Application of a PID Feedback Control Algorithm for Adaptive Queue Management to Support TCP Congestion Control

Seungwan Ryu and Christopher M. Rump

Abstract: **Recently, many active queue management (AQM) algorithms have been proposed to address the performance degradation of end-to-end congestion control under tail-drop (TD) queue management at Internet routers. However, these AQM algorithms show performance improvement only for limited network environments, and are insensitive to dynamically changing network situations. In this paper, we propose an adaptive queue management algorithm, called *PID-controller*, that uses proportional-integral-derivative (PID) feedback control to remedy these weaknesses of existing AQM proposals. The PID-controller is able to detect and control congestion adaptively and proactively to dynamically changing network environments using incipient as well as current congestion indications. A simulation study over a wide range of IP traffic conditions shows that PID-controller outperforms other AQM algorithms such as Random Early Detection (RED) [3] and Proportional-Integral (PI) controller [9] in terms of queue length dynamics, packet loss rates, and link utilization.**

Index Terms: **Congestion control, PID feedback control, queue management, TCP.**

I. INTRODUCTION

The current Internet uses end-to-end congestion control to avoid congestion collapse. Responsive traffic sources such as transmission control protocol (TCP) sources probe the available bandwidth through window based flow control, and adjust their sending rate based on the congestion information they detect. However, in spite of the introduction of improvements on the end-system such as slow-start, fast retransmission, and fast recovery, the current Internet still experiences performance degradation in forms of large delays, multiple packet losses, and unfair bandwidth allocation.

There are two main reasons for this performance degradation. One is the limitation of end-system based approaches [1]. For instance, when congestion occurs in the network, end-systems will detect the congestion only after a considerable time lag in which a large number of packets will be dropped. The other source of performance degradation is the lack of support from the network. In the current Internet, first-in-first-out (FIFO) based tail-drop (TD) queue management is widely used at network routers because it is simple and easy to implement. Since the Internet traffic is inherently bursty, TD is overprovisioned with large buffers to absorb this burstiness. However, when faced with the onset of persistent congestion, TD will yield large delays and unfair buffer resource allocation due to the well-known drawbacks associated with TD, full-queue, and lock-out [2].

To overcome the performance degradation of the end-to-end TCP congestion control over TD queue management, many active queue management (AQM) schemes have been proposed. In [2], the Internet Research Task Force (IRTF) recommends the deployment of AQM as a router-based mechanism to support end-to-end congestion control. The idea behind AQM is to drop packets early before buffers overflow. By doing so, a router can inform sources early as to the expected congestion so that they can adjust their sending rate accordingly. Consequently, problems such as large delays and multiple packet losses can be avoided.

Two main functions are used in AQM: One is the congestion indicator (to detect congestion) and the other is the congestion control function (to avoid and control congestion). Random early detection (RED) [3] enhanced the control function by the introduction of probabilistic early packet dropping to avoid the full queue phenomena, global synchronization, and bias against bursty traffic. RED also enhanced the congestion indicator by introducing an exponentially-weighted moving average (EWMA) of the queue length not only to detect the incipient congestion but also to smoothen the bursty incoming traffic and its resulting transient congestion. Since RED was proposed in 1993, many AQM algorithms have been proposed including BLUE [4], Adaptive-RED [5], stabilized-RED (SRED) [6], random early marking (REM) [7], adaptive virtual queue (AVQ) [8], and proportional-integral (PI)-controller [9].

Although these AQM algorithms improve end-to-end TCP congestion control over simple TD, they are also revealed to have weaknesses [1], [10]–[13]. One of the main weaknesses is the difficulty in detection and control of the incipient congestion. Since these algorithms detect and control congestion *reactively* based on the current (i.e., instantaneous) or the historical (i.e., exponentially weighted moving average (EWMA)) queue occupancy, they are not able to detect and control the incipient congestion effectively. In addition, most AQM algorithms are designed to maintain the average queue length at a certain level to provide a satisfactory overall long-term behavior at a router.

However, many studies have shown different queue length dynamics between the actual queue and the average queue [6], [10], [11]. This phenomenon is due to the mismatch between the *microscopic* behavior of the actual queue length and the *macro-*

Manuscript received December 20, 2002; approved for publication by Thomas Hou. Division III Editor, November 1, 2003.

S. Ryu is with the Mobile Telecommunication Laboratory, Electronics and Telecommunications Research Institute(ETRI), 161 Gajong-Dong, Yusung-Gu, Taejon, Korea, email: rush@etri.re.kr.

C. M. Rump is with the Applied Statistics and Operations Research department at Bowling Green State University, Bowling Green, OH 43403, USA, email: cmrump@cba.bgsu.edu.

1229-2370/04/$10.00

scopic design goals of using the average queue length with a small weight for smoothing the total traffic. Moreover, these AQM algorithms have been designed under limited assumptions on the network traffic such as the existence of a fixed number of persistent *elephant* flows and a constant round-trip time (RTT) of sources. Therefore, most AQM algorithms are insensitive to the dynamically changing network traffic situations [1], [10], [14], and thus show successful performance improvement over TD only under restricted traffic environments.

To address these weaknesses, an AQM algorithm should be able to detect and control congestion *adaptively* to the changing network situations while giving satisfactory control performances such as a stabilized queue length, a bounded delay, and a high link utilization. In this paper, we first propose an adaptive queue management algorithm, called *PID-controller*, using proportional-integral-derivative (PID) feedback control. The goals of PID-controller are to control congestion proactively, to stabilize the queue length around a desired level, to give a smooth and low packet loss rates to each flow, and to maintain a high link utilization. PID-controller can detect and control congestion proactively based on the past (I), current (P), and future (D) congestion. We examine the adaptability and the robustness of AQM algorithms including PID-controller under a wide range of network traffic situations. Additionally, we evaluate the control performance of AQM algorithms in terms of many different performance metrics.

Recently, Ren *et al.* [15] proposed an AQM algorithm based on the concept of a PID feedback control. In [15], the frequency response method was used in designing an AQM algorithm whereas the time response method is used in our paper. In general, the choice of a design method depends on the designer's preference [16]. Since the control performance specifications such as the rise time, the settling time, the maximum overshot, and the steady state error are available analytically in time domain. PID-controllers are commonly designed using time domain methods when the plant model is a second order system [16]. We adopt the time domain method in designing of the PID-controller. In this paper, we design a PID-controller as an AQM algorithm using the time domain design method based on the linearized TCP flow dynamics as a plant dynamic model obtained in [17].

This paper is organized as follows. In Section II, we introduce a feedback control modeling approach for TCP/AQM dynamics. Then, using control theory, we analyze the TCP flow dynamics under TD and existing AQM algorithms such as RED, BLUE [4], and PI-controller. In doing so, we address the limitations of currently proposed AQM algorithms. In Section III, we propose the PID-controller designed using PID feedback control to not only have anticipatory congestion detection and control capability but also to achieve a short-term (i.e., transient) and a long-term (i.e., steady-state) control performance simultaneously. In Section IV, we verify the performance of PID-controller and compare with RED and PI-controller under various traffic environments via simulation using the `ns-2` simulator [18]. In Section V, we provide a summary of this study and suggest directions for future study.

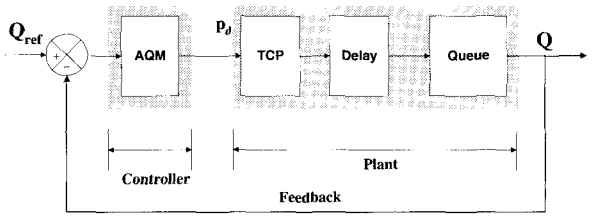

Fig. 1. Feedback control modelling of TCP congestion control with an AQM algorithm.

II. CONTROL THEORETIC MODELING OF TCP/AQM

A. Feedback Control and TCP Congestion Control with AQM

TCP congestion control dynamics with an AQM algorithm can be modeled as a feedback control system (Fig. 1). In this model, the feedback control system consists of: 1) A desired queue length at a router (i.e., *reference input*) denoted by Q_{ref}, 2) the queue length at a router as a plant variable (i.e., a *controlled variable*) denoted by Q, 3) a *plant* which represents a combination of subsystems such as TCP sources, routers, and TCP receivers that send, process, and receive TCP packets respectively, 4) an *AQM controller* which controls the packet arrival rate to the router queue by generating a packet drop probability, p_d, as a control signal, and 5) a *feedback signal* which is a sampled system output (i.e., queue length) used to obtain the error term, $Q_{ref} - Q$.

In [19], a system of nonlinear differential equations for the TCP/AQM dynamics was developed using fluid-flow analysis while ignoring the TCP time-out and slow-start mechanisms. In particular, differential equations for the behavior of the window size of source i, $W_i(t)$, and the equation for the queue length at a router, $q(t)$, were derived to be

$$\begin{aligned} \dot{W_i}(t) &= \frac{1}{R_i(q)} - \frac{W_i(t)W_i(t-\tau)}{2R_i(q(t-\tau))}p(x(t-\tau)), \\ \dot{q}(t) &\simeq -C + \sum_{i=1}^{N}\frac{W_i(t)}{R_i(q)}, \end{aligned} \tag{1}$$

where $p(x)$ is a packet drop function when the queue length is x, and $R_i(t) = a_i + q(t)/C$ is the round-trip time (RTT) of a TCP flow i at time t, where a_i is a fixed propagation delay, $q(t)/C$ is the queue delay, τ is a feedback delay, and C is the link capacity.

Then, in [17], a linearized[1] and simplified TCP/AQM dynamic model was developed and analyzed in terms of (feedback) control theory. There, the forward-path (open-loop) transfer function of the plant, $P(s)$, was given by

$$P(s) = P_{TCP}(s)P_{Queue}(s) = \left(\frac{\frac{R_0C^2}{2N^2}}{s+\frac{2N}{R_0^2C}}\right)\cdot\left(\frac{\frac{N}{R_0}}{s+\frac{1}{R_0}}\right), \tag{2}$$

where N is the load factor (i.e., the number of TCP connections), R_0 is the round trip time, and C is the link capacity. In general, an open-loop and a closed-loop transfer function of a

[1](1) was linearized about an operating point in equilibrium, (W_0, q_0, p_0), where $\dot{W}_0 = 0$, $\dot{q}_0 = 0$, and p_0 is a constant.

standard second-order system, $G(s)$ and $G_C(s)$, are

$$G(s) = \frac{\omega_n^2}{s^2 + 2\xi\omega_n s} \text{ and } G_C(s) = \frac{\omega_n^2}{s^2 + 2\xi\omega_n s + \omega_n^2}, \quad (3)$$

where ξ is the damping ratio and ω_n is the undamped frequency of the system [16].

B. Control Theoretic Analysis of Queue Management Algorithms

B.1 The TCP Flow Dynamics over Tail-Drop (TD)

The linearized TCP flow dynamics ($P(s)$) are shown stable when $N \geq N^-$ and $R_0 \leq R^+$ [17]. Since $P(s)$ has two non-zero poles, i.e., $s = -2N/(R_0^2 C)$ and $s = -1/R_0$, it is a type 0 system, so there always exists constant steady-state error to a step-function input [16]. Since $\lim_{s\to 0} P(s) = (R_0 C)^3/(4N^2)$, the steady-state error of the open-loop transfer function is $e_{ss} = M/(1 + (R_0 C)^3/(4N^2))$, where M is the reference input to eliminate this constant steady-state error.

■ Example 1: Sample network configuration

If we set $C = 3750$ packets/second, $N = 60$, and $R = 0.246$ seconds similar to the network configuration in [9] and [17], we find

$$\omega_r = 342.3 \text{ rad./sec.} \quad \text{and} \quad \xi = 4.5793/(2\omega_n) = 0.0067.$$

Then, the time domain performance specifications of the TCP flow dynamics are

- e_{ss} = the steady-state error = $M/(1 + \lim_{s\to 0} P(s)) = M/54985.1 > 0$,
- Maximum overshoot (MOS) (%) = $100e^{-\pi\xi/\sqrt{1-\xi^2}} = 97.92\%$,
- Rise time (t_r) $\cong 1.8/\omega_n = 0.00526$ sec.,
- Settling time (t_s) $\cong 4/(\xi\omega_n) = 1.75$ sec.

Fig. 2 shows a *MATLAB*® simulation of the unit-step response of the TCP flow dynamics under TD in this case. From the time domain performance specifications and the unit-step response, we conclude that because of the longer settling time relative to the very short rise time with very small damping ratio, $\xi = 0.0067$ (i.e., large maximum overshoot), the TCP flow dynamics under TD show a severely oscillating output signal.

In terms of control theory, TD queue management can be considered as an ON-OFF controller [1], where the control action, i.e., packet dropping, is activated when the buffer becomes full (ON phase) and is turned off otherwise (OFF phase). Since the ON-OFF controller does not take into account the magnitude of control action to correct the error, TD results in an oscillatory queue length [20]. In addition, there exists non-zero steady-state error with the TCP flow dynamics. Therefore, a well designed AQM controller should be able to compensate for the oscillatory TCP dynamics and give satisfactory control performance such as fast and stable control dynamics.

B.2 RED Controller

With RED [3], a link maintains the EWMA queue length, $Q_{avg} = (1 - w_Q) \times Q_{avg} + w_Q \times Q$, where Q is the current queue length and w_Q is a weight parameter, $0 \leq w_Q \leq 1$. When Q_{avg} is less than the minimum threshold (min_{th}), no packets are dropped (or marked). When it exceeds the maximum threshold (max_{th}), all incoming packets are dropped. When it is in between, a packet is dropped with a probability p_a that is increasing function with Q_{avg}.

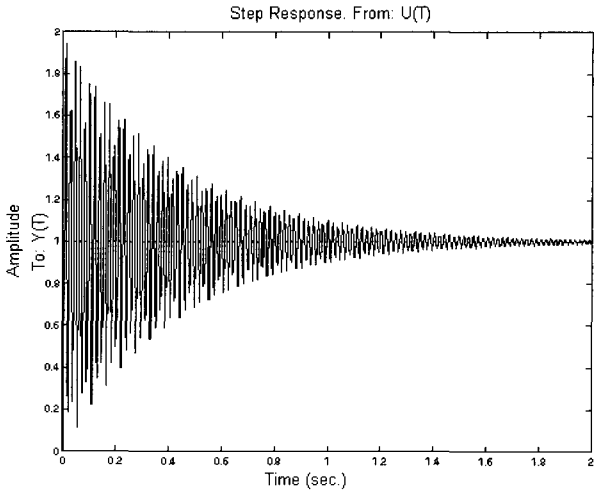

Fig. 2. Unit step response of the linearized TCP flow dynamics to the unit-step function.

RED attempts to eliminate the steady-state error by introducing the EWMA error terms[2] as an integral (I)-control to the TCP flow dynamics. However, since RED introduces a range of reference input values, i.e., $[min_{th}, max_{th}]$, rather than an unique reference input (or set-point) for I-control, the TCP/RED model shows oscillatory system dynamics. Moreover a very small recommended weight factor value ($w_Q = 1/512 \cong 0.002$) for the current queue length in calculation of the EWMA queue length is one of the main reasons for queue length oscillation and bias against bursty sources [12], [13]. In addition, the small value of w_Q brings an effect of large integral time in I-control, and may be accompanied by a large overshoot [16]. As a result, RED shows oscillatory queue length dynamics and gives poor performance under a wide range of traffic environments [1], [6]. In particular, RED has been shown to be unfair in bandwidth allocation [21] and unable to give low packet loss and high link utilization at the same time [5], [8].

B.3 BLUE

BLUE [4] was proposed to overcome the drawbacks of RED and other AQM algorithms that use the (EWMA) queue length as a congestion indicator. BLUE uses packet loss and link idle events as a congestion indicator and maintains a single packet drop probability, p, to control congestion. BLUE adjusts p in every fixed time interval, called the $freeze\_time$, when a packet loss or a link idle event occurs. In particular, BLUE increases p by a small amount of d_inc in response to the buffer overflow (i.e., packet loss) and decreases p by a small amount of d_dec when the link becomes idle. Thus, BLUE can be considered as

[2]The filtering of the queue length is equivalent to the filtering (integration) of the error terms.

a modified multi-positional ON-OFF controller or a *relay controller* with hysteresis $freeze\_time$ and two control outputs: If there is a packet loss (ON), the relay sends a control signal of d_inc units, and if there is a link idle (OFF), the relay sends a control signal of d_dec [22]. However, since BLUE adjusts p only after the packet loss or link idle event occurs, BLUE controls congestion reactively based on the current or past congestion information. Thus, some degree of performance degradation such as multiple packet losses and link under-utilization are not avoidable. Moreover, because of fixed values of the control parameters, d_inc, d_dec, and $freeze\_time$, BLUE is unable to provide adaptive control to the changing network traffic situations.

Fig. 3 shows concepts of TD, BLUE, and RED as a P-controller[3] in terms of feedback control.

B.4 Proportional-Integral(PI)-Controller

Proportional (P) control generates a control signal linearly proportional to the amount of the current queue length deviation from an unique desired queue length (i.e., the reference input) to minimize the control error. The P control signal at time t is

$$u(t) = K_P e(t) = K_P(Q(t) - Q_{ref}),$$

where K_P is a proportional gain. With P-control, the system can achieve fast response to the error. However, P-control tends to cause a large overshoot because of an inaccurate control signal and is not able to eliminate the steady-state error [16]. Thus, an AQM algorithm with P-control has been shown to have limitations for implementation [9]. For example, the control performance of AQM with P-control is very sensitive to the value of K_P, and thus easy results in queue length oscillation [9]. Thus, I-control is generally combined with P-control to eliminate the steady-state error and to improve oscillatory queue length dynamics simultaneously, albeit at the expense of responsiveness.

The Proportional-Integral (PI)-controller with a constant desired queue length has been proposed to overcome the drawbacks of RED and P-control based approaches [9]. Using the linearized TCP/AQM dynamics (2), PI-controller has been proposed not only to improve responsiveness of the TCP/AQM dynamics but also to stabilize the router queue length around Q_{ref}. The latter can be achieved by means of I-control, while the former can be achieved by means of P-control using the instantaneous queue length rather than using the EWMA queue length. The resulting PI-controller was suggested to be capable of eliminating the steady-state error regardless of the load level. The digitized packet drop probability at time $t = kT$ [9] is

$$p(kT) = p((k-1)T) + a\delta_Q(kT) - b\delta_Q((k-1)T), \quad (4)$$

where $f_s = 1/T$ is the sampling frequency, which is recommended to be $10 - 20$ times the link speed [9], a and b are constants, and $\delta_Q = Q - Q_{ref}$ is a deviation of the queue length from Q_{ref}. The recommended design rules [9] are

$$w_g = \frac{2N^-}{(R^+)^2 C}, \quad K_{PI} = w_g \left| \frac{(\frac{jw_g}{P_{queue}} + 1)}{\frac{(R^+C)^3}{(2N^-)^2}} \right|, \quad (5)$$

[3]In [9], RED was designed as a P controller.

where w_g is the unity-gain crossover frequency and K_{PI} is a PI-control gain.

The advantage of the above discretized PI-controller is that it requires less computation because the sampling frequency is much less than the link speed control of RED. However, since the congestion detection and control function of PI-controller is dependent on the current queue status or the history of the queue status (e.g., average queue length), PI-controller is *reactive* to current or past congestion not *proactive* to incipient congestion.

C. Limitations of Currently Proposed AQM Algorithms

Since each TCP source controls its sending rate through window size adjustment, the aggregate input traffic load (the offered load) at time $t \geq 0$, λ_t, is proportional to the total window size of all connections, W, i.e., $W \propto \lambda_t(R + Q_t/C)$, where R is the average propagation delay of all connections, Q_t/C is the queueing delay at a router, C is the output link capacity, and Q_t is the current queue length. Because of limited buffer size and output link capacity, the *carried traffic load*, λ'_t, will be a fraction of offered traffic load that is not dropped at a router, i.e., $\lambda'_t = \lambda_t(1 - P_d)$, where, P_d is the packet drop probability.

In a time-slotted model[4], Q_t is a function of λ'_t, i.e, $Q_t = (\lambda'_t - C)\Delta t + Q_{t-1}$, where Δt is the unit length of a time slot. However, the incipient congestion will be a function of the queue length of the next time slot, Q_{t+1}, not a function of Q_t. Therefore, to detect *incipient* congestion *proactively*, not *current* congestion *reactively*, P_d should be an increasing function of Q_{t+1} (or λ_{t+1} equivalently). Unfortunately, most AQM algorithms such as RED [3], REM [23], or PI-controller [9] use only the past traffic history such as Q_t (or the average queue length $\overline{Q}$) as a congestion indicator. As a result, these AQM algorithms are unable to detect incipient congestion adaptively to the traffic load variations.

III. PROPORTIONAL-INTEGRAL-DERIVATIVE(PID)-CONTROLLER

A. Managing the Future: Proportional-Integral-Derivative(PID) Control

To avoid congestion by detection of incipient congestion, an efficient mechanism that is able to predict and control expected error is required. The derivative (D)-control is capable of predicting future error and generating a predictive control action proportional to the rate of change of the error, i.e., $\frac{d}{dt}e(t)$ [16], [24]. Thus, the D-control signal at time t is

$$u(t) = K_D \frac{d}{dt} e(t),$$

where K_D is a derivative time (or the rate time). In addition, a fast response and better damping (i.e., less oscillation) can be achieved simultaneously with D-control [16], [20].

The proportional (P), integral (I), and derivative (D) feedback in the PID control is based on the *past (I)*, *current (P)*, and *future*

[4]In this model, time is divided into small time slots. At the end of each time slot, the queue size, Q_t, and total amount of queued input traffic, λ'_t, is calculated. For example, the simplest time-slotted model is a model that uses packet inter-arrival time as a unit time slot.

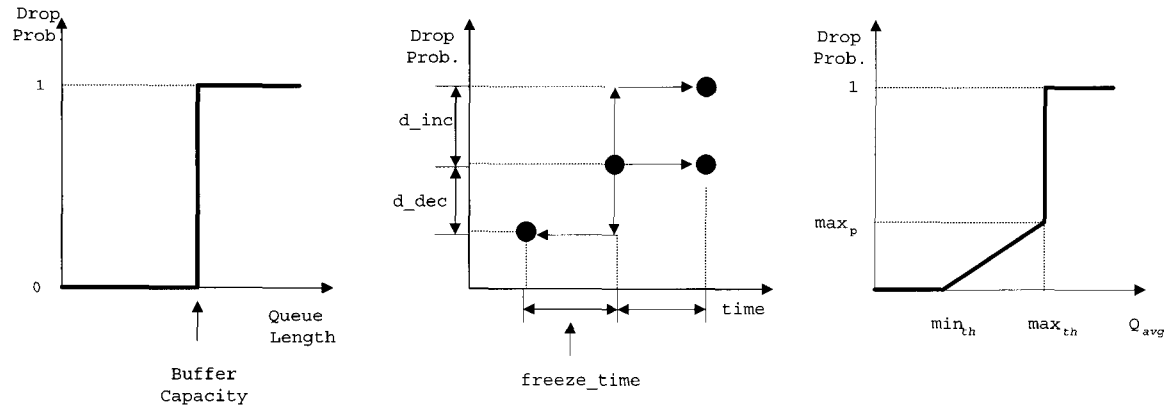

Fig. 3. TD, BLUE, and RED as a P controller.

D) control error [25]. PID control generates a control signal which is a linear function that combines the current error (P), the integral of previous errors (I), and the changing rate of current error (D). A generic control signal of PID control is

$$u(t) = K_P e(t) + K_I \int e(\tau)d\tau + K_D \frac{d}{dt}e(t), \qquad (6)$$

where K_P is a proportional gain, K_I is an integral gain[5], and K_D is the derivative time. With PID control, an AQM algorithm is able to detect and control the onset of persistent congestion, to control the current congestion, and to avoid the expected congestion effectively and proactively.

In this section, we first design a continuous PID-controller as an AQM algorithm based on the linearized TCP flow dynamics (2). The design procedure consists of two steps: 1) Design of a proportional-derivative (PD) control and 2) design of a proportional-integral (PI) control. Then, we digitize PID-controller for practical implementation at a router. Finally, we propose a PID control law and control parameters for digital implementation at a router.

B. Design of a PID-Controller

The PID-controller (6) can be expressed as a serial combination of a PD control and a PI control. PD control can improve damping (i.e., oscillation) and increase the rise time (i.e., the speed of response) of a control system, but can not eliminate the steady-state error. In contrast, PI control can reduce the steady-state error at the expense of a sluggish response. Since the controlled system, i.e., the TCP flow dynamic model (2), is a second-order system, time domain performance specifications such as the rise time, the settling time, the maximum overshoot, the time constant, and the steady-state error are available analytically [16]. Thus, we design PID-controller using the time-domain design and analysis method.

B.1 Performance Specifications

The time domain performance specifications can be classified into two categories: The *transient* performance specifications such as the maximum overshoot, the rise time, and the settling time, and the *steady-state* performance specification such as the steady-state error. In general, a faster speed of response of a system is accomplished by reduction of the *time constant*, T_C, which was suggested to be bounded by $R_0/2$ in [17] when assuming the existence of only persistent *elephant* FTP flows in equilibrium. However, to accomplish a fast decay of transient response in the presence of short-lived connections in size and lifetime (so called *mice*) [10], the bound for T_C should be smaller than $R_0/2$. Thus, we choose a proper value for T_C that provides satisfactory control dynamics in terms of both speed of response and steady-state error.

[5] $T_I = 1/K_I$ is the integral time (or the reset time).

B.2 Design Procedure of a PID Controller

Consider a PID-controller consisting of a PI control portion connected in serial with a PD control portion as shown in Fig. 4. Then, the PID control signal (6) in the Laplace transform is

$$\begin{aligned} D(s) &= K_P + K_D s + \frac{K_I}{s} = D_{PD}(s)D_{PI}(s) \\ &= (K_{P1} + K_{D1}s)\left(K_{P2} + \frac{K_{I2}}{s}\right), \end{aligned} \qquad (7)$$

where, $K_P = K_{P1}K_{P2} + K_{D1}K_{I2}$, $K_D = K_{D1}K_{P2}$, and $K_I = K_{P1}K_{I2}$. The PD control part is designed first. In general, proper K_{P1} and K_{D1} for a PD control are decided so that a portion of the desired *relative stability* is achieved [16]. In the time domain, the relative stability is achieved by obtaining a desired maximum overshoot. Next, we select the parameters K_{I2} and K_{P2} for the PI control part so that the performance specifications of the PID-controller can be achieved. Finally, we obtain a PID-controller with three PID control parameters, K_P, K_I, and K_D.

B.3 PD Control

PD control can give fast decay of transient response by maintaining a small time constant and damping ratio. Thus, for the design of a PD control the transient performance specifications

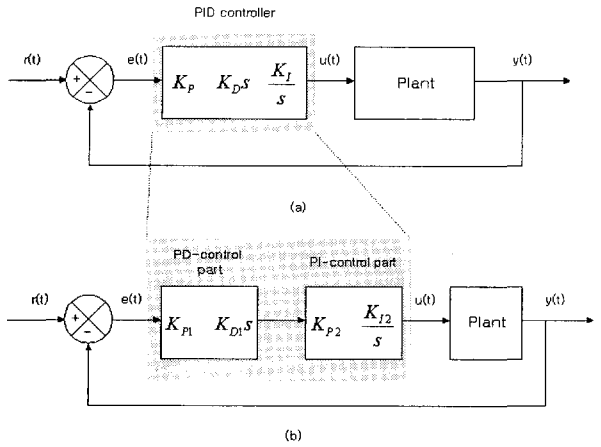

Fig. 4. PID control configurations for design: (a) a PID control configuration, (b) an equivalent PD-PI series control configuration.

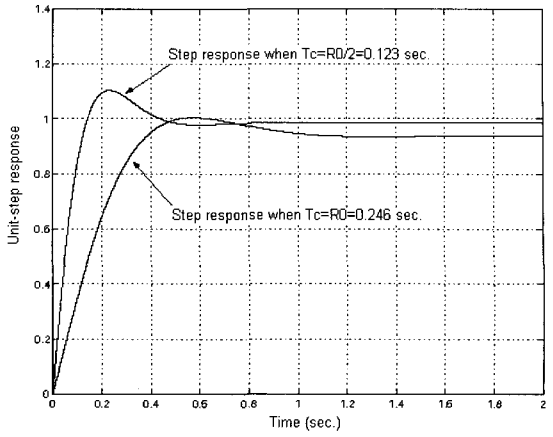

Fig. 5. Unit-step responses of a PD control system when $T_C = R_0 = 0.246$ sec. and $T_C = R_0/2 = 0.123$ sec.

are used. From (7), the transfer function of a PD control is

$$D_{PD}(s) = K_{P1} + K_{D1}s = K_{D1}\left(s + \frac{K_{P1}}{K_{D1}}\right).$$

Then, the open-loop and the closed-loop transfer function of a PD control system, $G(s)$ and $G_C(s)$, are

$$G(s) = D_{PD}(s)P(s) = \frac{\left(\frac{C^2}{2N}\right)K_{D1}\left(s + \frac{K_{P1}}{K_{D1}}\right)}{s^2 + \left(\frac{2N}{R_0^2C} + \frac{1}{R_0}\right)s + \frac{2N}{R_0^3C}}, \tag{8}$$

$$G_C(s) = \frac{G(s)}{1 + G(s)} \tag{9}$$

$$= \frac{\left(\frac{C^2}{2N}\right)K_{D1}\left(s + \frac{K_{P1}}{K_{D1}}\right)}{s^2 + \left(\frac{2N}{R_0^2C} + \frac{1}{R_0} + \frac{C^2}{2N}K_{D1}\right)s + \frac{2N}{R_0^3C} + \frac{C^2}{2N}K_{P1}}.$$

Since performance specifications of a PD control are functions of ξ and ω_n and (9) is a second-order system, PD control parameters, K_{P1} and K_{D1}, can be determined from the relation between characteristic equations of (9) and (3) for given acceptable bounds on the performance specifications.

$$q(s) = s^2 + \left(\frac{2N}{R_0^2C} + \frac{1}{R_0} + \frac{C^2}{2N}K_{D1}\right)s + \frac{2N}{R_0^3C} + \frac{C^2}{2N}K_{P1} = s^2 + 2\xi\omega_n s + \omega_n^2 = 0. \tag{10}$$

We first determine the desired maximum overshoot (i.e., the damping ratio ξ). Then, the system frequency (ω_n) is decided by fixing the desired speed of response in terms of the time constant $T_C = 1/(\xi\omega_n)$. Finally, K_{P1} and K_{D1} are obtained from ξ, ω_n, and (10).

■ Example 2:

Using the parameters of Example 1, we set the desired MOS to 5% (i.e., $\xi = 0.6901$ equivalently) and the settling time (t_s) to the time to reach and stay within $\pm 2\%$ of the steady-state value in the design of a PD control part for a given sample network configuration (Example 1). As shown in Fig. 5, the unit step response of a PD control system with $T_C = R_0/2$ shows better control dynamics than with $T_C = R_0$ in terms of speed of response and the steady-state error. Thus, we set $R_0/2$ as the time constant for design of the PD control part. Then, corresponding bounds of the transient performance specifications and PD control parameters are

- $\omega_n = 2/(R_0\xi) = 11.78$ rad/sec.,
- $t_r \simeq 0.153$ sec.,
- $t_s \simeq 0.492$ sec.,
- $K_{P1} = 1.166 \times 10^{-3}$,
- $K_{D1} = 9.97 \times 10^{-5}$.

B.4 PI Control

The next design procedure involves design of the PI control part based on the PD control system. From (7) and (8), the open-loop and the closed-loop transfer functions, $P_F(s)$ and $P_C(s)$, are

$$P_F(s) = G(s)D_{PI}(s) = \left(\frac{\left(\frac{C^2}{2N}\right)K_{D1}\left(s + \frac{K_{P1}}{K_{D1}}\right)}{\left(s + \frac{2N}{R_0^2C}\right)\left(s + \frac{1}{R_0}\right)}\right) \cdot \left(\frac{K_{P2}\left(s + \frac{K_{I2}}{K_{P2}}\right)}{s}\right)$$

$$= \frac{\left(\frac{C^2}{2N}\right)K_{D1}K_{P2}\left(s + \frac{K_{P1}}{K_{D1}}\right)\left(s + \frac{K_{I2}}{K_{P2}}\right)}{s\left(s + \frac{2N}{R_0^2C}\right)\left(s + \frac{1}{R_0}\right)}, \tag{11}$$

$$P_C(s) = \frac{P_F(s)}{1 + P_F(s)}, \tag{12}$$

where K_{D1} and K_{P1}/K_{D1} are constants obtained in the design of the PD control.

In (11), the dominant pole is $\frac{2N}{R_0^2C}$ because $\frac{2N}{R_0^2C} \ll \frac{1}{R_0}$ in general [17]. Since the TCP flow dynamics show severe oscillatory step response due to the dominant pole, proper location of the corner frequency, K_{I2}/K_{P2}, should be selected to reduce the transient of the TCP flow dynamics [26]. Thus, we select the location of K_{I2}/K_{P2} to cancel the dominant pole, $s = -\frac{2N}{R_0^2C}$, of

$P_F(s)$, i.e., by setting $K_{I2}/K_{P2} = (2N)/(R_0^2 C)$. Then, $P_F(s)$ and $P_C(s)$ become second-order systems.

$$P_F(s) = \frac{\left(\frac{C^2}{2N}\right) K_{D1} K_{P2} \left(s + \frac{K_{P1}}{K_{D1}}\right)}{s\left(s + \frac{1}{R_0}\right)}, \quad (13)$$

$$P_C(s) = \frac{\left(\frac{C^2}{2N}\right) K_{D1} K_{P2} \left(s + \frac{K_{P1}}{K_{D1}}\right)}{s^2 + \left(\frac{1}{R_0} + \frac{C^2}{2N} K_{D1} K_{P2}\right) s + \frac{C^2}{2N} K_{P1} K_{P2}}. \quad (14)$$

The PI control parameters can be obtained from the relation between characteristic equations of (14) and (3) via.

$$q(s) = s^2 + \left(\frac{1}{R_0} + \frac{C^2}{2N} K_{D1} K_{P2}\right) s + \frac{C^2}{2N} K_{P1} K_{P2} = s^2 + 2\xi\omega_n s + \omega_n^2 = 0. \quad (15)$$

Finally, the PID control parameters are

$$K_P = K_{P1} K_{P2} + K_{D1} K_{I2}, \quad K_D = K_{D1} K_{P2}, \text{ and } K_I = K_{P1} K_{I2} = 1/T_I. \quad (16)$$

■ **Example 3:**

Since $K_{P1} = 1.166 \times 10^{-3}$ and $\omega_n = 11.78$ rad./sec. for 5% of target MOS,

$$\omega_n^2 = 11.78^2 = \left(\frac{C^2}{2N}\right) K_{P1} K_{P2} = (117187.3)(1.166 \times 10^{-3}) K_{P2} = 136.7 K_{P2}.$$

Then, $K_{P2} = 1.02$ and $K_{I2} = 0.5258 \times K_{P2} = 0.534$. Finally, PID control parameters for 5% of target MOS are

$$K_P = 1.24 \times 10^{-3}, \quad K_D = 1.02 \times 10^{-4}, \text{ and } K_I = 6.23 \times 10^{-4}. \quad (17)$$

Fig. 6 shows unit-step response of the PID control system for 5% of target MOSs.

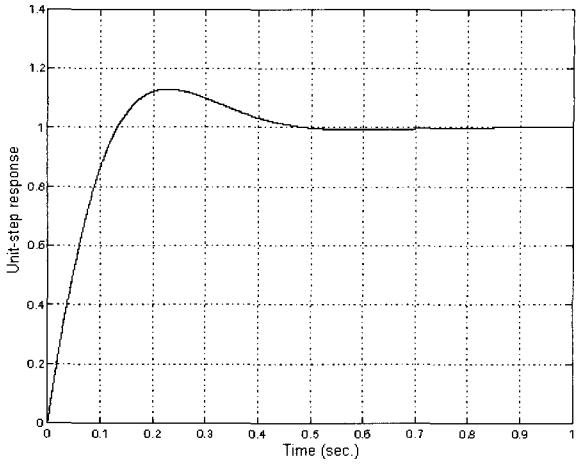

Fig. 6. Unit-step responses of the PID control system for 5% of target MOSs.

C. Digitization

Once continuous control systems are well established, we can take advantage of good continuous design features by finding an equivalent discrete controller. This method is called *emulation* [27]. First, we find the sampling frequency (f_s) analytically or empirically from the system frequency (f) of a continuous system. The sampling frequency is recommended to be $10 - 30$ times the system frequency [27]. Then, we find a digitized PID control equation using a digitization method. Finally, the digitized PID control equation is obtained using $\mathcal{Z}$-transform, i.e., *Tustin's method* for integration and the *backward* rectangle method for differentiation. Hence,

$$u_k = u_{k-1} + \Delta u_k = u_{k-1} + a_1 e_k - b_1 e_{k-1} + c_1 e_{k-2}, \quad (18)$$

where, $a_1 = \left(K_P + \frac{K_D}{T_s} + \frac{T_s}{2T_I}\right)$, $b_1 = \left(K_P + \frac{2K_D}{T_s} - \frac{T_s}{2T_I}\right)$, and $c_1 = \frac{K_D}{T_s}$.

D. Tuning of PID Control Parameters

Although we can explore the key factors governing the TCP/AQM dynamics through simplified theoretical analysis, this simplification may introduce substantial error [28]. In general, three external signals, the reference input (Q_{ref}), load disturbance, and measurement noise, affect a control system [20]. However, the simplified TCP dynamic model (2) does not include load disturbances such as the slow-start and time-out mechanisms. In detail, since (2) was developed under the assumption of existence of the long-lived FTP *elephant* flows only, we could see the effect of the load disturbance mainly caused from the mice flows under realistic traffic environments consisting of mice flows (2/3) and elephant flows (1/3). As a result, the analytically designed PID-controller generates a more aggressive control signal (i.e., aggressive packet drop probability) than necessary, and shows poor control performance under realistic IP traffic environments via `ns-2` [18] simulation. Thus, the PID control parameters (17) need to be tuned to generate proper control signals.

However, as mentioned in [15], we could not use existing tuning methods such as Ziegler-Nichols method for design of a PID controller. Since simulation is used not only to check the correctness of analytic approaches but also for allowing exploration of complicated network situations that are either difficult or impossible to analyze [28], PID control parameters can be tuned and a proper sampling frequency found empirically via extensive simulation studies. First, we use simple tuning methods for PID control parameters in which these parameters are tuned by multiplying a fractional value (called a tuning constant), κ, to make the PID-controller generate a less aggressive control signal. Second, a proper sampling frequency is chosen to provide better control performance with tuned control parameters. After extensive simulation study, we selected $\kappa = 0.05$ and $f_s = 10f = 29.5$ Hz as tuning constant and sampling frequency, respectively, for digital implementation.

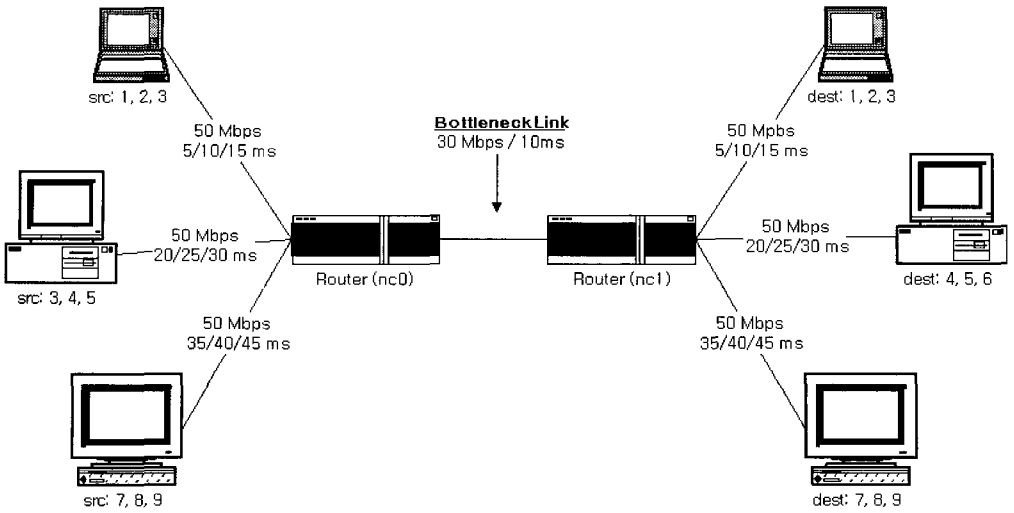

Fig. 7. Simulation network topology.

IV. SIMULATION STUDY

In this section, we examine the control performance of the PID-controller and other AQM algorithms such as RED and PI-controller via simulation study over a wide range of traffic environments using the `ns-2` [18] simulator.

A. Simulation Setup

We use a simple bottleneck network topology as shown in Fig. 7[6]. The network consists of two routers, `nc0` and `nc1`, with 9 TCP/Reno sources and 9 logically connected destinations. All TCP connections are connected to the routers, `nc0` and `nc1`, with link speeds of 50 Mbps. The propagation delay between TCP source i (`srci`) and `nc0` as well as between `nc1` and destination i (`desti`) is $5i$ ms, $i = 1, \cdots, 9$. The bottleneck link between `nc0` and `nc1` is assumed to have a link speed of 30 Mbps and a propagation delay of 10 ms. Thus, the logically connected source and destination pairs are connected with propagation delays ranging from 40 ms to 200 ms. All sources and destinations are assumed to use TD queue management with sufficient buffer capacity. The buffer at the bottleneck link uses an AQM algorithm and has capacity of 800 packets, which is about twice the bandwidth-delay product (BDP). With this network configuration, the capacity of bottleneck link C is set to 3750 packets/sec. and the round-trip time (RTT) ranges from 93.3 ms to 253.3 ms.

We consider two types of traffic flows: *Elephants* (long-lived FTP flows) and *mice* (short-lived flows). There are n elephant flows connected to each TCP source. Also, mice flows are connected to each TCP sources at the rate of $2n$ flows per connection with 3 seconds of average lifetime. Each packet is assumed to have an average size of 1000 bytes.

[6]In general, Internet flows go through many routers, and may experience congestion at some of those routers on the way from a source to a destination. However, if a congestion at a router is more severe than congestions at other routers, we can simplify the network topology into a simple dumbbell shape by assuming that there is a single congested router. In this simplification, congestion at other routers can be considered as additional propagation delays for each flow. Thus, in our simulation study, we used a simple dumbbell topology with propagation delays ranging from 40 ms to 200 ms (see Fig. 7).

Table 1. A summary of parameter setting of each AQM algorithms.

RED	$w_Q = 0.002, max_p = 0.1, max_{th} = 200,$ $min_{th} = 70$
PI-controller	$a = 1.822 \times 10^{-5}, b = 1.816 \times 10^{-5},$ $Q_{ref} = 200, f_s = 160$ Hz
PID-controller	$K_P = 6.2 \times 10^{-5}, K_D = 5.1 \times 10^{-5},$ $K_I = 3.12 \times 10^{-5}, f_s = 10f = 29.5$ Hz

We compare the control performance and the sensitivity of PID-controller with other AQM algorithms such as RED and PI-controller under packet drop mode. In RED, recommended parameter values [29] are used. In the PI-controller, we use the parameters, a, b, and T, used in [9]. The desired queue length (Q_{ref}) for PID-controller and PI-controller is set to 200 packets. A summary of the parameters setting of each AQM algorithm is shown in Table 1.

In [9], the system frequency of PI-controller, $f = \omega_g/(2\pi)$, where $\omega_g = 0.53$ rad./sec., has been derived analytically, and the sampling frequency (f_s) for implementation has been recommended at $10 \sim 20$ times the system frequency. For example, under the network configuration shown in Fig. 7 with the number of background flows, $N^- = 60$, the recommended sampling frequency is about $3 \sim 6$ Hz. However, a much faster sampling frequency, 160 Hz, was used in [9] for implementation of PI-controller via simulation. Through extensive simulation it turns out that PI-controller with a lower sampling frequency, i.e., $3 \sim 6$ Hz, shows poor control performance. Therefore, in this study, we use the high sampling frequency (160 Hz) to examine control performance of PI-controller.

B. Performance Metrics

B.1 The Queue Length

Control performance of an AQM controller can be measured by two measures: the *transient performance* (i.e., speed of response) and the *steady-state error control* (i.e., stability). We

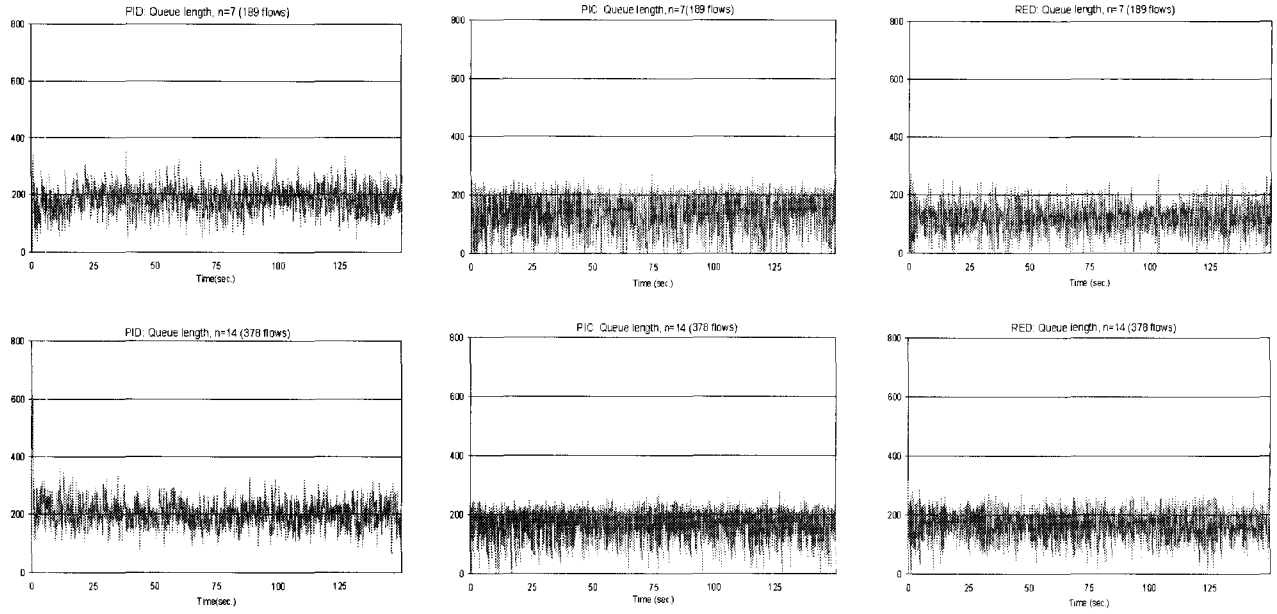

Fig. 8. The queue lengths of PID-controller, PI-controller, and RED under 189 (upper) and 378 (lower) flows.

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)* [30] defined as

$$S_e = \sqrt{\frac{1}{N+1}\sum_{i=0}^{N} e_i^2} = \sqrt{\frac{1}{N+1}\sum_{i=0}^{N}(Q_i - Q_{ref})^2}, \quad (19)$$

where Q_{ref} is the desired queue length, Q_i is the ith sampled queue length, $i = 1, \cdots, N$, and N is the number of sampling intervals.

B.2 The Packet Loss Rates

Since one of the goals of an AQM algorithm is to remove the bias against bursty sources, maintaining stable packet loss rates are important. High and bursty packet losses involve many packet losses at about the same time. If these packets belong to different flows, these flows experience losses at about the same time, and then experience global synchronization as a result. On the other hand, if these packets belong to bursty sources, there exists a bias against bursty sources. In general, the bias against bursty sources and the global synchronization can be eliminated effectively by achieving stable packet loss rates over time. In addition, to achieve higher throughput (or goodput) or to accommodate more traffic, maintaining low average packet loss rates is also important.

C. Control Performance of AQM Algorithms

In general, the control performance of an AQM algorithm is affected by several network components such as the buffer size (B), the link capacity (C), the traffic load factor (i.e., the number of flows (N)), and the round-trip time (RTT). Since the buffer size and the link capacity are fixed when an AQM is installed in a router, these parameters are *static* (i.e., time invariant) factors. In contrast, the traffic load factor (N) and the RTT are *dynamic* (i.e., time-varying) factors that change dynamically over time. Thus, it is important for an AQM algorithm to have adaptive and robust control performance to the dynamic factors. In this section, we examine the sensitivity of the control performance of PID-controller, PI-controller, and RED to the changes of the network environment, particularly on the dynamic factors, the traffic load factor (N), and the round-trip time (RTT).

C.1 The Effect of the Traffic Load Factor (N)

We first examine the control performance of PID-controller, PI-controller, and RED under two different traffic load levels, i.e., $n = 7$ (189 flows) and $n = 14$ (378 flows).

The queue length dynamics

Fig. 8 shows the queue length dynamics of PID-controller, PI-controller, and RED, respectively, under 189 flows (left) and 378 flows (right). PID-controller shows good control performance under two different traffic load levels in terms of the queue length dynamics staying around $Q_{ref} = 200$ packets. As shown in Fig. 8, PI-controller fails to maintain the queue length around Q_{ref}. Instead, the queue length stays below Q_{ref} most of time under both traffic load levels. Thus, PI-controller behaves like a tail-drop (TD) with buffer size of Q_{ref}. The main reason is the difference between properties of traffic assumed in analytical modeling, i.e., existence of only the elephant FTP flows, with those used in this simulation study. As shown in Fig. 8, RED maintains the queue length between $min_{th} = 70$ and $max_{th} = 200$ effectively under 189 flows. However, RED behaves like a TD with buffer size of Q_{ref} under 378 flows, similar to PI-controller.

Fig. 9 shows distributions of frequencies of the queue length of PID-controller, PI-controller, and RED under 189 and 378

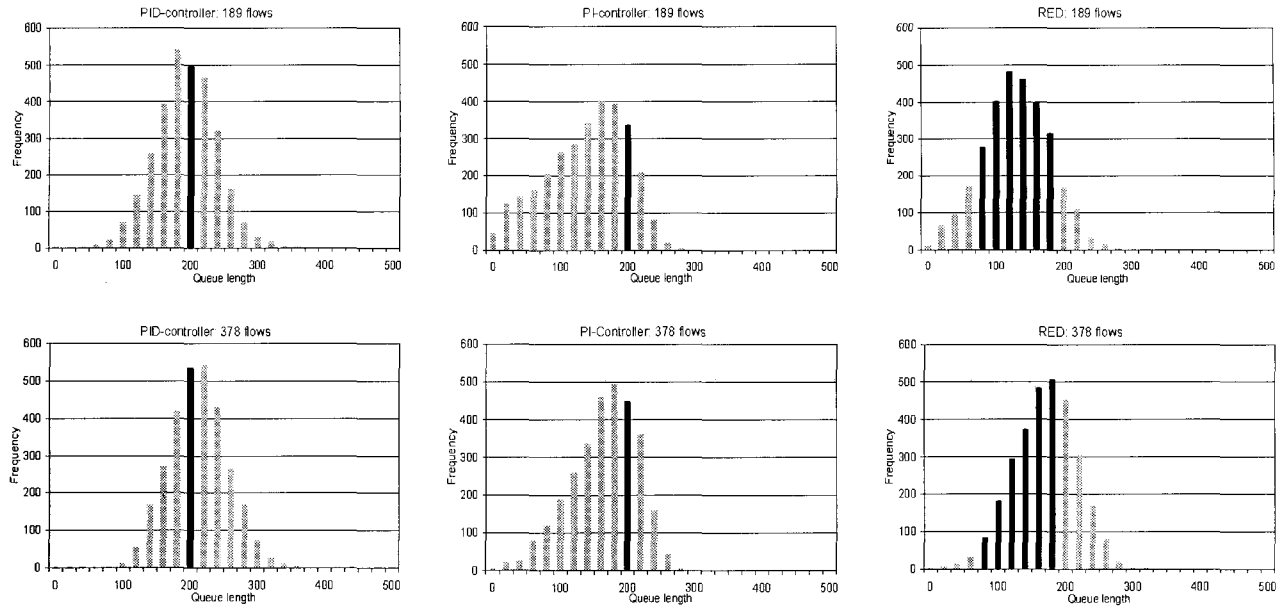

Fig. 9. Distributions of the queue lengths of PID-controller, PI-controller, and RED under 189 flows (upper) and 378 flows (lower).

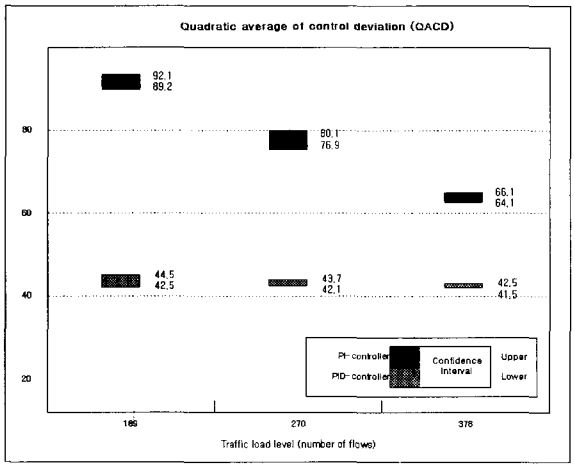

Fig. 10. The quadratic average of control deviation (QACD) of AQM algorithms for different traffic load levels.

flows. PID-controller shows invariant distributions of frequencies of the queue length $Q_{ref} = 200$, independent of the traffic load. Most of the queue lengths under RED are distributed between min_{th} and max_{th} for 189 flows, but are distributed around $max_{th} = 200$ for 378 flows. In PI-controller, most of the queue length are located below Q_{ref} under 189 and 378 flows.

The steady-state control performance of PID-controller and PI-controller can be evaluated in terms of the *QACD*[7] for three different traffic load levels 189, 270, and 378 flows. Table 2 shows the mean and variance of QACD for PID-controller and PI-controller under three different traffic load levels. As shown in Table 2, PID-controller shows robust steady-state control performance to changes of traffic load in terms of QACD. In contrast, the QACD of PI-controller is sensitive to traffic load, i.e., it is significantly reduced as the traffic load increases but higher than PID-controller. Fig. 10 shows 95% confidence intervals of QACD of PID-controller and PI-controller under three different traffic load levels.

[7]Since only PID-controller and PI-controller maintain unique desired queue length, Q_{ref}, QACD can be used for these two AQMs to compare the steady-state control performance.

Table 2. Summary of mean and variance of QACD of PID-controller and PI-controller under three different traffic load levels.

Number of (flows)	PID-controller		PI-controller	
	mean	variance	mean	variance
189	43.5	1.89	90.6	4.31
270	42.9	1.14	78.5	5.05
378	42.0	0.56	65.1	2.14

The packet loss rate

Fig. 11 shows the packet loss rates of PID-controller, PI-controller, and RED over time under the same traffic load levels used in Fig. 8. PID-controller shows stable and low packet loss rates over time, and can remove bias against bursty sources effectively as a result. As shown in Fig. 11, under 189 flows, RED shows stable and low packet loss rates whereas PI-controller shows high and severely fluctuating packet loss rates. However, both PI-controller and RED show fluctuating high packet loss rates under 378 flows. Thus, PI-controller and RED drop multiple packets, and may give a bias against bursty sources as a result.

Fig. 12 shows the average packet loss rate and the average link utilization of each AQM algorithm under different traffic load levels. PID-controller shows significantly lower average packet loss probabilities for all traffic load levels than other AQM algorithms. The average link utilization with PID-controller is higher than with RED and PI-controller for all traffic load levels.

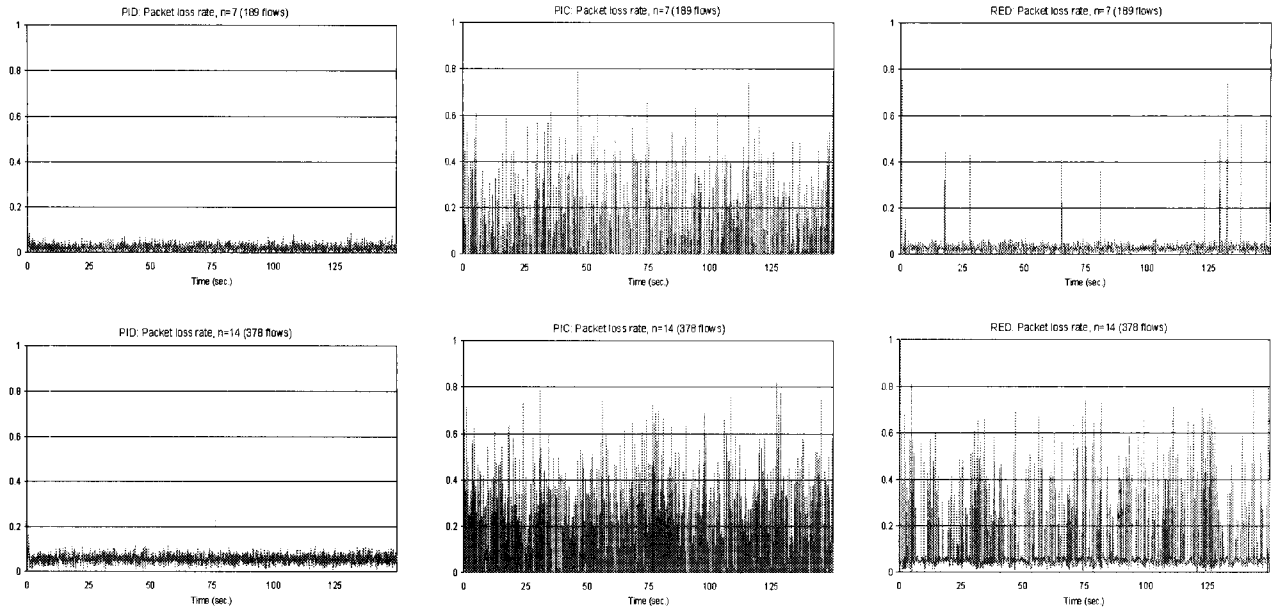

Fig. 11. The packet loss rate of PID-controller, PI-controller, and RED under 189 (left) and 378 flows (right).

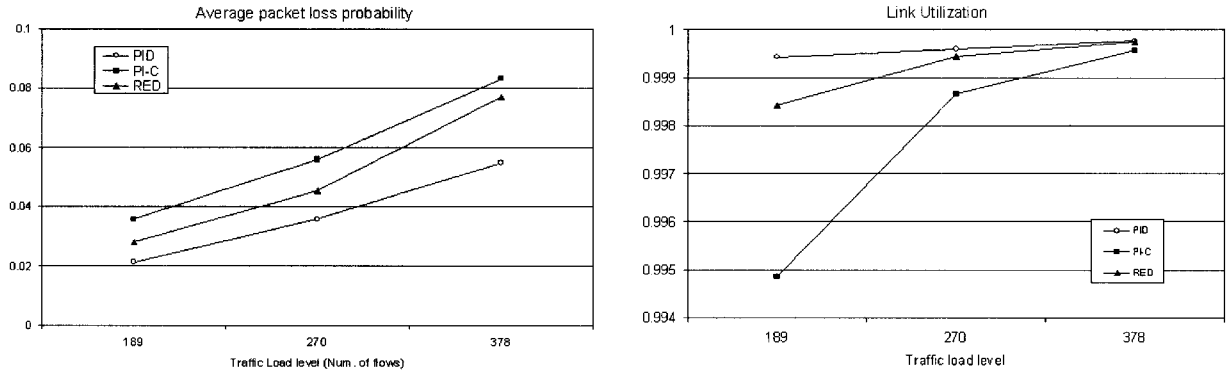

Fig. 12. The average packet loss probabilities and the link utilization of AQM algorithms under several different load levels.

C.2 Sensitivity to the Sudden Increased Traffic Load

In [31], it is shown that recently proposed AQM algorithms such as PI-controller perform poorly in situations where traffic consists of mostly web-like short-lived mice traffic and the traffic load varies over time. Thus, in this experiment, we examine the control performance of PID-controller, PI-controller, and RED under traffic situations where the traffic consists of 75% of mice flows and 25% of elephant flows, and the traffic load suddenly varies over time. The simulation starts with a light traffic load consisting of 3 sources at level $n = 10$ spawning 30 elephant and 90 mice flows. Thus, the initial total number of flows is 120. Then, an additional 3 sources, i.e., 30 elephant and 90 mice flows, are added at time 50.0 seconds. Thus, from time 50.0, the traffic consists of 6 sources with 240 flows (60 elephant and 180 mice). Finally, an additional 3 sources, i.e., 30 elephant and 90 mice flows, are added at time 100.0 seconds. Thus, from time 100.0, traffic consists of 9 sources with 360 flows (90 elephant and 270 mice). Fig. 13 summarizes the traffic load over time in this experiment.

Fig. 14 shows the queue length dynamics and the packet loss rates of PID-controller, PI-controller, and RED under the traffic scenario shown in Fig. 13. The queue length of PID-controller stays around Q_{ref} except during the time interval [0.0, 50.0] seconds because the number of elephant (FTP) flows (30) in this time interval is smaller than the lower bound ($N^- = 60$) assumed in (2). PID-controller shows fast response to the sudden increase of the traffic load. The queue length of PI-controller stays below Q_{ref} and fluctuates over time. In particular, under a light traffic load (i.e., in time interval $[0.0, 50.0]$ sec.), PI-controller shows severely fluctuating queue length dynamics. It also shows under-utilization of the link by allowing the link to idle occasionally. Moreover, PI-controller allows multiple packet losses over time, which become more frequent as the traffic load increases. RED maintains the queue length within the range between $min_{th} = 70$ and $max_{th} = 200$. RED is able to give stable and low packet loss rates under light traffic. However, as the traffic load increases, RED allows multiple packet losses more frequently.

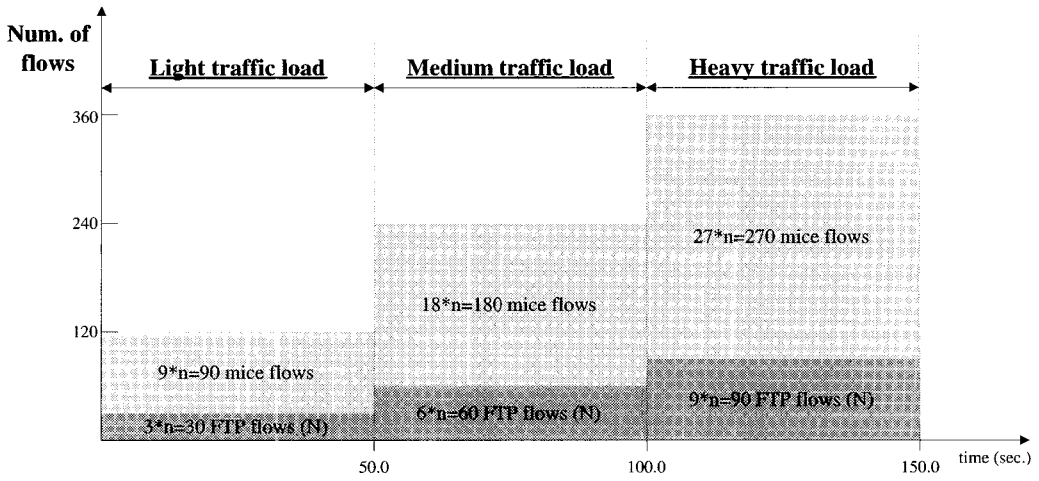

Fig. 13. A traffic scenario of varying traffic load.

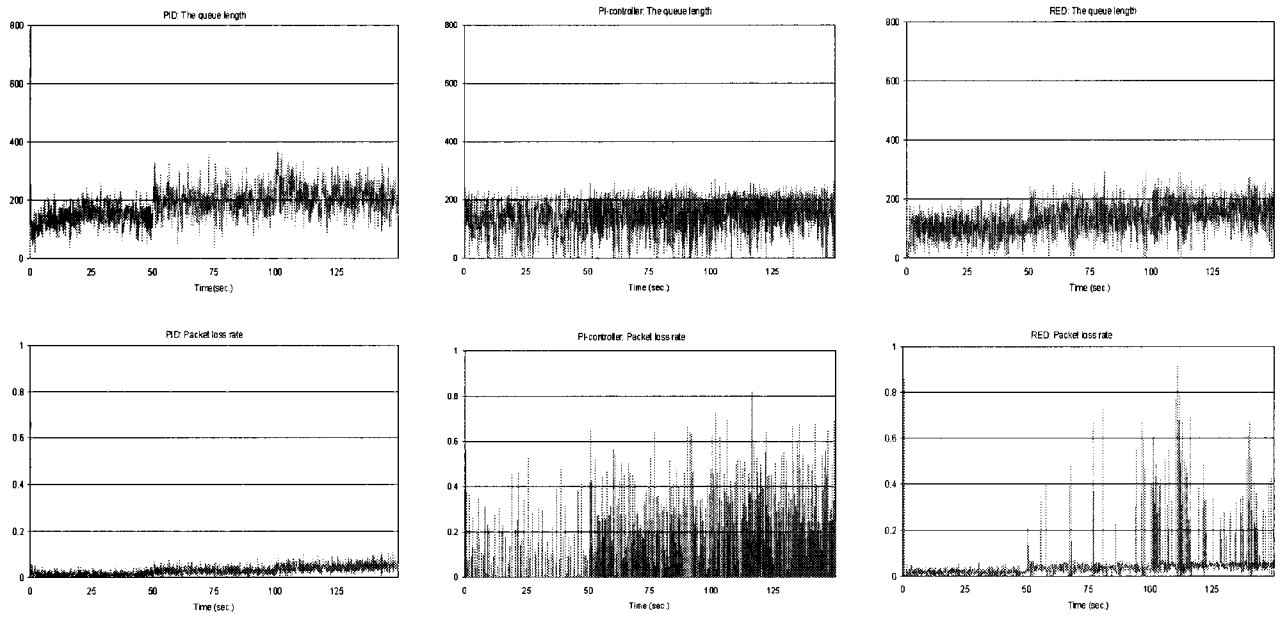

Fig. 14. The queue length and the packet loss rates of PID-controller, PI-controller, and RED over time.

Fig. 15 shows the average packet loss rates of the PID-controller, PI-controller, and RED algorithms for the three different time intervals, $[0.0, 50.0]$, $[50.0, 100.0]$, and $[100.0, 150.0]$. The average packet loss rates of PID-controller is lower than PI-controller and RED in all time intervals. Therefore, for the same average packet loss rates, the network can accommodate more traffic with PID-controller than with PI-controller and RED.

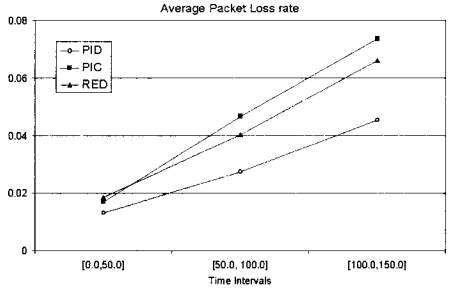

Fig. 15. The average packet loss probability in each time interval.

C.3 The Effect of the Round-Trip Time (RTT)

An increase of RTT not only degrades the control performance of an AQM algorithm but also leads the system to instability. Thus, the effect of RTT should be taken into account for robust design of an AQM, especially under a wide area networks (WAN) environment. To examine the effect of RTT on the control performance of AQM algorithms, we increase the link delay between routers `nc0` and `nc1` from 10 ms to 60 ms. Thus, RTTs

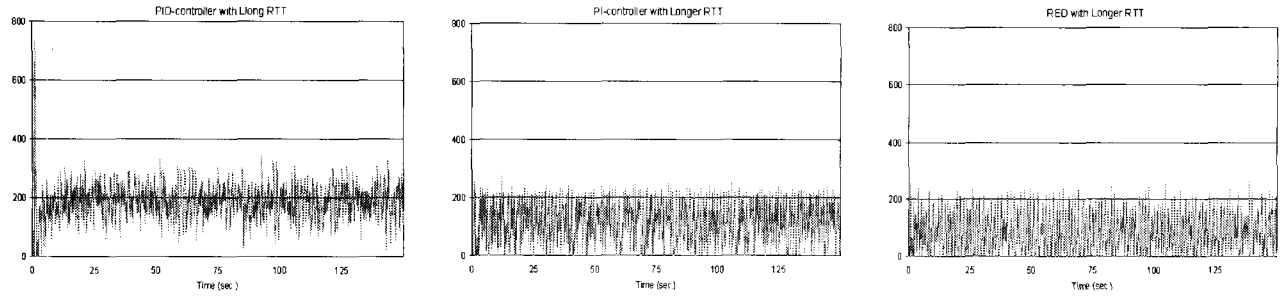

Fig. 16. The queue lengths of PID-controller, PI-controller, and RED with 378 flows under increased RTT.

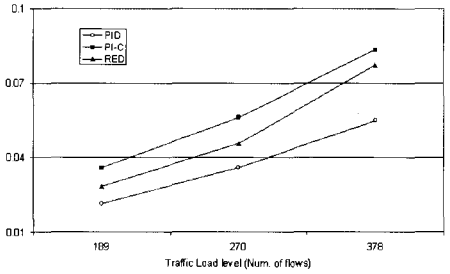

Fig. 17. The average packet loss rates of PID-controller, PI-controller, and RED under increased RTT.

of all sources are increased by 100 ms from [93.3, 253.3] ms to [193.3, 353.3] ms.

Since the pipe size of the round-trip path between sources and destinations (i.e., the network capacity) is increased due to the increase of RTT, we examine the control performance of AQM algorithms under 378 flows. Fig. 16 shows the queue length dynamics of PID-controller, PI-controller, and RED in this case. As shown in Fig. 16, PID-controller shows stable queue length dynamics after the increase in RTT while having slightly larger deviation from Q_{ref} than under the smaller RTT (Fig. 8). In contrast, the queue length dynamics of PI-controller and RED show very sensitive control performance to the RTT when compared to the queue dynamics shown in Fig. 8.

Fig. 17 shows the average packet loss rates of the PI-controller and RED algorithms to the increased RTT under different traffic load levels. PID-controller shows the lowest average packet loss rate for all traffic load levels, and difference of the average packet loss rate between PI-controller/RED and PID-controller becomes larger as the traffic load increases.

V. CONCLUSIONS AND FURTHER STUDY ISSUES

In this paper, we argue that, under dynamically changing traffic loads, an AQM algorithm should be able to detect and control congestion *proactively* based on incipient congestion, not *reactively* only based on the current congestion. Thus, to overcome the reactive congestion control of existing AQM algorithms, we designed a predictive and robust AQM algorithm, called PID-controller, using classical PID feedback control using the simplified TCP flow dynamics model (2) as a plant model. PID-controller showed robust and adaptive congestion control performance under various traffic situations via extensive `ns-2` [18] simulation studies. In particular, PID-controller showed stable queue length regulation around a desired level while giving low and smooth packet loss rates. PID-controller outperformed other AQM algorithms such as RED and PI-controller in terms of the queue length dynamics, the packet loss rates, and the link utilization.

In practical implementation, PID-controller has comparably less computational overhead than other AQM algorithms such a RED and PI-controller. In terms of the sampling overhead, the EWMA queue length and the packet drop probability are calculated at every packet arrival with RED. In contrast, PID-controller can be easily implemented with a smaller sampling frequency as compared to the link speed (i.e., 3750 Hz) implementation of RED and 160 Hz of PI-controller. In addition, PID-controller maintains small number of parameters that can be tuned easily and which give robust control performance to the dynamically changing network environment. In RED, for instance, more parameters such as w_Q, min_{th}, max_{th}, and max_p are maintained, and tuning of these parameters under dynamically changing traffic situations is extremely difficult [1], [6], [13]. Therefore, the computational complexity and overhead at a router can be reduced significantly with PID-controller.

There are some issues for further study however. First, the traffic process assumed in the analytic design of TCP flow dynamics (2) is different from complicated real IP traffic. Thus, the control performance of PID-controller under realistic IP traffic environments has been shown quite different from the control performance designed analytically. Therefore, we are trying to find a more precise TCP flow dynamic model than (2) to reduce the effect of the load disturbance and the measurement noise. The standard approach to control system design is to develop a linear model of the process for an operation condition and to design a controller having constant parameters. However, the control parameter for an operating point will not be optimal after some time because the network traffic is changing dynamically over time [14]. To take dynamically changing network traffic into account, we are trying to apply *adaptive control* techniques such as the gain scheduling and the auto-tuning PID-controller [26], [32] in the design of an AQM algorithm.

REFERENCES

[1] S. Ryu, C. Rump, and C. Qiao, "Advances in internet congestion control," *IEEE Commun. Surveys*, vol. 5, pp. 28–39, 2003.

[2] B. Braden *et al.*, "Recommendations on queue management and congestion avoidance in the internet," *IETF RFC2309*, Apr. 1998.

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

[4] W. Feng *et al.*, "The BLUE active queue management algorithms," *IEEE/ACM Trans. Networking*, vol. 10, pp. 513–528, Aug. 2002.

[5] W. Feng *et al.*, "A self-configuring RED gateway," in *Proc. INFOCOM'99*, (New York, NY), Mar. 1999, pp. 1320–1328.

[6] T. J. Ott, T. V. Lakshman, and L. Wong, "SRED: Stabilized RED," in *Proc. INFOCOM'99*, (New York, NY), Mar. 1999, pp. 1346–1355.

[7] S. H. Low and D. Lapsley, "Random early marking: An optimization approach to internet congestion control," in *Proc. IEEE ICON'99*, (Brisbane, Australia), 1999, pp. 67–74.

[8] S. Kunniyur and R. Srikant, "Analysis and design of an adaptive virtual queue (AVQ) algorithm for active queue management," *Tech. Rep.*, UIUC, Feb. 2001.

[9] C. V. Hollot *et al.*, "On designing improved controllers for AQM routers supporting TCP flows," in *Proc. INFOCOM 2001*, (Anchorage, AK), Apr. 2001, pp. 1726–1734.

[10] M. Christiansen *et al.*, "Tuning RED for web traffic," *IEEE/ACM Trans. Networking*, vol. 9, pp. 249–264, June 2001.

[11] M. May *et al.*, "Influence of active queue parameters on aggregate traffic performance," *Tech. Rep.*, INRIA-$n^0$3995, 2000.

[12] A. Misra, T. Ott, and J. Baras, "Effect of exponential averaging on the variability of a RED queue," in *Proc. IEEE ICC 2001*, June 2001, pp. 1817–1823.

[13] T. Ziegler, "On averaging for active queue management congestion avoidance," in *Proc. ISCC 2002*, (Naxos, Italy), July 2002, pp. 867–873.

[14] R. Fengyuan *et al.*, "A robust active queue management algorithm based on sliding mode variable structure control," in *Proc. INFOCOM 2002*, (New York, NY), June 2002, pp. 13–20.

[15] F. Ren and C. Lin, "Speed up the responsiveness of active queue management system," *IEICE Trans. Commun.*, vol. E86-B, pp. 630–636, 2003.

[16] B. C. Kuo, *Automatic Control Systems*, 7th ed., John Wiley & Sons, Inc., 1995.

[17] C. V. Hollot *et al.*, "A control theoretic analysis of RED," in *Proc. INFOCOM 2001*, (Anchorage, AK), Apr. 2001, pp. 1510–1519.

[18] S. McCanne and S. Floyd, "Network simulator - ns (version 2)," http://www.isi.edu/nsnam/ns, 1996.

[19] V. Misra, W. Gong, and D. Towsley, "Fluid-based analysis of a network of AQM routers supporting TCP flows with an application to RED," in *Proc. ACM SIGCOMM 2000*, (Stockholm, Sweden), Sept. 2000, pp. 151–160.

[20] K. Åstrom and T. Hagglund, *PID Controllers: Theory, Design, and Tuning*, 2nd ed., Instrument Society of America, 1995.

[21] D. Lin and R. Morris, "Dynamics of random early detection," in *Proc. ACM SIGCOMM'97*, (Cannes, France), Sept. 1997, pp. 127–137.

[22] D. I. Wilson, "Advanced control," http://www.ee.kau.se/forskning /ModSim/, Oct. 2001.

[23] D. E. Lapsley and S. H. Low, "Random early marking for internet congestion control," in *Proc. GLOBECOM'99*, (Rio de Janeiro, Brazil), Dec. 1999, pp. 1747–1752.

[24] G. Franklin, J. Powell, and A. Emami-Naeini, *Feedback Control of Dynamic Systems*, 3rd ed., Addison-Wesley Publishing Co., 1995.

[25] K. Åstrom and T. Hagglund, "The future of PID control," *Control Engineering Pratice*, vol. 9, pp. 1163–1175, 2001.

[26] K. Åstrom *et al.*, "Automatic tuning and adaptation for PID controllers - A survey," *Control Engineering Pratice*, vol. 91, pp. 699–714, 1993.

[27] G. Franklin, J. Powell, and M. Workman, *Digital Control of Dynamic Systems*, 3rd ed., Addison-Wesley Publishing Co., 1998.

[28] S. Floyd and V. Paxson, "Difficulties in simulating the internet," *IEEE/ACM Trans. Networking*, vol. 9, pp. 392–403, Aug. 2001.

[29] S. Floyd, "Notes on testing RED implementation," http://www.icir.org /floyd/papers/redtesting, 1996.

[30] R. Isermann, *Digital Control Systems Volumn I: Fundamentals, Deterministic Control*, 2nd Revised, Springer-Verlag, 1989.

[31] S. Floyd and E. Kohler, "Internet research needs better models," in *Proc. First Workshop on Hot Topics in Networks (HotNets-I)*, (Princeton, NJ), http://www.acm.org/sigcomm/HotNets-I, Oct. 2002.

[32] K. Åstrom and T. Wittenmark, *Adaptive Control*, 2nd ed., Addison-Wesley Publishing Co., 1995.

Seungwan Ryu received a B.S. and a M.S. degrees from Korea University, Seoul, Korea, in 1988 and 1991, respectively, and Ph.D. from University at Buffalo (SUNY), Buffalo, NY, USA, in 2003, in Industrial Engineering with a concentration in Operations Research. He joined the Electronics and Telecommunications Research Institute (ETRI), Teajon, Korea, in 1993, and he is currently a senior member of engineering staff at the Mobile Telecommunications Research Laboratory, ETRI, Teajon, Korea. His research interests include modeling and analysis of communication network performance, modeling and control of the Internet traffic, and design and analysis of beyond the third generation (B3G) and the fourth generation (4G) wireless communication systems.

Christopher M. Rump is an Assistant Professor in the Department of Applied Statistics and Operations Research within the College of Business Administration at Bowling Green State University (BGSU). He received a B.S. in applied mathematical sciences from Texas A&M University and a Ph.D. in operations research from The University of North Carolina at Chapel Hill. He served as an Assistant Professor of Industrial Engineering at the University at Buffalo (SUNY) before joining BGSU. Professor Rump's research interests lay mainly in applied probability modeling of computer & communication networks, transportation networks, and other service systems. He has published nearly twenty refereed articles in a variety of journals. Professor Rump is an active member of the Institute for Operations Research and the Management Sciences (INFORMS) and treasurer for Omega Rho, the international honor society for operations research and management science.