Published in IET Communications
Received on 31st October 2009
Revised on 27th March 2010
doi: 10.1049/iet-com.2009.0700

ISSN 1751-8628

Effective adaptive virtual queue: a stabilising active queue management algorithm for improving responsiveness and robustness

H. Wang[1] *C. Liao*[2] *Z. Tian*[1]
[1]*Department of Automation, Shanghai Jiao Tong University, Shanghai 200240, People's Republic of China*
[2]*Department of Mechanical and Aerospace Engineering, University of Florida, Gainesville, FL 32611, USA*
E-mail: haowang.sjtu@gmail.com

Abstract: Adaptive virtual queue (AVQ) algorithm is an effective method aiming to achieve low loss, low delay and high-link utilisation at the link. However, it is difficult to guarantee fast response, strong robustness and good trade-off over a wide range of network dynamics. The authors propose a stabilising active queue management (AQM) algorithm – effective-AVQ, as an extension of AVQ, to improve the responsiveness and robustness of the transmission control protocol (TCP)/AQM system. Specifically, a proportional integral derivative (PID) neuron is introduced to tune the virtual link capacity dynamically. Also we derive the parameter self-tuning mechanism for the PID neuron from the Hebbian learning rule and gradient descent approach. The stability condition of the closed-loop system is presented based on the time-delay control theory. The performance of effective-AVQ is validated in the NS2 platform. Simulation results demonstrate that effective-AVQ outperforms AVQ in terms of steady-state and transient performance. It achieves fast response, expected link utilisation, low queue size and small delay jitter, being robust against dynamic network changes.

1 Introduction

Unpredictable number of users is growing rapidly in today's Internet, which results in network congestion and even congestion collapse. To keep the network stable and to guarantee satisfactory network performance, Internet congestion control mechanisms have been intensively studied over the last decades. Window-based flow control strategy [1] was added in transmission control protocol (TCP) to handle congestion. Also active queue management (AQM) mechanism [2] was introduced to assist TCP congestion control and to provide high-link utilisation with low-queuing delay in TCP/IP networks. The key idea of AQM is to give an early notification of congestion degree to end hosts by dropping or marking packets before the buffer overflows in routers. In the past few years, a series of AQM schemes have been proposed. In terms of the congestion indicators, AQM algorithms can be categorised into four classes: (i) queue-based schemes such as random early detection (RED) [3], proportional integral controller (PI) [4]; (ii) rate-based schemes like adaptive virtual queue (AVQ) [5]; (iii) buffer event-based algorithms like BLUE [6]; and (iv) hybrid methods such as random exponential marking (REM) [7].

The first well-known AQM scheme, RED [3], was introduced to routers for solving the synchronisation problem and keeping the average queue length low. RED randomly drops incoming packets with a probability proportional to the average queue length. However, RED is sensitive to traffic loads and parameter configurations, thus, several modified RED schemes [8–12] have been proposed to alleviate the above problems. These schemes were designed on the basis of empirical observation and simulation analysis, so they are only valid under a narrow range of network conditions. Recently, some systematic approaches, like loss ratio-based RED (LRED) [13], self-tuning proportional and integral RED (SPI-RED) [14] and auto-parameterisation RED (AP-RED) [15], were developed to improve the integrated performance of original RED. They outperform the heuristics-based variants of RED in terms of stability, robustness and trade-off between link utilisation and queuing delay.

PI [4] is another queue-based method derived from control theory. It is superior to RED, but suffers from sluggish response and parameter configuration problems. The PI controller with fixed parameters cannot maintain stable queue length over a wide range of network dynamics. Specifically, the buffer overflows, and the transient period becomes quite long in the case of heavy traffic loads. Some enhanced algorithms were proposed to improve the transient and steady-state performance of the original PI controller. PID controller [16] is able to detect and control incipient and current congestion by adding a differential segment, which improves the response of PI effectively. Q-SAPI controller [17] aims to improve the transient performance of the original PI controller, while maintaining its steady-state performance over a wide range of uncertainties in various network scenarios. Chen *et al.* used classical control theory to design a pole placement technique-based PI controller [18], which could achieve desirable transient response and stability by choosing proper configurations of damping ratio and undamped natural frequency. Hong and Yang proposed a self-tuning TCP traffic controller [19] and an adaptive PI rate controller [20] based on Nyquist stability criterion to guarantee the stability and robustness of the TCP/AQM system.

AVQ [5] is a rate-based algorithm, which is able to detect incipient congestion information. It is able to maintain the queue length at a low level, but its responsiveness and robustness cannot be guaranteed with fixed parameters. BLUE [6] uses buffer overflow and emptiness events rather than the queue length or traffic rate to detect and control congestion. However, it cannot avoid buffer overflow, which inevitably causes consecutive packet drop and large queuing delay. REM [7] takes queue length and input rate as congestion indicators. It has similar control property to the PI control and suffers from slow convergence and big queue oscillations. Some modified methods, such as intelligent price-based controller (IPC) [21] and self-tuning price-based controller (SPC) [22], have been proposed to improve the performance of REM. They provide the guidelines to design enhanced price and parameter tuning rules for improving responsiveness, adaptability and robustness.

In this paper, we aim at the AVQ scheme and propose the effective-AVQ algorithm (E-AVQ for short) to improve its responsiveness and robustness. In particular, a PID neuron is employed to update the virtual link capacity, which has significant influence on the stability and robustness of AVQ. Then, a parameter self-tuning mechanism for the weights and gain of the neuron is derived from Hebbian learning rule and gradient descent method. The stability analysis is also conducted based on the time-delay control theory. Simulation results demonstrate that E-AVQ is responsive and robust in dynamic networks. It is superior to the original AVQ scheme in achieving fast convergence rate, stable queue length and expected high-link utilisation.

The remainder of the paper is organised as follows. Section 2 introduces the related work about the AVQ scheme. Our proposed algorithm, E-AVQ, is presented in Section 3, including algorithm description, stability analysis and implementation. Section 4 presents the performance evaluation of the proposed algorithm under different network configurations. Finally, we draw our conclusions in Section 5.

2 Related work

Kunnuyur and Srikant [5] proposed a novel AQM algorithm called AVQ. The motivation behind AVQ is to develop an AQM scheme that provides a high-link utilisation with low loss and low delay. Specifically, AVQ maintains a virtual queue, whose capacity (named virtual capacity) is less than the real capacity of the congested link. When a packet arrives at the actual queue, the virtual capacity is also updated. Once the virtual queue overflows, the packet in the real buffer is marked. Let C be the real link capacity, $\tilde{C}$ be the virtual queue capacity and γ be the desired utilisation at the link. At arrival time of each packet, the virtual link capacity is updated according to the following equation

$$\dot{\tilde{C}}(t) = \alpha(\gamma C - \lambda(t)) \tag{1}$$

where λ is the aggregate arrival rate and α is the smoothing parameter.

Then, the number of bytes Vq in the virtual queue is calculated as

$$Vq = Vq + b - \tilde{C}(t-k) \tag{2}$$

where b is the bytes of the arrival packet and k and t are the arrival time of the previous packet and current time, respectively.

AVQ could successfully keep the queue length low, which is regarded as an attractive merit. However, it suffers from some shortcomings in terms of stability and robustness. In particular, the literature in [23] indicated that the rule for configuring AVQ's parameters is impractical because the stability condition given in [5, 24] is unsolvable for almost all practical network environments. Additionally, AVQ has poor robustness against dynamic networks, resulting in sluggish responsiveness and drastic queue oscillations. To alleviate the above problems, scalable parameter tuning rules have been introduced in [25, 26]. But the control parameters are tuned in particular network conditions, causing lack of flexibility and adaptability.

3 Proposed algorithm – E-AVQ

3.1 Algorithm description

In light of the previous investigations, the updating rule shown in (1) has great influence on the integrated performance of AVQ. Furthermore, it is difficult to guarantee desirable quality of service using constant parameter configuration. Hence, we propose an E-AVQ scheme with self-adjusting parameters to improve the responsiveness and robustness of AVQ.

In our work, the updating rule of the virtual capacity is designed as

$$\dot{\tilde{C}}(t) = K_i e(t) + K_p \dot{e}(t) + K_d \ddot{e}(t) \tag{3}$$

where $e(t) = \gamma C - \lambda(t)$ and $K_i, K_p, K_d > 0$. Being different from (1), our updating rule is able to detect congestion more efficiently; moreover, its parameters are adjusted online in dynamic networks.

The marking probability is expressed as follows

$$p(\lambda, \tilde{C}) = \left(\frac{\lambda - \tilde{C}}{\lambda}\right)^{+} \tag{4}$$

where $(*)^{+} = \max[0, \min(*, 1)]$.

Then, we employ a single neuron [27] to update the virtual link capacity, because of its easy implementation, self-learning ability and good adaptability. Rewriting (3) to the discrete-time form yields

$$\tilde{C}(t) = \tilde{C}(k) + g(t)\sum_{i=1}^{3} w_i(t)x_i(t) \tag{5}$$

where $\boldsymbol{X} = \begin{bmatrix} x_1 & x_2 & x_3 \end{bmatrix}^{\mathrm{T}} = \begin{bmatrix} e & \Delta e & \Delta^2 e \end{bmatrix}^{\mathrm{T}}$ is the input vector, $\Delta e(t) = e(t) - e(k)$ is the deviation of the error and $\Delta e^2(t) = \Delta e(t) - \Delta e(k)$. The weight coefficients corresponding to x_1, x_2 and x_3 are defined as $w_1 = K_i T$, $w_2 = K_p$ and $w_3 = K_d/T$. $g(t)$ is the gain of the neuron and let $T = t - k$ denotes the sampling interval.

In control theory, the weight coefficient and the gain of the neuron have crucial impact on the system performance. To ensure desirable performance in dynamic scenarios, they should be adjusted automatically. Considering that the weights w_i should be concerned with the correlative functions of input, output and error signal, supervising Hebbian learning rule is used for tuning weight coefficients as

$$\begin{aligned} w_i(t) &= w_i(k) + \eta_i v_i(k) \\ v_i(k) &= z(k)\tilde{C}(k)x_i(k) \end{aligned} \tag{6}$$

where $v_i(k)$ is the recursive signal and hushes gradually during the process. $z(k) = e(k)$ is the error signal and $i = 1, 2, 3$.

To ensure the convergence and robustness of the neuron-based updating method shown in (5) and (6), the learning algorithm should be disposed canonically, then we obtain

$$\begin{aligned} \tilde{C}(t) &= \tilde{C}(k) + g(t)\sum_{i=1}^{3} \bar{w}_i(t)x_i(t) \\ \overline{w_i}(t) &= \frac{w_i(t)}{\sum_{i=1}^{3} |w_i(t)|} \\ w_i(t) &= w_i(k) + \eta_i z(k)\tilde{C}(k)x_i(k) \end{aligned} \tag{7}$$

We derive the self-tuning rule for the gain of the neuron via gradient descent method. The performance index function is defined as follows

$$J(t) = \frac{1}{2}e^2(t) \tag{8}$$

The gain of the neuron is adjusted along the minus direction of J as

$$g(t) = g(k) - \eta_4 \frac{\partial J(k)}{\partial g(k)} \tag{9}$$

where η_4 is the learning rate, which should be small enough to keep the system stable.

The partial differential equation in the right side of (9) is expanded via the chain rule as

$$\frac{\partial J(k)}{\partial g(k)} = \frac{\partial J(k)}{\partial e(k)} \frac{\partial e(k)}{\partial \Delta\tilde{C}(k)} \frac{\partial \Delta\tilde{C}(k)}{\partial g(k)} \tag{10}$$

According to (8) and (7), each term is calculated as follows

$$\frac{\partial J(k)}{\partial e(k)} = e(k) \tag{11}$$

$$\frac{\partial e(k)}{\partial \Delta\tilde{C}(k)} = \mathrm{sgn}\frac{e(t) - e(k)}{\Delta\tilde{C}(t) - \Delta\tilde{C}(k)} \tag{12}$$

$$\frac{\partial \Delta\tilde{C}(k)}{\partial g(k)} = \sum_{i=1}^{3} \bar{w}_i(k)x_i(k) \tag{13}$$

where (12) is an approximate derivative determined by changing the output $e(k)$ and the input $\Delta\tilde{C}(k)$.

Submitting (10–13) into (9) yields the adjustment rule of the gain g as

$$g(t) = g(k) - \eta_4 e(k)\mathrm{sgn}\frac{e(t) - e(k)}{\Delta\tilde{C}(t) - \Delta\tilde{C}(k)}\sum_{i=1}^{3} w_i(k)x_i(k) \tag{14}$$

3.2 Stability analysis

Considering a single bottleneck link network, a non-linear dynamic model for the congestion avoidance phase of the additive increase and multiplicative decrease algorithm (AIMD) has been presented in [28]. The bottleneck link with capacity C has N homogeneous TCP sources traversing it with the same RTT delay d. Through using the $-1/(d^2\lambda)$ utility function that was proposed in [28], the dynamic behaviours of the jth TCP user can be modelled as

$$\dot{\lambda}_j(t) = \frac{1}{d^2} - \beta\lambda_j(t)\lambda_j(t-d)p\left(\sum_{l=1}^{N}\lambda_l(t-d),\ \tilde{C}(t-d)\right) \tag{15}$$

where λ_j and p denote the traffic rate of the jth TCP user and packet marking probability, respectively. According to [24], setting β as 2/3 would be appropriate that gives the steady-state throughput of TCP of $\sqrt{3/(2p^*)}/d$, where p^* is the steady-state marking probability. Hence, we set $\beta = 2/3$ in all calculations.

Let λ_j^*, λ^*, $\tilde{C}^*$ and p^* denote the equilibrium values of λ_j, λ, $\tilde{C}$ and $p(\lambda, \tilde{C})$. The equilibrium point of the non-linear TCP/AQM model is given by

$$\begin{aligned} \sum_{j=1}^{N}\lambda_j^* &= \lambda^* = \gamma C \\ \lambda_j^* &= \gamma C/N \\ p^* &= p(\gamma C, \tilde{C}^*) = N^2/\beta(d\gamma C)^2 \end{aligned} \tag{16}$$

Then assume that

$$\begin{aligned} \lambda(t) &= \lambda^* + \delta\lambda(t) \\ \tilde{C}(t) &= \tilde{C}^* + \delta\tilde{C}(t) \end{aligned} \tag{17}$$

According to (15), (3) and (4), the linearised model of the non-linear TCP/AQM model can be written as

$$\begin{aligned} \delta\dot{\lambda}(t) &= -K_{11}\delta\lambda(t) - K_{12}\delta\lambda(t-d) + K_2\delta\tilde{C}(t-d) \\ \delta\dot{\tilde{C}}(t) &= -K_i\delta\lambda(t) - K_p\delta\dot{\lambda}(t) - K_d\delta\ddot{\lambda}(t) \end{aligned} \tag{18}$$

where $K_{11} = N/\gamma Cd^2$, $K_{12} = K_2 = \beta\gamma C/N$.

Let $\Lambda(s)$ and $\Theta(s)$ denote the Laplace transforms of $\lambda(t)$ and $\tilde{C}(t)$. Taking the Laplace transform of (18) yields

$$\begin{aligned} s\Lambda(s) &= -K_{11}\Lambda(s) - K_{12}\mathrm{e}^{-sd}\Lambda(s) + K_2\mathrm{e}^{-sd}\Theta(s) \\ s\Theta(s) &= -K_i\Lambda(s) - K_p s\Lambda(s) - K_d s^2\Lambda(s) \end{aligned} \tag{19}$$

From (19), we have the following equation

$$P(s)\Lambda(s) = 0 \tag{20}$$

where $P(s)$ is the characteristic polynomial given by

$$\begin{aligned} P(s) = {}& (1 + K_2K_d\mathrm{e}^{-sd})s^2 + (K_{11} + K_{12}\mathrm{e}^{-sd} + K_2K_p\mathrm{e}^{-sd})s \\ & + K_2K_i\mathrm{e}^{-sd} \end{aligned} \tag{21}$$

In the following, the stability of the TCP/E-AVQ system is analysed on the basis of control theory [29].

Theorem: System (18) is asymptotically stable if and only if the control parameters satisfy

$$(m_0m_3 + m_1m_2)^2 - 4(m_1m_3 + m_4)(m_0m_2 - m_4) < 0 \tag{22}$$

or

$$\begin{cases} (m_0m_3 + m_1m_2)^2 - 4(m_1m_3 + m_4)(m_0m_2 - m_4) \geq 0 \\ (m_0 - m_1)(m_2 - m_3) > 0 \end{cases} \tag{23}$$

where $m_0 = 2K_{11} + 2(K_{12} + K_2K_p)K_2K_d$, $m_1 = 2(K_{12} + K_2K_p + K_{11}K_2K_d)$, $m_2 = 2K_2K_i(K_{12} + K_2K_p)$, $m_3 = 2K_{11}K_2K_i$, $m_4 = 4K_2^2K_i^2$.

Proof: Letting $\mathrm{e}^{-sd} = z$, where z is a complex number, we rewrite (21) as

$$\mathrm{P}(s) = a_0s^2 + a_1s + a_2 \tag{24}$$

where $a_0 = 1 + K_2K_dz$, $a_1 = K_{11} + K_{12}z + K_2K_pz$ and $a_2 = K_2K_iz$.

Then we construct a Hermitian matrix [29] as

$$H = \begin{bmatrix} (0,1) & (0,2) \\ (0,2) & (1,2) \end{bmatrix}$$

where

$$\begin{aligned} (0,1) &= \mathrm{i}^{0+1+1}((-1)a_0\bar{a}_1 + (-1)\bar{a}_0a_1) = a_0\bar{a}_1 + \bar{a}_0a_1 \\ &= 2K_{11} + (K_{12} + K_2K_p)(z + \bar{z}) + K_{11}K_2K_d(z + \bar{z}) \\ &\quad + 2(K_{12} + K_2K_p)K_2K_d|z|^2 \end{aligned} \tag{25}$$

$$\begin{aligned} (0,2) &= \mathrm{i}^{0+2+1}((-1)^2a_0\bar{a}_2 + (-1)\bar{a}_0a_2) = -\mathrm{i}(a_0\bar{a}_2 - \bar{a}_0a_2) \\ &= \mathrm{i}K_2K_i(z - \bar{z}) \end{aligned} \tag{26}$$

$$\begin{aligned} (1,2) &= \mathrm{i}^{1+2+1}((-1)^2a_1\bar{a}_2 + (-1)^2\bar{a}_1a_2) = a_1\bar{a}_2 + \bar{a}_1a_2 \\ &= K_{11}K_2K_i(z + \bar{z}) + 2K_2K_i(K_{12} + K_2K_p)|z|^2 \end{aligned} \tag{27}$$

Let $z = \mathrm{e}^{\mathrm{i}\omega} = \cos\omega + \mathrm{i}\sin\omega$ and rewrite (25)–(27) as follows

$$(0,\ 1) = 2K_{11} + 2(K_{12} + K_2K_p)K_2K_d + 2(K_{12} + K_2K_p + K_{11}K_2K_d)\cos\omega$$

$$(0,\ 2) = -2K_2K_i \sin\omega$$

$$(1,\ 2) = 2K_2K_i(K_{12} + K_2K_p) + 2K_{11}K_2K_i\cos\omega$$

Then we denote (see equation at the bottom of the page)

Condition 1: The matrix $H(1) = H(e^{i0})$ is positive definite, in other words, the sequential principal minors of $H(1)$ are all positive, where (see equation at the bottom of the page)

We have

$$\begin{cases} 2K_{11} + 2(K_{12} + K_2K_p)K_2K_d + 2(K_{12} + K_2K_p + K_{11}K_2K_d) > 0 \\ \det H(1) > 0 \end{cases}$$

Note that all the parameters, such as K_{11}, K_{12}, K_2, K_p, K_i and K_d, are positive.

Condition 2: $\det H(e^{i\omega}) > 0$, for all $\omega \in [0, 2\pi]$

$$\begin{aligned} \det H(e^{i\omega}) &= [2K_{11} + 2(K_{12} + K_2K_p)K_2K_d \\ &\quad + 2(K_{12} + K_2K_p + K_{11}K_2K_d)\cos\omega] \\ &\quad \cdot [2K_2K_i(K_{12} + K_2K_p) + 2K_{11}K_2K_i\cos\omega] \\ &\quad - 4K_2^2K_i^2\sin^2\omega > 0 \\ &= (m_0 + m_1\cos\omega)(m_2 + m_3\cos\omega) \\ &\quad - m_4\sin^2\omega > 0 \end{aligned} \tag{28}$$

Submitting $\sin^2\omega = 1 - \cos^2\omega$ to (28), we rewrite the inequality with respect to $\cos\omega$ as

$$\det H(e^{i\omega}) = (m_1m_3 + m_4)\cos^2\omega + (m_0m_3 + m_1m_2)\cos\omega + m_0m_2 - m_4 > 0 \tag{29}$$

Notice that $m_1m_3 + m_4 > 0$. According to the property of the quadratic inequality (29), we discuss the following two cases.

Case 1:

$$(m_0m_3 + m_1m_2)^2 - 4(m_1m_3 + m_4)(m_0m_2 - m_4) < 0 \tag{30}$$

In this case, (29) is fulfilled for all $\cos\omega$, where $\omega \in [0, 2\pi]$.

Case 2:

$$(m_0m_3 + m_1m_2)^2 - 4(m_1m_3 + m_4)(m_0m_2 - m_4) \geq 0 \tag{31}$$

In this case, we have the following requirement from inequality (29), either

$$\cos\omega < \frac{-(m_0m_3 + m_1m_2) - \sqrt{(m_0m_3 + m_1m_2)^2 - 4(m_1m_3 + m_4)(m_0m_2 - m_4)}}{2(m_1m_3 + m_4)} \tag{32}$$

or

$$\cos\omega > \frac{-(m_0m_3 + m_1m_2) + \sqrt{(m_0m_3 + m_1m_2)^2 - 4(m_1m_3 + m_4)(m_0m_2 - m_4)}}{2(m_1m_3 + m_4)} \tag{33}$$

Considering that $\cos\omega \in [-1, 1]$, for all $\omega \in [0, 2\pi]$, conditions (32) and (33) can be reduced to the following inequalities

$$\sqrt{(m_0m_3 + m_1m_2)^2 - 4(m_1m_3 + m_4)(m_0m_2 - m_4)} < -(m_0m_3 + m_1m_2) - 2(m_1m_3 + m_4) \tag{34}$$

or

$$\sqrt{(m_0m_3 + m_1m_2)^2 - 4(m_1m_3 + m_4)(m_0m_2 - m_4)} < m_0m_3 + m_1m_2 - 2(m_1m_3 + m_4) \tag{35}$$

It is obviously observed that there is no solution for inequality (34), because of m_0, m_1, m_2, m_3, $m_4 > 0$. As for (35), we obtain a simple form as

$$(m_0 - m_1)(m_2 - m_3) > 0 \tag{36}$$

Synthesising Condition 1 as well as Condition 2, reaches the stability conditions for the TCP/E-AVQ system shown as (22) and (23).

$$H(e^{i\omega}) = \begin{bmatrix} 2K_{11} + 2(K_{12} + K_2K_p)K_2K_d + 2(K_{12} + K_2K_p + K_{11}K_2K_d)\cos\omega & -2K_2K_i\sin\omega \\ -2K_2K_i\sin\omega & 2K_2K_i(K_{12} + K_2K_p) + 2K_{11}K_2K_i\cos\omega \end{bmatrix}$$

$$H(1) = \begin{bmatrix} 2K_{11} + 2(K_{12} + K_2K_p)K_2K_d + 2(K_{12} + K_2K_p + K_{11}K_2K_d) & 0 \\ 0 & 2K_2K_i(K_{12} + K_2K_p) + 2K_{11}K_2K_i \end{bmatrix}$$

3.3 Implementation of E-AVQ

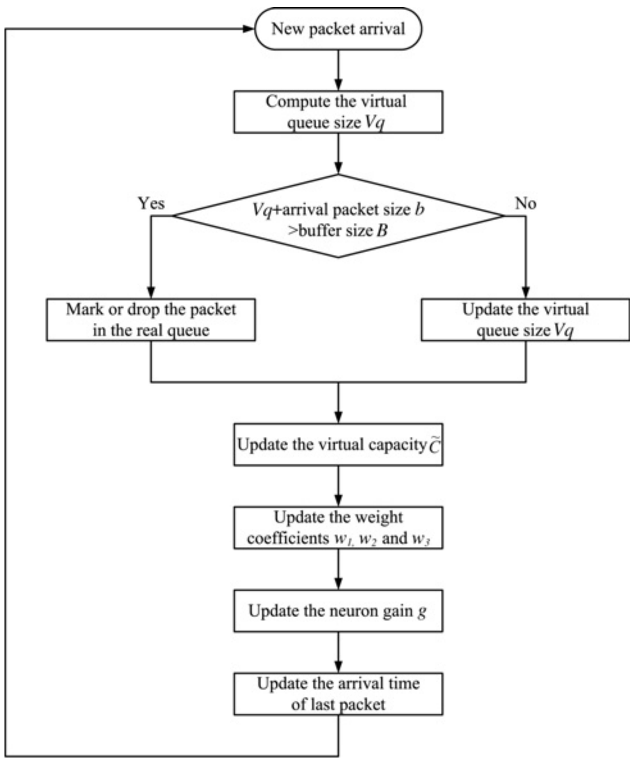
The E-AVQ algorithm is briefly presented through a flowchart shown in Fig. 1. Upon the arrival of each packet, the virtual queue size is being computed. If the virtual queue length overflows, the packet in the real queue is being marked/dropped, otherwise the virtual queue length is going to be updated. Then the virtual capacity and control parameters, including the weight coefficients and the neuron gain, are updated according to (5), (7) and (14), respectively.

4 Performance evaluation

The E-AVQ algorithm is implemented in the NS2 platform [30] to evaluate its effectiveness and robustness. Original AVQ scheme is also simulated as comparison. We conduct simulation experiments in a series of network scenarios with both single-bottleneck (shown in Fig. 2) and multiple-bottleneck topology (shown in Fig. 8). In these simulations, we focus on the evolution of queue length, which is regarded as the key performance measure of AQM. If an AQM scheme controls the queue length with fast convergence and maintains stability with small jitter, the satisfactory network performance with high-link utilisation and low-queuing delay is obtained.

4.1 Simulation in the single-bottleneck topology

Simulations have been conducted in the single bottleneck topology in Fig. 2, which is a dumbbell network topology



New packet arrival
Compute the virtual queue size Vq
Yes
Vq+arrival packet size b >buffer size B
No
Mark or drop the packet in the real queue
Update the virtual queue size Vq
Update the virtual capacity $\tilde{C}$
Update the weight coefficients w_1, w_2 and w_3
Update the neuron gain g
Update the arrival time of last packet

Figure 1 *Flowchart of the E-AVQ algorithm*

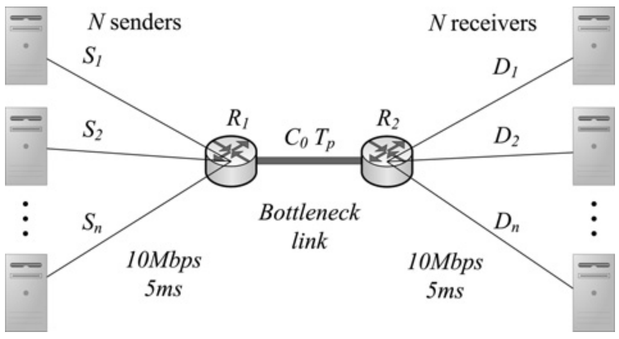


Figure 2 *Simulation network topology*

with multiple TCP connections from $S_{1,2,\ldots,n}$ to $D_{1,2,\ldots,n}$, sharing the single bottleneck link between the router R_1 and R_2.

Each link capacity, as well as the propagation delay, is depicted in Fig. 2. Unless stated elsewhere, the common link capacity C_0 is set to 10 Mbps, the propagation delay $Tp = 10$ ms, the maximum buffer size of each router is set at 100 packets. The links between the TCP sources or sinks and routers are configured as 10 Mbps with 5 ms propagation delay. In the simulations, N greedy FTP flows share the bottleneck link, where each packet has a size of 1000 bytes. In addition, the explicit congestion notification (ECN) technique [31] is enabled, and the packets get marked instead of getting dropped. The original AVQ algorithm is taken as the comparison with the proposed algorithm. For AVQ, the default parameters are configured as in [5]. For E-AVQ, $w_1 = 0.8$, $w2 = w3 = 0.1$, $g = 2.0$, $\eta_1 = \eta_2 = \eta_3 = \eta_4 = 1.0 \times 10^{-18}$.

4.1.1 Queue evolution with constant TCP flows:
In this experiment, we evaluate the steady-state performance of our proposed algorithm with different configurations of parameter γ. The number of TCP connections N is set as 50. Fig. 3 depicts the queue evolution of AVQ and E-AVQ with different γ.

It is evidently observed that E-AVQ outperforms the original AVQ algorithm in achieving fast convergence rate and low queue size with small jitter. Specifically, in the case of $\gamma = 0.98$, E-AVQ quickly stabilises the queue length whereas AVQ spends more than 20 s to keep the queue length stable. In the case of $\gamma = 1$, both algorithms exhibit fast response, but the average queue length of AVQ is much larger, which causes bigger queuing delay. The accurate performance statistics are summarised in Table 1.

From Table 1, we observe that E-AVQ provides accurate link utilisation to the expected value, and achieves lower queue length with smaller jitter than that of AVQ.

4.1.2 Queue evolution under dynamic traffics:
In this experiment, the number of long-live TCP flows is set

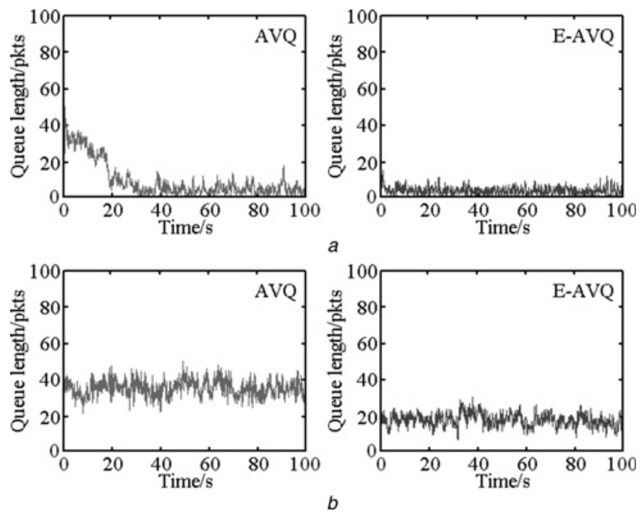


Figure 3 *Evolution of queue length with constant TCP flows*

a $\gamma = 0.98$
b $\gamma = 1.0$

Table 1 Performance statistics of link utilisation, average queue length and standard deviation

γ	AVQ		E-AVQ	
	0.9800	1.0000	0.9800	1.0000
link utilisation	0.9856	0.9982	0.9798	0.9991
average queue length (pkts)	5.4249	35.7856	4.7189	17.7326
standard deviation (pkts)	2.6691	4.5169	1.9299	3.3866

to 50 at the beginning. Additional 100 TCP flows are equally divided into two groups, which are enabled at time $t = 50$ and 100 s, respectively. Fig. 4 depicts the queue evolution with different γ under dynamic traffics.

Sudden changes of traffic flows can be regarded as system disturbances. Whether the queue length can achieve fast responsiveness and maintain stability is an important criterion to evaluate AQM schemes. As we can see from Fig. 4, AVQ suffers from sluggish response when the parameter γ increases. Its queue length oscillates drastically, which inevitably results in big delay jitter. In contrast, in all the three cases, the queue length of E-AVQ keeps stable at a low level against the sudden changes of TCP flows. Hence, the responsiveness and robustness of our proposed algorithm are obviously superior to those of AVQ.

4.1.3 Queue evolution with different link capacities: We conduct this experiment to study the performance of E-AVQ under different network configurations, where the link capacities are set as 2 and

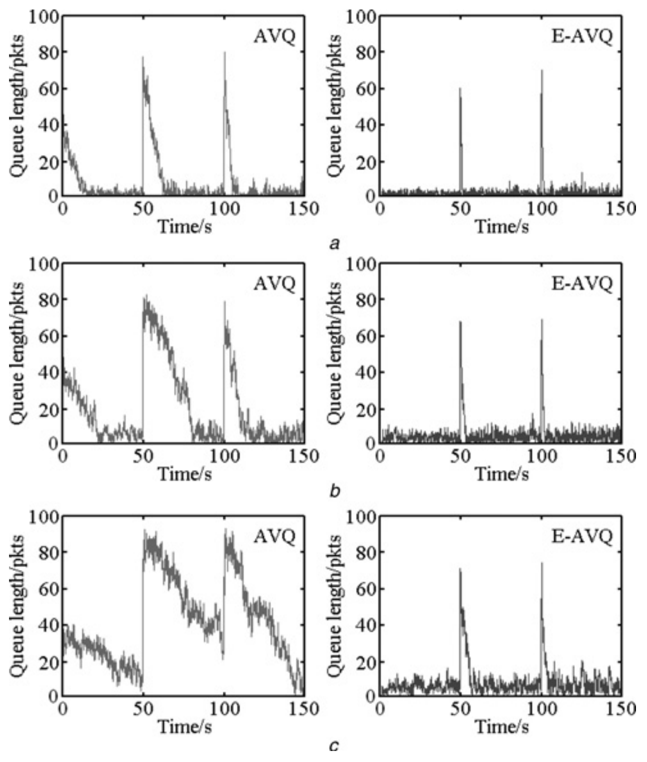


Figure 4 *Evolution of queue length under dynamic traffics*

a $\gamma = 0.95$
b $\gamma = 0.98$
c $\gamma = 0.99$

20 Mbps, respectively. The other network parameters follow the configurations, which are listed in Section 4.1.2. Fig. 5 depicts the simulation results as follows.

In the case of $C_0 = 2$ Mbps, the queue size of AVQ is big and its oscillation is also remarkable. Both steady-state and transient performance deteriorate. On the contrary, E-AVQ successfully stabilises the queue length low and responds fast. In the other case with $C_0 = 20$ Mbps, the result is similar to the previous one, demonstrating the good responsiveness and robustness of our proposed algorithm.

4.1.4 Queue evolution with unresponsive flows: We conduct this simulation to study the responsiveness and robustness of E-AVQ in dynamic networks mixed with unresponsive flows. Fifty TCP flows are started at the beginning and stopped at the end. Additional 100 TCP flows as well as 50 UDP flows are both equally divided into two groups, which are started at 50 and 100 s, respectively, and stopped at the end. The inter-packet gap of a CBR flow is set to 0.1 s. Fig. 6 depicts the queue evolution with different γ under unresponsive flows.

From Fig. 6, the queue lengths of AVQ and E-AVQ increase when the traffic loads change abruptly. E-AVQ can keep fast response and stable queue length in all

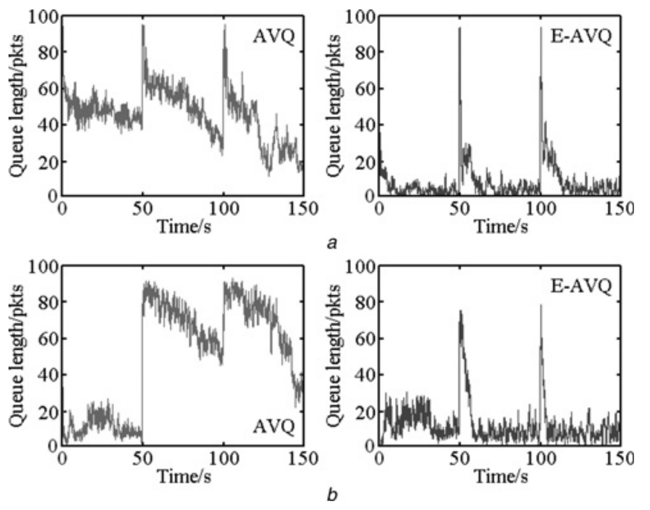


Figure 5 *Evolution of queue length with different link capacities*

a $C_0 = 2$ Mbps
b $C_0 = 20$ Mbps

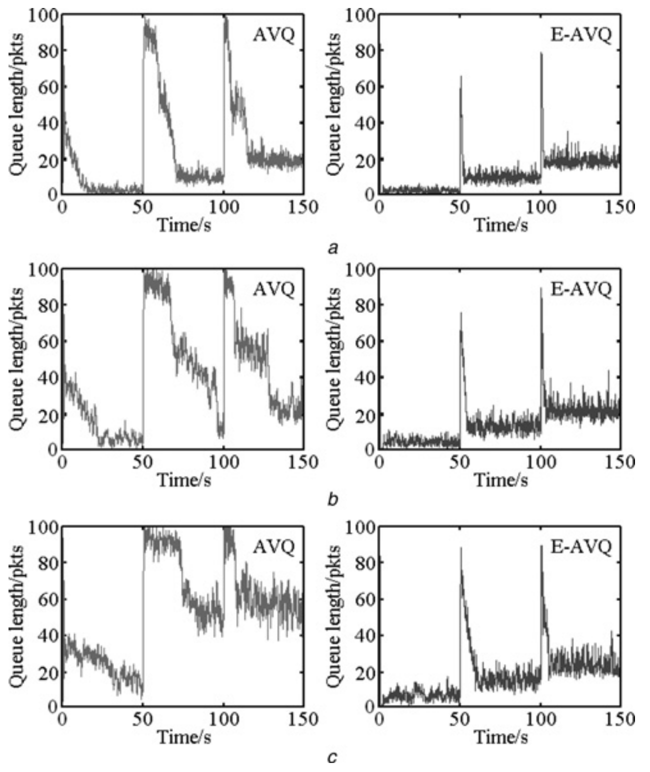


Figure 6 *Evolution of queue length with unresponsive flows*

a $\gamma = 0.95$
b $\gamma = 0.98$
c $\gamma = 0.99$

circumstances tested. However, the integrated performance of AVQ, including transient and steady-state performance, dramatically deteriorates when the parameter γ increases. The settling time becomes longer and the fluctuation of queue size gets larger. Also AVQ suffers from big queue overshoots, which will cause consecutive packet drops.

4.1.5 Queue evolution with HTTP flows:

Short-lived HTTP traffic can influence the network performance as a result of queue oscillation or even unstable queue length. In this simulation, we test the performance of E-AVQ under disturbances of additional HTTP flows. The number of long-lived TCP connections is set as 100. The HTTP traffic is generated by 'PagePool/Web-Traf' provided by NS2. The page pool attached to each of the congested link contains five servers and five clients. The number of pages per session is 1000 and the other parameters are configured as the default values. Fig. 7 depicts the evolution of queue length with HTTP flows.

From Fig. 7, it is observed that E-AVQ can robustly stabilise the queue length with fast response and small oscillation in each scenario. On the contrary, AVQ exhibits big overshoots and requires much longer time to decrease the queue length from the buffer top. This set of simulation results demonstrates that E-AVQ is effective to regulate the queue length against HTTP disturbances.

4.2 Simulation in the multiple-bottleneck topology

In this subsection, we consider a multiple-bottleneck topology in Fig. 8. There are three traffic groups. The first group has N_1 TCP connections traversing all links; the second group has N_2 TCP connections traversing the link

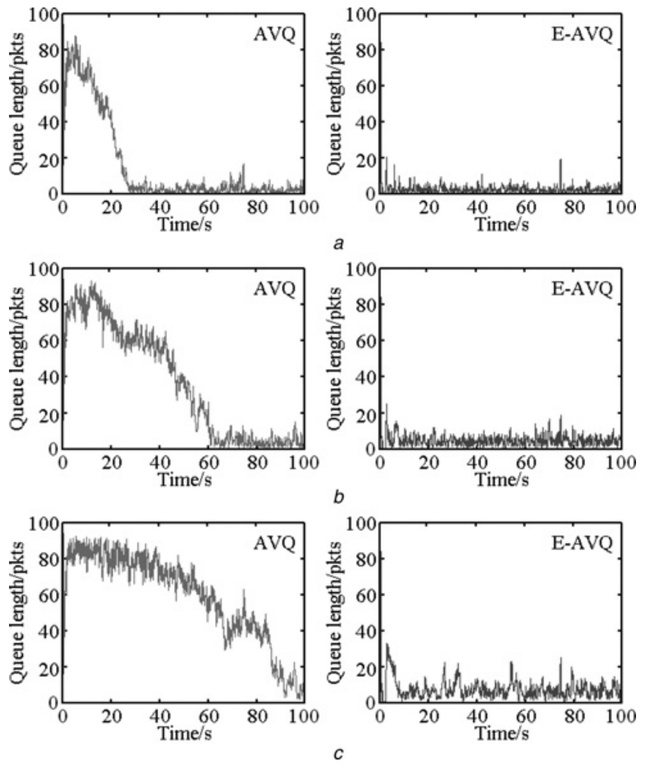


Figure 7 *Evolution of queue length with HTTP flows*

a $\gamma = 0.95$
b $\gamma = 0.98$
c $\gamma = 0.99$

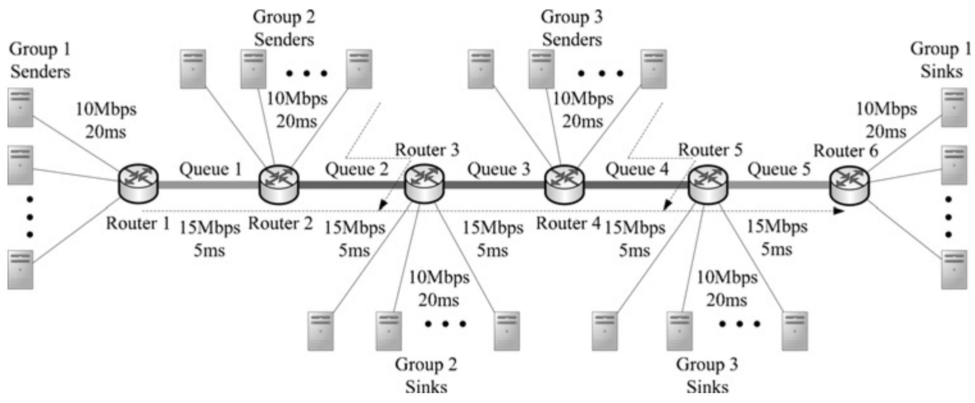


Figure 8 *Multiple-bottleneck topology*

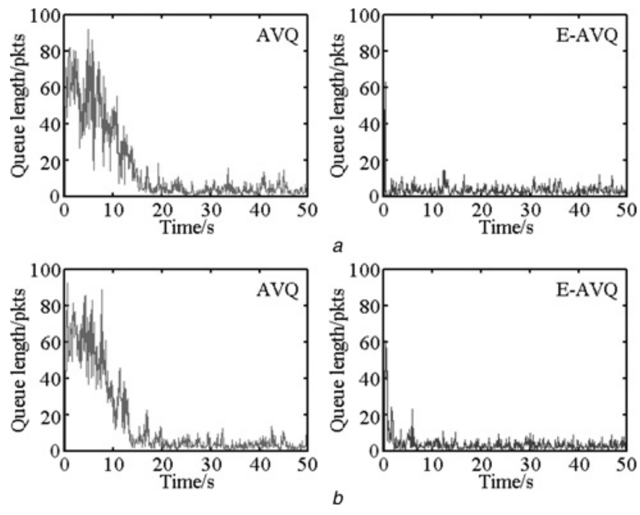


Figure 9 *Evolution of queue length under multiple-bottleneck topology ($\gamma = 0.95$)*

a Queue 2 (R2–R3)
b Queue 4 (R4–R5)

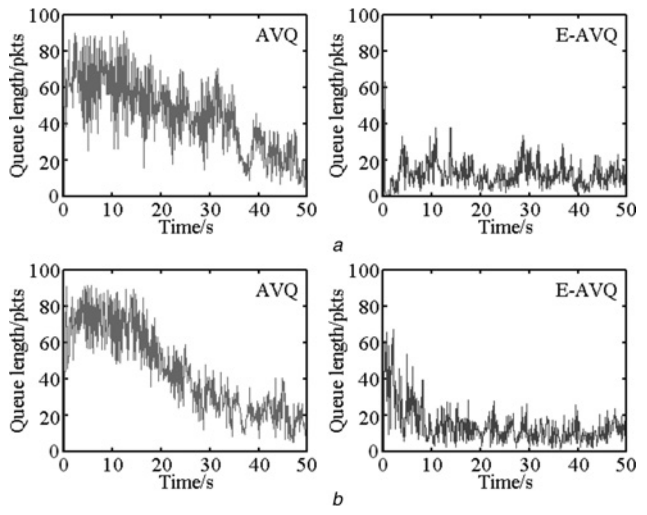


Figure 10 *Evolution of queue length under multiple-bottleneck topology ($\gamma = 0.99$)*

a Queue 2 (R2–R3)
b Queue 4 (R4–R5)

between Router 2 and Router 3 and the third group has N_3 Reno-TCP connections traversing the link between Router 4 and Router 5. The buffer size is 100 packets; the link capability and propagation delay of each link are indicated in Fig. 8. The performance of E-AVQ is compared with AVQ. The parameters of both algorithms are set to the same values as in the above subsection.

In the experiment, we set $N_1 = N_2 = N_3 = 100$ and ECN mechanism is also enabled. Figs. 9 and 10 plot the queue evolution with different configurations of γ.

Observed from Fig. 9, the queue evolution of Queue 2 (R2–R3) and Queue 4 (R4–R5) exhibits similarly. E-AVQ can fast regulate the queue size low and maintain

Table 2 Performance statistics of average queue length, standard deviation, packet loss ratio and link utilisation

γ	0.95				0.99			
Algorithms	AVQ		E-AVQ		AVQ		E-AVQ	
Queue	Q2	Q4	Q2	Q4	Q2	Q4	Q2	Q4
average queue length (pkts)	4.4874	3.9853	3.6912	3.3428	21.5493	19.9720	11.0314	11.1850
standard deviation (pkts)	2.8637	2.5228	2.0311	1.7862	9.0259	7.9553	5.3531	4.8262
packet loss ratio	0.0033	0.0024	0.0015	0.0018	0.0036	0.0039	0.0015	0.0018
link utilisation	0.9523	0.9490	0.9499	0.9496	0.9985	0.9985	0.9872	0.9905

small jitter. However, the initial queue length of AVQ is big and the response is slow. When we increase γ from 0.95 to 0.99, Fig. 10 reveals that the transient processes of both algorithms get longer, and the queue length jitters become bigger. Compared to AVQ, E-AVQ still shows its superior performance in terms of stability, responsiveness and robustness. The accurate performance statistics are summarised in Table 2.

From Table 2, we observe that E-AVQ provides lower average queue length, smaller queue length jitter and less packet loss in all cases. The only unattractive performance of E-AVQ is that its link utilisations are lower than those of AVQ. But we can see that the link utilisations of E-AVQ are much closer to γ regarding its anticipated values, which demonstrate E-AVQ's capability of providing accurate link utilisation to the expected value.

5 Conclusions

AVQ scheme is developed to provide high-link utilisation with low delay. However, it suffers from parameter configuration problem, sluggish response and poor robustness. To address these problems, we have proposed a stabilising AQM scheme via automatic control theory. Specifically, an adaptive PID neuron is employed to adjust the virtual link capacity. Scalable parameter setting rule is also presented via the time-delay control theory to guarantee the stability of the TCP/E-AVQ system. Simulation experiments have been conducted to evaluate the integrated performance of our proposed algorithm. The results demonstrate the superiority of our proposed algorithm in achieving fast system response and strong robustness.

In future work, the stability condition in multiple-bottleneck topology needs further investigation. It is also very interesting to conduct simulation experiments with modified TCP versions or in wireless networks. Additionally, the fairness issue against non-response flows will be studied.

6 References

[1] PAGANINI F., WANG Z., DOYLE J.C., LOW S.H.: 'Congestion control for high performance, stability and fairness in general networks', *IEEE/ACM Trans. Netw.*, 2005, **13**, (1), pp. 43–56

[2] BRADEN B., CLARK D., CROWCROFT J., *ET AL.*: 'Recommendations on queue management and congestion avoidance in the internet'. Technical Report RFC 2309, IETF, 1998

[3] FLOYD S., JACOBSON V.: 'Random early detection gateways for congestion avoidance', *IEEE/ACM Trans. Netw.*, 1993, **1**, (4), pp. 397–413

[4] HOLLOT C.V., MISRA V., TOWSLEY D., GONG W.: 'On designing improved controllers for AQM routers supporting TCP flows'. Proc. IEEE Conf. on Computer Communications, Anchorage, AK, April 2001, pp. 1726–1734

[5] KUNNIYUR S., SRIKANT R.: 'Analysis and design of an adaptive virtual queue (AVQ) algorithm for active queue management'. Proc. ACM SIGCOMM, San Diego, CA, USA, August 2001, pp. 123–134

[6] FENG W., SHIN K.G., KANDLUR D.D., SAHA D.: 'The blue active queue management algorithms', *IEEE/ACM Trans. Netw.*, 2002, **10**, (4), pp. 513–528

[7] ATHURALIYA S., LOW S.H., LI V.H., YIN Q.: 'REM: active queue management', *IEEE Netw.*, 2001, **15**, (3), pp. 48–53

[8] FLOYD S., GUMMADI R., SHENKER S.: 'Adaptive RED: an algorithm for increasing the robustness of RED's active queue management', http://www.icir.org/floyd/papers/adaptiveRed.pdf, accessed 2001, pp. 1–12

[9] AWEYA J., OUELLETTE M., MONTUNO D.Y.: 'A control theoretic approach to active queue management', *Comput. Netw.*, 2001, **36**, (2–3), pp. 203–235

[10] SUN J., KO K., CHEN G., CHAN S., ZUKERMAN M.: 'PD-RED: to improve the performance of RED', *IEEE Commun. Lett.*, 2003, **7**, (8), pp. 406–408

[11] ZHOU K., YEUNG K.L., LI V.O.K.: 'Nonlinear RED: a simple yet efficient active queue management scheme', *Comput. Netw.*, 2006, **50**, (18), pp. 3784–3794

[12] ABBASOV B., KORUKOGLU S.: 'Effective RED: an algorithm to improve RED's performance by reducing packet loss rate', *J. Netw. Comput. Appl.*, 2009, **32**, (3), pp. 703–709

[13] WANG C., LIU J., LI B., SOHRABY K., HOU Y.T.: 'LRED: a robust and responsive AQM algorithm using packet loss ratio measurement', *IEEE Trans. Parall. Distr. Syst.*, 2007, **18**, (1), pp. 29–43

[14] XIONG N., PAN Y., JIA X., PARK J.H., LI Y.: 'Design and analysis of a self-tuning feedback controller for the Internet', *Comput. Netw.*, 2009, **53**, (11), pp. 1784–1797

[15] CHEN W., YANG S.H.: 'The mechanism of adapting RED parameters to TCP traffic', *Comput. Commun.*, 2009, **32**, (13–14), pp. 1525–1530

[16] RYU S., RUMP C., QIAO C.: 'A predictive and robust active queue management for Internet congestion control'. Proc. IEEE Symp. on Computers and Communications, Kemer-Antalya, Turkey, June 2003, pp. 991–998

[17] CHANG X., MUPPALA J.K.: 'A stable queue-based adaptive controller for improving AQM performance', *Comput. Netw.*, 2006, **50**, (13), pp. 2204–2224

[18] CHEN Q., YANG O.W.W.: 'AQM controller design for IP routers supporting TCP flows based on pole placement', *IEE Proc. Commun.*, 2004, **151**, (4), pp. 347–354

[19] HONG Y., YANG O.W.W.: 'Self-tuning TCP traffic controller using gain margin specification', *IET Commun.*, 2007, **1**, (1), pp. 27–33

[20] HONG Y., YANG O.W.W.: 'Design of adaptive PI rate controller for best-effort traffic in the internet based on phase margin', *IEEE Trans. Parall. Distrib. Syst.*, 2007, **18**, (4), pp. 550–561

[21] WANG H., TIAN Z.: 'Intelligent price-based congestion control for communication networks'. Proc. IEEE IWQoS, Tsinghua, Beijing, China, June 2010, pp. 1–5

[22] WANG H., TIAN Z., ZHANG Q.: 'Self-tuning price-based congestion control supporting TCP networks'. Proc. ICCCN, ETH Zurich, Switzerland, August 2010, pp. 1–6

[23] KATABI D., BLAKE C.: 'A note on the stability requirement of adaptive virtual queue', http://publications.csail.mit.edu/lcs/pubs/pdf/MIT-LCS–TM-626, accessed 2002, pp. 1–5

[24] KUNNIYUR S., SRIKANT R.: 'An adaptive virtual queue (AVQ) algorithm for active queue management', *IEEE/ACM Trans. Netw.*, 2004, **12**, (2), pp. 286–299

[25] TAN L., YANG Y., LIN C., XIONG N., ZUKERMAN M.: 'Scalable parameter tuning for AVQ', *IEEE Commun. Lett.*, 2005, **9**, (1), pp. 90–92

[26] YANG Y., TAN L., XIONG N.: 'PDAVQ: an adaptive virtual queue algorithm based on the proportional and differential control', *J. Commun.*, 2005, **26**, (3), 39–44

[27] HAYKIN S.: 'Neural networks: a comprehensive foundation' (Prentice-Hall, 2nd edn., 1999)

[28] KUNNIYUR S., SRIKANT R.: 'End-to-end congestion control schemes: utility functions, random losses and ECN marks', *IEEE/ACM Trans. Netw.*, 2003, **11**, (5), pp. 689–702

[29] LEHNIGK S.H.: 'Stability theorems for linear motions' (Englewood Cliffs, Prentice-Hall, NJ, 1966)

[30] Network Simulator-2. Available at http://www.isi.edu/nsnam/ns/

[31] RAMAKRISHNAN K., FLOYD S., BLACK D.: 'The addition of explicit congestion notification (ECN) to IP'. Technical Report RFC 3168, IETF, 2001