# Proportional-Integral Active Queue Management with an Anti-Windup Compensator

Hyuk Lim, Kyung-Joon Park, Eun-Chan Park, and Chong-Ho Choi
School of Electrical Engineering and Computer Science
Seoul National University, Seoul, Korea
{hyuklim, kjpark, ecpark, chchoi}@csl.snu.ac.kr

*Abstract* — **In this paper, we apply an anti-windup scheme for improving the performance of a conventional Proportional-Integral (PI) controller for Active Queue Management (AQM) supporting TCP flows. When a PI controller is used for AQM, the "windup" phenomenon of the integral action can cause performance degradation because the packet drop probability is limited between 0 and 1. Therefore we model TCP/AQM as a system with a saturator and apply an anti-windup method to the conventional PI AQM scheme. We compare the performance of the proposed controller with the conventional PI controller through ns simulations. The simulation results show that our proposed control scheme outperforms the conventional PI controller when the traffic load is varying, which is always the case in the real network environment.**

## I. Introduction

The current TCP congestion control with drop-tail queues has some problems: First of all, the TCP sources of drop-tail queues reduce their rates only after detecting packet loss due to queue overflow. Therefore considerable time may have passed between the packet drop and its detection. At the same time, a large number of packets may be dropped as the sources continue to transmit at a rate that the network cannot support. In addition, packet drop at drop-tail queues could result in the global synchronization of sources [1].

To alleviate these problems, Random Early Detection (RED) gateways were proposed for Active Queue Management (AQM) [2][3]. However, the original RED algorithm also has several shortcomings. First, the parameter tuning remains an inexact science [4][5][6]. Furthermore, there exist some arguments on the deployment of RED, especially in the case of small buffers [4]. Many variants of RED were proposed to resolve these problems [6][7][8][9]. Recently, several researchers have proposed system theoretic approaches [10][11][12][13][14][15][16]. The authors of [10][11][12] proposed an optimization-based view of networks and suggested the Random Early Marking (REM) algorithm, which is actually a PI controller. In [13] and [14], the authors gave a control theoretic analysis of RED and designed a PI controller which outperformed RED significantly. The authors of [15] and [16] focused on stabilizing the queue and proposed an integral controller for AQM.

In this paper, we model AQM as a system with a saturator because the packet drop probability is limited between 0 and 1 [17][18]. With this constraint, the output of the integral controller would increase and become large if the queue size remains below the target value over some period. Once this happens, the integral controller cannot regulate the queue size properly as the queue size changes and this could result in a significant performance degradation. This kind of problem is known to control engineers as the "windup" phenomenon of an integral controller, which exists when a system consists of an integral cotroller and a saturator at the control input [18][19].

To resolve this problem, we add a saturator to the TCP/AQM model and apply an anti-windup method to the conventional PI AQM scheme. We compare the performance of the proposed scheme with the conventional PI controller through ns simulations. The simulation results show that our proposed control scheme outperforms the convetional PI controller when the traffic is fluctuating, which is always the case in the real network environment. Our main contribution in this paper is as follows:

*(i)* We consider the limitation of the packet drop probability between 0 and 1 and introduce an TCP/AQM model with a saturating actuator. This saturation phenomenon has not been considered earlier in the literature.

*(ii)* To compensate the saturation, We apply an anti-windup scheme to the conventional PI controller. Also we compare the performance of our proposed scheme with the PI AQM through ns simulations.

The rest of the paper is organized as follows. In section II, we present the details of the TCP/AQM system with emphasis on the saturating actuator. In Section III we describe the proposed anti-windup algorithm. In Section IV, we compare the proposed algorithm with the conventional PI controller via simulation using an ns-2 network simulator [20]. Finally, we present the conclusion in Section V.

## II. Feedback control of active queue management

RED is an AQM algorithm, which controls network congestion by randomly dropping or marking packets with a probability $p_d$. When TCP sources detect that their packets are dropped or marked, they reduce their sending rates, and the queue size of the router decreases. This process constitutes a closed loop feedback control system as shown in Figure 1 [15]. The system consists of TCP sources, a router queue, and a congestion controller. The congestion controller regulates the queue size of the router by changing the probability $p_d$. Because the PI controller ensures that the steady state error becomes zero, it is better than RED for active queue management [14]. Figure 2 depicts the conventional PI controller. The controller input is the queue size error $q_e$, which is defined as the difference between the target queue size and the queue size of the router as follows:

$$q_e(t) = q(t) - q_{ref} \tag{1}$$

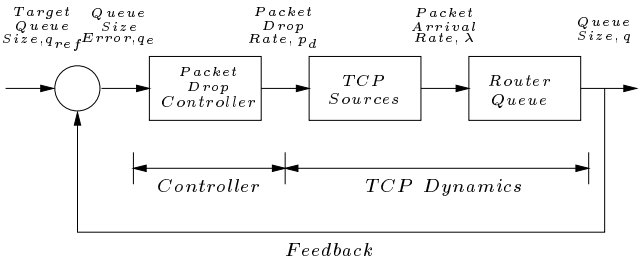



Figure 1: TCP congestion avoidance as a closed-loop feedback control system

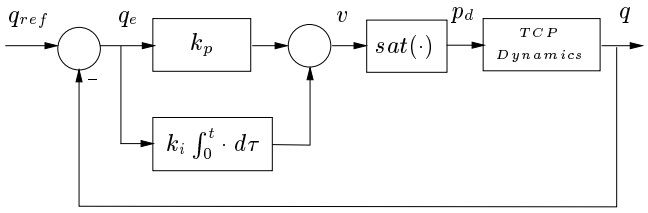



Figure 2: Conventional PI control

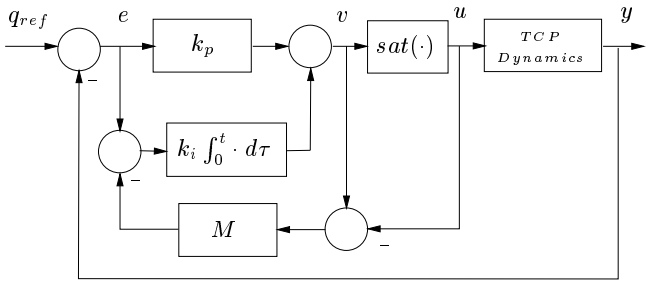



Figure 3: PI control with the proposed anti-windup compensator

where $q_{ref}$ is the target queue size of the router. The drop probability is obtained by limiting the controller output within the range of [0, $p_{max}$] as the following equation:

$$p_d(t) = \text{sat}(v(t)) \tag{2}$$

$$\text{sat}(x) = \begin{cases} p_{max} & x > p_{max} \\ 0 & x < 0 \\ x & \text{otherwise} \end{cases} \tag{3}$$

where $p_{max}$ is the maximum drop probability. According to the drop probability, packets are dropped or marked. When the packets are dropped or marked, each of the TCP sources adjusts its window size to reduce its sending rates, and consequently the queue size decreases.

A PI controller consists of a proportional and an integral controller as follows:

$$v(t) = k_p q_e(t) + k_i \int_0^t q_e(\tau) d\tau \tag{4}$$

where $k_p$, $k_i$ are the proportional gain and the integral gain respectively [21]. A PI controller regulates the queue size of the router properly when it operates around the target queue size. However, real network traffic load varies rapidly due to the bursty nature, and sometimes can be much lighter than what is required to maintain the target queue size. If the queue size is smaller than the target queue size for a certain period of time, then the state of the integral controller will become a large negative value, and $p_d$ will become zero. If this situation is followed by rapid traffic increase, a buffer overflow will occur because the state of the integral controller will be negative for a considerable time. Consequently the performance of the congestion control degrades significantly.

## III. PI CONTROL WITH AN ANTI-WINDUP COMPENSATOR

Here we adopt an anti-windup scheme in [22] [23]. Actually this scheme is a special case of a more general method, i.e. a dynamic anti-windup scheme in [24] [25] [26].

First we formulate the state-space representation of the AQM system. We use a linearized version of the TCP dynamics in [13]. Here we use the incoming rate $r_{in}(t)$ instead of the window size as a sate variable. Further we ignore the delay in the control input. The TCP model is as follows:

$$\dot{\boldsymbol{x}}_p(t) = A\boldsymbol{x}_p(t) + Bu(t) \tag{5}$$
$$y(t) = C\boldsymbol{x}_p(t) + Du(t) \tag{6}$$
$$u(t) = \text{sat}(v(t)) \tag{7}$$

where $\boldsymbol{x}_p(t) = (\delta q(t), \delta r_{in}(t))^T$, $u(t) = p_d(t)$, $A = \left(\begin{smallmatrix} 0 & 1 \\ 0 & a \end{smallmatrix}\right)$, $B = (0, b)^T$, $C = (1, 0)$, $D = 0$, $a = -\frac{2N}{R^2 C_l}$, and $b = -\frac{2C_l^2}{3N}$. Here $R$, $N$, and $C_l$ are the round-trip time, the number of TCP sessions, and the link capacity, respectively.

Also we represent the conventional PI controller as follows:

$$\dot{x}_c(t) = Fx_c(t) + Ge(t) \tag{8}$$
$$v(t) = Hx_c(t) + Le(t) \tag{9}$$
$$e(t) = r(t) - y(t) \tag{10}$$

where $F = 0$, $G = 1$, $H = k_i$, $L = k_p$, and $r(t) = q_{ref}$.

### *A. Dynamics of the closed-loop system in the absence of the saturating actuator*

From (15) and (16) in [23] we can get the following as dynamics of the closed-loop TCP/AQM system in the absence of the saturating actuator. This corresponds to the case when $u = v$ in the Figure 3.

$$\begin{bmatrix} \dot{x}_c \\ \dot{\boldsymbol{x}}_p \end{bmatrix} = A_l \begin{bmatrix} x_c \\ \boldsymbol{x}_p \end{bmatrix} + \begin{bmatrix} 1 \\ 0 \\ k_p b \end{bmatrix} r \tag{11}$$

where

$$A_l := \begin{bmatrix} 0 & -1 & 0 \\ k_i & 0 & 1 \\ k_i b & -k_p b & a \end{bmatrix} \tag{12}$$

### *B. Dynamics of the closed-loop system in the presence of the saturating actuator*

By similar manner, from Figure 3 we get the following as dynamics of the closed-loop TCP/AQM system in the presence of the saturating actuator.

$$\begin{bmatrix} \dot{x}_c^{sat} \\ \dot{\boldsymbol{x}}_p^{sat} \end{bmatrix} = A_{sat} \begin{bmatrix} x_c^{sat} \\ \boldsymbol{x}_p^{sat} \end{bmatrix} + B_{sat} \tag{13}$$

where

$$A_{sat} := \begin{bmatrix} k_i M & -(1 - k_p M) & 0 \\ 0 & 0 & 1 \\ 0 & 0 & a \end{bmatrix} \tag{14}$$

$$B_{sat} := \begin{bmatrix} (1 - k_p M) r + M \cdot \text{sat}(v) \\ 0 \\ b \cdot \text{sat}(v) \end{bmatrix} \tag{15}$$

Note that we do not consider the operating modes as in [23]. The reason is that the upper saturation is not possible in physical sense. If the drop probability is equal to one, it means that all the incoming packets will be dropped. Actually in this case the contol input is infinite, which is not with the model in (5), (6), and (7). This comes from the fact that the model is valid only when the drop probability is small enough. The physical meaning of the lower saturation is that it is impossible at the router to generate packets when the incoming traffic is less than the link capacity. Consequently we only need to consider the lower saturation case.

Now we need the following assumptions for determining the value of $M$.
(A1) The plant is open-loop stable.
(A2) The controller provides acceptable nominal performance in the absence of the saturating actuator.
(A3) $A - B(I + LD)^{-1}LC$ is nonsingular.

We can easily check that assumption (A1) is satisfied from (5). Assumption (A2) is also satisfied if we assume that we have designed a PI controller by following the guideline in [14]. Also we can check Assumption (A3) easily from (5) and (6).

Now let $(\bar{x}_c, \bar{\boldsymbol{x}}_p^T)^T$ denote the equilibirum point of (11) and $(\bar{x}_c^{sat}, (\bar{\boldsymbol{x}}_p^{sat})^T)^T$ denote the virtual equilibrium point of (13). The design objective is to choose $M$ such that the distance between these two equilibrium points is as close as possible. If we let $(\tilde{x}_c, \tilde{\boldsymbol{x}}_p^T)^T := (\bar{x}_c - \bar{x}_c^{sat}, (\bar{\boldsymbol{x}}_p - \bar{\boldsymbol{x}}_p^{sat})^T)^T$, then we have the following result.

**Theorem 1.** If Assumptions (A1)–(A3) are satisfied, the solution $M$ of

$$\min_M J = [\tilde{x}_c^2 + \tilde{\boldsymbol{x}}_p^T \tilde{\boldsymbol{x}}_p]^{1/2} \tag{16}$$

is uniquely determined by

$$M = 1/k_p. \tag{17}$$

*Proof.* We have the following objective function

$$\begin{aligned} J &= \|(\tilde{x}_c, \tilde{x}_p)\| \\ &= [\tilde{x}_c^2 + \tilde{\boldsymbol{x}}_p^T \tilde{\boldsymbol{x}}_p]^{1/2}. \end{aligned} \tag{18}$$

From Theorem 1 in [23] , the solution $M$ of minimizing (18) is uniquely determined by

$$\begin{aligned} M &= G(D - CA^{-1}B)[(I + LD) - LCA^{-1}B]^{-1} \quad (19) \\ &= GD(I + LD)^{-1} \\ &\quad -G[I - D(I + LD)^{-1}L]C \\ &\quad \cdot[A - B(i + LD)^{-1}LC]^{-1}B(I + LD)^{-1}. \quad (20) \end{aligned}$$

By (5), (6), (7), (8), (9), and (10) we get the following after some simple algebraic calculation.

$$M = 1/k_p.$$

This completes the proof. □

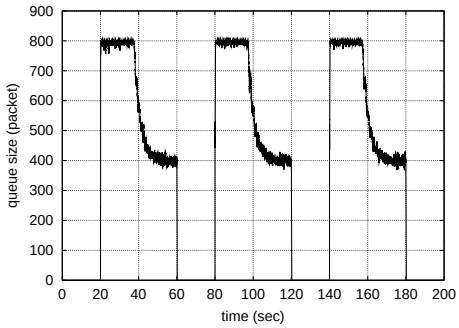
*Remark 1:* The proposed scheme happens to be same with the so-called "conditioning" technique [17]. However the design objective of the proposed scheme is quite different from the conditioning technique.

*Remark 2:* Here we adopted a static compensation method in [23] and designed an anti-windup compensator for nominal system, i.e. given values of $N$ and $R$. However, there is also a dynamic compensation method proposed by Park and Choi [24] [25] [26]. The deployment of the dynamic compensation method will be future work.

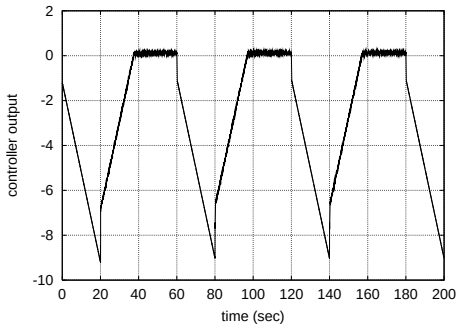
*Remark 3:* While designing the anti-windup compensator, we did not consider the delay in the control input. Consequently the performance of the proposed scheme will degrades in the presence of the delay. The consideration of the delay will also be future work.

Now consider the stability of the proposed scheme. Here we address the *total–stability* of the system [27].

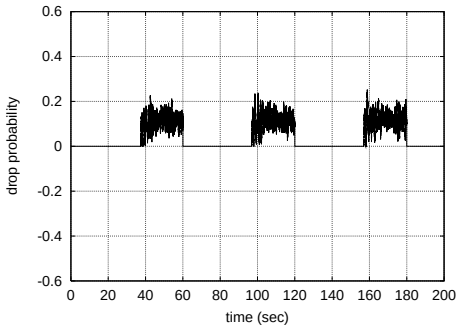
**Theorem 2.** The overall system is *totally stable* under Assumptions (A1) and (A2).



(a) Queue size



(b) Controller output



(c) Drop probability

Figure 4: Experiment I; PI control

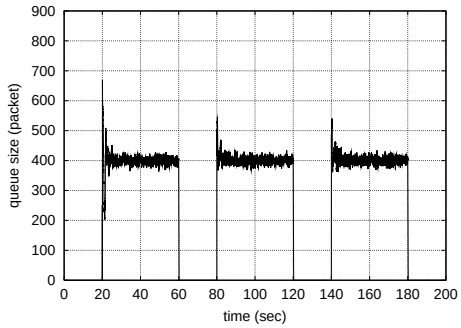
*Proof.* Suppose the overall system satisfy Assumptions (A1) and (A2). Then, from Theorem 2 in [23] the overall system is stable if $F - MH$ is Hurwitz. With the TCP/AQM system $F - MH = -k_i/k_p$ where $k_i$ and $k_p$ are positive constants. This completes the proof. □

## IV. Simulations

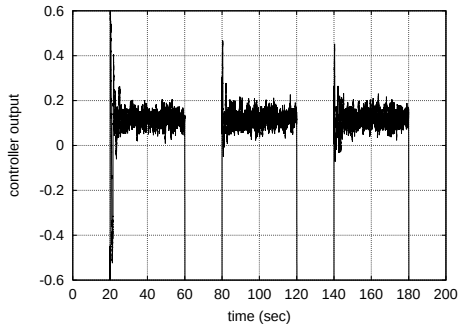
### A. Experiment I

In the first experiment, we evaluate the performance of recovery from an empty queue size. Because of the bursty nature of TCP flows, the flows can suddenly change from zero to maximum link capacity. This fluctuation is modeled by square waves of TCP flows. The number of TCP connections is 500, and these connections are alternatively on and off for 40 sec and 20 sec, respectively.

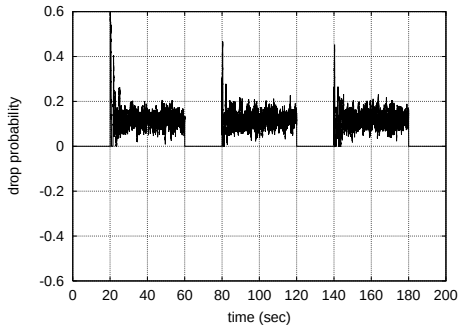
Figure 4 shows the result of the conventional PI control. Because TCP flows begin at t = 20 sec., there is no incoming packet in the router during the first 20 sec. Because the target queue size is 400 packets as given in Table I, and this queue size error makes the output of the PI controller become a large negative value during the first 20




(a) Queue size

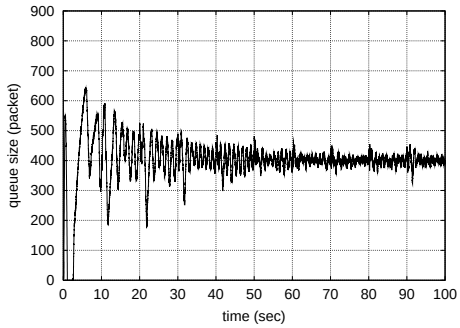



(b) Controller output

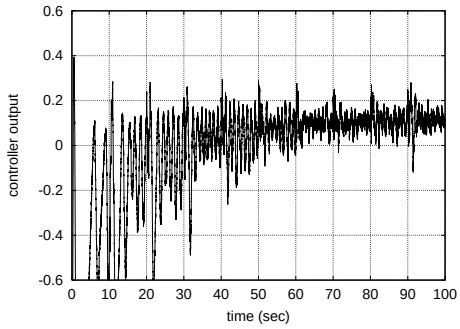



(c) Drop probability

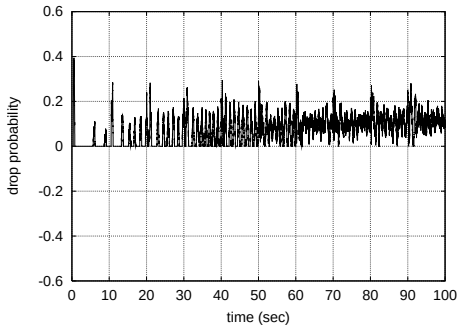
Figure 5: Experiment I; PI control with the proposed anti-windup compensator




(a) Queue size




(b) Controller output




(c) Drop probability

Figure 6: Experiment II; PI control

sec. As shown in Figure 4 (b), the controller output decreases linearly before the TCP flows begin, while the drop probability is limited to zero. After the flows begin at t = 20 sec., the controller output still remains negative for about another 20 seconds, and consequently the drop probability remains zero until about t = 40 sec. This results in a buffer overflow.

When we apply the proposed scheme to the PI controller, we can prevent the integral term from increasing during saturation period. Before TCP flows begin at t = 20 sec, the controller output is bounded above -0.6 as we can see in Figure 5 (b). The proposed anti-windup compensator helps the PI controller operate in the linear region after the underflow finishes. The anti-windup algorithm yields a short settling time at the cost of producing an overshoot of the queue size.

The experiment shows that a low TCP flow induces inevitable queue size error in the input of the PI controller, and this results in buffer overflow. This buffer overflow can be avoided by using an anti-windup method. The proposed scheme shows fast recovery from the empty queue size with the overshoot of queue size. In the following experiment, we will evaluate the effect of the slow response on the performance of AQM.

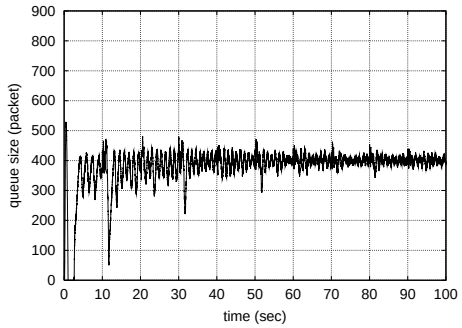
### *B. Experiment II*

In the second experiment, we dynamically increase the number of active TCP connections by 100 after each 10-second interval. The number of connections is increased from 100 to 1000.

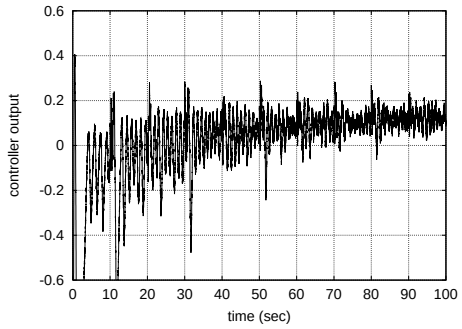
Figure 6 is the result of the conventional PI controller. The controller waits for the queue size to be higher than the target queue size with the drop probability being zero. When the queue size surpasses the target queue size, the controller tries to reduce the incoming rate by dropping or marking the incoming packets. Because the incoming rate is not high enough, the initial packet drop, whose probability is proportional to the queue size error, results in the empty buffer for about 1.5 seconds. After the under-utilization, the controller produces a large overshoot. The PI controller with the anti-windup compensator produces no overshoot after the initial under-utilization as shown in Figure 7. The experiment shows that PI controller with the proposed scheme performs better than the conventional PI controller.

## V. CONCLUSION

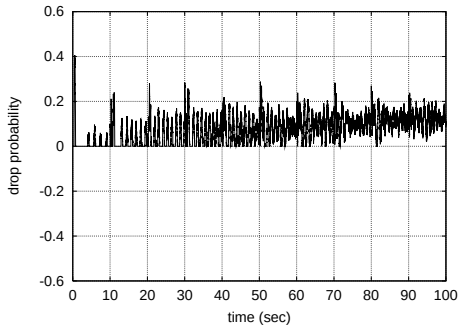
The conventional PI controller outperforms RED significantly. However, the drop probability is limited between 0 and 1, and this can degrade the performance of the conventional PI controller. To resolve




(a) Queue size




(b) Controller output




(c) Drop probability

Figure 7: Experiment II; PI control with the proposed anti-windup compensator

this problem, we have modeled AQM as a system with a saturating actuator and have applied an anti-windup scheme.

We compared the proposed scheme with the conventional PI controller via simulations using an ns-2 network simulator. The simulation results show that our proposed controller performs better than the PI controller. Here we adopted a static anti-windup scheme in [23] for simple implementation. A more advanced scheme, i.e. a dynamic compensation method was also proposed in [24][25][26]. We will apply this dynamic compensation method as an extension of this work. Also we ignored the delay in the control input for simplification. The delay in TCP network is crucial and we will also consider the delay for better perfromance. These remain as future work.

## REFERENCES

[1] S. Floyd and V. Jacobson, "On traffic phase effects in packet switched gateways," *Internetworking: Research and Experience*, vol. 3, pp. 115–156, 1992.

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

[3] B. Braden, "Recommendation on queue management and congestion avoidance in the Internet," *IETF RFC 2309*, 1998.

[4] M. May, J. Bolot, C. Diot, and B. Lyles, "Reasons not to deploy RED," in *Proceedings of IWQoS*, 1999, pp. 260–262.

[5] W. Feng, D. Kandlur, D. Saha, and K. Shin, "Blue: A new class of active queue management algorithms," *Tech. Rep., UM CSE-TR-387-99*, 1999.

[6] W. Feng, D. Kandlur, D. Saha, and K. Shin, "A self configuring RED gateway," in *Proceedings of IEEE INFOCOM*, 1999, vol. 3, pp. 1320–1328.

[7] T. J. Ott, T. V. Lakshman, and L. H. Wong, "SRED: Stabilized RED," in *Proceedings of IEEE INFOCOM*, 1999, vol. 3, pp. 1346–1355.

[8] H. Wang and K. Shin, "Refined design of random early detection gateways," in *Proceedings of GLOBECOM*, 1999, vol. 1b, pp. 769–775.

[9] S. D. Cnodder, O. Elloumi, and K. Pauwels, "RED behavior with different packet sizes," in *Proceedings of ISCC*, 2000, pp. 793–799.

[10] S. Athuraliya, S. H. Low, V. H. Li, and Q. Yin, "REM: Active queue management," *IEEE Network*, vol. 15, pp. 48–53, 2001.

[11] Q. Yin and S. H. Low, "Convergence of REM flow control at a single link," *IEEE Comm. Letters*, vol. 5, no. 3, March 2001.

[12] S. H. Low, F. Paganini, and J. Doyle, "Internet congestion control," *IEEE Control Systems Magazine*, Feb. 2002.

[13] C. V. Hollot, V. Misra, D. Towsley, and W. Gong, "A control theoretic analysis of RED," in *Proceedings of IEEE INFOCOM*, 2001, vol. 3, pp. 1510–1519.

[14] C. V. Hollot, V. Misra, D. Towsley, and W. Gong, "On designing improved controllers for AQM routers supporting TCP flows," in *Proceedings of IEEE INFOCOM*, 2001, vol. 3, pp. 1726–1734.

[15] J. Aweya, M. Ouellette, and D. Y. Montuno, "A control theoretic approach to active queue management," *Computer Networks*, vol. 36, pp. 203–235, 2001.

[16] J. Aweya, M. Ouellette, and D. Y. Montuno, "An optimization-oriented view of random early detection," *Computer Communications*, vol. 24, pp. 1170–1187, 2001.

[17] Y. Peng, D. Vrancic, and R. Hanus, "Anti-windup, bumpless and conditioned transfer techniques for PID controllers," *IEEE Control Systems*, vol. 16, pp. 48–57, 1996.

[18] K.J Astrom and L. Rundqwist, "Integrator windup and how to avoid it," in *Proceedings of ACC*, 1989, pp. 1693–1698.

[19] M.V. Kothare, P.J. Campo, M. Morari, and C.N. Nett, "A unified framework for the study of anti-windup designs," *Automatica*, vol. 30, pp. 1869–1883, 1994.

[20] J. Fall and K. Varadhan, "The ns manual," http://www.isi.edu/nsnam/ns/ns-documentation, 2001.

[21] G. F. Franklin, J. D. Powell, and A. Emami-Naeini, 3 Eds., *Feedback control of dynamic systems*, Addsion-Wesley, 1995.

[22] J.-K. Park and C.-H. Choi, "Performance enhancement of control systems with saturating actuators," *Trans. KIEE*, vol. 38, pp. 380–387, 1989.

[23] J.-K. Park and C.-H. Choi, "A compensation method for improving the performance of multivariable control systems with saturating actuators," *Control–Theory and Advanced Technology*, vol. 9, pp. 305–323, 1993.

[24] J.-K. Park and C.-H. Choi, "Dynamical anti-reset windup method for discrete-time saturating systems," in *Proceedings of CDC*, 1994, pp. 2926–2931.

[25] J.-K. Park and C.-H. Choi, "Dynamic compensation method for multivariable control systems with saturating actuators," *IEEE Trans. Automatic Control*, vol. 40, pp. 1635–1640, 1995.

[26] J.-K. Park and C.-H. Choi, "Dynamical anti-reset windup method for discrete-time saturating systems," *Automatica*, vol. 33, pp. 1055–1072, 1997.

[27] C.-T. Chen, 3 Eds., *Linear system theory and design*, Oxford University Press, 1999.