{117187.5}{(s+.28)(s+4.17)} \quad (13)$$

With no controller, the closed loop behavior of the plant will be as:

$$w_n = 342.33, \qquad \xi = 6.495e-3,$$
$$t_r = \frac{\pi - \theta}{w_d} = \frac{\pi - \cos^{-1}\xi}{w_n\sqrt{1-\xi^2}} = 4.608e-3,\ \text{sec},$$
$$t_s = \frac{4}{\xi w_n} = 1.8\ \text{sec},$$
$$MOS\% = 97.98\%,$$
$$e_{ss} = \frac{1}{1 + \lim_{s\to 0} P(s)} = 9.96e-6$$

So, because of the long settling time relative to the very short rise time with very small damping ratio, the TCP flows show severely oscillating dynamics. Therefore, a well-designed AQM controller should be able to compensate the oscillatory dynamics and give satisfactory control performance.

5.2 Simulation results

The P, PD, PI, and PID controllers are designed for this network and the resulted parameters are summarized in Table. 1. To improve the controller performances, the exponentially weighted moving average (EWMA) queue length with averaging parameter w_Q = 0.05 is fed back. The controllers are implemented in ns-2 and results are seen in Figs. 2-4. The above mentioned metrics are also calculated and tabled in Table. 2. Adaptive RED and RED are also included for comparison. In Fig. 2 and 3, the steady state behavior of controllers in queue length regulation is investigated. As seen, the PID controller performance is the best in queue regulating, while P controller cannot do regulation at all. PD performs better than P, but suffers from large steady state error and multimode queue distribution. PI performs still better, but with more variations around the set point. Aggregate loss rate and congested link utilization are depicted in Fig. 4, which shows increasing loss rate in nearly full link utilization in ARED and RED methods as well as in PI, PD, and PID controllers. So, the PID controller does the queue regulation much better than others without increasing loss or decreasing link utilization considerably.

6. COCLUSION

The performance of several modified control theory -based AQM methods are investigated comparatively. The simulation scenario is considered to be more realistic than common, which consists mainly of http sessions. That is, the simulation results show the best performance of PID controller with respect to the others in both queue regulation and performance metrics.

REFERENCES

[1] V. Misra, Wei-Bo Gong, and D. Towsley, *"Fluid-based Analysis of a Network of AQM Routers Supporting TCP Flows with an Application to RED,"* in Proc. ACM/SIGCOMM, 2000.

[2] C. V. Hollot, V. Misra, D. Towsley, and W. Gong. *"On Designing Improved Controllers for AQM Routers Supporting TCP Flows,"* in Proc. of INFOCOM′ 2001, pp. 1726-1734, April 2001.

[3] S. Ryu, C. Rump, and C. Qiao, *"A Predictive and Robust Active Queue Management for Internet Congestion Control,"* in Proc. of the Eighth IEEE International Symposium on Computers and Communication (ISCC′ 03), pp. 1530-1346, 2003.

[4] S. Floyd, E. Kohler, *"Internet Research Needs Better Models, "* ACM Computer Communication Review, vol. 33, no. 3, pp. 261-280, June 2003.

[5] G. Franklin, J. Powell, and M. Workman, *Digital Control of Dynamic Systems*, Addison-Wesley, third edition, 1998.

[6] A. Bitorika, M., and M. Huggard, *“A Framework For Evaluating Active Queue Management Schemes,”* IEEE/ACM Int. Symp. Modeling, Analysis and Simulation of Computer Telecommu. Sys**,** Orlando, Florida, Oct. 12 - 15, 2003.

[7] R. Jain, D. Chiu, and W. Hawe, *"A quantitative measure of fairness and discrimination for resource allocation in shared computer systems,"* Digital Equipment Corporation, Tech. Rep. DEC-TR-301, Sept. 1984.

Table. 1: Designed controller parameters.

RED	w_Q = 1.49e-6, max_p = 4.963e-3 , min_{th} = 75, and max_{th} = 150.
PID	w_Q = 0.05, K_p = 1.44924e-3, K_D = 1.07e-4, K_I = 3.974e-4, and T_s = 0.02sec.
P	w_Q = 0.05, K_p = 4.0832e-5, and T_s = 0.075sec.
PI	w_Q = 0.05, K_p= 1.741e-4,K_I = 4.874e-5, and T_s = 0.0068sec.
PD	w_Q = 0.05, K_p = 2e-4, K_D = 0.08, and T_s = 0.01sec.

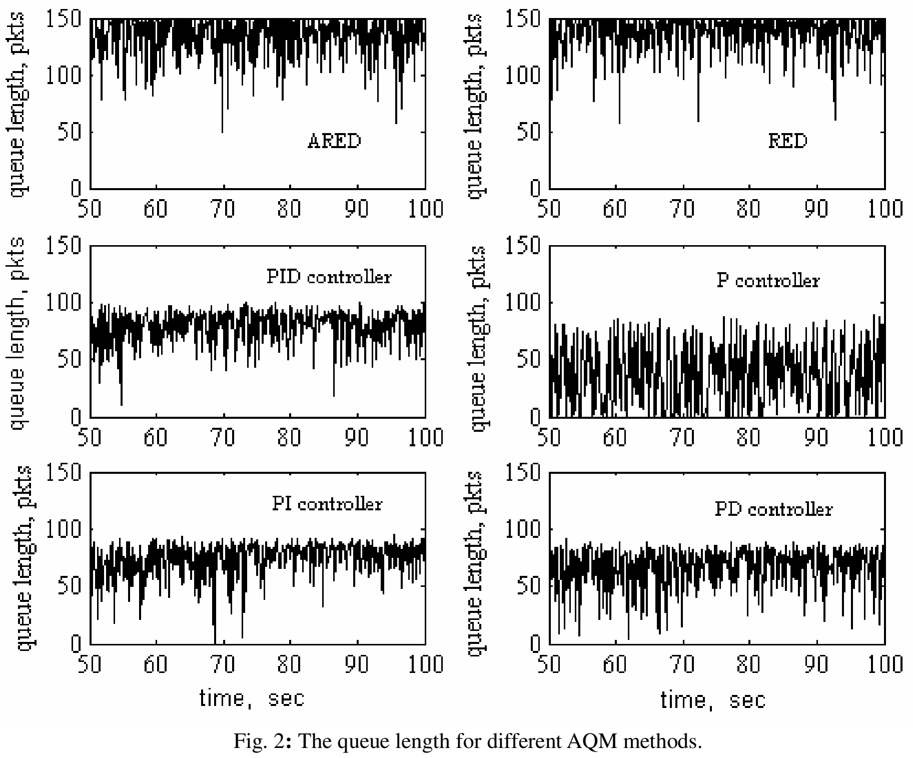

Fig. 2**:** The queue length for different AQM methods.

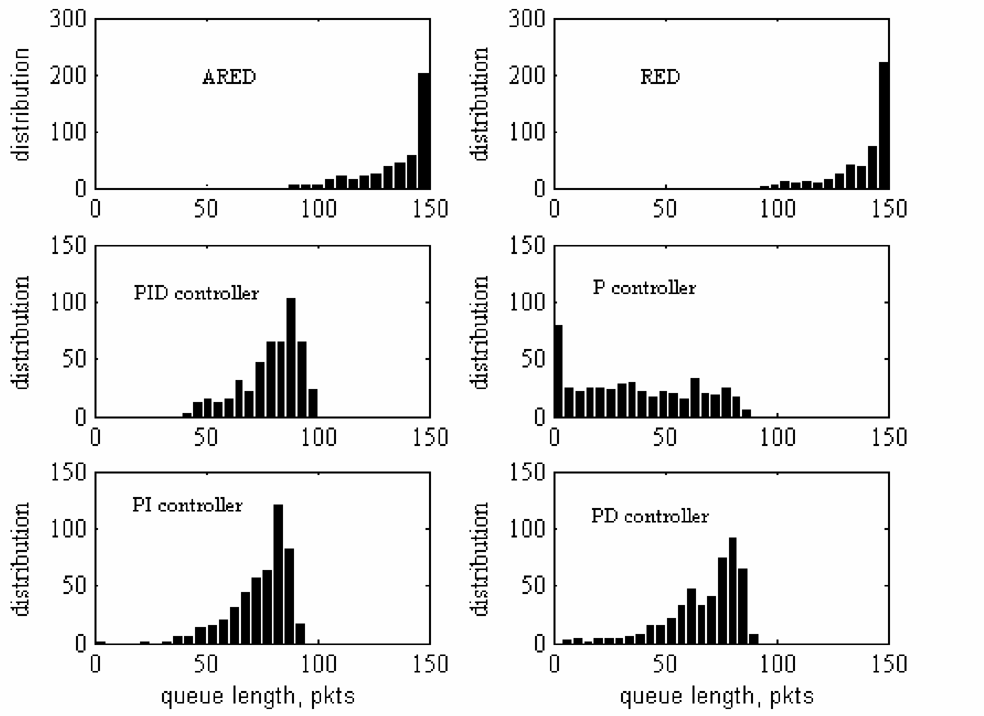

Fig. 3: The queue length distribution for different AQM methods.

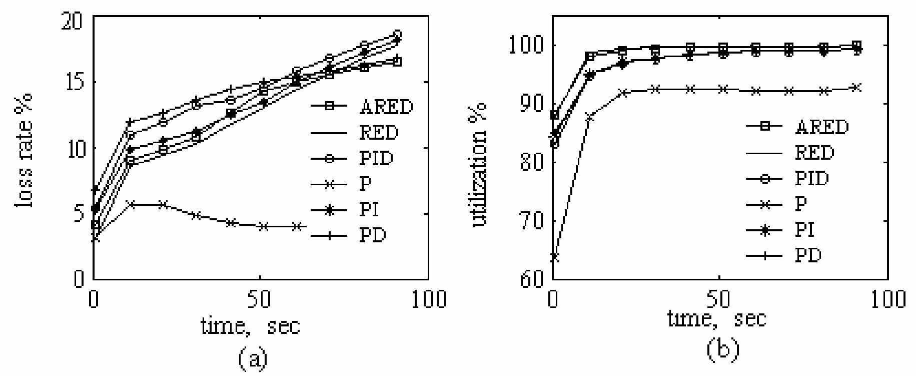

Fig. 4: a) The aggregate loss rate and b) the link utilization for different AQM methods.

Table.2: Different metrics for evaluating AQM methods.

	ARED	RED	PID	P	PI	PD
Queue avg.	130.4691	134.4351	89.4032	36.3014	85.0160	71.5549
Queue stan dev.	21.6902	18.1643	11.2661	26.9781	12.1290	14.4211
Utility metric%	75.39	75.78	75.4	67.73	73.69	67.7
Drop metric%	86.92	86.22	85.34	95.91	82.83	82.66
Delay metric	0.2375	0.2531	0.1711	0.1930	0.2231	0.2203
Jitter metric	0.0036	0.0091	0.0053	0.0033	0.0070	0.0138
Fairness metric	0.7367	0.8256	0.7930	0.6538	0.8443	0.9054