Publications

Available Online

J. Sci. Res. **2** (2), 273-284 (2010)

JOURNAL OF
SCIENTIFIC RESEARCH

www.banglajol.info/index.php/JSR

A Fuzzy Logic Based Network Congestion Control Using Active Queue Management Techniques

I. K. Tabash[1], M. A. A. Mamun[2], and A. Negi[1]

[1]Department of Computer and Information Sciences, University of Hyderabad, Gachibowli, Hyderabad, 500046, India

[2]Department of Computer Science & Information Technology, Islamic University of Technology (IUT), Board Bazar, Gazipur-1704, Bangladesh

Received 9 July 2009, accepted in revised form 18 April 2010

Abstract

Conventional IP routers are passive devices that accept packets and perform the routing function on any input. Usually the tail-drop (TD) strategy is used where the input which exceeds the buffer capacity are simply dropped. In active queue management (AQM) methods routers manage their buffers by dropping packets selectively. We study one of the AQM methods called as random exponential marking (REM). We propose an intelligent approach to AQM based on fuzzy logic controller (FLC) to drop packets dynamically, keep the buffer size around desired level and also prevent buffer overflow. Our proposed approach is based on REM algorithm, which drops the packets by drop probability function. In our proposal we replace the drop probability function by a FLC to drop the packets, stabilize the buffer around the desired size and reduce delay. Simulation results show a better regulation of the buffer.

Keywords: Random exponential marking; Active queue management; Fuzzy logic controller; Pro-active queue management.

DOI: 10.3329/jsr.v2i2.2786 J. Sci. Res. **2** (2), 273-284 (2010)

1. Introduction

Active queue management (AQM) techniques [1-3] provide mechanisms to control the queue length (i.e. the number of packets in a router's buffer) by actively discarding arriving packets before the router's buffer becomes full. For instance, one of typical AQM mechanisms [4-7] called random early detection (RED) [8, 9] randomly drops an arriving packet with a probability proportional to its average queue length. However, it is known that RED's effectiveness is heavily dependent on the setting of its control parameters. Moreover, another problem that the average queue length of RED in steady state depends on the number of active TCP connections [10, 11]. Hence, in the literature, several

[2] *Corresponding author*: raseliut@yahoo.com, ibrahim3100@yahoo.com

variants of RED-H ERED (exponential RED) [12], RED (gentle RED) [13], DRED (dynamic RED) [14] and SRED (Stabilized RED) [15] have been proposed for solving the problems of RED.

To address these problems, it is necessary for an AQM algorithm to have a more efficient congestion indicator and control function. To avoid or to control congestion proactively before it becomes a problem, both the congestion indicator and the control function of an AQM algorithm should be adaptive to changes in the traffic environment such as the amount of traffic, the fluctuation of traffic load, and the nature of traffic.

Two adaptive and proactive AQM algorithms [16], *proportional-integral-derivative* (PID)-*controller* and the *pro-active queue management* (*PAQM*) [18], have been proposed to detect and control the incipient as well as the current congestion effectively and proactively. The goal of these algorithms is to control congestion proactively, to make the queue length agree with a desired level and to give smooth and low packet loss rates to each flow so as to remove the bias against bursty sources.

In this paper we use fuzzy logic controller (FLC) to design active queue management of IP networks, because fuzzy control does have better suitability to dynamic network environment without need for a precise model. There have been some fuzzy-based control algorithms for buffer management under ATM networks. The fuzzy control in our proposal is a static algorithm with static fuzzy rule and static parameters in membership functions.

This paper is organized as follows: in section 2 we discuss the random exponential marking (REM) algorithm, in section 3 we present an overview of fuzzy logic controller, in section 4 the details of our algorithm (FREM), in section 5 simulation study and analysis. Finally some conclusions and future works are given out in section 6.

2. REM Algorithm

REM attempts to obtain high utilization, low loss, and low queuing delay. The key insight is that REM [9, 18] uses a congestion measure called *price* that is decoupled form performance measures such as packet loss or queue length [14]. REM periodically samples the router queue and updates the congestion measure to reflect any mismatch between packet arrival and departure rates at the link (i.e., the difference between the demand and the service rate), and any queue size mismatch (i.e., the difference between the actual queue length and its target value). Given the k^{th} samples of the router queue and the mismatch between packet arrival and departure rate, the congestion measure $p(kT)$ at time kT is computed by [18]:

$$p(kT) = \max(0, p((k-1)T) + \gamma(\alpha(q(kT) - q_{ref}) + x(kT) - c)) \tag{1}$$

where c is the link capacity (in packet departures per unit time), $q(kT)$ is the queue length, and $x(kT)$ is the packet arrival rate, all determined at time *kT*. The mark or drop probability in REM is defined as [18]:

$$prob(kT) = 1 - \phi^{-pkT} \tag{2}$$

where $\phi > 1$ is a constant.

The parameters for REM are summarized in Table 1.

Table 1. REM parameters.

Parameters	Description
q_{ref}	Target queue reference for the instantaneous queue.
α and γ	Constants for computing the "congestion price"
ϕ	Constant for computing the mark or drop probability
T	Sampling interval for the instantaneous queue

When there is a positive rate mismatch, i.e., the packet arrival rate is higher than the link capacity, over a time interval, more packets are backlogged at the router and cause the router queue to increase. Conversely, a negative rate mismatch over a time interval will drain the queue. Thus, REM can be detecting the rate mismatch by comparing the instantaneous queue length with its previous sampled value.

3. Overview of Fuzzy Logic Controller (FLC)

Fuzzy logic controllers, like expert systems, can be used to model human experiences and human decision making behaviors [17]. In FLC the input-output relationship is expressed by using a set of linguistic rules or relational expressions. A FLC basically consists of four important parts including a fuzzifier, a defuzzifier, an inference engine and a rule base. As in many fuzzy control applications, the input data are usually crisp, so a fuzzification is necessary to convert the input crisp data into a suitable set of linguistic value that is needed in inference engine. Singleton fuzzifier is the general fuzzification method used to map the crisp input to a singleton fuzzy set. In the rule base of a FLC, a set of fuzzy control rules, which characterize the dynamic behavior of system, are defined. The inference engine is used to form inferences and draw conclusions from the fuzzy control rules. Fig. 1 shows the fuzzy logic controller architecture. The output of inference engine

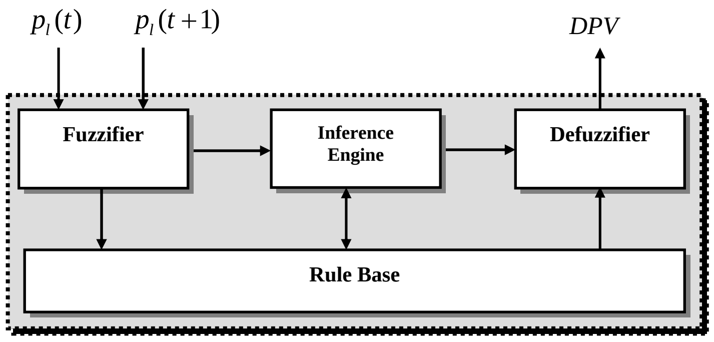

Fig. 1. Fuzzy logic controller architecture.

is sent to defuzzification unit. Defuzzification is a mapping from a space of fuzzy control actions into a space of crisp control actions. Suppose the FLC has *n* input variables including $x_1, x_2, ..., x_n$. The input vector X is defined as $X = (x_1, x_2, ..., x_n)^T$. Furthermore, suppose the rule base consists of *K* rules with the following general form:

$$Rule-1: \text{if } X = (A_{11}, A_{12}, ..., A_{1n}), \text{ then } y \text{ is } B_1$$
$$Rule-2: \text{if } X = (A_{11}, A_{12}, ..., A_{1n}), \text{ then } y \text{ is } B_2$$
$$Rule-k: \text{if } X = (A_{11}, A_{12}, ..., A_{1n}), \text{ then } y \text{ is } B_k$$

where in the j^{th} rule A_{ij} and B_j are fuzzy sets of linguistic variables $x_1, x_2, ..., x_n$ and *y*, respectively. The output *f(X)* of this fuzzy controller with singleton fuzzifier, *product* inference engine and center-average defuzzifier can be calculated as [17]:

$$f(X) = \sum_{j=1}^{K} y_0^j \prod_{i=1}^{n} \mu_i^j(x_i) \Big/ \sum_{j=1}^{K} \prod_{i=1}^{n} \mu_i^j(x_i) \tag{3}$$

where y_0^j is the center value of the output fuzzy set b_j, in the j^{th} rule. $\mu(x)$ is the membership function for fuzzy sets. In our proposed model we use two input variables to fuzzy controller which present the congestion measures for the current and previous time intervals and the output will be the drop probability value.

4. Fuzzy Random Exponential Marking (FREM)

Our algorithm FREM based on congestion measure's variable for REM algorithm, this variable call *Price* as we explained it in the section II, we will use this variable as input variable to fuzzy controller during two time intervals, current and previous sampled value. In our algorithm we have two input variables, once for the current time sample $p_l(t+1)$ and once for the previous time sample $p_l(t)$, based on these two inputs values the controller will decide the packet drop probability value (*DPV*) which presents the output for fuzzy controller. In order to calculate the packet drop probability value according to fuzzy logic controller (FLC) we need to define fuzzy sets, membership functions, and fuzzy control rules. In the following we will first present the full details of FLC components. The control symbol used in FREM is only *price* variable, which needs to be fuzzified firstly and so we should define the term set for it. In this paper we use $p_l(t+1)$ to represent the level of congestion in the current time sample and $p_l(t)$ to represent the level of congestion in the previous time sample. The term set of input linguistic variable $p_l(t+1)$ can be defined as:

$p_l(t+1)$ = {NLS, NVL, NL, NS, ZO, PS, PL, PVL, PLS} (4)

Similarly for $p_l(t)$:

$p_l(t)$ = {NLS, NVL, NL, NS, ZO, PS, PL, PVL, PLS} (5)

The following equations used to calculate *price* periodically for the current time sample [8]:

$$p_l(t+1) = [p_l(t) + \gamma(\alpha_l(b_l(t) - b_l^*) + x_l(t) - c_l(t)]^+ \tag{6}$$

Similarly for the previous time sample:

$$p_l(t) = [p_l(t-1) + \gamma(\alpha_l(b_l(t-1) - b_l^*) + x_l(t-1) - c_l(t-1)]^+ \tag{7}$$

where $\gamma > 0$ and $\alpha_l > 0$ are small constants, and $[z]^+ = \max\{z,0\}$. Here, $b_l(t)$ is the aggregate buffer occupancy at queue l in period t and $b_l^* \geq 0$ is target queue length, $x_l(t)$ is the aggregate input rate to queue l in period t, and $c_l(t)$ is the available bandwidth to queue l in period t. The difference $x_l(t) - c_l(t)$ measures rate mismatch and the difference $b_l(t) - b_l^*$ measures queue mismatch. The term set of output linguistic variable drop probability value (*DPV)* can be defined as:

$$DPV = \{0, 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 1\}$$

The membership functions for the fuzzy inputs are given below figures 2 and 3. (Note that a triangular distribution is assumed for ease of computation and can be changed depending on the choice of the designer).

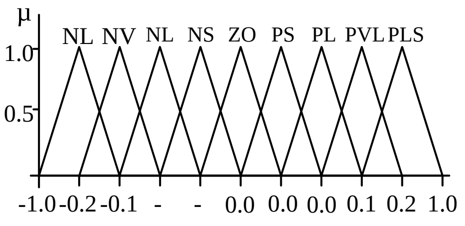

Fig. 2. The membership functions for $p_l(t+1)$.

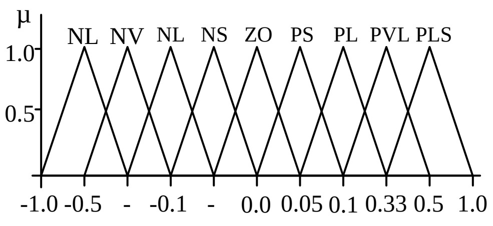

Fig. 3. The membership functions for $p_l(t)$.

The Rule Base component contains a set of *If-Then* rules that is the basis for the decision making process of the inference mechanism. This set of rules is designer-dependent and can be modified to provide better performance. For example,

if pr_{t+1} is *NLS* and pr_t is *PL* then *DPV* is 0.4

where "*DPV*" generated by the fuzzy controller due to the combination of inputs. The following rule bases Table 2 summarizes all of the outputs generated by the fuzzy controller.

The inference engine is a two-step process that outputs the certainty that the input to the AQM controller should take on various values. The first step is to determine which sets of rules apply to the most current situation. This process involves determining the certainty that each rule applies and is highly variable depending on the choice of the membership functions and the number of inputs to the fuzzy controller. The second step of the Inference Mechanism is to determine which conclusion should be reached when the rules that are "ON" are applied to decide what should be the drop probability value to the controller should be.

Table 2. Summary of all the outputs generated by the fuzzy controller.

Drop Pro. value DPV		pr_t								
		NLS	NVL	NL	NS	Zo	PS	PL	PVL	PLS
pr_{t+1}	NLS	0.0	0.0	0.0	0.0	0.0	0.4	0.4	0.5	0.6
	NVL	0.0	0.0	0.0	0.0	0.1	0.4	0.5	0.6	0.7
	NL	0.0	0.0	0.0	0.0	0.2	0.5	0.6	0.7	1.0
	NS	0.0	0.0	0.0	0.0	0.3	0.5	0.6	0.7	1.0
	Zo	0.0	0.0	0.0	0.0	0.4	0.5	0.6	0.7	1.0
	PS	0.0	0.0	0.0	0.0	0.5	0.6	0.7	1.0	1.0
	PL	0.0	0.0	0.0	0.0	0.5	0.6	0.7	1.0	1.0
	PVL	0.0	0.0	0.0	0.0	0.6	0.7	0.7	1.0	1.0
	PLS	0.0	0.0	0.0	0.0	0.6	0.7	1.0	1.0	1.0

In defuzzification component we use the center of average to calculate the output value for fuzzy controller which presents the packet drop probability value (*DPV*), by using Eq. (3).

5. Simulation Study and Analysis

In this section, we compare the control performance of our fuzzy AQM algorithm (FREM) with existing AQM proposals that support fixing of a desired queue length value such as PID-controller and random exponential marking (REM) via simulation study over a wide range of traffic environments using NS-2 [20].

5.1. *Simulation setup*

We use a simple bottleneck network topology as shown in Fig. 4. The network consists of two routers, R1 and R2, with *n* TCP/Reno sources and *n* logically connected destinations. All TCP connections are connected to routers, R1 and R2, with link speeds 100 Mbps. The propagation delay is generated randomly. The bottleneck link between R1 and R2 is assumed to have a link speed of 0.7 Mbps and propagation delay 20 ms, all sources and destinations are assumed to use drop tail queue management with sufficient buffer capacity. The buffer at the bottleneck uses some AQM algorithm for example FREM and has capacity of 200 packets and the desired queue length value is set to 80 packets. Each packet is assumed to have an average size of 512 bytes. We considered only a single type of traffic flow (long-lived FTP flows), since each proposed AQM proposal is only good for a particular traffic condition; not for realistic IP traffic nor heterogeneous traffic environment. We specify simulation time 100 sec.

5.2. *Performance metrics: The queue length*

Control performance of an AQM algorithm can be measured by two measures: the transient performance (i.e., speed of response) and the steady-state error control (i.e., stability). We use the instantaneous queue length as a performance metric for the transient performance. For the steady-state control performance, we use the *quadratic average of control deviation* (QACD) [19] defined as:

$$s_e = \sqrt{\frac{1}{M+1}\sum_{i=0}^{M} e_i^2} = \sqrt{\frac{1}{M+1}\sum_{i=0}^{M} (Q_i - Q_{ref})^2} \tag{8}$$

where Q_{ref} is the desired queue length, Q_i is the i^{th} sampled queue length, $i = 1,\dots,M$, and M is the number of sampling intervals.

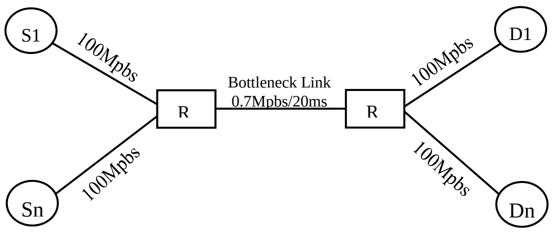

Fig. 4. Network topology.

5.3. *Analysis of control performance*

We examine the sensitivity of control performance of our FREM AQM algorithm, PID-controller and REM AQM algorithms.

5.3.1. *Sensitivity to traffic load*

We examine the control performance in terms of queue length, packet loss, delay and throughput under three different traffic load conditions i.e., number of sources (*n* = 100, 200 and 300).

5.3.2. *The dynamic queue length*

Figs. 5, 6 and 7 show the queue length dynamics of FREM and REM respectively under 100, 200 and 300 sources. FREM shows good control performance independent of traffic load levels in terms of queue length dynamics around $Q_{ref} = 80$ packets. Traditional REM algorithm fails to maintain the queue length around Q_{ref} .

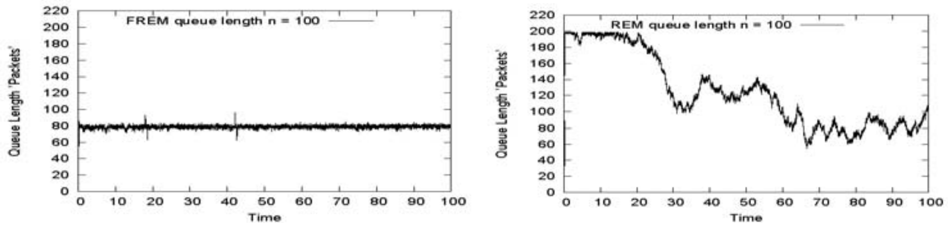

Fig. 5. The queue length of FREM and REM for 100 sources.

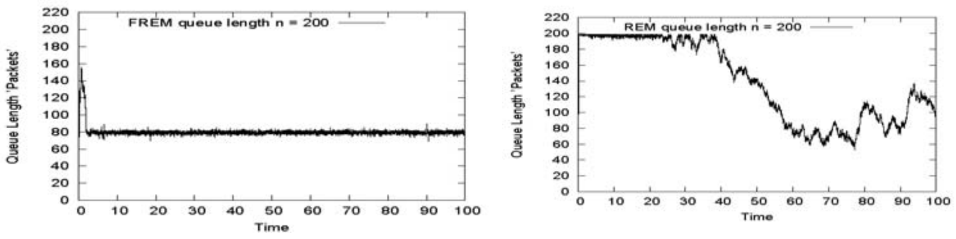

Fig. 6. The queue length of FREM and REM for 200 sources.

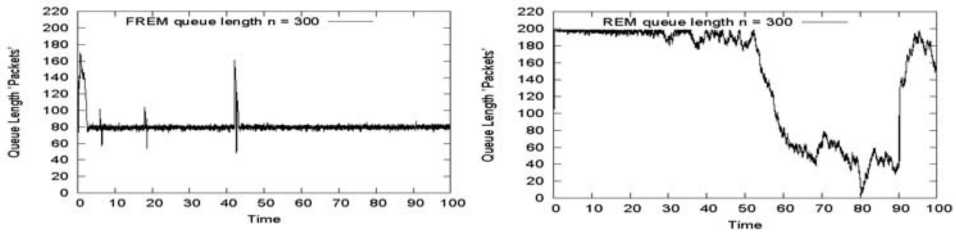

Fig. 7. The queue length of FREM and REM for 300 sources.

Also Figs. 8, 9 and 10 show the queue length dynamics of FREM and PID-controller respectively under 100, 200 and 300 flows. FREM shows good control performance independent of traffic load levels in terms of the queue length dynamics around $Q_{ref} = 80$ packets. PID-controller algorithm also shows good control performance to maintain the queue length around Q_{ref} .

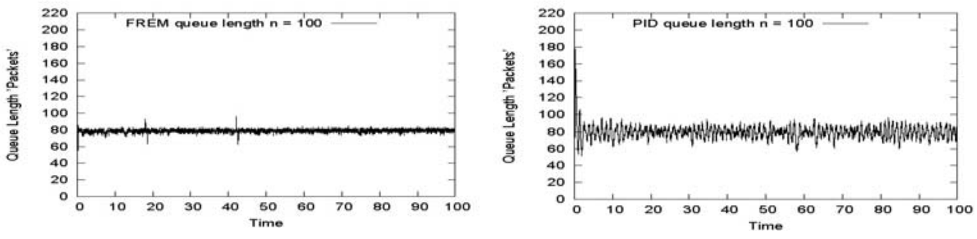

Fig. 8. The queue length of FREM and PID for 100 sources.

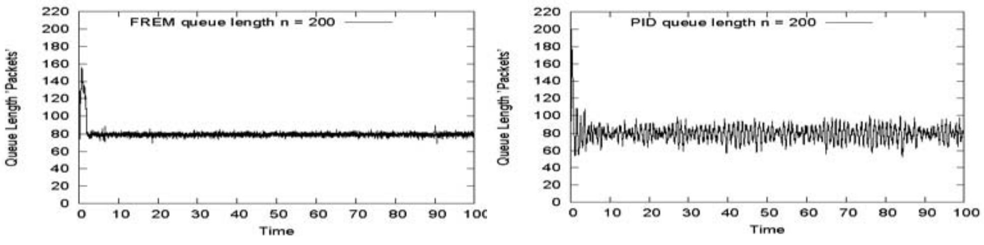

Fig. 9. The queue length of FREM and PID for 200 sources.

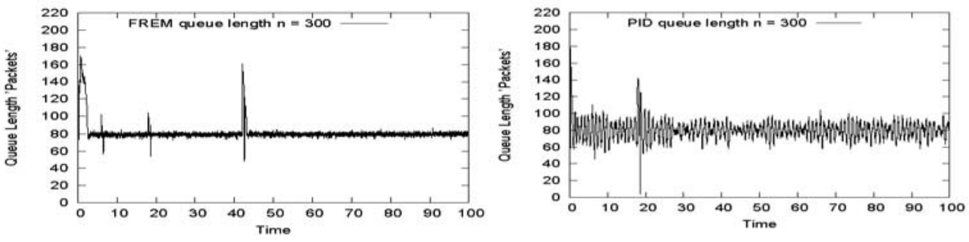

Fig. 10. The queue length of FREM and PID for 300 sources.

5.3.3. *The steady-state control performance*

The steady-state control performance of FREM, REM and PID-controller can be evaluated in terms of QACD at three different traffic load levels: 100, 200, 300 flows. Table 3 shows the mean of QACD of the AQM algorithm respectively under three different traffic load levels for FREM, REM and PID-controller. FREM and PID controllers show robust steady-state control performance independent of traffic load levels in terms of relatively small mean for the QACD. However, the steady-state control performance of REM algorithm is highly dependent on traffic load level. The steady-state control error of REM algorithm is much higher than those of FREM algorithm and PID controller for all traffic load levels. Table 4 shows summary of average rates of delay of AQM algorithms under several traffic load levels. Our algorithm reduces the delay compared to traditional REM algorithm.

Table 3. Summary of mean of QACD of AQM algorithms (packets).

No. of sources	FREM scheme	PID-controller	REM algorithm
100	3.45	8.42	62.80
200	7.21	10.45	77.75
300	10.90	11.45	90.38

Table 4. Summary of average delay rates of AQM algorithms in “ms”.

No. of sources	FREM scheme	PID-controller	REM algorithm
100	300.58	302.70	445.99
200	288.14	300.19	488.31
300	305.35	299.49	483.30

Table 5 shows summary of average rates of packet loss and Table 6 shows summary of average of throughput rates. It can be seen from the tables that throughput rates are comparable for all these schemes. REM performs best as far as packet loss is concerned followed by PID controller. FREM has more packet loss than either of the other schemes. On the other hand, FREM has the least average delay for packet delivery since the queue length is maintained constant, thus reducing queuing delays, PID controller is almost as good and under heavy load conditions, it is slightly better than FREM. REM, on the other hand, performs much worse than the other two schemes. In summary, FREM does as good as the other schemes for throughput, has less delay and has slightly higher packet loss than other schemes.

Table 5. Summary of average packet loss rates of AQM algorithms “%”.

No. of sources	FREM scheme	PID-controller	REM algorithm
100	15.77	12.39	9.67
200	23.56	17.30	14.43
300	24.23	19.78	19.26

Table 6. Summary of average throughput rates of AQM algorithms “kpbs”.

No. of sources	FREM scheme	PID-controller	REM algorithm
100	3.74	3.75	3.75
200	1.96	1.87	1.87
300	1.25	1.25	1.27

6. Conclusions

We have presented a FUZZY AQM scheme to provide congestion control in TCP best effort networks. Our proposal is based on random exponential marking (REM) AQM algorithm. We replace the drop probability function for (REM) algorithm and use a fuzzy controller to drop the packets dynamically based on two inputs, pr_{t+1} which represents a congestion measure the rate mismatch (i.e., difference between input rate and link capacity) and queue mismatch (i.e., difference between queue length and target) for current time interval and pr_t for previous time interval. Our scheme stabilizes the queue length to the desired value and reduces the delay compared to traditional REM algorithm. The disadvantage of our scheme is that there is more packet loss. Our schemes show robust steady-state control performance independent of traffic load levels in terms of relatively small mean for QACD [19]. However, the steady-state control performance of REM algorithm is highly dependent on traffic load level. Our scheme is based on output link capacity to calculate the inputs pr_{t+1} and pr_t. So we need manual re-configuration of fuzzy logic controller based on the output link capacity. To configure or adapt the fuzzy controller dynamically to the network topology and traffic conditions, we need another controller such as PID-controller.

References

1. T. Bonald, et al., "Analytic Evaluation of RED Performance" - IEEE INFOCOM, (2000).
2. C. V. Hollot, V. Misra, D. Towsley, W.-B. Gong, IEEE Transactions on Automatic Control, **47** (6), 945 (June 2002). doi:10.1109/TAC.2002.1008360
3. V. Hollot, Y. Liu, V. Misra, and D. Towsley,"Unresponsive Flows and AQM Performance" - IEEE INFOCOM , **2** ,.85 (April 2003).
4. W. Feng, D. Kandlur, D. Saha, K. Shin, "A Self-Configuring RED Gateway", INFOCOM, (March 1999).
5. W. Feng, D. Kandlur, D. Saha, K. Shin, "Blue: A New Class of Active Queue Management Algorithms" U. Michigan CSE **387** 99 (April 1999).
6. W. Feng, W. Feng, "The Impact of Active Queue Management on Multimedia Congestion Control," IC3N (October 1998).
7. T. B. Reddy, A. Ahammed, R. banu, "Performance Comparison of Active Queue Management Techniques"-IJCSNS, **9** (2) February 2009).
8. S. Floyd and V. Jacobson, "Random early detection gateways for congestion avoidance" - IEEE/ACM Transactions on Networking **1,** 397 (Aug. 1993). doi:10.1109/90.251892
9. W. Feng, D. Kandlur, D. Saha, and K. Shin, "Techniques for Eliminating Packet Loss in Congested TCP/IP Networks" U. Michigan CSE **349** (November 1997).
10. Y. Pan, W. K. Tsai, and T. Suda, "Improving TCP Throughput in AQM Queues with Unresponsive Traffic", University of California at Irvine (August 2007).
11. T. Eguchi, H. Ohsaki and M. Murata, "On Control Parameters Tuning for Active Queue Management Mechanisms using Multivariate Analysis,"- SAINT, (2003).
12. Shao Liu, Tamer Basar, and R. Srikant, "Exponential RED: A Stabilizing AQM Scheme for Low- and High-speed TCP Protocols" (October 2005).
13. S. Floyd, "Recommendations on using the gentle variant of RED" (May 2000). http://www.aciri.org/ floyd/red/gentle.html
14. J. Aweya, M. Ouellette, and D. Y. Montuno, Computer Networks **36** 203, (2001). doi:10.1016/S1389-1286(00)00206-1
15. T. j. Ott, T. V. Lakshman, and L. Wong, "SRED: Stabilized RED"- IEEE INFOCOM (March 1999) pp. 1346-1355.
16. M. Shin, S. Chong, and I. Rhee, "Dual-Resource TCP/AQM for Processing-Constrained Networks" (April 2008).
17. C. Wang, Bo Li, K.Sohraby, and Y. Peng, "AFRED: An Adaptive Fuzzy-based Control Algorithm for Active Queue Management" - IEEE LCN (2003).
18. S. H. L. Sanjeewa Athuraliya, IEEE Network, **15** (3), 48 (2001). doi:10.1109/65.923940
19. S. Ryu, C. Rump, and C. Qiao, Telecommunication Systems **25** (3-4), 317 (2004).

doi:10.1023/B:TELS.0000014788.49773.70
20. Network simulator-ns2. http://www. mash .cs.berkeley.edu/ns