A NEW ACTIVE QUEUE MANAGEMENT ALGORITHM BASED ON NEURAL NETWORKS PI

M. Yaghoubi Waskasi
MYaghoubi@ece.ut.ac.ir

M. J. Yazdanpanah
Yazdan@ut.ac.ir

N. Yazdani
Yazdani@ut.ac.ir

Control and Intelligent Processing Center of Excellence
Electrical and Computer Engineering Department
University of Tehran
P.O. Box: 14395/515, Tehran, IRAN

Abstract: Active Queue Management (AQM) applies a suitable control policy upon detecting congestion in networks. In this paper, an adaptive Proportional-Integral (PI) controller based on Artificial Neural Networks (ANN) is applied to AQM for the objective of congestion avoidance and control in middle nodes. The proposed controller is simple and can be easily implemented in high-speed routers. Neural Network PI (NNPI) dynamically adapts its parameters with respect to changes in the system. It is anticipated that this results in better response compared to linear controllers due to the nonlinear nature of NNPI. We simulated our method in *ns2* and compared its performance with the conventional PI controller. The simulation results show NNPI yields better performance. *Copyright © 2005 IFAC*

Keywords: Neural Networks, Queuing Network Models, Nonlinear Systems, Computer Networks, PI Controllers.

1. INTRODUCTION

Congestion is a major problem in data networks and it is considered an active and a hot research area for researchers (Low, *et al.*, 2002). Initially, this problem introduced by Jacobson (1988), in the last 80's, it caused proposing different improved versions of TCP in order to have more control on congestion. Most of the proposed methods for congestion are implemented on end nodes while there is no special control policy in middle nodes and packets are simply dropped when the queues are full, known as DropTail (Low, *et al.*, 2002). In 1993, RED (Random Early Detection) was introduced to regulate queues' lengths in middle nodes based on a desired policy (Floyd, Jacobson, 1993). This approach, named Active Queue Management (AQM), manages the queue length by dropping packets randomly before the queue buffer overflows. Researchers have found that the basic RED does not meet their requirements and it causes some instability in the network when the number of sources increases or the parameters of the network change (Hollot, *et al.* 2001a). Even in some cases its performance is worse than the simple DropTail policy (Christiansen, *et al.,* 2000). Some new approaches based on the primary RED have been introduced to improve the performance of RED and solve its challenges (Ott, *et al.*, 1999; Feng, *et al.*, 1999a). Some new methods also have been proposed such as BLUE (Feng, *et al.*, 1999b) and GREEN (Feng, *et al.*, 2002) to achieve better performance and provide more sophisticated control policy.

Definitely, using a mathematical model can give a more general view of a dynamical model. We can also better analyze the behavior of the system and its transient response to an input based on a mathematical model. Due to the nonlinear and time variant dynamics of the global computer network, some researcher attempted to describe its behavior by some mathematical formula (Floyd, 2001; Misra, *et al.*, 2000). One of the best models of this type about TCP is a model that was developed by Misra, *et al.* (2000), which is based on the fluid flow model. In this model, the probability of packet drop is calculated according to the control law. Therefore, one's primary goal is to design a controller that produces appropriate control signals. The first controller of this model was a simple PI controller proposed by Hollot, *et al.* (2001a) which gives a better performance than RED.

Unfortunately, studies proved that AQM with PI controller is not robust in response to uncertainties in the network and increasing the number of sources (Fengyuan, *et al.*, 2002). Therefore, designing a more robust algorithm became a hot research topic (Fengyuan, *et al.*, 2002). Some researchers tried to introduce non-model-based controllers because of their robustness in parameters changing (Fengyuan and Xiuming, 2002; Fatta, *et al.*, 2002).

Certainly, any new AQM algorithm should have better performance and easy to implement. In this paper, we apply an adaptive PI controller based on ANN to AQM in order to control traffic in middle nodes and avoid congestion in the network. Our algorithm is simple to implement and have a more satisfactory performance in comparison to the existing mechanisms. NNPI dynamically adapts its parameters with changes in the system. Due to the nonlinear nature of NNPI, it is anticipated that our controller results in better response than those of linear controllers designed before.

The rest of the paper is organized as follows: In Section 2, we introduce the active queue management scheme. Section 3 presents the mathematical modeling of congestion in computer networks. In Section 4, the structure of NNPI is discussed. Simulation result is presented in Section 5. Finally, Section 6 concludes the paper.

2. ACTIVE QUEUE MANAGEMENT

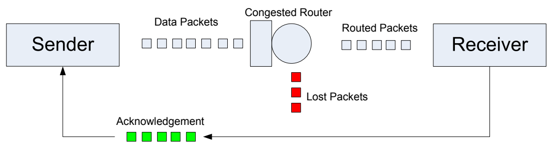

Fig. 1. A middle node router

Figure 1 shows a middle node router that applies the active queue management policy for packets arriving to its queue.

In the algorithms previously proposed, the router was dropping packets when its queue buffer was full. This policy is known as DropTail. One of the most important weaknesses of DropTail is that the router drops any arriving packets even a short-lived flow, known as mice, when long-lived flows, known as elephants, fill the buffer. This conflicts with the well respected fairness principle of computer networks. Meanwhile, the variation of the queue length in middle nodes causes jitter, variation in the delay, in packet delivery, which is not desired for real-time applications. The amount of delay experienced in the network depends on the amount of round trip time of packets. Equation 1 describes this relevance.

$$R_i(t) = a_i + \frac{q(t)}{C} \qquad (1)$$

Index i indicates flow number and a_i is a constant indicating the propagation delay of the link. C and $q(t)$ are line capacity and the queue length respectively. The second part of equation 1 represents the time wasted to process incoming packets.

With fixing the queue length, since a_i and C are constants, we can guarantee a constant round trip time delay. Therefore, AQM can help to provide Quality of Service (QoS) in networks.

RED was the first proposed AQM policy which drops all arriving packets if the average queue length exceeds the maximum threshold. In this model routers processes and sends all packets if the average queue length was smaller than the minimum threshold. When the average queue length lies between the maximum and minimum thresholds, packets are dropped with a probability of p, which is proportional to the average queue length. The RED behavior is illustrated in figure 2.

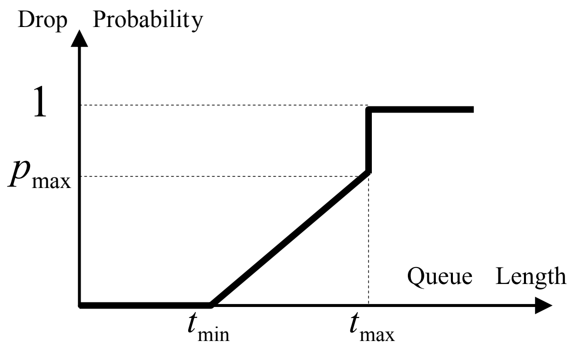

Fig. 2. RED dropping policy

Indeed, AQM acts as a regulator for the queue length regarding the control theory point of view. The input of this controller is the degree of congestion measured by the average and the current queue lengths. The output is the probability of packet dropping or marking. The control block diagram of computer networks is illustrated in figure 3.

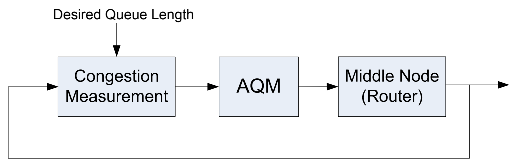

Fig. 3. AQM as a controller

There are many control theoretic-based approaches for AQM such as PI, FPI, REM, AVQ and etc. In the next section, we introduce a mathematical model for computer networks which helps us to design a controller for AQM.

3. MATHEMATICAL MODEL OF COMPUTER NETWORKS

A practical model of active queue management based on fluid flow was proposed in 2000 and verified by some computer simulations, which has introduced an acceptable accuracy for the proposed model (Misra, *et al.*, 2000). The behavior of packet loss in this model was considered as a Poisson process. Then, a stochastic differential equation is derived based on some assumptions.

Consider a router with N sources. As explained previously, the round trip time of packets in each flow is as below:

$$R_i(t) = a_i + \frac{q(t)}{C}$$

Now, we can determine the congestion window size by:

$$dW_i = \frac{dt}{R_i(t)} - \frac{W_i}{2} dN_i \qquad (2)$$

Where dN_i is equal to 1 when the packet loss occurs. Equation 2 consists of two increasing and decreasing parts. The increasing part indicates the additive increase of the AIMD algorithm while the decreasing part indicates the multiplicative decrease. Chiu and Jain (1989) try to prove that AIMD achieves fair and efficient network utilization. AIMD increases congestion window size by one in each round trip time. This occurs in the congestion avoidance phase of TCP. Equation 2 only considers the congestion avoidance phase of TCP. This is true for elephant flows such as FTP flows, since the time of slow start in long-lived flows is small. To model this phase, we have:

$$dW_i = \frac{W_i dt}{R_i(t)} \qquad (3)$$

In this equation, we ignore the effect of the packet loss in the slow start stage. The decrease section halves the congestion window size in packet loss due to the multiplicative decrease in AIMD.

Equation 2 models an end to end system. We should also model the middle node's behavior or the queue length variation dynamic. So we have:

$$\frac{dq(t)}{dt} \approx -C + \sum_{i=1}^{N} \frac{W_i}{R_i(q)} \qquad (4)$$

The decreasing part of this equation corresponds to the line capacity, while the increasing part is the number of packets arriving to the queue.

Hollot, *et al.*, (2001), assume a constant round trip time and suppose same sources and rewrite equation 2 and 4 as the following (Hollot, *et al.* 2001b):

$$\dot{W}(t) = \frac{1}{R_0} - \frac{W^2(t)}{2R_0} p(t - R_0) \qquad (5)$$
$$\dot{q}(t) = \frac{W(t)}{R_0} N - C$$

Indeed, they ignored some delays in the model parameters and clearly stated that these assumptions are acceptable. Equation 5 is nonlinear with time delay in its input. The AQM control law should produce a suitable probability, $p(t)$, which eventually yields a good performance. Figure 4 represents an overall view of the AQM controller position.

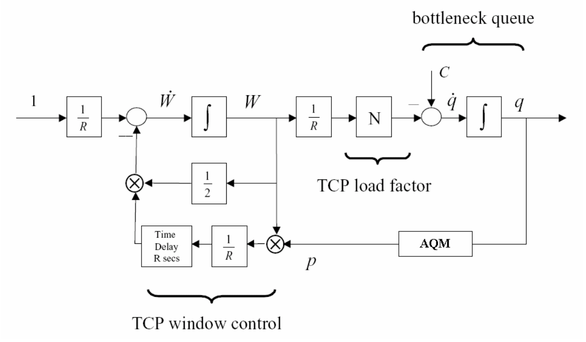

Fig. 4: AQM block diagram

4. PI AND NNPI CONTROLLERS

The proposed PI controller in (Hollot, *et al.* 2001a) calculates the probability of drop using the error and integral of the error between the current queue size and the desired queue length. A formal PI controller is as follows (Franklin, *et al.*, 1995):

$$p(t) = K_p e(t) + K_i \int_0^t e(\tau) d\tau \qquad (6)$$

Where K_p and K_i are the proportional and integral coefficients and the equation of error is:

$$e(t) = q(t) - q_{desired} \qquad (7)$$

PI controller yields a zero steady state error because of the integral term. The most challenging problem in designing a PI controller is choosing the appropriate K_p and K_i. In (Hollot, *et al.* 2001a), the authors linearized the nonlinear equations and designed a robust and stable PI controller using this linear model. They assumed a minimum number of sources and a maximum amount of round trip time for their uncertainty boundaries. However, it can be shown that the PI controller dose not operate well when the number of flows increases or the parameters of network changes.

We introduce a neural network-based PI (Matsukuma, *et al.*, 1997) controller that adapts its parameters with respect to the difference between the current queue length and the desired queue length.

Consider the one layer feedforward neural network that is shown in figure 5.

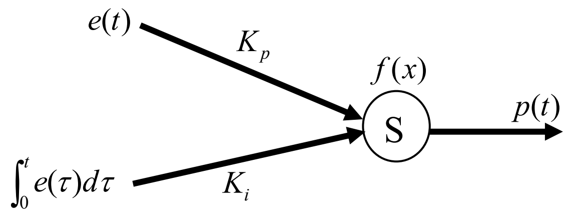

Fig. 5. A simple neural network-based PI (NNPI)

$e(t)$ is the error function and $f(x)$ is the activation function of neuron. We have:

$$\begin{aligned} &p(t) = f(x) \\ &While \\ &x = K_p e(t) + K_i \int_0^t e(\tau)d\tau \end{aligned} \quad (8)$$

We choose the activation function as a hyperbolic tangent. The maximum value of the hyperbolic tangent is 1, so it is suitable for probability function that its maximum is 1.

$$f(x) = \tanh(kx) = \frac{1-e^{-2kx}}{1+e^{-2kx}} \quad (9)$$

In this equation, k instances for the slope of hyperbolic tangent function.

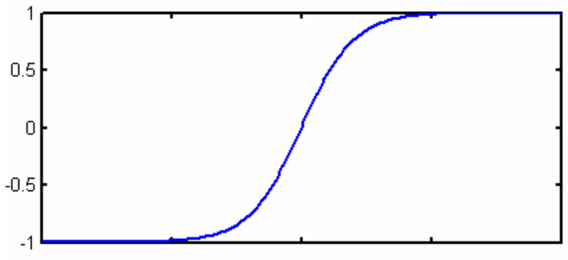

Fig. 6. The activation function of neural network

A lower k yields a smoother activation function and a higher k yields a sharper activation function. The hyperbolic tangent function for $k=1$ is shown in figure 6.

To update the weight of neural networks, we use the error back propagation rule. Using the steepest decent, we have:

$$\begin{aligned} K_{p_{new}} &= K_{p_{old}} - \eta_p \frac{\partial E}{\partial K_p} \\ K_{i_{new}} &= K_{i_{old}} - \eta_i \frac{\partial E}{\partial K_i} \end{aligned} \quad (10)$$

Where E is the cost function. The cost function can be defined as the sum of square errors, and so we have:

$$E = \frac{1}{2}\Sigma(e(t))^2 \quad (11)$$

Therefore,

$$\begin{aligned} \frac{\partial E}{\partial K_p} &= \frac{\partial E}{\partial q}\frac{\partial q}{\partial p}\frac{\partial p}{\partial x}\frac{\partial x}{\partial K_p} = -(q - q_{desired})\frac{\partial q}{\partial p} f'(x)e(t) \\ \frac{\partial E}{\partial K_i} &= \frac{\partial E}{\partial q}\frac{\partial q}{\partial p}\frac{\partial p}{\partial x}\frac{\partial x}{\partial K_i} = -(q - q_{desired})\frac{\partial q}{\partial p} f'(x)\int_0^t e(\tau)d\tau \end{aligned} \quad (12)$$

We choose $\frac{\partial q}{\partial p} = 1$ for convenience.

For the initial weight of the neural network, we use the proposed PI coefficients mentioned by Hollot, *et al.*, (2001). Neural network will change their weight that acts as PI coefficient according to system properties. Our experiments show that the learning rates of order of 10^{-12} are suitable for this problem.

5. SIMULATION RESULTS

We evaluated the effectiveness and performance of the proposed method via simulations using *ns2* simulator. The network topology for our simulations is shown in figure 7. In these simulations, we consider a network with a link capacity of 15Mbps. The only bottleneck link lies between node 1 and node 2. Our simulation duration was 100 seconds. PI controller's gains were chosen as the coefficient derived in (Hollot, *et al.* 2001a). The learning rates of the neural network were equal to 2e-10.

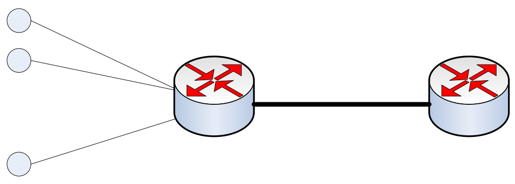

Fig. 7. Network topology of our simulations

Simulation 1: For the first experiment, we set the round trip time to 246 milliseconds and considered

400 ftp sources connected to node 1. The mean packets size of TCP packets was 500 bytes. The desired queue length was 200 packets while the physical limit of our queue was 800 packets. The queue length responses for PI and NNPI are depicted in figure 8. Considering this figure, it is clear that the proposed method should results in a better performance and queue regulation than that of the PI controller.

Our proposed method guarantees the maximum queue length of 200 packets while the AQM with PI oscillates around 200 packets. Variation of the queue length in the PI algorithm causes jittering in delay that is not desired in multimedia applications.

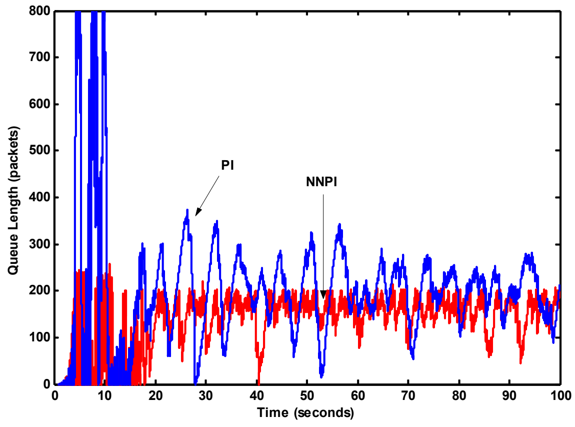

Fig 8. N=60, RTT=246ms, Q_{Refrence}=200 pkts

Simulation 2: Now, we increase the number of sources to 400. While PI has a sluggish response, our method achieves a more acceptable and faster response (figure 9).

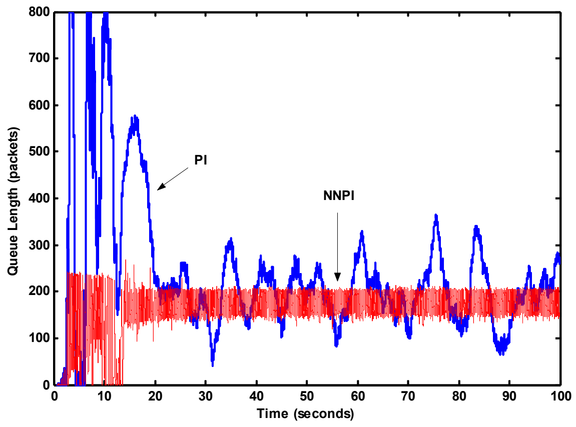

Fig. 9. N=400, RTT=246ms, Q_{Refrence}=200 pkts

Simulation 3: We tested the controllers at one of the end of stability spectrum by reducing the number of sources to 20. As illustrated in figure 10, PI controller exhibits oscillations while the NNPI controller operates in a relatively stable mode.

Simulation 4: Figure 11 shows the queue length plots when we increase our desired queue length to 400 packets. While PI controller oscillates heavily, the NNPI regulates the queue length to the desired value.

Simulation 5: In this experiment we decrease the time delay of the link to 70 milliseconds. It is obvious from figure 12 that NNPI controller acts better in this test bed in comparison to PI controller.

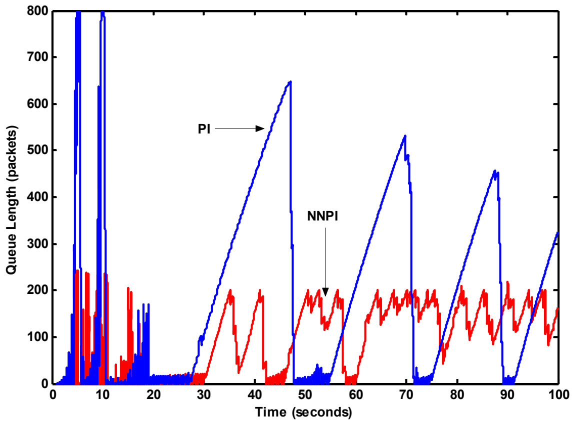

Fig. 10. N=20, RTT=246ms, Q_{Refrence}=200 pkts

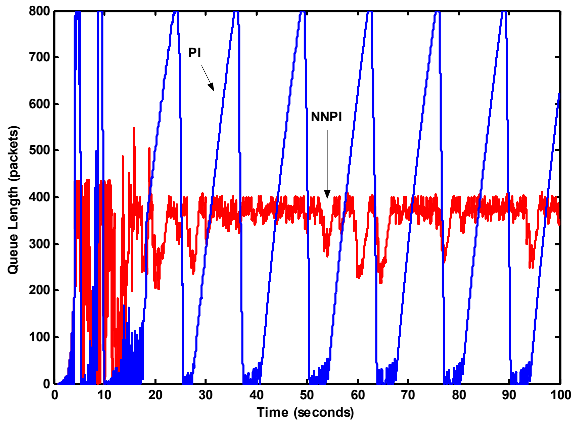

Fig. 11. N=60, RTT=246ms, Q_{Refrence}=400 pkts

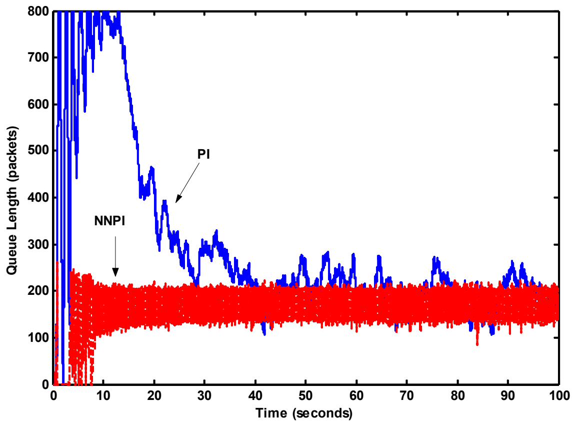

Fig 12: N=400, RTT=70ms, Q_{Refrence}=200 pkts

6. CONCLUSION

We propose a new method for active queue management based on Neural Network PI controller. Our algorithm adapts itself in response to uncertainties in network parameters. Our method is simple to implement and deploy in high speed routers. To investigate the performance of the proposed controller, some computer simulations have been done using *ns2* simulator. The simulations' results show that NNPI controller has a superior performance than that of the classical PI controller in different networks.

Our next step will be studying the effect of unresponsive flows like UDP on the AQM.

REFERENCES

Chiu D. and R. Jain (1989). Analysis of the Increase/Decrease Algorithms for Congestion Avoidance in Computer Networks, *Journal of Computer Networks and ISDN*, **Vol. 17**, No. 1, pp. 1-14.

Christiansen M., K. Jeffay, D. Ott, and F.D. Smith (2000). Tuning RED for web traffic, *Proceedings of ACM/SIGCOMM*.

Fatta G. Di, G. Lo Re, and A. Urso (2002). A Fuzzy Approach for the Network Congestion Problem, *Computer Communications,* **vol. 25**, pp. 874-883, Elsevier Science.

Feng W., Dilip D. Kandlur, Debanjan Saha, and Kang G. Shin (1999a). A Self-Configuring RED Gateway, *Proceedings of IEEE/INFOCOM*.

Feng W., *et al.* (1999b). Blue: A New Class of Active Queue Management Algorithms, *Technical Report* CSE-TR-387-99, University of Michigan .

Feng W., A. Kapadia, and S. Thulasidasan (2002). GREEN: Proactive Queue Management over a Best-Effort Network, *Proc. of IEEE Globecom 2002*.

Fengyuan R., L. Chuang, Y. Xunhe, S. Xiuming, and W. Fubao (2002). A Robust Active Queue Management Algorithm Based on Sliding Mode Variable Structure Control, *Proc. Of IEEE INFOCOM.* New York US: IEEE, pp. 13-20.

Fengyuan R. Y. R. and S. Xiuming (2002). Design of a Fuzzy Controller for Active Queue Management, *Computer Communications*, **vol. 25**, pp. 847-883, Elsevier Science.

Floyd S. and V. Jacobson (1993). Random Early Detection Gateways for Congestion Avoidance, *IEEE/ACM Trans. Net.,* **vol. 1**, no. 4, pp. 397–413.

Floyd S. (2001). A Report on Recent Developments in TCP Congestion Control, *IEEE Commun. Mag.*, **vol. 39**, no. 4, pp. 84–90.

Franklin G. F., J. David Powell, and Abbas Emami-Naeini (1995). Feedback Control of Dynamic Systems, *Addison-Wesley.*

Hollot C. V., Vishal Misra, Don Towsley, and Wei-Bo Gong (2001a). On Designing Improved Controllers for AQM Routers Supporting TCP Flows, *Proc. of INFOCOM 2001*.

Hollot C. V., Vishal Misra, Don Towsley, and Wei-Bo Gong (2001b). A Control Theoretic Analysis of RED, *Proceedings of IEEE/INFOCOM*.

Jacobson V. (1988). Congestion Avoidance and Control, *Proc. ACM SIGCOMM '88*, pp. 314–29.

Low S., F. Paganini and J. Doyl (2002). Internet Congestion Control, *IEEE Control System Magazine*, **vol. 22**, no. 6, pp. 954-959.

Matsukuma T., A. Fujiwara, M. Namba, and Y. Ishida (1997). Non-Linear PID Controller Using Neural Networks, *IEEE 1997*.

Misra V., W. Gong, and D. Towsley (2000). Fluid-Based Analysis of a Network of AQM Routers Supporting TCP Flows with an Application to RED, *Proc. of ACM SIGCOMM 2000*, available at ftp://gaia.cs.umass.edu/pub/

Ott T. J., T. V. Lakshman, and L. H.Wong (1999). SRED: Stabilized RED, *Proceedings of IEEE/INFOCOM* .