Available at
www.ElsevierComputerScience.com
POWERED BY SCIENCE DIRECT®

Computer Networks 44 (2004) 17–41

COMPUTER NETWORKS

www.elsevier.com/locate/comnet

Analysis and design of the virtual rate control algorithm for stabilizing queues in TCP networks ☆

Eun-Chan Park, Hyuk Lim, Kyung-Joon Park, Chong-Ho Choi *

School of Electrical Engineering and Computer Science, Seoul National University, Seoul 151-742, South Korea

Received 24 October 2002; received in revised form 8 April 2003; accepted 11 June 2003

Responsible Editor: M. Ajmone Marsan

Abstract

The virtual rate control (VRC) algorithm has been proposed for active queue management (AQM) in TCP networks. VRC, a rate-based control mechanism, responds quickly to traffic changes, thus allowing for high utilization and small loss. It can effectively stabilize both the input rate and the queue length around their target levels. In this paper, we analyze the stability of the VRC algorithm based on a linearized TCP model with time delay and provide a design guideline for parameter setting to make the overall system stable. Finally, we confirm the validity of our analysis and the effectiveness of VRC compared to RED, PI, REM, and AVQ through extensive *ns*-2 simulations.
© 2003 Elsevier B.V. All rights reserved.

Keywords: Active queue management; TCP networks; Congestion control; Queue regulation; Rate-based control

1. Introduction

A considerable amount of research has been carried out on TCP congestion control mechanisms, in order to enhance the performance of the Internet. Several TCP congestion control mechanisms, based on the congestion window (*cwnd*), have been proposed such as *slow-start, congestion avoidance, fast-recovery*, and *fast-retransmit* [1]. By adjusting the size of *cwnd*, TCP sender controls the number of in-flight packets that have been sent and not yet acknowledged. The key idea behind TCP congestion control is to probe the available bandwidth on the network and to adjust the transmission rate accordingly. To achieve this objective, TCP adopted the additive increase multiplicative decrease (AIMD) mechanism [1,2]. By additively increasing its *cwnd* until detecting packet loss, which is regarded as an implicit notification of congestion, TCP sender probes available bandwidth and tries to use the bandwidth most effectively. When it detects packet loss, TCP sender decreases its *cwnd* multiplicatively, and reduces the transmission rate. Hence, TCP is a *best-effort* service. However, current TCP congestion control mechanisms with drop-tail queues have some drawbacks, such as global synchronization of flows and low utilization of network resources [3]. A TCP sender reduces its transmission rate only after detecting packet loss, which results from

☆ This work was supported in part by POSCO, South Korea.
* Corresponding author.
E-mail addresses: ecpark@csl.snu.ac.kr (E.-C. Park), hyuklim@csl.snu.ac.kr (H. Lim), kjpark@csl.snu.ac.kr (K.-J. Park), chchoi@csl.snu.ac.kr (C.-H. Choi).

overflow of the drop-tail queue that the packet traverses. Therefore considerable time may have passed between the packet drop and its detection because of round-trip time (RTT) delay. A large number of packets may be dropped, because sources continue to transmit at a rate that the network cannot support.

To alleviate the problems associated with current TCP congestion control algorithms which use drop-tail queues, active queue management (AQM) has been proposed. While drop-tail queues are passive and reactive to congestion, queues under AQM are proactive to congestion. To avoid global synchronization, AQM drops packets randomly before buffer-overflow occurs, not only on the event of buffer-overflow. And the probability of random drop is proportional to the queue size because as the queue size increases, there is more possibility of buffer-overflow. The basic rationale of AQM is to provide advance notification of congestion to the sender, so that the sender can reduce its transmission rate before buffer overflow occurs. It is also imperative for the AQM algorithm to properly regulate the queue for high utilization and consistent queuing delay.

One of the most prevalent AQM algorithms is RED [3]. RED can prevent global synchronization, reduce packet loss rates, and minimize bias against bursty sources. However, obtaining appropriate values of RED parameters that show good performance under different network scenarios remains an inexact science [4–6].

To overcome the shortcomings of RED and to design a better AQM controller, the proportional–integral (PI) control algorithm [7], random exponential marking (REM) [8], and dynamic random early detection (DRED) [9] methods have recently been proposed. The PI algorithm is based on classical control system techniques, while REM is derived from an optimization standpoint. The PI algorithm attempts to regulate queue length and drops [1] packets with a probability proportional to the queue length error, i.e., the difference between the current length and the desired length, and to its accumulation (time integral). REM aims to stabilize both the input rate and queue length (match rate, clear buffer) regardless of the number of users sharing the link. In order to achieve this, REM introduces a variable, *price*, which acts as a measure of congestion. DRED is essentially an integral controller, and can be regarded as a special case of PI and REM. Note that PI, REM, and DRED all perform queue-length-based control and that they use instantaneous queue length, while RED uses average queue length. Queue-length-based control exhibits slower response than rate-based control. Another AQM algorithm, the adaptive virtual queue (AVQ) algorithm [11] utilizes a virtual queue. Using the difference between the desired utilization and the input rate, the virtual capacity and the virtual queue length are adjusted so as to make the input rate match the desired level of utilization. AVQ is basically a rate-based controller, while RED, DRED, PI, and REM are queue-length-based controllers. This explains in part the promising performance of AVQ compared to prior schemes under dynamic traffic scenarios.

In a recent study, the basic idea of the virtual rate control (VRC) algorithm has been proposed to tightly regulate the queue and to achieve high utilization with small packet loss, under various network conditions [12]. The notion of a *virtual target rate* was adopted to maintain the equilibrium input rate near to the link capacity and to stabilize the queue length. In [13], VRC was shown to be a proportional–integral–derivative (PID) controller. Another form of PID controller [14] and a PID controller based on the state-space approach [15] have also been proposed recently. The current paper is an extended version of the work done in [12,13]. The contributions of this study are as follows: (i) We present further analysis regarding the steady-state behavior and stability conditions of the VRC algorithm using linearized TCP dynamics from a control-theoretic standpoint. (ii) We validate our analysis by comparing the theoretical and simulation results. (iii) We present a comparative study with various AQM schemes such as RED, PI, REM, AVQ, and VRC through extensive simulations under various network scenarios.

[1] Hereafter, we use the terms *drop* (*dropping probability*) and *mark* (*marking probability*) interchangeably. If the network supports ECN [10], AQM schemes mark packets as congestion notification instead of dropping packets.

The rest of the paper is organized as follows: Section 2 introduces the VRC algorithm in detail. We analyze the stability of the VRC algorithm using the linearized model of TCP and derive stability conditions in Section 3. In Section 4, we present a comparative study of various AQM schemes. Section 5 shows the simulation results in order to validate the analysis and compare the performances of VRC with RED, PI, REM, and AVQ. The conclusion follows in Section 6.

2. Virtual rate control algorithm

To maximize link utilization and regulate the queue length effectively, it is desirable that the aggregate input rate be kept equal to the output link capacity. Any instance of the input rate exceeding the output link capacity should be quickly compensated for, before it increases the queue length and possibly leads to buffer overflow, which can result in low utilization. Furthermore, any difference between input rate and output link capacity results in the variation of the queue length and this variation causes delay jitter. Rate control can regulate the queue length and reduce delay jitter. Instead of queue control, consider a proportional rate control mechanism which can keep the input rate near a given target rate as follows:

$$p(t) = [\alpha(r(t) - r_t(t))]^+, \quad \alpha > 0. \tag{1}$$

Here $p(t)$ is the marking probability, $r(t)$ is the aggregate input rate of the queue, $r_t(t)$ is the target rate, and $[\cdot]^+ = \max(\min(\cdot, 1), 0)$. One can simply take the target rate $r_t(t)$ as the link capacity C to maximize the throughput. However, instead of a constant target rate, a modified target rate is adopted in order to keep the queue length $q(t)$ close to the target queue length q_{ref}. The target rate is set as the sum of the link capacity and the difference between q_{ref} and $q(t)$.

$$r_t(t) = C + \gamma(q_{\mathrm{ref}} - q(t)), \quad \gamma > 0. \tag{2}$$

If the queue length becomes smaller than its target length, there is more room to accommodate packets and so the target rate increases. Otherwise, if the queue length grows so as to exceed the target length, the target rate decreases. Combining (1) and (2), the marking probability becomes

$$p(t) = [\alpha((r(t) - C) + \gamma(q(t) - q_{\mathrm{ref}}))]^+. \tag{3}$$

The marking probability (3) is the weighted sum of the rate difference $(r(t) - C)$ and the queue length difference $(q(t) - q_{\mathrm{ref}})$, which is similar to the price of REM; REM updates the price to minimize the rate mismatch and queue mismatch. However, REM responds slowly to network congestion compared to (3) because the update rule of REM depends on the accumulation of these mismatches, while (3) depends on current mismatches.

However, the rate control mechanisms (1) and (2) cannot ensure that the input rate converges to the link capacity at equilibrium. To explain this discrepancy, the overall steady-state TCP behavior is considered using a graphical method. Fig. 1 illustrates the equilibrium of the marking probability and the input rate when rate control (1) is applied for congestion control. In Fig. 1, the throughput $U(p)$ is considered to be a strictly decreasing function of p, i.e., the throughput decreases as the marking probability increases. The equilibrium point (p^*, r^*) is at the intersection of (1) and $U(p)$. In Fig. 1, the equilibrium input rate r^* is greater than the equilibrium target rate r_t^*, and consequently there is always a rate error at equilibrium.

To compensate for this rate error, the concept of a *virtual target rate* is introduced. Instead of

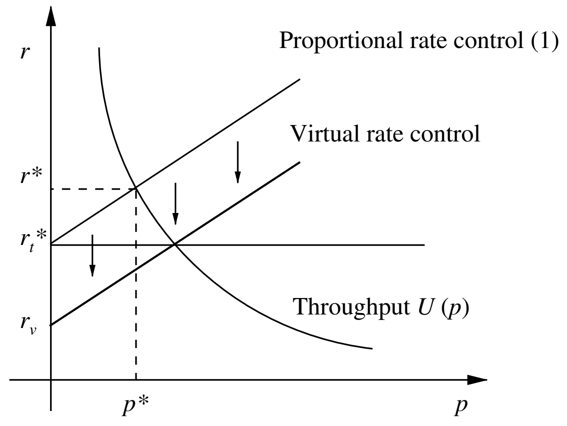


Fig. 1. Equilibrium points for proportional control and virtual rate control.

using $r_t(t)$ in (1), the virtual target rate $r_v(t)$ is adopted to match r^* with r_t^*.

$$p(t) = [\alpha(r(t) - r_v(t))]^+. \tag{4}$$

The virtual target rate $r_v(t)$ is updated to minimize the difference between $r(t)$ and $r_t(t)$ as follows:

$$\begin{aligned} &r_v(t) = r_t(t) - \Delta r_v(t), \quad t = nT_s, \\ &\Delta r_v(t + T_s) = \Delta r_v(t) + \beta T_s(r(t) - r_t(t)), \quad \beta > 0, \end{aligned} \tag{5}$$

where T_s is the sampling interval and $r(t)$ can be estimated using the algorithm in [16].

In [12], it is shown that if the throughput of TCP is a strictly decreasing function of the dropping probability, p, i.e., $\dot{U}(p) < 0$, then the input rate of VRC converges to the target rate at the steady-state. This has been proven without resorting to any TCP dynamic model.

3. Analysis of the VRC algorithm

In this section, we will analyze the stability of the VRC algorithm when combined with TCP dynamics. First, we present the dynamic model of VRC with TCP. Next, using this model we derive the stability condition of the VRC algorithm, which is represented in terms of its control parameters and can be used as an effective design guideline for parameter setting.

3.1. Dynamic model of VRC in TCP networks

We adopt the fluid-based TCP dynamic model [17] which was developed using fluid-flow and stochastic differential equation analysis. For the sake of simplicity, we ignore the slow-start and time-out mechanism of TCP. The simplified model is as follows:

$$\dot{W}(t) = \frac{1}{R(t)} - \frac{W(t)W(t - R(t))}{aR(t - R(t))} p(t - R(t)). \tag{6}$$

Here, $W(t)$ [packets] and $R(t)$ [s] are the window size and RTT, respectively, and $a(> 0)$ is a scaling factor. The queue length, $q(t)$ [packets], changes depending on the queue occupancy rate, which is the difference between the incoming rate and the outgoing link capacity, i.e., $r(t) - C$. Taking the finite buffer size, B [packets], into consideration we can model the queue dynamics as

$$\dot{q}(t) = \begin{cases} r(t) - C & \text{if } 0 < q(t) < B, \\ \max(0, r(t) - C) & \text{if } q(t) = 0, \\ \min(0, r(t) - C) & \text{if } q(t) = B. \end{cases} \tag{7}$$

The round-trip delay $R(t)$ is the sum of propagation delay T_p [s] and queuing delay $q(t)/C$,

$$R(t) = T_p + q(t)/C. \tag{8}$$

We assume that there are N homogeneous TCP connections. Then, the aggregate incoming rate $r(t)$ [packet/s] can be represented as a function of $W(t)$ and $q(t)$:

$$r(t) = N\frac{W(t)}{R(t)} = N\frac{W(t)}{T_p + q(t)/C}. \tag{9}$$

On the other hand, the marking probability of VRC can be represented as

$$p(t) = [\alpha(r(t) - C) + \alpha(\beta + \gamma)(q(t) - q_{\text{ref}}) + \alpha\beta\gamma z(t)]^+, \tag{10}$$

$$z(t) = \int_0^t (q(\tau) - q_{\text{ref}})\mathrm{d}\tau \tag{11}$$

from (2), (4), (5) and (7).

The block diagram of the whole system is shown in Fig. 2. The system consists of three blocks:

- VRC (congestion controller): Regulating the queue length by adjusting $p(t)$, which is calculated using queue length error, $e(t) = q(t) - q_{\text{ref}}$.
- TCP dynamics: Controlling the window size $W(t)$ and the input rate $r(t)$ based on the information of $p(t)$ and $R(t)$.
- Queue dynamics: Acting as integrator with respect to the queue occupancy rate, $(r(t) - C)$.

Ignoring the limit of buffer size, (10) can be written as

$$p(t) = K_D\dot{e}(t) + K_P e(t) + K_I \int_0^t e(\tau)\,\mathrm{d}\tau. \tag{12}$$

Note that (12) is a proportional–integral–derivative (PID) type control with respect to queue length error with $K_D = \alpha$, $K_P = \alpha(\beta + \gamma)$ and

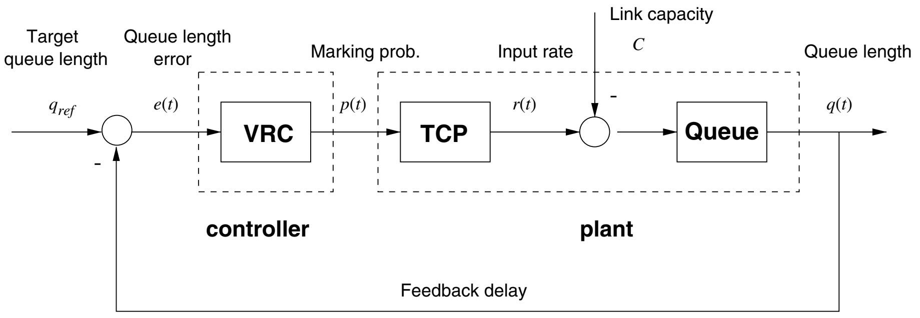


Fig. 2. Block diagram of the system.

$K_{\rm I} = \alpha\beta\gamma$. As will be shown in Section 4, the proposed algorithm is a generalized form of the RED, DRED, PI, and REM algorithms. Compared to these algorithms, the proposed algorithm can respond quickly to traffic changes, due to the introduction of derivative control. In the derivative control, the control depends on the rate of change of the error, exhibiting an anticipatory response. Hence, the derivative control tends to improve stability. On the other hand, the integral control improves the steady-state response and provides robustness with respect to parameter variations. However, it tends to reduce the stability of the system. The effect of derivative control on the stability will be shown via numerical analysis in this section and extensive simulations in Section 5.

3.2. Stability analysis

In order to linearize the system described in (6), (7) and (11), we find the equilibrium point (W^*, q^*, z^*). By setting $\dot{W}(t) = 0$, $\dot{q}(t) = 0$, and $\dot{z}(t) = 0$, we can obtain equilibrium point from (6), (7), (10) and (11):

$$\begin{aligned} W^* &= \sqrt{\frac{a}{p^*}} = \frac{R^*C}{N}, \\ q^* &= q_{\rm ref}, \\ z^* &= \frac{p^*}{K_{\rm I}} = \frac{a}{K_{\rm I}}\left(\frac{N}{R^*C}\right)^2. \end{aligned} \tag{13}$$

Here, the equilibrium RTT R^* is calculated as $R^* = T_{\rm p} + q_{\rm ref}/C$ from (8). In order to make r^* agree with the steady-state throughput of TCP [18], i.e., $r^* = \sqrt{3/2}/(\sqrt{p^*}R^*)$ for a single source, the constant a is set to 3/2. Note that $r^* = NW^*/R^* = C$ and $q^* = q_{\rm ref}$ from (9) and (13). This means that, in equilibrium, the VRC algorithm matches the input rate and the queue length to the link capacity and the target queue length, respectively.

Now, we linearize the nonlinear model at the equilibrium point (W^*, q^*, z^*). Let $W(t) = W^* + \delta W(t)$, $q(t) = q^* + \delta q(t)$, $z(t) = z^* + \delta z(t)$. Then, the linearized model around (W^*, q^*, z^*) can be written as follows:

$$\begin{aligned} \delta\dot{W}(t) &= -\tau_1\delta W(t) - (\tau_1 + K_{\rm D}\tau_2)\delta W(t - R^*) \\ &\quad - \frac{\tau_1}{N}\delta q(t) - \left(-\frac{\tau_1}{N} - \frac{K_{\rm D}\tau_2}{N} + \frac{K_{\rm P}\tau_2 R^*}{N}\right) \\ &\quad \times \delta q(t - R^*) - \frac{K_{\rm I}\tau_2 R^*}{N}\delta z(t - R^*), \\ \delta\dot{q}(t) &= \frac{N}{R^*}\delta W(t) - \frac{1}{R^*}\delta q(t), \\ \delta\dot{z}(t) &= \delta q(t), \end{aligned} \tag{14}$$

where $\tau_1 = N/(R^{*2}C)$ and $\tau_2 = C^2/(aN)$. The derivation of (14) is given in Appendix A.1.

We analyze the stability of the linearized model (14) using its characteristic equation. By taking the Laplace transform of (14), we obtain the characteristic equation

$$\begin{aligned} & s^3 + \left(\tau_1 + \frac{1}{R^*}\right)s^2 + \frac{2\tau_1}{R^*}s \\ & \quad + {\rm e}^{-sR^*}((\tau_1 + K_{\rm D}\tau_2)s^2 + K_{\rm P}\tau_2 s + K_{\rm I}\tau_2) = 0. \end{aligned} \tag{15}$$

For the simplicity of analysis, we approximate the time delay as the first-order lag, i.e., ${\rm e}^{-sR^*} \approx 1/(1 + R^*s)$ [19]. In the low frequency region, the approximation is effective [2], and the resulting approximation of (15) is

$$s^4 + a_1 s^3 + a_2 s^2 + a_3 s + a_4 = 0, \tag{16}$$

where

$$a_1 = \tau_1 + 2/R^*,$$
$$a_2 = \frac{1}{R^*}\left(4\tau_1 + K_D\tau_2 + \frac{1}{R^*}\right),$$
$$a_3 = \frac{1}{R^{*2}}(2\tau_1 + K_P\tau_2 R^*),$$
$$a_4 = \frac{K_I\tau_2}{R^*}.$$

Theorem. *The approximated system of* (14) *is locally stable* **if and only if** *the control parameters* K_D, K_P, *and* K_I *satisfy*

$$(2\tau_1 + \tau_2 R K_P)[(2 + \tau_1 R)(4\tau_1 + \tau_2 K_D + 1/R) - 2\tau_1 + \tau_2 R K_P] - (2 + \tau_1 R)^2 \tau_2 R K_I > 0. \tag{17}$$

Proof. The details of the proof are given in Appendix A.2.

We can investigate the stability region in terms of system parameters using this theorem. We set the control parameters to be $K_D = 0.0001$ and $K_P = K_I = 0.001$, and calculate and plot the stability region in Fig. 3. From Fig. 3, we can see that the region of system stability decreases in size as the RTT and capacity increase or the number of flows decreases. Note that the stability region does not shrink, but rather widens, as the number of TCP connections increases. These results agree well with those reported in [11,20,21]. Using the theorem, we can check whether the control parameters that are used can guarantee the system stability or not. However, it is not easy to find three control parameters satisfying Eq. (17). Hence, we need to find closed-form conditions of control parameters for system stability. The next

[2] For the typical round-trip delay of TCP networks (i.e., $R \approx 0.05 \sim 0.2s$), the approximation shows little error up to a frequency of several rad/s, which is sufficient for this study. Hence we can say that the approximation is effective for AQM controller.

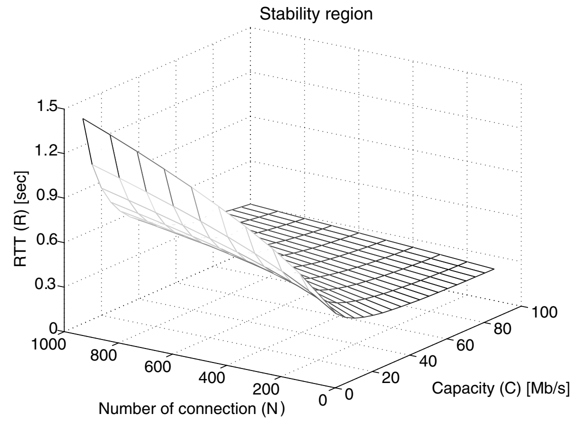


Fig. 3. Stability region of R, C, and N at fixed parameters K_D, K_P, and K_I: the region below the surface is stable.

theorem gives a practical design guideline for parameter setting to make the system stable.

Corollary. *The approximated system of* (14) *is stable* **if**, *for any non-negative* K_D, *the control parameters* K_P *and* K_I *satisfy the following*:

$$0 < K_I < K_I^{\max}, \tag{18}$$

$$\max(0, K_{P1}) < K_P < K_{P2}, \tag{19}$$

where

$$K_I^{\max} = \frac{(4\tau_1 + \tau_2 K_D + 1/R)^2}{4\tau_2 R}, \tag{20}$$

$$K_{P1}, K_{P2} = \frac{1}{2\tau_2 R^2}\left[(\tau_1 R + 2)(4\tau_1 R + \tau_2 R K_D + 1) - 4\tau_1 R \mp (\tau_1 R + 2)\sqrt{(4\tau_1 R + \tau_2 R K_D + 1)^2 - 4\tau_2 R^3 K_I}\right]. \tag{21}$$

Proof. The details of the proof are given in Appendix A.3.

Example. We present a simple numerical example that shows how to employ Corollary when setting control parameters. Consider the system parameters are given as $R = 0.1$ s, $N = 100$, and $C = 1250$

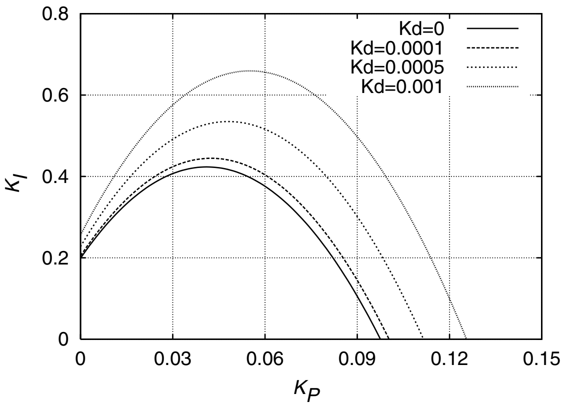


Fig. 4. Parametric region of K_D, K_P, and K_I for system stability: the regions below the curves are stable.

packet/s (corresponds to a 10 Mb/s with an average packet size of 1 KByte). First, we can take an arbitrary positive value for $K_D = 0.001$. Then, (18) gives K_I range: $0 \leqslant K_I < 0.659$. After taking $K_I = 0.01$, then we can choose K_P from (19), which gives K_P region: $0 < K_P < 0.125$.

Using the Corollary, we can calculate and plot the parametric region of K_D, K_P, and K_I required for system stability. The parametric region is drawn in Fig. 4, when the system parameters are given as $R = 0.1$ s, $N = 100$, and $C = 1250$ packet/s. Note that Fig. 3 shows the stability region, which is represented in terms of system parameters (R, N, and C), at fixed control parameters (K_D, K_P, and K_I). Conversely, Fig. 4 shows the region of control parameters when the system parameters are given. As shown in Fig. 4, small value of K_I extends the range of K_P and large value of K_D also extends the ranges of K_P and K_I for stability. This result in Fig. 4 implies that stable range of the parameters in the VRC algorithm, which has the derivative control ($K_D > 0$), is wider than that of the conventional AQM algorithms such as RED, DRED, PI, and REM algorithms, which do not have the derivative control ($K_D = 0$).

The validity of this analysis and the impact of the number of connections, round-trip delay, and the link capacity on the stability of the network will be investigated through extensive simulations in Section 5.

4. Comparative analysis of AQMs

In this section, we give a brief explanation of various AQM schemes from a control-theoretic viewpoint. The RED algorithm manages the queue length by randomly dropping packets with increasing probability as the average queue length increases. The dropping probability p can be represented as a function of the average queue length $\bar{q}(t)$:

$$p(t) = \begin{cases} \max(0, s(\bar{q}(t) - \min_{\text{th}})), & \bar{q}(t) < \max_{\text{th}}, \\ 1, & \text{otherwise}, \end{cases} \tag{22}$$

where $s = p_{\max}/(\max_{\text{th}} - \min_{\text{th}})$. This shows that the RED is basically a *proportional* controller. When the network is lightly congested, this proportional controller can keep $\bar{q}(t)$ at around $\min_{\text{th}}$. However, when the traffic load increases, $\bar{q}(t)$ increases, too. This load-dependent property of RED causes a scalability problem, which will be shown via simulations in Section 5.

To resolve this drawback of RED, dynamic RED (DRED) has been proposed [9]. The objective of DRED is to stabilize queue length $q(t)$ at a predetermined target queue length q_{ref}. To do this, DRED adapts the dropping probability $p(t)$ using the error signal $e(t) = q(t) - q_{\text{ref}}$, so that $q(t)$ can be kept as close to q_{ref} as possible. DRED updates $p(t)$ periodically with a fixed time interval T_s and the update rule for $p(t)$ is given as

$$p(t) = \left[p(t-1) + \alpha \frac{\hat{e}(t)}{B} \right]^+. \tag{23}$$

Here, α and B are a control parameter and the maximum buffer size, respectively, and $\hat{e}(t)$ denotes the estimated value of $e(t)$, which can be obtained by using a first-order low-pass filter. We can rewrite (23) as

$$p(t) = \frac{\alpha}{B} \int_0^t \hat{e}(\tau)\,\mathrm{d}\tau.$$

Hence we can see that DRED is actually an *integral* controller.

The PI AQM [7] consists of a *proportional* and an *integral* controller, which can be regarded as being more general than DRED. The update rule for $p(t)$ is

$$p(t) = [p(t-1) + a(q(t) - q_{\mathrm{ref}}) - b(q(t-1) - q_{\mathrm{ref}})]^{+}, \tag{24}$$

where a, b are control parameters. By rewriting the recursive equation (24), $p(t)$ can be represented as

$$\begin{aligned} p(t) &= b(q(t) - q_{\mathrm{ref}}) + (a-b)\int_0^t (q(\tau) - q_{\mathrm{ref}})\,\mathrm{d}\tau \\ &= K_{\mathrm{P}}^{\mathrm{PI}} e(t) + K_{\mathrm{I}}^{\mathrm{PI}} \int_0^t e(\tau)\,\mathrm{d}\tau. \end{aligned} \tag{25}$$

The PI controller integrates the queue length error in order to make the dropping probability vary according to the traffic load. However, the integral controller can induce a large overshoot and windup phenomenon, which may result in buffer overflow.

Another AQM scheme, REM [8], introduces a new variable, *price* μ, as a measure of congestion and calculates the dropping probability as a function of the price $\mu(t)$ such that

$$p(t) = 1 - \phi^{-\mu(t)}, \tag{26}$$

$$\mu(t) = [\mu(t-1) + \gamma[\alpha(q(t-1) - q_{\mathrm{ref}}) + (r(t) - C)]]^{+}. \tag{27}$$

Considering the queue dynamics, $(r(t) - C)$ is the rate at which the queue length grows and it can be approximated to the change in queue length, i.e., $q(t) - q(t-1)$. Then, $\mu(t)$ can be written as

$$\begin{aligned} \mu(t) &= [\mu(t-1) + \gamma(q(t) - q_{\mathrm{ref}}) \\ &\quad - (1-\alpha)\gamma(q(t-1) - q_{\mathrm{ref}})]^{+} \\ &= [(1-\alpha)\gamma(q(t) - q_{\mathrm{ref}}) \\ &\quad + \alpha\gamma \int_0^t (q(\tau) - q_{\mathrm{ref}})\,\mathrm{d}\tau]^{+}. \end{aligned} \tag{28}$$

Comparing (28) and (24), we can see that the main update rule for REM is essentially identical to that of the PI controller. If $\mu(t)$ is small, (26) can be approximated as $p(t) = (\ln\phi)\mu(t)$, hence, $p(t)$ is the scaled version of $\mu(t)$. Combining this with (28), $p(t)$ is represented as

$$p(t) = K_{\mathrm{P}}^{\mathrm{REM}} e(t) + K_{\mathrm{I}}^{\mathrm{REM}} \int_0^t e(\tau)\,\mathrm{d}\tau, \tag{29}$$

where, $K_{\mathrm{P}}^{\mathrm{REM}} = (1-\alpha)\gamma\ln\phi$ and $K_{\mathrm{I}}^{\mathrm{REM}} = \alpha\gamma\ln\phi$.

AVQ [11] is a novel scheme compared to the previous AQM algorithms. Its derivation is based on an optimization-based foundation [22]. The AVQ algorithm utilizes a virtual queue whose capacity $\widetilde{C}$ is less than the actual link capacity C, i.e., $\widetilde{C} = \beta C$ $(0 < \beta < 1)$. The differential equation of the virtual capacity $\widetilde{C}$ is

$$\dot{\widetilde{C}}(t) = -\alpha(r(t) - \gamma C), \tag{30}$$

where α and γ are control parameters, which determine the convergence speed and the desired level of utilization, respectively.

If we set $\gamma = 1$ for simplicity, then (30) becomes $\dot{\widetilde{C}}(t) = -\alpha(r(t) - C) = -\alpha\dot{q}(t)$. Taking the initial conditions as $\widetilde{C}(0) = C$ and $q(0) = 0$, $\widetilde{C}$ can be obtained from $q(t)$ as

$$\widetilde{C}(t) = C - \alpha q(t). \tag{31}$$

From (31), we can see that the virtual capacity $\widetilde{C}$ is adjusted according to the backlog in the queue, i.e., $\widetilde{C}$ decreases as the backlog grows larger and vice versa. Packets in the real queue start to be dropped when the virtual queue overflows. In [11], the dropping probability is given as

$$p(t) = \left[1 - \frac{\widetilde{C}(t)}{r(t)}\right]^{+}. \tag{32}$$

This equation shows that AVQ is approximately an SPF (simple probabilistic forwarding) algorithm [16] based on the virtual capacity instead of the link capacity. In this way, AVQ adapts the input rate to achieve the desired level of utilization and keeps the queue length short.

The above algorithms, with the exception of AVQ, can be viewed as being a special case of the VRC algorithm.

5. Simulation study

In this section, we conduct extensive and profound simulations using the *ns*-2 network simulator [23], in order to validate our analysis and compare the performance of the VRC algorithm with other AQM algorithms such as RED, PI, REM, and AVQ.

The simulation setup is briefly explained in the first sub-section. In the next sub-section, we confirm the validity of the stability analysis for the VRC algorithm and compare our theoretical results with the simulation results in Cases A1 and A2. Additionally, we study the effect of the target queue length on the level of utilization and the round-trip delay in Case A3.

Next, Case B1 and B2 evaluate and compare the transient behaviors of various AQM schemes when the traffic load changes dynamically. We observe and compare the responsiveness of AQM schemes under a dynamic traffic scenario by investigating the queue length and its variation.

One important issue for practical implementation of the AQM scheme is that it should maintain its level of performance regardless of the network conditions. Hence, in the following sub-section we perform simulations in order to evaluate the robustness of the various AQM schemes in relation to variations in certain system parameters, such as the number of flows (Case C1), the round-trip propagation delay (Case C2), the bottleneck link capacity (Case C3), and the buffer size (Case C4). Finally, we evaluate and compare the performance of various AQM schemes under more realistic network scenarios, i.e.,

(i) when TCP connections have different RTT's, different access link capacities, and different transport protocols (Case D1),
(ii) when TCP flows, UDP flows, and web-like short-flows coexist (Case D2),
(iii) when there exist multiple bottleneck links (Case D3).

5.1. Simulation setup

We implement a simple bottleneck network configuration consisting of two routers and a number of pairs of TCP senders and receivers as shown in Fig. 5. The routers are connected by a link with a capacity of 10 Mb/s. The capacities of all the other links are set to 100 Mb/s, so that only the link between the two routers generates a bottleneck, while the other links do not. The propagation delay of the bottleneck link is set to be uniformly distributed between 5 and 15 ms [3], and those of the other links are set to [10–30] ms. Thus, the round-trip propagation delay is distributed between 50 and 150 ms.

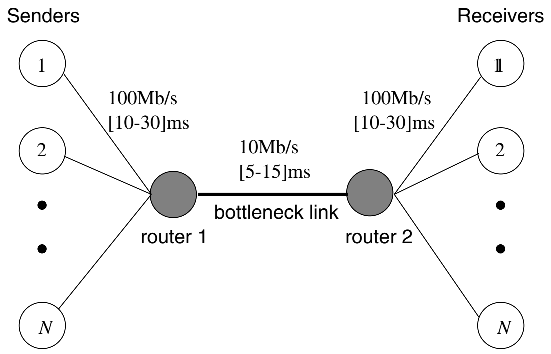


Fig. 5. Simple bottleneck topology.

The allocated buffer size, B, is 100 packets, and the target queue length at the router is one half of the buffer size, i.e., $q_{\text{ref}} = 0.5B = 50$ packets. The average packet length is set to 1 Kbytes and the simulation time is set to 100 s. These parameters are not changed unless otherwise stated. We use TCP-Reno as the default transport protocol and we use ECN marking [10] instead of packet dropping. We assume that the senders always have data to send.

In the simulations, the performance is evaluated in terms of the average queue length and the standard deviation (STD) of queue length, the link utilization, and the packet loss rate. By observing the average and the STD of queue length, which are the first-order and the second-order statistics of queue length distribution, respectively, we can evaluate the stabilization performance of the queue. Furthermore, the queue length and its variation are closely related to the queuing delay and the amount of delay jitter, respectively, the latter two parameters constituting important performance indices for delay-critical real-time traffic. The link utilization is defined as the throughput

[3] Hereafter, we denote the propagation delay that is uniformly distributed between T_1 and T_2 as $[T_1 - T_2]$.

Table 1
The parameters of RED, PI, REM, AVQ and VRC algorithms

RED		PI		REM		AVQ		VRC	
$\min_{th}$	$0.2B$	a	0.002445	α	0.0184	α	0.15	K_D	0.0003
$\max_{th}$	$0.8B$	b	0.0024	γ	0.00175	β	0.8	K_P	0.0024
$p_{\max}$	0.1	T_s	10 ms	ϕ	1.15	γ	1.0	K_I	0.0045

normalized by the link capacity and the loss rate is defined as the ratio of the number of lost packets to the number of packets transmitted by the senders.

The control parameters of the various AQM schemes and their values are listed in Table 1. For RED, $\max_{th}$ and $\min_{th}$ are set to $0.2B$ and $0.8B$, respectively, in order to keep the average queue length at around $0.5B$ $(= q_{\text{ref}})$, and $p_{\max}$ is set to the default value of 0.1 and the *gentle_* option [24] is enabled. For PI and REM, the control parameters are set so that K_P^{PI} in (25) and K_P^{REM} in (29) have the same value of K_P, which is the proportional gain of VRC, and similarly, K_I^{PI} in (25) and K_I^{REM} in (29) have the same value of K_I, which is the integral gain of VRC. Also, we set the update interval T_s for PI, REM, and VRC to be equal, 10 ms. By taking $K_P^{PI} = K_P^{REM} = K_P$ and $K_I^{PI} = K_I^{REM} = K_I$, we can evaluate the effect of introducing the derivative control in VRC over PI and REM. For AVQ, we set the desired utilization, γ, to one for fair comparison with the other AQM schemes. The other parameters of AVQ are set to the values recommended in [11]. Finally, the parameters of VRC, K_P, K_I and K_D are chosen to satisfy the stability conditions (18) and (19). Those parameters which are not specified are set to the values recommended in the original papers and in the *ns*-2. Note that the values of the parameters remain unchanged throughout the whole process of simulation unless otherwise stated.

5.2. Validity of analysis and characteristics of VRC

5.2.1. Case A1: Validity for critical delay bound

In this simulation, we determine the critical delay bound required for system stability with a fixed set of control parameters, and compare the simulation results with the analytical results. Let $R^*_{\max}$ be the maximum round-trip delay that does not violate (17) with the given control parameters K_D, K_P and K_I; if the round-trip delay is smaller than $R^*_{\max}$ the system remains stable with the given parameters.

For fixed $N = 100$, $C = 10$ Mb/s and control parameters in Table 1, $R^*_{\max}$ is calculated to be 361 ms from (17). The buffer size is set to be about half of the value of the maximum delay-bandwidth product and the target queue length is set to half of the buffer size, i.e., $B = 0.5R^*_{\max}C \approx 200$ packets and $q_{\text{ref}} = 0.5B = 100$ packets. Assuming the average queuing delay is $q_{\text{ref}}/C = 80$ ms, then maximum propagation delay $T^*_{p,\max}$ is 280 ms. For two different round-trip propagation delay $T_{p,1}(< T^*_{p,\max})$ and $T_{p,2}(> T^*_{p,\max})$, we compare the corresponding performance via simulations. Fig. 6 (a) and (b) show the input rate and instantaneous queue length for $T_{p,1} = 250$ ms $< T^*_{p,\max}$ and $T_{p,2} = 300$ ms $> T^*_{p,\max}$, respectively. While both the input rate and queue length are controlled in the case of Fig. 6(a), they cannot be controlled properly and fluctuate severely in the case of Fig. 6(b), which is the case wherein T_p exceeds $T^*_{p,\max}$, hence, the given control parameters do not satisfy the stability condition in the theorem. These results show the validity of the theorem and the performance can be degraded significantly if the control parameters do not satisfy the stability condition.

Similarly, we also calculate $T^*_{p,\max}$ for several sets of control parameters and compare them with the critical delay bounds obtained via simulation. Let $T_{p,\max}$ denote the maximum round-trip delay that allows the system to remain stable in the simulation. $T_{p,\max}$ is obtained by increasing T_p until the queue length consistently oscillates between zero and the maximum buffer size as shown in Fig. 6(b).

Fig. 7 shows both $T^*_{p,\max}$, obtained from analysis, and $T_{p,\max}$, obtained from simulation, versus K_P $(= K_I)$ for $K_{D1} = 0$ or $K_{D2} = 0.0003$. Note that

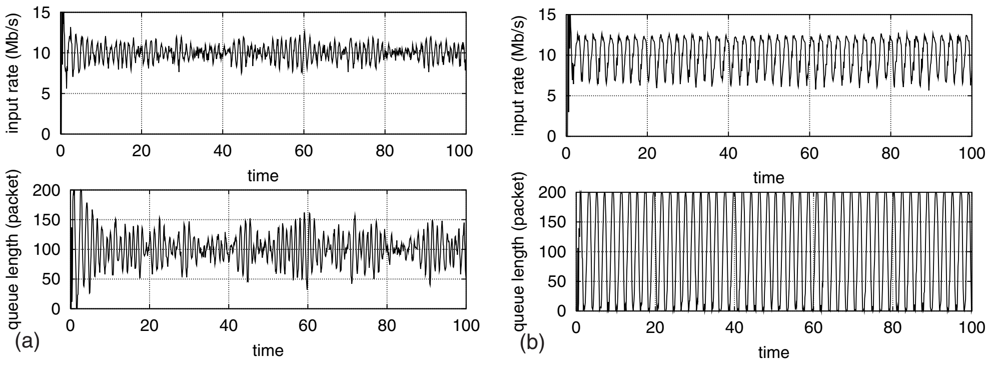


Fig. 6. Comparison of input rate and queue length with different round-trip propagation delay $T_{p,1}$ ($< T^*_{p,max}$) and $T_{p,2}$ ($> T^*_{p,max}$) (Case A1): (a) $T_{p,1} = 250$ ms ($< T^*_{p,max}$), (b) $T_{p,2} = 300$ ms ($> T^*_{p,max}$).

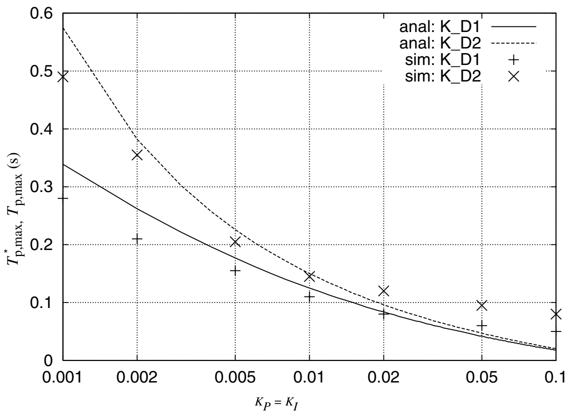


Fig. 7. Maximum propagation delay $T^*_{p,max}$ (obtained from analysis) and $T_{p,max}$ (obtained from simulation) with respect to several sets of $K_P = K_I$ and two constant K_Ds, $K_{D1} = 0$ and $K_{D2} = 0.0003$ (Case A1).

the cases with $K_{D1} = 0$ and $K_{D2} = 0.0003$ correspond to the PI controller and the VRC algorithm, respectively. Here, for the sake of convenience, we set $K_P = K_I$. Both $T^*_{p,max}$ and $T_{p,max}$ for other sets of K_P and K_I also show a similar trend. As shown in Fig. 7, both $T^*_{p,max}$ and $T_{p,max}$ show the similar trends and the difference between them is little, which shows the validity of the analysis in Section 3. When K_P ($= K_I$) is small, the maximum delay bound of T_p with K_{D1} is much smaller than that of T_p with K_{D2}. As K_P ($= K_I$) increases, the difference between the two delay bounds decreases. Comparing $T_{p,max}$ with K_{D1} and $T_{p,max}$ with K_{D2}, we can confirm that VRC is effective even for the range of long round-trip delay where the PI controller fails to make the system stable. Fig. 7 also shows that the maximum delay bound exponentially decreases as K_P and K_I increase, whereas it increases as K_D increases. Conversely, by reducing the values of K_P and K_I, or by increasing the value of K_D, we can guarantee system stability for a longer round-trip delay. This result agrees with the classical control theory in that, for the PID control, decreasing the proportional gain (K_P) and the integral gain (K_I) or increasing the derivative gain (K_D) tends to improve the stability.

5.2.2. *Case A2: Validity of parametric region for system stability*

In the pervious simulations, we focused on the critical delay bound with fixed control parameters K_D, K_P, and K_I. Conversely, in this simulation, we evaluate the stability region of the control parameters, with fixed sets of system parameters, such as R, N, and C. For fixed $N = 100$, $C = 10$ Mb/s and two different values of $R = 0.2$ and 0.3 s, we plot the boundaries of K_P and K_I that satisfy (18) and (19) in Fig. 8. Compared to Fig. 4, which shows the boundaries of K_P and K_I for different values of K_D, Fig. 8 shows the boundaries for different values of R. Here, we set $K_D = 0.0003$ to simplify the simulation. Also, via simulation, we measure several sets of boundaries of K_P and K_I for the corresponding R's and plot them together with our analysis results in Fig. 8. Here, the set of K_P

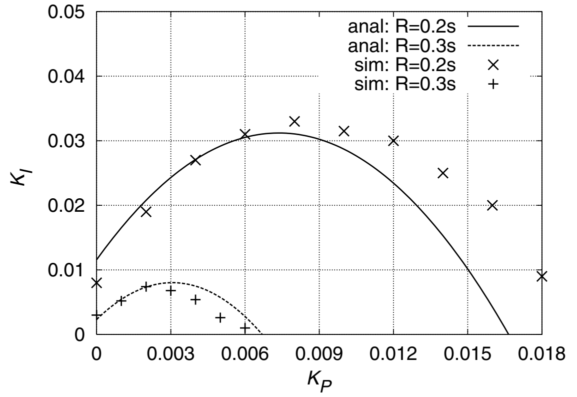


Fig. 8. K_P and K_I regions required for system stability with fixed K_D and two different RTTs: for each R, the region above the curve is unstable and the one below is stable (Case A2).

and K_I is obtained by increasing K_I at fixed K_P until the queue length consistently oscillates between zero and the maximum buffer size.

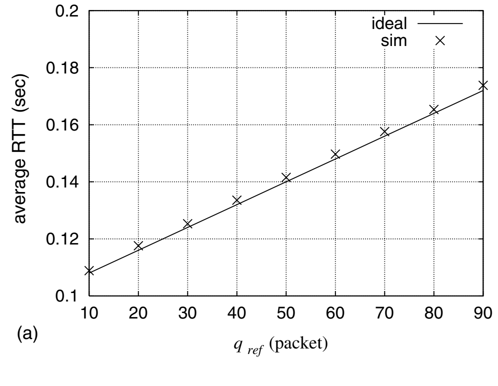
As shown in Fig. 8, the stability region of the control parameters rapidly contracts as the delay increases (compare the cases of $R = 0.2$ and 0.3 s). Some differences in the stability regions are observed between the results of analysis and simulation, which may be attributed to the linearization and approximation of the delay. However, the relationship between K_P and K_I as determined by the simulation is similar to the relationship predicted by the analysis. Thus, the conditions (18) and (19) in the corollary can be used as an effective guideline for setting parameter values.

5.2.3. Case A3: Effect of target queue length

In this simulation, we investigate the characteristics of VRC focusing on the effect of target queue length on performance indices, such as delay and utilization. If we set q_{ref} to be small, the queue length will be maintained around this small value of q_{ref}. Hence we can intuitively expect that the queuing delay would be small, however, this small target length leads to the possibility of under-utilization occurring, since a small value of q_{ref} increases the possibility of the queue being empty. On the other hand, a large value of q_{ref} can result in high utilization, with a longer queuing delay. Thus, there is a trade-off between small queuing delay and high utilization.

Both the ideal average RTT's and the actual average RTT's are plotted in Fig. 9(a). The ideal average RTT, $\overline{R^*}$, is the sum of average end-to-end propagation delay $\overline{T_p}$ and average queuing delay, i.e., $\overline{R^*} = \overline{T_p} + q_{ref}/C = 100 + 0.8q_{ref}$ ms. On the other hand, the actual average RTT, $\overline{R}$, is obtained from the simulation using the time-stamp information of TCP packet. Fig. 9(a) shows that measured RTT's are very close to their ideal values, and it indicates that $\overline{R}$ has nearly linear relationship with q_{ref}. The contribution of change in the target queue length, Δq_{ref}, to the round-trip delay is almost the same as $\Delta q_{ref}/C$. These results show that the queuing delay and/or round-trip delay can be satisfactorily controlled by adjusting the value of q_{ref}.

Next, we observe the effect of q_{ref} on utilization. From Fig. 9(b), we can see that the utilization is



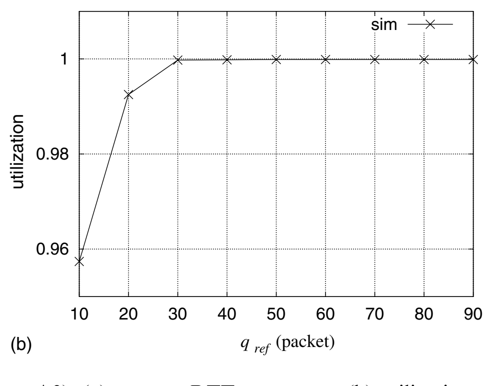


Fig. 9. Average RTT and utilization for various values of q_{ref} (Case A3): (a) average RTT versus q_{ref}, (b) utilization versus q_{ref}.

degraded when the value of q_{ref} is small, however, if q_{ref} is bigger than a certain value (i.e., 30 packets in this simulation), the utilization is almost one and varies very little. Thus, if we prefer short delay (high utilization), q_{ref} should be decreased (increased).

5.3. Transient behavior for dynamic traffic

5.3.1. Case B1: When TCP connections are abruptly established and dropped

We perform a simulation using a dynamic traffic scenario. Initially at $t = 0$ s, 50 TCP connections are established and then a half connections among them are dropped abruptly at $t = 50$ s. At $t = 100$ s, another 50 TCP connections are established.

Fig. 10 shows the corresponding queue lengths for the drop-tail, RED, PI, REM, AVQ, and VRC schemes. Since the drop-tail policy does not control queue length actively, the queue length is kept close to the buffer size and oscillates severely. Therefore, the drop-tail queue results in a long queuing delay and high jitter in the queuing delay, compared to those of the other algorithms. In the case of RED, the queue length depends on the traffic load; when the traffic load is light, the queue length is small, and when it becomes heavy, the queue length increase accordingly. However the queue length of the REM and VRC schemes are relatively constant, remaining close to the value of q_{ref} regardless of the traffic load. Although the average queue length of REM is close to q_{ref}, the queue length fluctuates more than that of VRC. Observing Fig. 10(d), especially at $t = 50$ s and $t = 100$ s when there are abrupt changes in the traffic load, we can see that the queue length of the REM scheme takes longer to settle down compared to VRC, i.e., the transient response of REM is slower than that VRC. As shown in Fig. 10(e), the queue length of AVQ does not change much when the traffic load decreases abruptly at $t = 50$ s, however, it is significantly affected by a sudden increase in traffic load at $t = 100$ s. On the other hand, the queue length of VRC remains near to the value of q_{ref}, even during the abrupt traffic changes, because the derivative control of VRC is able to compensate for the rate variation even before the variation affects the queue length.

5.3.2. Case B2: When TCP connections are randomly established and dropped

We consider another dynamic traffic scenario, i.e., when TCP connections are randomly established and dropped. There are initially 50 TCP connections that are persistent, and additional 50 connections are randomly established between $t = 0$ s and $t = 50$ s and randomly terminated between $t = 50$ s and $t = 100$ s.

The characteristics of the queue length in this simulation are similar to those of the previous simulation. We observe the performance indices of each AQM scheme and summarize them in Table 2. The level of utilization of all the AQM schemes is acceptable. Although the drop-tail queue provides almost full link utilization, its loss rate is particularly high. The reason is that the TCP senders reduce their sending rate only after detecting packet losses due to buffer-overflow. From a control-theoretic viewpoint, the drop-tail policy can be considered as *bang–bang* control. We exclude the drop-tail policy, hereafter, when comparing the performance of various AQM schemes, because it does not control the queue length actively. In the case of RED, the loss rate is high, because of its high average queue length, which causes the dropping probability to be high, and leads to the possibility of buffer-overflow. Compared with the other AQM schemes, VRC shows the smallest STD of queue length, the highest utilization, and the smallest loss rate. These results show that when the traffic load changes dynamically, the transient response of the VRC algorithm outperforms other AQM algorithms, and the queue length of VRC converges fast to the desired length, while maintaining a satisfactory utilization and loss rate.

5.4. Robustness to variations in system parameters

5.4.1. Case C1: Number of connections

We examine the network performance for different numbers of active connections, N. Since it is difficult to estimate N in a real environment, it is important that the AQM scheme not be sensitive to variations in the value of N. Here, we repeat the simulation with different values of N ranging from 10 to 500.

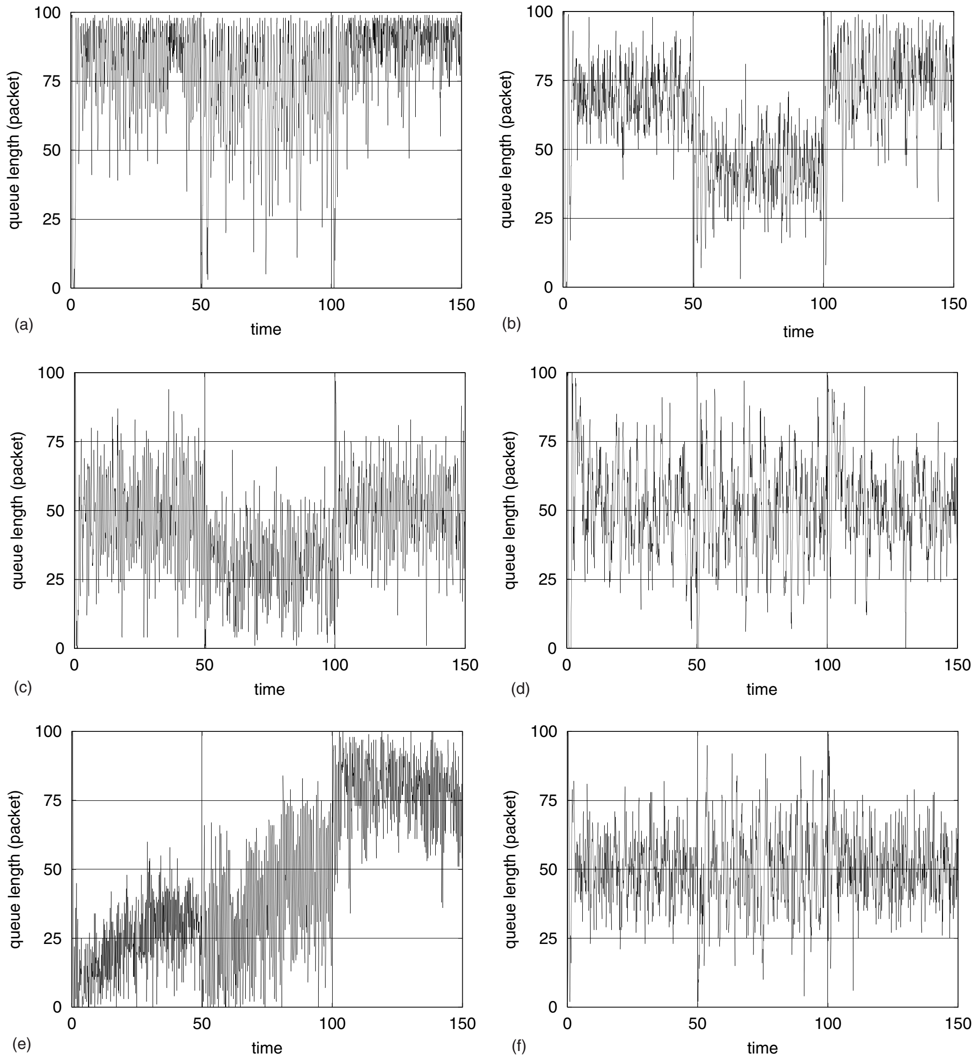


Fig. 10. Queue length of various AQM schemes when TCP connections are abruptly established and dropped (Case B1): (a) drop tail, (b) RED, (c) PI, (d) REM, (e) AVQ and (f) VRC.

Table 2
Performance comparison of various AQM schemes when TCP connections are randomly established and dropped (Case B2)

Criterion	Drop-tail	RED	PI	REM	AVQ	**VRC**
Average queue length	84.06	71.09	49.75	50.00	49.61	**49.83**
STD of queue length	14.83	15.81	13.50	15.88	29.78	**11.57**
Utilization (%)	99.70	99.61	99.79	99.52	99.03	**99.80**
Loss rate (%)	5.516	2.387	0.085	0.139	0.326	**0.049**

As shown in Fig. 11(a), in the case of RED and AVQ, the average queue length is sensitive to changes in the value of N. However, in the case of REM and VRC, the average queue length does not

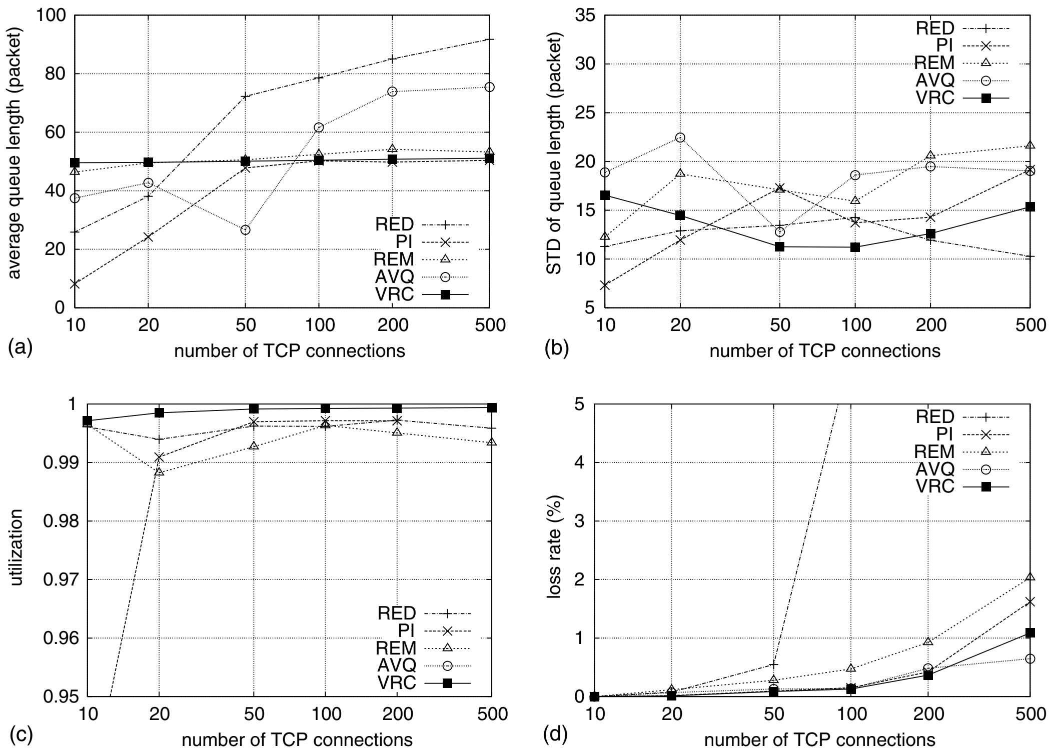


Fig. 11. Performance comparison of several AQM schemes for various numbers of TCP connections (Case C1): (a) average queue length, (b) STD of queue length, (c) utilization, (d) loss rate.

vary much from the target length regardless of the value of N, especially in the case of the VRC scheme. When N is less than 50, the PI algorithm is not able to regulate the queue length properly. In the case of RED, the average queue length increases as N increases, which confirms the load-dependent property of RED. The average queue length of AVQ varies irregularly with N, because AVQ does not control the queue length directly. Fig. 11(b) shows that the STD of queue length of VRC is relatively small compared to those of other algorithms.

Next, the throughput and loss rate are investigated. As shown in Fig. 11(c), the link utilization of VRC outperforms those of the other algorithms and remains relatively constant, whereas those of the other algorithms depend on the number of connections. The packet loss rates of all of the AQM schemes are satisfactory except for RED. As N increases, the loss rate of RED becomes substantially higher than those of the other algorithms, because the increased queue length of RED, due to heavy traffic load, results in buffer overflow. These simulation results verify that VRC has a consistent queuing delay, high utilization and a small loss rate for wide range of the traffic load.

5.4.2. Case C2: Round-trip propagation delay

Similarly, we investigate the impact of round-trip propagation delay T_p on the network performance indices. We set the average T_p to be in the range of 10–500 ms. For each T_p, we set the propagation delay to be distributed between $0.5T_p$ and $1.5T_p$. Here, N and C are set to 100 and 10 Mb/s, respectively.

Fig. 12(a) shows that in the case of VRC the average queue length is scarcely affected by T_p, while in the case of RED, PI, and REM its value decreases as T_p increases. In most cases, VRC

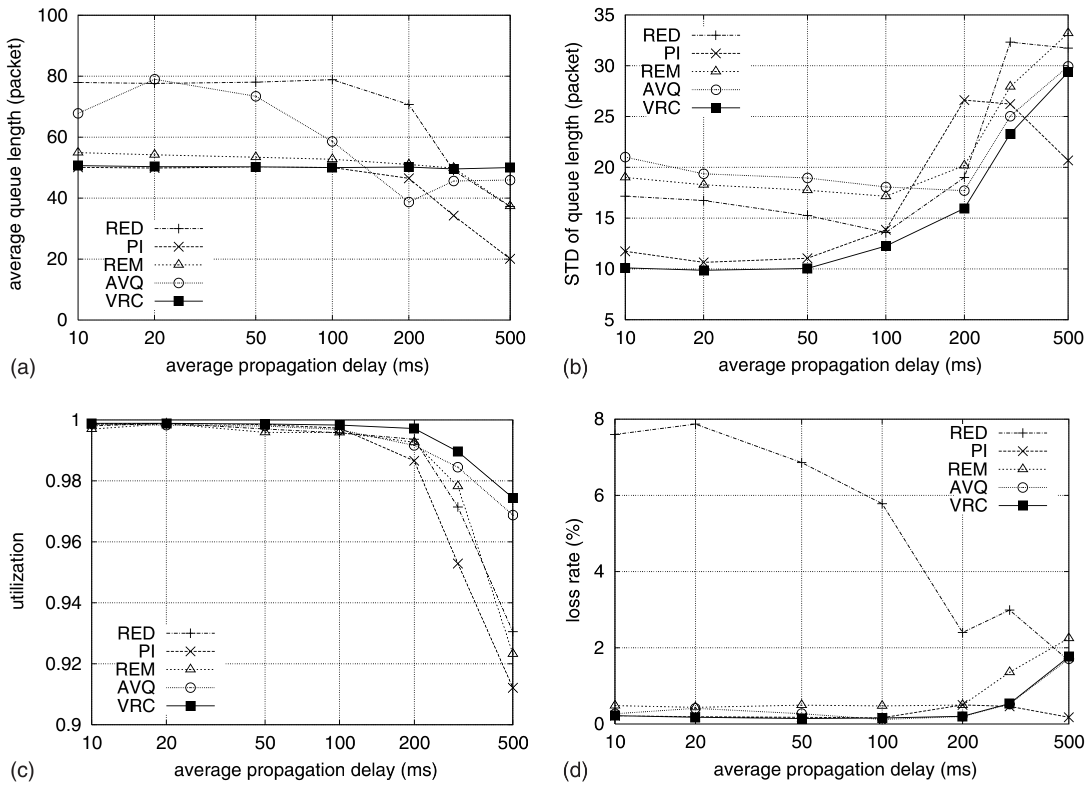


Fig. 12. Performance comparison of several AQM schemes for various round-trip propagation delays (Case C2): (a) average queue length, (b) STD of queue length, (c) utilization, (d) loss rate.

shows promising queue regulation performance with high utilization and small loss rate. We can investigate the stability of AQM schemes with respect to T_p by means of Fig. 12. When T_p is greater than about 200 ms, PI starts to fail to regulate the queue length at around q_{ref}, at which point the STD increases significantly and the utilization decreases abruptly. However, Fig. 12 (a) and (c) show that VRC regulates the queue length and maintains relatively high utilization, even for T_p values for which the utilization of the other AQM schemes deteriorates significantly. Thus, we can conclude that VRC is more robust with respect to long delay values than the other AQM schemes.

From our analysis, we can actually calculate the critical round-trip delay R^*_{max} for PI and VRC, which guarantees the stability of system, Furthermore, R^*_{max} for AVQ can be calculated from Theorem 4.1 in [11]. The results are as follows:

$R^*_{max} = 0.216$s for AVQ with $\alpha = 0.15$, $\gamma = 1.0$,
$R^*_{max} = 0.286$s for PI with $K_P^{PI} = 0.0024$, $K_I^{PI} = 0.0045$,
$R^*_{max} = 0.361$s for VRC with $K_P = 0.0024$, $K_I = 0.0045$, $K_D = 0.0003$.

Note that these results are represented in terms of the round-trip delay, R, while the results in Fig. 12 are represented in terms of the propagation delay, T_p. If we take into account the queuing delay, i.e., $T_p = R - q/C$, these results match those in Fig. 12. Recall that the proportional and integral gains of PI and VRC are both the same, i.e., $K_P^{PI} = K_P$ and $K_I^{PI} = K_I$, consequently, the addition of the derivative term in VRC enables the stability region to be expanded over a wider range of round-trip delay. This result also coincides with the fact that the critical delay bound can be increased with higher derivative gain K_D (compare the two cases of $K_{D1} = 0$ and $K_{D2} = 0.0003$ in Fig. 7). We also expect that the performance degradation due to long delay can be improved by using smaller values of K_P and K_I, as shown in Fig. 7.

5.4.3. Case C3: Bottleneck link capacity

To study the effect of bottleneck link capacity C, we set C to be in the range of 1–100 Mb/s. Here, N and T_p are set to 100, [50–150] ms, respectively. Fig. 13 shows and compares the performance indices of the various AQM schemes in relation to changes in the value of C. The effect of increasing the value of C on the performance index is similar to that produced by increasing the value of T_p (Compare Figs. 12 and 13). These trends can be explained by the fact that, the system tends to become unstable as T_p and C increase, as already stated in Section 3. The VRC scheme maintains the average queue length close to the value of q_{ref} and keeps the level of utilization high in comparison to the other schemes. However, the STD and utilization of all of the AQM schemes become degraded when C exceeds 20 Mb/s.

5.4.4. Case C4: Buffer size

In addition to the system parameters such as N, T_p, and C, we analyze the effect of buffer size on the performance of the AQM scheme. Note that the limit of the buffer size was ignored when analyzing the steady-state behavior of VRC in Section 3. We repeat the simulation with different values of the buffer size B ranging from 20 packets to 1000 packets. Here, N, T_p, and C are set to 100, [50–150] ms, and 10 Mb/s, respectively.

Fig. 14 compares the performance indices when the buffer size has different values. Fig. 14(a) and (b) represent the average and the STD of queue length normalized to buffer size, respectively, i.e.,

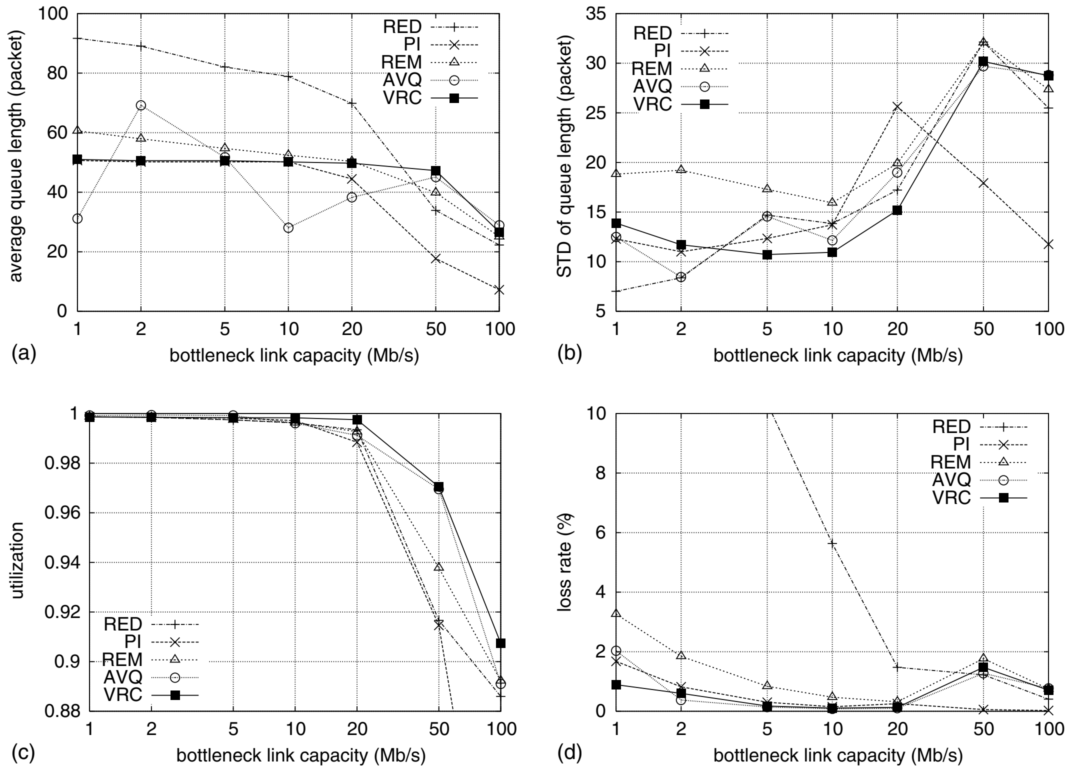


Fig. 13. Performance comparison of several AQM schemes for various bottleneck link capacities (Case C3): (a) average queue length, (b) STD of queue length, (c) utilization, (d) loss rate.

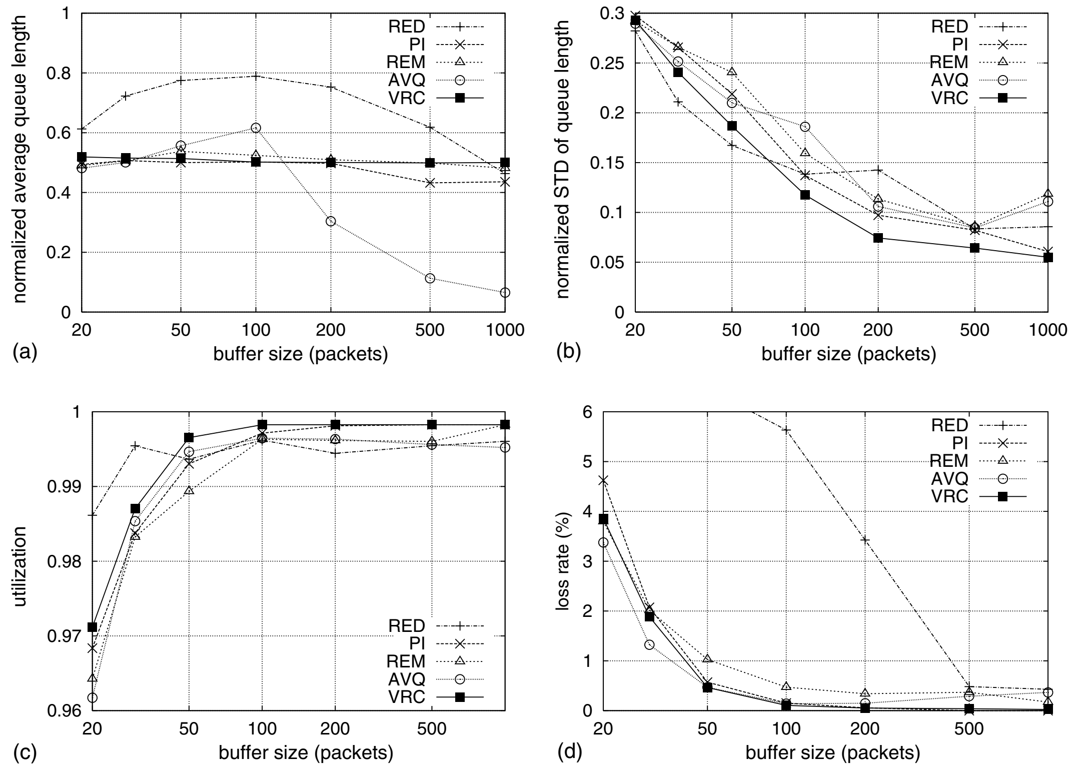


Fig. 14. Performance comparison of several AQM schemes for various buffer sizes (Case C4): (a) average queue length, (b) STD of queue length, (c) utilization, (d) loss rate.

$E[q(t)]/B$, $\mathrm{STD}[q(t)]/B$. As we can expect, when B has small value, the performance is degraded; the link utilization decreases and the loss rate increases because the small buffer size results in high probability of buffer-overflow and does not effectively alleviate bursty traffic of TCP. In spite of small buffer size, VRC outperforms the other algorithms for regulation of queue length. Also, in the case of VRC, the utilization is higher and the loss rate is lower than those of other AQM schemes in almost all the cases. RED provides high utilization but has high loss rate when the buffer size is small. As shown in Fig. 14, once the buffer size is larger than a certain values, 50 packets in this simulation, the effect of buffer size is not significant and the utilization is acceptable in all AQM's.

5.4.5. Discussion on the robustness of VRC

If we compare Figs. 11–13, we can see that in the case of RED and AVQ the average queue length depends on the system parameters, N, T_p, and C, while in the case of PI, REM, and VRC its value is kept close to the target queue length. Also, any change in T_p or C affects the STD of queue length and utilization more than a change in N. In almost all the cases, the queue length of VRC is tightly regulated to the target length and the utilization of VRC is higher than those of the other schemes.

Furthermore, the simulation results in this subsection show that the performance of VRC is less affected by the system parameters than the other AQM schemes. Thus, VRC can be expected to work well even when the system parameters vary or when it is difficult to estimate them. To design and implement VRC, i.e., to choose appropriate control parameters satisfying the stability conditions, it is necessary to know the values of N, R, and C. While the link capacity C is generally known to the network operator and the average round-trip delay R can be roughly estimated, the number of active connections, N, is somewhat harder to estimate. However, this is less of a problem in the case of VRC, because VRC is almost immune to changes in the system parameters. Recall that the control parameters of VRC, which are selected based on the stability condition in the corollary with $N = 100$, $R = 0.1$ s, and $C = 10$ Mb/s, are not changed during the course of the simulations. The simulation results in this subsection show that VRC can work for wide range of system parameters, i.e., $N = 10$–500, $T_p = 10$–300 ms, and $C = 1$–20 Mb/s. Furthermore, the analysis and simulation results show that the increase in the value of N does not lead to a deterioration in the performance of the system, while the increase in the values of R and C tends to worsen the system stability. Therefore, if we design VRC with a roughly estimated or over-estimated round-trip delay, R_0, and link capacity, C_0, and under-estimated N_0, the system would remain stable for $R < R_0$, $C < C_0$, and $N > N_0$, as stated in Section 3. Consequently, in practice, we expect VRC to work for a wide range of system parameters.

5.5. Using more realistic scenarios

5.5.1. Case D1: When the access links are heterogeneous

So far, we have investigated the performance of VRC for homogeneous links, i.e., all the links, with the exception of the bottleneck link, have the same propagation delay and link capacity. Here, we observe the performance of VRC for heterogeneous links, i.e., with different propagation delay or link capacity. In this simulation, pairs of TCP senders and receivers are grouped into four different subnets, with each subnet consisting of N_s pairs of TCP senders and receivers. Let us define C_s and $T_{p,s}$ as the capacity and propagation delay of the access link, respectively. The access link indicates the link between the TCP sender and the ingress router (router 1 in Fig. 5) and the link between the TCP receiver and the egress router (router 2 in Fig. 5).

We set the values of N_s, C_s, $T_{p,s}$ as well as the protocol of TCP in each subnet differently. The configuration of each subnet is summarized in Table 3. Hereafter, we use this configuration of subnet in the remaining simulations.

We compare the performance of VRC when the access links are heterogeneous and the different transport protocols are used and when the links are homogeneous and the same protocol is used. In the homogeneous case, we set C_s and $T_{p,s}$ for all subnets to have same values, 50 Mb/s and [11–19] ms, respectively. Note that the average capacities and the average propagation delays for the homogeneous case and the heterogeneous case are nearly the same.

Table 4 summarizes and compares the performance of VRC. It shows that the performance indices observed when the links are heterogeneous

Table 3
Configuration of subnet for more realistic scenarios: each subnet has different values of N_s, C_s, and $T_{p,s}$ and different transport protocol

Subnet	N_s	C_s	$T_{p,s}$ (ms)	Protocol
1	35	100 BaseT (100 Mb/s)	[5–8]	TCP/Reno
2	20	10 BaseT (10 Mb/s)	[14–25]	TCP/NewReno
3	25	T3 (45 Mb/s)	[11–16]	TCP/Tahoe
4	20	T1 (1.54 Mb/s)	[21–34]	TCP/Reno

Table 4
Performance comparison of VRC for the homogeneous case and heterogeneous case (Case D1)

Criterion	Hetero.	Homo. (Tahoe)	Homo. (Reno)	Homo. (NewReno)
Average queue length	50.46	50.43	50.43	50.42
STD of queue length	10.84	10.67	10.72	10.50
Uilization (%)	99.94	99.92	99.92	99.92
Loss rate (%)	0.104	0.085	0.083	0.094

are almost the same as those obtained when the links are homogeneous. It also shows that the difference between the performance indices with different protocol are negligible. These results show that the VRC algorithm is effective for heterogeneous links as well as for homogeneous links and that the performance of VRC is little affected by the transport protocol.

5.5.2. Case D2: When web-like short-flows and UDP flows exist

Until now, we have tested the performance of VRC with persistent long-lived TCP flows. In this simulation, we investigate the effect of web-like short-lived flows and unresponsive UDP flows when they coexist with long-lived TCP flows. We generate web-like mice traffic using *on/off* traffic, whose burst time and idle time are taken from the Pareto distributions [25]. Both the average burst time and idle time are set to 1 s. During the *on* periods, packets are generated at a constant burst rate (e.g., 128 Kb/s), whereas no packets are generated during the *off* periods. We vary the number of web-like short-flows from 0 to 100. Here, we set the bottleneck link capacity to T3 rate (45 Mb/s). The maximum rate generated by these short-flows is 12.5 Mb/s representing up to about 28% of the bottleneck link capacity. We also generate greedy and unresponsive constant bit rate (CBR) traffic to test the effect of UDP traffic. We set the portion of UDP traffic load to 10 percent of the bottleneck link capacity. From a control-theoretic viewpoint, we can consider these web-like short flows and UDP flows as disturbance or noise.

Fig. 15 shows the performances of various AQM schemes when long-lived flows and web-like short-flows coexist. Fig. 15(a) and (c) show promising queue regulation performance and high

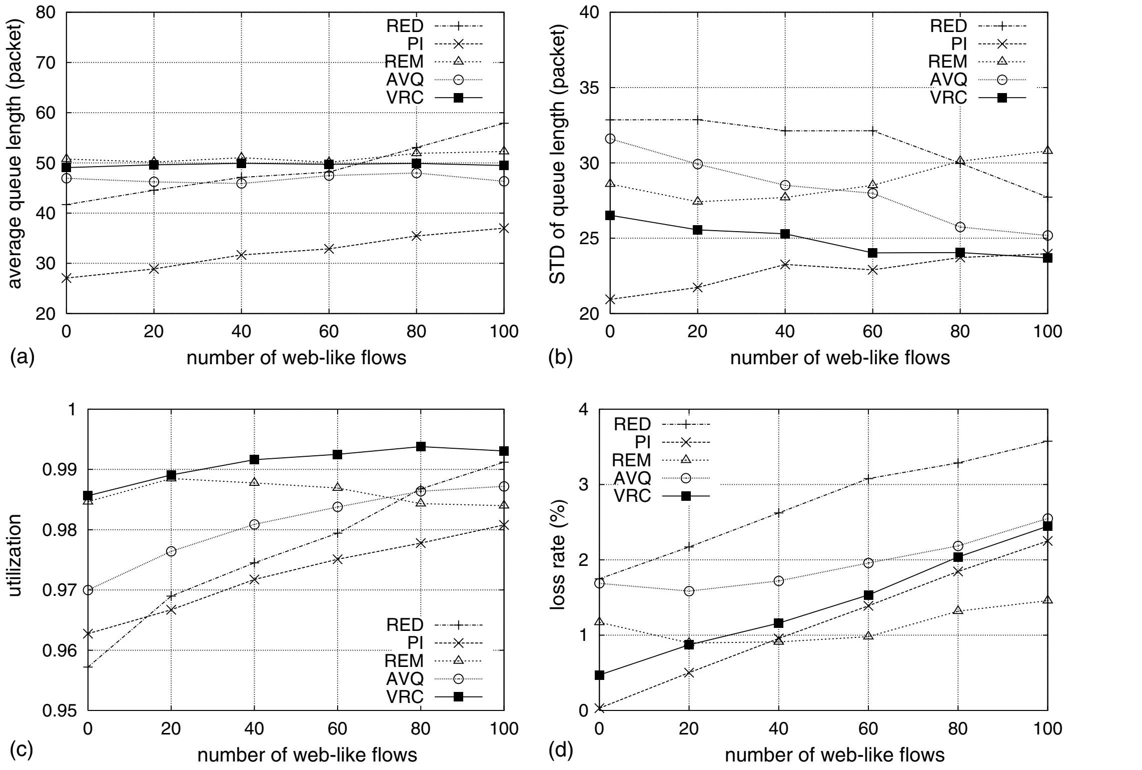


Fig. 15. Performance comparison of several AQM schemes when web-like short-flows exist (Case D2): (a) average queue length, (b) STD of queue length, (c) utilization, (d) loss rate.

Table 5
Performance comparison of various AQM schemes when UDP flows coexist with TCP flows (Case D2)

Criterion	RED	PI	REM	AVQ	**VRC**
Average queue length	41.70/42.59	27.06/29.88	50.75/49.02	46.93/48.37	**49.06/49.47**
STD of queue length	32.86/34.07	20.94/23.08	28.60/32.65	31.60/33.84	**26.52/27.85**
Utilization (%)	95.72/94.85	96.27/96.15	98.46/96.91	97.66/96.82	**98.56/98.01**
Loss rate (%)	1.748/2.374	0.031/0.754	1.171/2.054	1.688/1.832	**0.471/0.797**

(without UDP/with UDP).

utilization of VRC, regardless of the web-like mice traffic load. However, as the web-like traffic load increases, the packet loss rates of all the AQM schemes increase.

On the other hand, the effect of UDP flows on the performance indices can be investigated using Table 5. The average queue lengths and the STD's of all the AQM schemes are little affected by the introduction of the UDP flows. For all AQM schemes, the utilization is slightly reduced and the loss rate is slightly increased. In spite of the existence of UDP flows, VRC still maintains good regulation performance and outperforms the other schemes for utilization.

5.5.3. Case D3: Multiple bottleneck topology

We extend the simple single bottleneck topology to a multiple bottleneck topology, in order to create more realistic scenarios. This simulation uses a 5-level bottleneck topology as shown in Fig. 16. The capacity and propagation delay of each link between routers is set to T3 rate (45 Mb/s) and [5–10] ms, respectively, while the links between routers and cross-traffic senders/receivers have a capacity of 100 BaseT rate (100 Mb/s) and a propagation delay of [5–20] ms. There are N flows traversing all links and N_k flows traversing each individual link k. We set N to 100, and N_1, N_2, N_3, N_4, and N_5 to 50, 100, 150, 100, and 50, respectively. Link 3 is the most heavily congested link, and thereafter congestion becomes less heavy with each successive link.

We observe the performance indices at every link and plot them in Fig. 17. In the case of RED and PI, the average queue length varies according to the degree of congestion; as the link becomes more congested, the queue length increases accordingly, and vice versa. However, the queue lengths of VRC are kept close to the value of $q_{\rm ref}$ and STD's of VRC are relatively small, confirming excellent queue regulation performance of VRC, even in the case of a multiple congested link.

Fig. 17(c) shows that VRC maintains a outstanding high level of utilization in almost all the links, however, the utilization levels of the other AQM schemes deteriorate at the link 4 and the link 5, which are the links situated immediately after the most heavily congested link (link 3). As

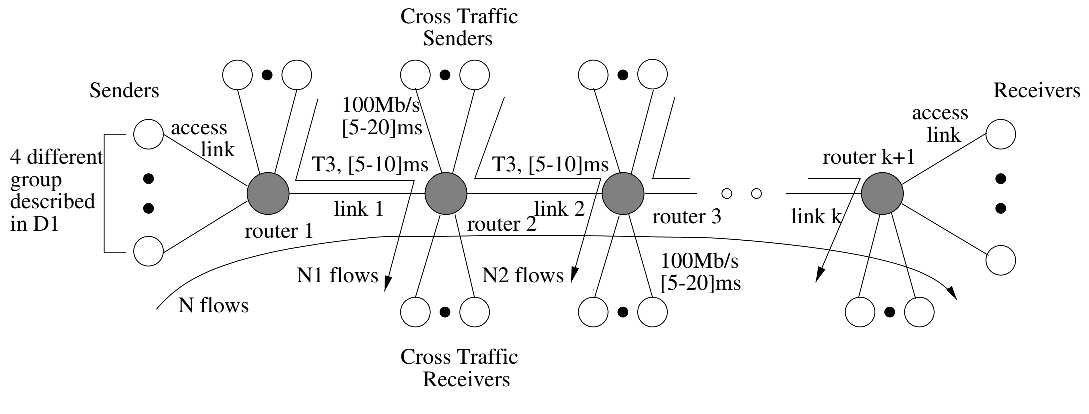


Fig. 16. Multiple bottleneck topology.

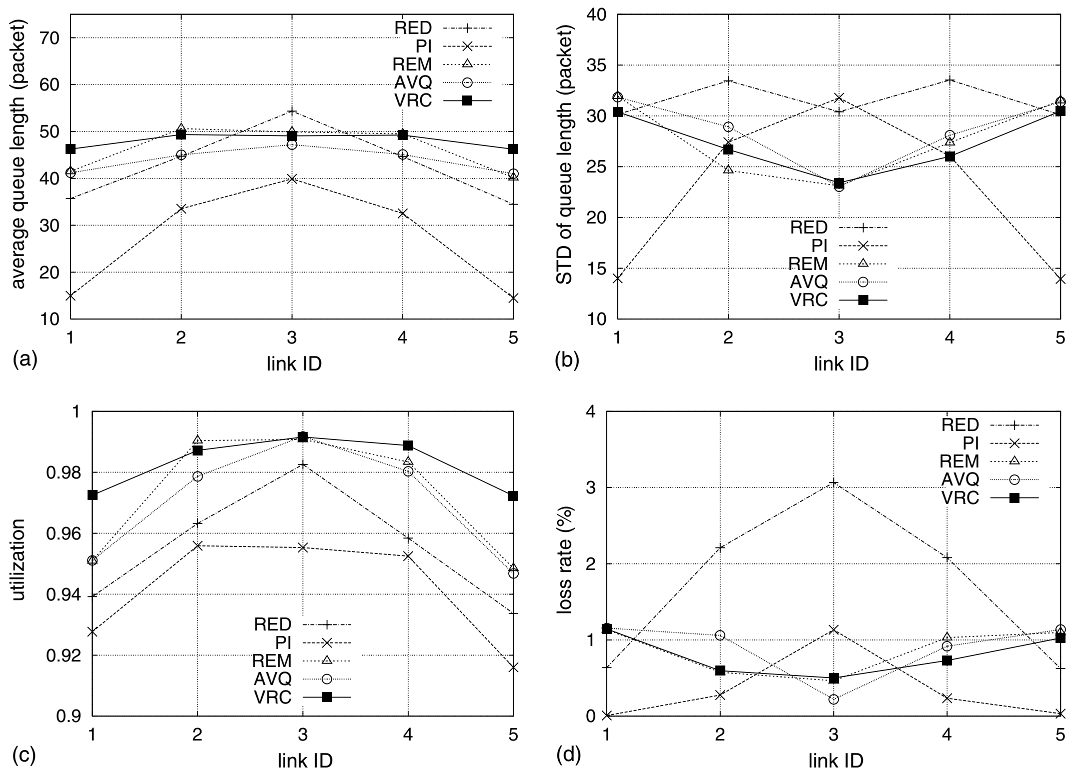


Fig. 17. The performance comparison of several AQM schemes under multiple bottleneck topology (Case D3): (a) average queue length, (b) STD of queue length, (c) utilization, (d) loss rate.

shown in Fig. 17(d), the packet loss rates of all of the AQM schemes are acceptable except for RED. As the link becomes more or less congested, the packet loss rate of RED also significantly increases or decreases, because of its load-dependent property.

We can see that the VRC algorithm is still effective in the case of a multiple congested link, with respect to queue regulation and utilization.

6. Conclusion

The stability of the VRC algorithm, which provides improved response to changing traffic load with high utilization and small loss rate, has been analyzed. It has been shown that the input rate converges to the target rate in the steady-state, with the simple assumption that the throughput of TCP decreases as the dropping probability increases. Using the linearized TCP model with time delay, we derived a sufficient and necessary condition for system stability in terms of the control parameters. For the purpose of providing practical design guideline for parameter setting, we also provided closed-form sufficient conditions for system stability. We validated our analysis via simulations and presented a comparative analysis of various AQM schemes, including RED, PI, REM, DRED, and AVQ.

The simulation results showed that VRC stabilizes the queue length at a value close to the target length with small variation, while maintaining high utilization and small loss rate in most of the cases. Through extensive and propound simulations, we also confirmed that (i) the transient response of VRC outperforms that of the other AQM schemes when the traffic load changes dynamically, (ii) the performance of VRC is almost immune to changes in the system parameters, such as the number of connections, the RTT, the link capacity, and the buffer size, and hence VRC works well for a wide range of system parameters, (iii) the promising performance of VRC holds up in the case of more realistic network scenarios, including heterogeneous links, links with both web-like short flows and UDP flows, and a multiple bottleneck topology.

These results show that VRC is a practical and realistic AQM scheme and is a strong candidate for future AQM scheme.

Appendix A

A.1. Derivation of (14)

We rewrite the dynamic equations (6), (7) and (11) as

$$\begin{aligned}\dot{W}(t) &= f(W(t), W(t-R(t)),\\ &\qquad q(t), q(t-R(t)), z(t-R(t))),\\ \dot{q}(t) &= g(W(t), q(t)),\\ \dot{z}(t) &= q(t) - q_{\mathrm{ref}}.\end{aligned} \tag{A.1}$$

Then, the linearized model around (W^*, q^*, z^*) can be written as follows:

$$\begin{aligned}\delta\dot{W}(t) &= -A_1\delta W(t) - A_2\delta W(t-R(t))|_{R(t)=R^*}\\ &\quad - A_3\delta q(t) - A_4\delta q(t-R(t))\Big|_{R(t)=R^*}\\ &\quad - A_5\delta z(t-R(t))|_{R(t)=R^*},\\ \delta\dot{q}(t) &= -B_1\delta W(t) - B_2\delta q(t),\\ \delta\dot{z}(t) &= \delta q(t),\end{aligned} \tag{A.2}$$

where

$$\begin{aligned}A_1 &= -\frac{\partial f(\cdot)}{\partial W(t)} = \tau_1,\\ A_2 &= -\frac{\partial f(\cdot)}{\partial W(t-R(t))|_{R(t)=R^*}} = \tau_1 + K_{\mathrm{D}}\tau_2,\\ A_3 &= -\frac{\partial f(\cdot)}{\partial q(t)} = \frac{\tau_1}{N},\\ A_4 &= -\frac{\partial f(\cdot)}{\partial q(t-R(t))|_{R(t)=R^*}} = -\frac{\tau_1}{N} - \frac{K_{\mathrm{D}}\tau_2}{N} + \frac{K_{\mathrm{P}}\tau_2 R^*}{N},\\ A_5 &= -\frac{\partial f(\cdot)}{\partial z(t-R(t))|_{R(t)=R^*}} = \frac{K_{\mathrm{I}}\tau_2 R^*}{N},\\ B_1 &= -\frac{\partial g(\cdot)}{\partial W(t)} = -\frac{N}{R^*},\\ B_2 &= -\frac{\partial g(\cdot)}{\partial q(t)} = \frac{1}{R^*}.\end{aligned}$$

A.2. Proof of the Theorem

Starting with (16), which is the characteristic equation of a linear time-invariant system, we can apply the Routh–Hurwitz stability criterion [19]. The necessary and sufficient conditions for the approximated system to be stable are represented as

$$(a_1a_2 - a_3) > 0, \tag{A.3}$$

$$a_3(a_1a_2 - a_3) - a_1^2a_4 > 0. \tag{A.4}$$

For the sake of simplicity, let us denote b_1 and b_2 as

$$b_1 = a_1a_2 - a_3, \tag{A.5}$$

$$b_2 = a_3b_1 - a_1^2a_4, \tag{A.6}$$

respectively. Note that a_1, a_2, a_3, and a_4 are all positive. If (A.4) is satisfied, i.e., $b_2 > 0$, then $b_1 > (a_1^2a_4)/a_3$ from (A.6), and (A.3) is satisfied. Therefore, (A.3) is redundant and only (A.4) is required for the stability condition. Rewriting (A.4) in terms of K_{D}, K_{P} and K_{I}, it becomes (17). □

A.3. Proof of the Corollary

We find the ranges of K_{D}, K_{P}, and K_{I} satisfying (17). Start with taking an arbitrary non-negative constant value for K_{D}. Then, we can rewrite (17) as a second-order inequality with respect to K_{P}:

$$(2\tau_1 + \tau_2RK_{\mathrm{P}})(a_1a_2R^2 - 2\tau_1 - \tau_2RK_{\mathrm{P}}) > a_1^2\tau_2R^3K_{\mathrm{I}}. \tag{A.7}$$

Note that a_1 and a_2 do not contain any terms of K_{P} and K_{I}. Let $f(K_{\mathrm{P}}) = (2\tau_1 + \tau_2RK_{\mathrm{P}})(a_1a_2R^2 - 2\tau_1 - \tau_2RK_{\mathrm{P}})$. In order to find the conditions for K_{P} and K_{I} to satisfy (A.7), we represent both sides of the inequality (A.7) in $(K_{\mathrm{P}}, f(K_{\mathrm{P}}))$ plane. We define

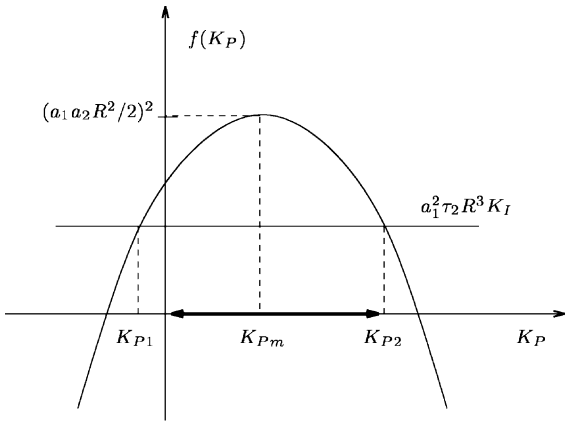


Fig. 18. Representation of (A.7) in $(K_P, f(K_P))$ plane.

K_{Pm} be the value of K_P maximizing $f(K_P)$, i.e., $K_{Pm} = arg(\max f(K_P))$. Note that, for any non-negative K_D, $K_{Pm} = (a_1 a_2 R^2 - 4\tau_1)/(2\tau_2 R)$ is positive and that $f(0)$ is also positive.

Since $K_{Pm} > 0$ and $f(0) > 0$, we can plot $f(K_P)$ as Fig. 18. From Fig. 18, if the horizontal line $a_1^2\tau_2 R^3 K_I$ lies below $\max(f(K_P)) = (a_1 a_2 R^2/2)^2$, i.e.,

$$K_I < K_I^{\max} = \frac{\max(f(K_P))}{a_1^2\tau_2 R^3} = \frac{(a_2 R)^2}{4\tau_2 R}, \tag{A.8}$$

which is equivalent to (20), then there exist K_P and K_I satisfying (A.7). Thus, if we set K_I to be smaller than $K_I^{\max}$, the corresponding range for K_P satisfying (A.7) can be obtained as $\max(0, K_{P1}) < K_P < K_{P2}$, where K_{P1} and K_{P2} are the solutions of the second-order equality with respect to K_P, $f(K_P) - a_1^2\tau_2 R^3 K_I = 0$.

Consequently, if the control parameters K_I and K_P satisfy $0 < K_I < K_I^{\max}$ and $\max(0, K_{P1}) < K_P < K_{P2}$ for any non-negative K_D, then the approximated system of (14) is stable. □

References

[1] V. Jacobson, Congestion avoidance and control, in: Proceedings of ACM SIGCOMM, 1988, pp. 314–329.

[2] D. Chiu, R. Jain, Analysis of the increase/decrease algorithms for congestion avoidance in computer networks, Computer Networks and ISDN 17 (1) (1989) 1–14.

[3] S. Floyd, V. Jacobson, Random early detection gateways for congestion avoidance, IEEE/ACM Transactions on Networking 1 (4) (1993) 397–413.

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

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

[6] M. Christiansen, K. Jeffay, D. Ott, F. Smith, Tuning RED for web traffic, in: Proceedings of ACM SIGCOMM, 2000, pp. 139–150.

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

[8] S. Athuraliya, S.H. Low, V.H. Li, Q. Yin, REM: active queue management, IEEE Network 15 (3) (2001) 48–53.

[9] J. Aweya, M. Ouellette, D.Y. Montuno, A control theoretic approach to active queue management, Computer Networks 36 (2–3) (2001) 203–235.

[10] K. Ramakrishnan, S. Floyd, A proposal to add Explicit Congestion Notification (ECN) to IP, RFC 2481, 1999.

[11] S. Kunniyur, R. Srikant, Analysis and design of an adaptive virtual queue (AVQ) algorithm for active queue management, in: Proceedings of ACM SIGCOMM, 2001, pp. 123–134.

[12] H. Lim, K.-J. Park, E.-C. Park, C.-H. Choi, Virtual rate control algorithm for active queue management in TCP networks, IEE Electronics Letters 38 (16) (2002) 873–874.

[13] E.-C. Park, H. Lim, K.-J. Park, C.-H. Choi, Analysis of the virtual rate control algorithm in TCP networks, in: Proceedings of IEEE GLOBECOM, vol. 3, 2002, pp. 2619–2623.

[14] H. Lim, K.-J. Park, E.-C. Park, C.-H. Choi, Active queue management algorithm with a rate regulator, in: Proceedings of the 15th IFAC World Congress on Automatic Control, Barcelona, Spain, 2002.

[15] K.B. Kim, S.H. Low, Analysis and design of AQM for stabilizing TCP, Tech. Rep., Caltech CSTR:2002.009.

[16] I. Stoica, S. Shenker, H. Zhang, Core-stateless fair queueing: achieving approximately fair bandwidth allocations in high speed networks, in: Proceedings of ACM SIGCOMM, 1998, pp. 118–130.

[17] V. Misra, W. Gong, D. Towsley, Fluid-based analysis of a network of AQM routers supporting TCP flows with an application to RED, in: Proceedings of ACM SIGCOMM, 2000, pp. 151–160.

[18] J. Padhye, V. Firoiu, D. Towsley, J. Kurose, Modeling TCP throughput: a simple model and empirical validation, in: Proceedings of ACM SIGCOMM, 1998, pp. 303–314.

[19] G.F. Franklin, J.D. Powell, A. Emami-Naeini, Feedback Control of Dynamic Systems, third ed., Addsion-Wesley, New York, 1995.

[20] S.H. Low, F. Paganini, J. Wang, S. Adlakha, J.C. Doyle, Dynamics of TCP/RED and a scalable control, in: Proceedings of IEEE INFOCOM, 2002, pp. 239–248.

[21] C.V. Hollot, V. Misra, D. Towsley, W. Gong, Analysis and design of controllers for AQM routers supporting TCP flows, IEEE Transactions on Automatic Control 47 (6) (2002) 945–959.

[22] S. Kunniyur, R. Srikant, End-to-end congestion control schemes: Utility functions, random losses and ECN marks, in: Proceedings of IEEE INFOCOM, 2000, pp. 1323–1332.
[23] J. Fall, K. Varadhan, The ns manual, http://www.isi.edu/nsnam/ns/ns-documentation (2001).
[24] S. Floyd, RED queue management, http://www.aciri.org/floyd/red.html (2001).
[25] M.E. Crovella, A. Bestavros, Self-similarity in world wide web traffic: evidence and possible causes, IEEE/ACM Transactions on Networking 5 (6) (1997) 835–846.

Eun-Chan Park received the B.S. and M.S. degrees from the School of Electrical Engineering and Computer Science, Seoul National University, Seoul, Korea, in 1999 and 2001, respectively. He is currently pursuing the Ph.D. degree at Seoul National University. His current research interests include congestion control, QoS in IP networks, and network performance analysis.

Hyuk Lim received the B.S. and M.S. degrees in 1996 and 1998, respectively, from the School of Electrical Engineering and Computer Science, Seoul National University, Seoul, Korea, where he is currently working towards the Ph.D. degree. His research interests include network modelling, measurement, and congestion control.

Kyung-Joon Park received the B.S. and M.S. degrees from the School of Electrical Engineering and Computer Science, Seoul National University, Seoul, Korea, in 1998 and 2000, respectively. He is currently pursuing his Ph.D. degree at Seoul National University. His current research interests include network modeling, bandwidth provisioning, and active queue management.

Chong-Ho Choi (S'77.M'78) received the B.S. degree from Seoul National University, Seoul, Korea, in 1970 and the M.S. and Ph.D. degrees from the University of Florida, Gainesville, in 1975 and 1978, respectively. He was a Senior Researcher with the Korea Institute of Technology from 1978 to 1980. He is currently a Professor in the School of Electrical Engineering and Computer Science, Seoul National University. His research interests include adaptive control, system identification, neural networks, network modelling, and their applications.