A State Feedback Control Approach to Stabilizing Queues for ECN-Enabled TCP Connections

Yuan Gao
Statistics Research, Math Center
Bell Labs, Murray Hill, NJ
yuangao@research.bell-labs.com

Jennifer C. Hou
Computer Science Department
University of Illinois, Urbana-Champaign
jhou@cs.uiuc.edu

***Abstract*— In this paper, we present an analytical TCP model that takes into account of several issues that were ignored in the other existing models (such as those in [15], [19]), i.e., (i) the congestion window is not gradually decreased at the rate of $\frac{w^2p}{2}$, but suddenly halved upon receipt of congestion indication and (ii) the congestion window is halved at most once during one round-trip time (RTT). We also include the delayed ACK option in the model. With the use of state feedback control theory, we then design, based on the enhanced TCP model, an AQM controller to stabilize the queue length at routers. The performance of the new controller is shown, via ns-2 simulation to outperform several other schemes under a variety of network scenarios and traffic loads, in terms of fluctuation in the queue length, link utilization, and packet loss ratio.**

***Index Terms*— AQM, state feedback control, TCP/IP, and ECN.**

I. Introduction

A central tenet of the current Internet architecture is that congestion control is performed mainly by TCP at end hosts. However, as new applications (which may not deploy TCP for congestion control), e.g. continuous media applications, become widely deployed on the Internet, it becomes difficult, if not impossible, to exclusively rely on end hosts to perform end-to-end congestion control. It has been agreed upon that the network itself must now participate in congestion control and resource management. To this end, the Internet Engineering Task Force (IETF) is advocating deployment of explicit congestion notification (ECN) [22] and active queue management mechanisms (AQM) at routers as a means to congestion control. By AQM, we mean core routers inside the network measure congestion level on their links and explicitly signal traffic sources by selectively marking/dropping packets, before congestion actually takes place. Traffic sources then response to the congestion signal by adjusting their control window sizes (or equivalently sending rates).

Several active queue management mechanisms have been proposed, e.g., random drop [11], early packet discard [18], early random drop [14], random early detection (RED) [8] and its variations (FRED [17], stabilized RED (SRED) [20], and balanced RED (BRED) [1]), BLUE [7], REM [2], PI controller [12], and AVQ [16], among which RED has perhaps received the most attention. RED has been shown to successfully prevent global synchronization , reduce packet loss ratios, and minimize the bias against bursty sources. The major pitfalls of RED are, however, that its performance is sensitive to the level of network load and the parameter settings, that the queue length usually oscillates significantly [3], and that it cannot achieve simultaneously high link utilization and low packet losses (due to the fact that RED directly couples the packet dropping probability with the queue length) [7], [2], [16].

In the past few years, significant research efforts have been made to study/improve the performance of RED (or, in general, AQM). These efforts can be roughly classified into three categories. In the first category, approaches are devised to adaptively, on-line adjust RED parameters according to the condition of network congestion [1], [17], [20], [7]. Most of the approaches proposed in this category are heuristic-based and validated through simulation. In the second category, interactive behaviors of TCP connections and AQM controllers are characterized as an gradient optimization problem [15], [16], [21], with the objective of maximizing the utility of the network. The optimization based approaches proposed in this category primarily focus on the steady state equilibrium, rather than the transient behavior, of the queue and TCP. The third category envisions a network that consists of TCP connections and AQM routers as a dynamic feedback control system, in which AQM routers act as controllers and TCP traffic sources act as plants [15], [19]. Several analytical models are proposed to approximate the dynamic additive increase and multiplicative decrease (AIMD) behaviors of TCP in conjunction with AQM [15], [19]. Automatic control theory is then used to analyze (with respect to controllability and stability) and design AQM controllers. Our proposed work falls in the third category.

The analytical models proposed in the third category provide new insights on designing better AQM controllers, choosing appropriate parameters, and detecting problematic parameter settings. All models try to describe the main dynamics of TCP in the congestion avoidance phase. Succinctly, in the TCP congestion avoidance phase, a TCP connection uses AIMD algorithm to adjust its congestion window size *cwnd*. That is, for each positive acknowledgment received, it increases its *cwnd* by $\frac{1}{cwnd}$, and in the case of congestion indication (i.e., receipt of three duplicate acknowledgments or ECN), it reduces its *cwnd* by half to $\frac{cwnd}{2}$ [13]. These analytical models characterize the AIMD behavior of TCP as follows [15]: as each positive acknowledgment increases *cwnd* by $1/cwnd$ and each congestion indication reduces the *cwnd* by half to $\frac{cwnd}{2}$, the rate at which the expected congestion window size

changes is expressed as $\frac{1-p}{\tau} - \frac{w^2 p}{2\tau}$, where τ is the round-trip time (RTT) of a TCP connection, w is the current congestion window size, and p is the dropping or marking probability. The changing rate is then taken as the TCP dynamic behavior for analysis and design.

Although the controllers designed under the aforementioned models are shown, via simulation, to perform well, two effects are not considered in these models. First, the congestion window is not *gradually* decreased at the rate of $\frac{w^2 p}{2}$, but *suddenly* halved upon receipt of congestion indication. Second, the congestion window is halved at most once during one RTT. In this paper, we present an enhanced model that takes into account of these effects as well as the TCP option of delay acknowledgment. We show that the new model characterizes the TCP dynamics more realistically and that under the new model $cwnd$ decreases faster. We then analyze the stability of its linearized model and design, with the use of state feedback control theory, a controller to stabilize the queue at an AQM router. The resulting AQM controller is called the *state feedback controller (SFC)*. All the algorithm implementation and parameter setting issues are carefully considered and validated through theoretical reasoning. We also evaluate via *ns-2* simulation the performance of the new controller and compare it against other existing schemes. The simulation results show that SFC outperforms other schemes in terms of fluctuation in the queue length, link utilization, and packet loss ratio. In particular, SFC can reduce packet loss by more than 50% as compared to other schemes and yet achieve full link utilization. As compared to PI controller proposed in [12], SFC achieves 10% more link utilization.

The rest of the paper is organized as follows. In Section II, we present an enhanced TCP model that considers effects aforementioned and ignored in the previous models. In Section III, we linearize our model and analyze its local stability. In Section IV, we design a state feedback controller based on our linearized model. In Section V, we discuss through theoretical reasoning, the algorithm implementation and parameter setting issues. Following that, we summarize and categorize related work in Section VI and present simulation results in Section VII. The paper concludes in Section VIII.

II. An Enhanced TCP Model

In this section, we take into account of (i) effects that were previously ignored in other analytical models and (ii) the TCP option of delay acknowledgment, and derive a new model to characterize the expected transient behavior of the TCP congestion window in the congestion avoidance phase. We also compare the new model against existing models.

Similar to the existing models [15], [19], we assume that (**A1**) TCP connections operate in the congestion avoidance phase[1]; (**A2**) the change in the packet dropping/marking probability is insignificant in one round trip time, τ; and (**A3**) packets are marked independently. While the other models take the expected rate at which *cwnd* changes over the interval between two acknowledgments as the approximate *cwnd* change rate, we calculate the expected *cwnd* change, $E(\Delta w)$, over one RTT τ, and use $\frac{E(\Delta w)}{\tau}$ as the *cwnd* change rate. As will be clearer below, with this subtle change we will be able to figure in effects ignored in the other models.

We model the TCP behavior in the congestion avoidance phase in terms of "cycles." An old cycle ends and a new cycle begins when all data packets in the previous congestion window are acknowledged. In the time axis, a cycle takes approximately one RTT, τ. Let the size of the current congestion window and the size of the congestion window one RTT before be denoted, respectively, as w and w'. By definition, totally w' packets are acknowledged in the current cycle. Let the number of packets that are acknowledged by a received acknowledgment (ACK) be denoted as b. If each ACK acknowledges only one packet, $b = 1$. On the other hand, if the delayed ACK option is used (i.e., one ACK is sent for every two data packets received), $b = 2$.

If the ECN bit of the kth acknowledgment is marked, the current congestion window size, $cwnd = w + \frac{k-1}{bw}$, will be halved. As the congestion window is halved at most once in a round trip time, after one cycle, the change in the congestion window size (in unit of MSS) is $\frac{1}{b} - \frac{w}{2} - \frac{k-1}{2bw}$. Let p denote the probability that the ECN bit of a packet is marked in the current cycle. Then, under assumptions (**A2**) and (**A3**), the probability that the kth data packet is the first with the ECN bit marked in the current congestion window (so that the kth acknowledgment carry the ECN indication) is $(1-p)^{k-1}p$. If no ECN bit is marked or three duplicate ACKs received, *cwnd* will be increased by $\frac{w'}{bw}$, and the corresponding probability is $(1-p)^{w'}$. By the end of each cycle, the expected change in the congestion window size can be expressed as

$$\begin{aligned} E(\Delta w) &= \sum_{k=1}^{w'} (1-p)^{k-1} p \left(\frac{2w' - k + 1}{2bw} - \frac{w}{2} \right) + \frac{w'(1-p)^{w'}}{bw} \\ &= -\frac{1 - (1-p)^{w'} w' p - (1-p)^{w'} - p + p(1-p)^{w'}}{2bwp} \\ &\quad -\frac{w}{2}\left[1 - (1-p)^{w'}\right] + \frac{w'}{bw}. \end{aligned} \tag{1}$$

When p is small, i.e. $wp \ll 1$, we can consider first-order items of p while ignoring high-order items and simplify Eq. (1) to

$$\begin{aligned} E(\Delta w) &\approx -\frac{w'(w'-1)}{2bw} p - \frac{ww'}{2} p + \frac{w'}{bw} \\ &= \frac{w'}{bw} + \left(\frac{w'}{2bw} - \frac{w'^2}{2bw}\right) p - \frac{ww'}{2} p. \end{aligned} \tag{2}$$

The rate at which *cwnd* changes can then be approximated as

$$\frac{dE(w)}{dt} \approx \frac{E(\Delta w)}{\tau}. \tag{3}$$

Notice that $w' = w(t-\tau)$ and the packet dropping probability that a TCP connection perceives also incurs a time delay τ, i.e., $p = p(t-\tau)$. Substituting w' and $p(t-\tau)$ into Eq. (3), we have

$$\begin{aligned} \frac{dE(w)}{dt} &= \frac{w(t-\tau)}{b\tau w(t)} + \left[\frac{w(t-\tau)}{2b\tau w(t)} - \frac{w^2(t-\tau)}{2b\tau w(t)}\right] p(t-\tau) \\ &\quad -\frac{w(t)w(t-\tau)}{2\tau} p(t-\tau). \end{aligned} \tag{4}$$

[1]It has been observed in [5] that the majority of Internet traffic is still dominated by long-lived TCP connections and most long-lived TCP connections operate in the congestion avoidance phase most of the time.

Comparison against other models: Now we compare the newly derived model with the other analytical models. Misra's model [19] (which is scaled by b to take into account of the delayed ACK option) is given below:

$$\frac{dE(w)}{dt} = \frac{1}{b\tau} - \frac{w(t)w(t-\tau)}{2\tau}p(t-\tau). \quad (5)$$

Comparing Eqs. (4) and (5), we can see that when $w(t-\tau) \approx w(t)$ and $w(t-\tau) > 1$ (which is always true, as we assume TCP connections operate in the congestion avoidance phase and hence $w(t-\tau) \geq 3$.),

$$\frac{w(t-\tau)}{2b\tau w(t)} - \frac{w^2(t-\tau)}{2b\tau w(t)} < 0. \quad (6)$$

Eq. (6) implies the second term in Eq. (4) is negative, and hence the congestion window size decreases faster in our model than in Misra's model.

Let the sending rate be $x(t) = \frac{w(t)}{\tau}$. Then Eq. (4) can be re-written as

$$\begin{aligned}\frac{dx}{dt} &= \frac{x(t-\tau)}{b\tau^2 x(t)} + \left[\frac{x(t-\tau)}{2b\tau^2 x(t)} - \frac{x^2(t-\tau)}{2b\tau x(t)}\right] p(t-\tau) \\ &\quad - \frac{x(t)x(t-\tau)}{2}p(t-\tau). \end{aligned} \quad (7)$$

Kelly's model [15] (which is again scaled by b to take into account of the delayed ACK option) is given below:

$$\frac{dx}{dt} = \frac{x(t-\tau)}{b\tau^2 x(t)} - \frac{x(t-\tau)}{b\tau^2 x(t)}p(t-\tau) - \frac{x(t)x(t-\tau)}{2}\cdot p(t-\tau). \quad (8)$$

Comparing Eqs. (7) and (8), we can see when $x(t-\tau) > \frac{3}{\tau}$, we have

$$\frac{x^2(t-\tau)}{2b\tau x(t)} - \frac{x(t-\tau)}{2b\tau^2 x(t)} > \frac{x(t-\tau)}{b\tau^2 x(t)}, \quad (9)$$

i.e., the second term in Eq. (7) is less than that in Eq. (8). By assumption (**A1**), the congestion window size should always be greater than 3 MSS, and hence $x(t-\tau) > \frac{3}{\tau}$ always holds. As a result, the sending rate decreases faster in our model than in Kelly's model.

This fact that the congestion window size decreases faster cautions us for the importance of designing appropriate AQM controllers as the impact of the packet dropping/marking probability on the congestion window change is larger than the other models expect.

III. Analysis of Interaction Between TCP and AQM

In this section, we consider a system in which N homogeneous TCP connections traverse a single bottleneck link with bandwidth C. By homogeneous, we mean all the TCP connections incur roughly the same round trip time and share the same bottleneck link, although they do not necessarily traverse the same end-to-end path. Let the queue length on the bottleneck link be denoted q and the congestion window size of each TCP connection w. The dynamic system can be described by

$$\begin{cases} \dot{q} &\triangleq g(w(t),q) = \frac{N}{\tau}w - C, \\ \dot{w} &\triangleq f(w(t),w(t-\tau),p) = \frac{w(t-\tau)}{b\tau w(t)} - \frac{w(t)w(t-\tau)}{2\tau}p(t-\tau) \\ &\quad + \left[\frac{w(t-\tau)}{2b\tau w(t)} - \frac{w^2(t-\tau)}{2b\tau w(t)}\right]p(t-\tau). \end{cases} \quad (10)$$

The first differential equation (Eq. (10)) states that the queue length is an integral of the difference between the packet arrival rate and the link capacity. The second differential equation (Eq. ((10)) describes the dynamic behavior of the TCP congestion window that is developed in Section II.

As the system model (Eq. (10)) is nonlinear with a time delay, it is impossible to analyze it analytically. Hence, we will first approximate the system model with its small-deviation linearized model around an operating point, say (w_0, p_0), to analyze its local stability. We assume that τ is constant. Let $\delta w \triangleq w - w_0$ and $\delta p \triangleq p - p_0$, in which δw and δp are, respectively, deviations of the congestion window and the dropping probability from the operating point. By setting $g(w(t),q) = 0$ and $f(w(t),w(t-\tau),p) = 0$, we have

$$\begin{aligned} w_0 &= \frac{\tau C}{N}, \\ p_0 &= \frac{2}{bw_0^2 + w_0 - 1} = \frac{2N^2}{b\tau^2C^2 + \tau CN - N^2}. \end{aligned} \quad (11)$$

Also,

$$\begin{aligned} \frac{\partial g}{\partial w} &= \frac{N}{\tau}, \\ \frac{\partial f}{\partial w} &= -\frac{p_0 + 2bw_0p_0}{2b\tau}, \\ \frac{\partial f}{\partial p} &= \frac{1}{2b\tau} - \frac{w_0}{2b\tau} - \frac{w_0^2}{2\tau} = -\frac{1}{b\tau p_0}. \end{aligned}$$

Hence, the equations that characterize the system dynamics around the operating point are

$$\begin{aligned} \delta\dot{q} &= \frac{\partial g}{\partial w}\delta w = \frac{N}{\tau}\delta w, \\ \delta\dot{w} &= \frac{\partial f}{\partial w}\delta w + \frac{\partial f}{\partial p}\delta p \\ &= -\frac{p_0 + 2bw_0p_0}{2b\tau}\delta w - \frac{1}{b\tau p_0}\delta p(t-\tau). \end{aligned} \quad (12)$$

For the system model to be meaningful around the operating point, we require that $w_0 \geq 0$ and $0 \leq p_0 \leq 1$. As $\tau > 0$, $C > 0$ and $N > 0$, w_0 is always greater than 0. To satisfy $0 \leq p_0 \leq 1$, we should have $0 \leq N < \frac{1+\sqrt{12b+1}}{6}\tau C$. All our analysis and design will be restricted to the parameter region mentioned above, because otherwise the system has no equilibrium and can not be stabilized according to the linear model. The transfer function of the system is

$$T(s) = \frac{W(s)}{P(s)} = -\frac{\frac{1}{b\tau p_0}}{s + \frac{p_0+2bw_0p_0}{2b\tau}}e^{-\tau s} \triangleq G(s)e^{-\tau s}, \quad (13)$$

i.e., $T(s)$ is a pure-delay system with a non-delay part $G(s)$. As $G(s)$ has a pole at $-\frac{p_0+2bw_0p_0}{2b\tau}$ that lies on the left half of the complex plane, $G(s)$ is stable.

In summary, the system in which N homogeneous TCP connections traverse a bottleneck link with capacity C can be approximated by the following linear differential equation:

$$\begin{cases} \delta\dot{q} &= \frac{N}{\tau}\delta w, \\ \delta\dot{w} &= -\frac{p_0+2bw_0p_0}{2b\tau}\delta w - \frac{1}{b\tau p_0}\delta p(t-\tau). \end{cases} \quad (14)$$

In the matrix form, the system can be represented as $\dot{x} = Ax + D\delta p(t-\tau)$, where $x = \begin{bmatrix} x_1 & x_2 \end{bmatrix}^T = \begin{bmatrix} \delta q & \delta w \end{bmatrix}^T$, $A = \begin{bmatrix} 0 & \frac{N}{\tau} \\ 0 & -\frac{p_0+2bw_0p_0}{2b\tau} \end{bmatrix}$ and $D = \begin{bmatrix} 0 & -\frac{1}{b\tau p_0} \end{bmatrix}^T$. Since $\begin{bmatrix} D & AD \end{bmatrix}$ is full ranked, the system is controllable. Hence by using the proper control law, we can take system's states, i.e. the queue length at the bottleneck link and the congestion window size, $cwnd$, of TCP connections, to some desirable equilibrium point.

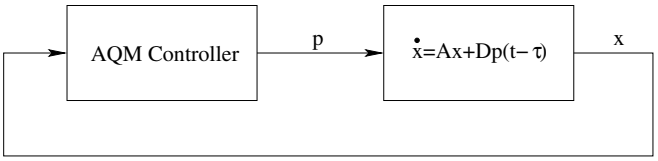


Fig. 1. The system diagram.

IV. State Feedback Controlled AQM

In this section, we design, based on the state feedback control theory, an AQM controller under the linearized model (formulated in Section III), and discuss how tunable parameters should be set to stabilize the system, i.e. to make δq and δw as close to zero as possible. The reasons for using state feedback control are: (i) it is desirable to remove the operation of averaging the queue length (and using it as a congestion index) in several AQM schemes, as it has been believed that this operation brings in more sluggish behaviors to a delay system; (ii) since all system states in state feedback control can be readily obtained or estimated, a state feedback controller can be easily implemented and can quickly respond to system dynamics.

Fig. 1 shows the block diagram of the feedback control system that characterize the interaction between TCP and AQM. The controllable plant is the linearized TCP model and the AQM controller marks the arrived packets with probability p (which is a function of system states) and is the entity to be designed. With the use of state feedback control, we can express the marking/dropping probability as a linear combination of system's states, i.e. $x_1 = \delta q$ and $x_2 = \delta w$. Specifically, let $p(t) = K \cdot x(t)$. We have

$$\begin{aligned} \dot{x} &= Ax + Dp(t-\tau) = Ax + D \cdot K \cdot x(t-\tau), \\ &= \begin{bmatrix} 0 & \frac{N}{\tau^2 C} \\ 0 & \frac{-1-2b w_0}{2b\tau} p_0 \end{bmatrix} x + \begin{bmatrix} 0 & 0 \\ \frac{-k_1}{b\tau p_0} & \frac{-k_2}{b\tau p_0} \end{bmatrix} x(t-\tau) \end{aligned} \quad (15)$$

After Laplace transform, the characteristic polynomial, $D(s)$, of the system can be expressed as:

$$\begin{aligned} D(s) &= \det(sI - A - D \cdot K e^{-s\tau}) \\ &= \frac{p_0 (1+2bw_0)}{2b\tau} \left[\frac{2Nk_1 + 2k_2\tau s}{(1+2bw_0) p_0^2 \tau} \right] e^{-s\tau} \\ &\quad + \frac{p_0 (1+2bw_0)}{2b\tau} \left[1 + \frac{2b\tau}{(1+2bw_0) p_0} s \right] s. \end{aligned} \quad (16)$$

Stable region of k_2: To make the feedback control system stable, we should choose parameters k_1 and k_2 such that the roots of $D(s)$ all lie in the left half complex plane. Here we leverage the results reported in [24]: the sufficient and necessary condition for the system to be stable is that k_2 fulfills the following inequality (after the value of k_2 is determined, k_1 is chosen accordingly; we will elaborate on how to choose k_1 below):

$$\begin{aligned} 0 &\le k_2 \le bp_0 \sqrt{\theta^2 + \frac{(1+2bw_0)^2 p_0^2}{4b^2}} \\ &= \frac{2bN^2}{b\tau^2 C^2 + \tau CN - N^2} \sqrt{\theta^2 + \frac{(1+2bw_0)^2 p_0^2}{4b^2}}, \end{aligned}$$

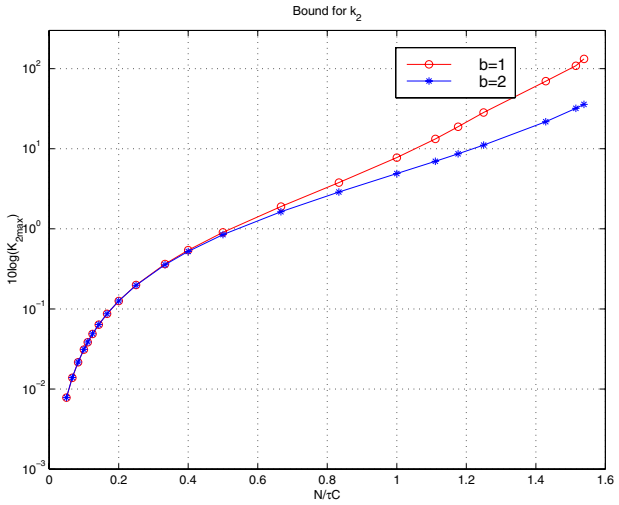


Fig. 2. The bound of k_2.

in which θ is solution to $\tan(\theta) = -\frac{b\left(b\tau^2 C^2 + \tau CN - N^2\right)}{N^2\left(1+2b\frac{\tau C}{N}\right)}\theta$ in the interval of $\left(\frac{\pi}{2}, \pi\right)$.

In summary, let $\alpha = \frac{\tau C}{N}$. The sufficient and necessary condition for system stabilization is that $0 \le k_2 \le \frac{2b}{\alpha^2 b + \alpha - 1} \sqrt{\theta^2 + \frac{1}{b^2}\left(\frac{2ab+1}{\alpha^2 b+\alpha-1}\right)^2} = K_{2\max}$ in which θ is a solution to $\tan(\theta) = -b\frac{b\alpha^2+\alpha-1}{2ab+1}\theta$ in the interval of $\left(\frac{\pi}{2}, \pi\right)$.

The bound of k_2 as a function of $\frac{N}{\tau C}$ in the case of $b = 1$ and $b = 2$ is depicted in Fig. 2. From the figure, we can see that the larger the value of $\frac{N}{\tau C}$, the larger the bound of k_2. This implies when the number, N, of connections gets larger or the round trip time, τ, gets smaller, we can choose a larger value of k_2. Alternatively, if we choose the value of k_2 for a minimal number of connections, say N_{min} and a maximal value of τ, say τ_{max}, then for all $N \ge N_{min}$ and $\tau \le \tau_{max}$, the system is still stable, as $\frac{N}{\tau C} \ge \frac{N_{min}}{\tau_{max} C}$. Also, as shown in Fig. 2, when the delayed ACK option is used, i.e. $b = 2$, the stable region of k_2 is smaller, so we should choose a smaller value of k_2.

Stable region of k_1: After the value of k_2 is determined, the region of k_1 to stabilize the system can be determined by finding the roots of the following equation [24]:

$$k_2 + \frac{(2\alpha b + 1)\cos(z)}{(b\alpha^2 + \alpha - 1)^2} - \frac{bz\sin(z)}{(b\alpha^2 + \alpha - 1)} = 0. \quad (17)$$

Specifically, let the non-negative roots of Eq. (17) be arranged in the increasing order of magnitude and denoted as $z_i, i = 0, 1, 2, \ldots$ (of which $z_0 = 0$). Next we compute $a_i = a(z_i), i = 0, 1, 2, \ldots$ using the following equation:

$$a(z) = \left(\frac{\sqrt{4ab+2}}{\sqrt{N}\left(ba^2 + a - 1\right)} \right)^2 z \left[\sin(z) + \frac{b\left(ba^2 + a - 1\right)}{(1+2ab)} \cos(z) \right]. \quad (18)$$

The lower and upper bound of k_1 will be $0 < k_1 < \min_{i=1,3,5,\ldots} a_i$. In fact, we do not need to find all the roots of Eq. (17). After finding z_{2j+1} that makes $\cos(z_{2j+1}) > 0$, the algorithms can stop and proceeds to find the bound for k_1. In

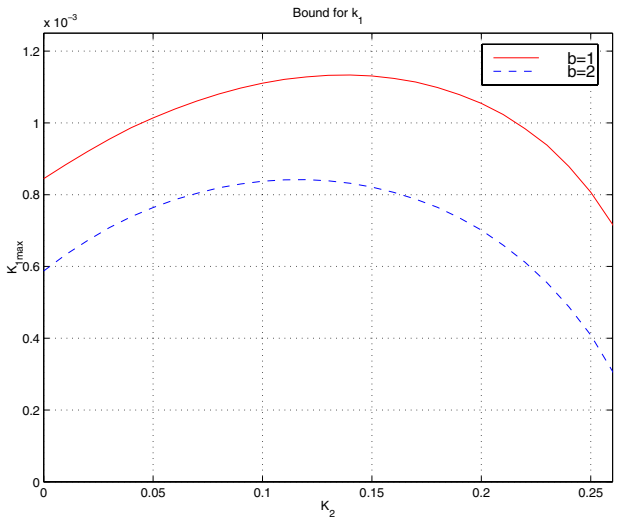


Fig. 3. The bound of k_1.

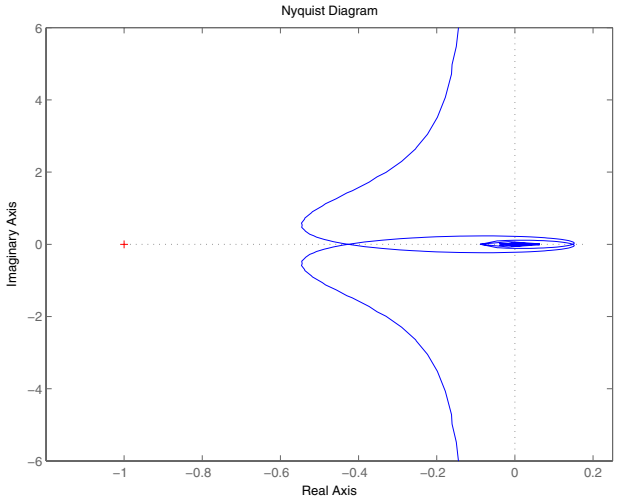


Fig. 4. The Nyquist diagram of the system of interest.

what follows, we give an example to illustrate how to choose these parameters.

Example 1: Given the network parameters: C = 10Mbps = 1250 packets/second with the average packet size 1000 bytes, $N_{min} = 300$, $\tau_{max} = 0.6sec$, and $b = 2$ (i.e., the delayed ACK option is used). As mentioned above, only when $N < \frac{1+\sqrt{12b+1}}{6}\tau C = 750$, the system equilibrium is meaningful. Also, $\frac{N_{min}}{\tau_{max}C} = 0.4$. As shown in Fig. 2, if we choose $0 < k_2 < 0.4$, the system will be stable for $N \geq N_{min}$ and $\tau \geq \tau_{max}$. The bound of k_1 as a function of k_2 in the cases of $b = 1$ and $b = 2$ is shown in Fig. 3. Also as shown in Fig. 3, when the delayed ACK option is used, i.e. $b = 2$, we should choose a smaller value of k_1. Given all the above criteria, we can choose $k_2 = 0.2$ and then $k_1 = 0.0005$. The open-loop Nyquist plot of the system is shown in Fig. 4. As the Nyquist plot does not encircle point (-1, 0), the closed-loop system is stable. ■

/* Called upon arrival of a new packet */
/* Q_{lim} is the buffer size at the router, C_0 is the capacity of the
* outgoing link, q_0 is the desired queue length, and k_1 and $a = \frac{k_2}{C}$
* are parameters chosen to stabilize the queue length */
1. **if** $(q \geq Q_{lim})$ {
2. Drop the packet;
3. Return;
4. }
5. $R \leftarrow$ *R_estimate*();
6. $\delta q \leftarrow q - q_0$;
7. $p \leftarrow k_1 \cdot \delta q + a \cdot (R - C_0)$;
8. **if** $(p < 0)$
9. $p = 0$;
10. **else if** $(p > 1)$
11. $p = 1$;
12. $drop \leftarrow$ Random_uniform(0,1);
13. **if** $(drop > p)$ {
14. Put the packet into the queue;
15. } **else if** (ECN is enabled) {
16. Mark the ECN bit of the packet;
17. Put the packet into the queue;
18. } **else**
19. Drop the packet;
20. Return;

Fig. 5. Enqueue Procedure.

V. Algorithm Implementation and Parameter Setting

The algorithm of the AQM controller is outlined in Fig. 5. Lines 1–4 determine if the queue length already exceeds the limit. If so, the incoming packet is discarded. Lines 5–11 calculate the packet marking/dropping probability, p, according to the current system states, and reset the probability to 0 or 1, if the calculated value exceeds region $[0, 1]$. In Lines 12–19, the packet is marked/dropped with probability p.

Line 7 in the algorithm is worthy of further discussion, as it is related to the issues of how to practically implement the AQM controller and how to measure/gather all the parameters such as k_1 and k_2. One key objective of choosing these parameters is that we should ensure that the controller is robust to a wide variety of network condition changes. The value of k_1 can be determined as described in Section IV, after the value of k_2 is determined. On the other hand, to practically implement the algorithm, we replace $k_2\delta w$ with $a(R - C_0)$. The value of δw cannot be directly obtained at the router, but can be estimated by $\frac{\tau}{N}\delta\dot{q}$, where $\delta\dot{q}$ is the difference between the incoming rate and the link capacity, i.e. $R - C_0$. The remaining problem is how to determine the values of τ and N. As mentioned in Section IV, the equilibrium state of the system is meaningful only when $N < \frac{1+\sqrt{12b+1}}{6}\tau C$. Hence, we can use $N_{max} = \frac{1+\sqrt{12b+1}}{6}\tau C$ (which is $0.77\tau C$ and τC, respectively, when b is 1 and 2.). Thus, in the case of $b = 2$, we have $|\frac{\tau}{N_{max}}\delta\dot{q}| = |\frac{1}{C}\delta\dot{q}| \leq |\frac{\tau}{N}\delta\dot{q}| = |\delta w|$. Because $|\frac{k_2}{C}\delta\dot{q}| \leq |k_2\delta w|$, the net effect of setting $a = \frac{k_2}{C}$ is that we choose a smaller value of k_2 than that determined according to the method described above, and hence the system is still stable.

$p = k_1 \cdot \delta q + a \cdot (R - C_0)$ implies that the packet dropping/marking probability in SFC consists of two parts: the

first part is proportional to the difference between the current queue length and the target queue length; and the second part is proportional to the ratio of the difference between the incoming rate and the link capacity to the link capacity. Continuing Example 1, we discuss in the following example how the parameters are practically chosen:

Example 2: As shown in Fig. 3, for the nominal value of $k_2 = 0.2$ obtained in Example. 1, the appropriate value of k_1 ranges from 0 to 0.0007. As described above, we estimate the number, N, of connections to be N_{max}, the net effect of which is that we actually use a smaller value of k_2 than the chosen value of k_2. Consequently, we should choose the value of k_1 so that the system remains stable given a smaller value of k_2. From Fig. 3, we can see that $k_1 = 0.0005$ is in the stable region for all values of $k_2 \in [0, 0.2]$ and hence is a safe choice. ■

VI. Related Work

As discussed in Section I, several AQM schemes have been proposed, e.g., FRED [17], balanced RED (BRED) [1], BLUE [7], stabilized RED (SRED) [20], random exponential marking (REM) [2], PI controller [12], and AVQ [16]. These schemes differ in (1) the performance objectives (in addition to that of notifying end hosts of incipient congestion by dropping/marking packets); (2) the parameters used as an indicator of congestion; and (3) the policies used to detect (incipient) congestion and to drop/mark packets. In what follows, we summarize these schemes, and give in Table I a taxonomy with respect to the above three aspects.

Schemes that aim to achieve fairness*:* In FRED, a router monitors, not only the global average queue length, but also the average queue length, $qlen_i$, of each individual active connection i. Moreover, two minimum and maximum thresholds are defined for the per-flow average queue length. When a packet from flow i arrives, $qlen_i$ is compared against these two thresholds. A flow with $qlen_i$ less than the minimum limit is not subject to random early dropping even if *minth* $\leq$ *avg_queue* $\leq$ *maxth*. On the other hand, a flow which consistently exceeds the maximum threshold is subject to more aggressive dropping.

BRED extends FRED and imposes three thresholds, ℓ_1, ℓ_2, and ℓ_3, on per-flow queue length, $qlen_i$. The three thresholds divide the space of $qlen_i$ into 4 regions: (0, ℓ_1), (ℓ_1, ℓ_2), (ℓ_2, ℓ_3), and (ℓ_3, ∞), each of which is associated with a dropping probability of 0, p_1, $p_2 (> p_1)$, and 1, respectively. A router keeps, for each active flow i, the queue length, $qlen_i$, and the number of its packets accepted into the queue since last drop, gap_i. The dropping probability for a packet from flow i is then a function of (i) the region $qlen_i$ is in and (ii) gap_i. The reason for figuring gap_i into the dropping probability is to prevent consecutive multiple drops from a flow. In essence, both FRED and BRED aim to improve the fairness of RED at the expense of keeping per-active-flow state information.

Schemes that decouples the congestion index and the performance index*:* Schemes in this category aim at achieving both high utilization and low packet delay (queue length). The key idea is to decouple the congestion measure from the performance measure. Specifically, these schemes either use additional measures (e.g., link utilization, input rate) as congestion indices, or introduce an intermediate entity (the price function in REM or the virtual queue in AVQ) so that calculation of the dropping probability is not *directly* related to the actual queue length.

In BLUE, the instantaneous queue length and the link utilization are used as the indices of traffic load, and a single dropping probability p is maintained and used to mark or drop packets upon packet arrival. If the instantaneous queue length exceeds a pre-determined threshold, L, a BLUE router increases p by an amount of *delta* (which is a system parameter). To avoid dropping packets too aggressively, BLUE keeps a minimum interval, *freeze_time*, between two successive updates of p. Conversely, if the link is idle (i.e., the queue is empty), the BLUE gateway decreases p by an amount of $delta$ periodically (once every *freeze_time*). By adjusting p with respect to the instantaneous queue length and link utilization (idle events), BLUE is shown through simulation to make the instantaneous queue length converge to an operational point with small buffer sizes, while retaining all the desirable features of RED.

REM decouples the congestion measure from the performance measure by defining the price function, $c(k+1)$, as

$$c(k+1) = \max(0, c(k) + \gamma(\alpha(Q(k) - Q_{opt}) + x(k) - R)), \tag{19}$$

where $x(k)$ is the aggregate input rate and R is the capacity of the outgoing link. The $(\alpha(Q(k) - Q_{opt})$ term is the queue mismatch, and the $x(k) - R$ term the rate mismatch. Since $x(k) - R$ measures the rate at which the queue length grows, it can be approximated as $Q(k+1) - Q(k)$, and Eq. (19) reduces to

$$c(k+1) = \max(0, c(k) + \gamma(Q(k+1) - (1-\alpha)Q(k) - \alpha Q_{opt})). \tag{20}$$

The price increase if the weighted sum of these mismatches is positive, and decrease otherwise. A REM router calculates the marking probability periodically as $p(k) = 1 - \phi^{-c(k)}$, where ϕ is an arbitrary constant that is greater than 1.

The AVQ scheme, on the other hand, takes a dramatically different approach, and uses solely the input rate, $x(t)$, as the congestion index. An AVQ router maintains a virtual queue whose capacity, $\hat{R}$, is adjustable. Upon packet arrival, the virtual queue capacity is updated according to

$$\frac{d\hat{R}}{dt} = \alpha(\gamma R - x(t)). \tag{21}$$

where γ is the desired utilization. The rationale behind Eq. (21) is to mark/drop packets more aggressively when the arrival rate exceeds the desired utilization (γR) and vice versa. Also, a fictitious packet is enqueued in the virtual queue if space is available. Otherwise, the fictitious packet is not enqueued and the real packet in the real queue is marked/dropped. The rule for choosing the parameter α is rigorously analyzed using a control theoretic approach to ensure system stability. Through simulation in [16], AVQ is shown to outperform REM in terms of reducing the packet drop rate and average queue length and achieving high utilization.

Category	Scheme	Congestion index	Policies used to detect congestion and to drop/mark packets
Achieving fairness	FRED [17]	queue length	Monitors the per-flow queue length and fine-tunes the dropping/marking decision w.r.t. the per-flow queue length.
	BRED [1]	queue length	Defines three thresholds and divides the state of per-flow queue length into 4 regions. Fine-tunes the dropping/marking decision w.r.t. the per-flow state.
Achieving high utilization and low packet loss	BLUE [7]	queue length, link idle event	Increases p if the instantaneous queue length exceeds L and has not been updated for over *freeze_time*. Decreases p if the link is idle for over *freeze_time*.
	REM [2]	queue length, input rate	Defines the price function, $c(k)$, as in Eq. (19), and calculates the marking probability as $p(k) = 1 - \phi^{-c(k)}, \phi > 1$.
	AVQ [16]	input rate	Maintains a virtual queue. At each packet arrival, enqueue a fictitious packet and update the virtual queue capacity using Eq. (21). Mark/drop a real packet only if the virtual queue overflows.
Stabilizing queue	SRED [20]	queue length	keeping a zombie list to keep track of recently seen flows, to detect misbehaving flows, and to estimate the number, N, of active flows. Figures in N in the packet dropping probability.
	PI [12]	queue length, input rate	Calculates the marking probability as in Eq. (22).
	Scalable control [21]	input rate	Router updates its price function, $p_l(t)$, as in Eq. (23), and marks packets with probability as $1 - \phi^{-p_l(t)}, \phi > 1$. Source sets its rate as $x_i(t) = x_{max,i} e^{-\frac{\alpha_i q_i(t)}{M_i r_i}}$.

TABLE I
A TAXONOMY OF AQM SCHEMES.

Schemes that stabilize the instantaneous queue length: SRED argues that the instantaneous queue length may fluctuate dramatically under RED if the number of active flows varies. To stabilize the instantaneous queue, SRED equips each queue with a zombie list that keeps a list of M recently seen flows. When a packet arrives, it is compared with a randomly chosen zombie in the zombie list. The result of a hit or a miss is used to detect potential misbehaving flows for more aggressive dropping and to estimate the number of active flows. The estimated value of N is then figured into the calculation of the dropping probability p (e.g., p is an increasing function of N) so as to avoid, upon packet loss, the situation of significant system throughput decrease in the case that there are only a few active flows. The simulation results indicated that SRED keeps the buffer occupancy close to the specified value and away from overflow or underflow.

The PI controller also aims to stabilize the instantaneous queue length, but it is built upon on a fluid model proposed in [19] and takes a more systematic approach. The PI controller marks each packet with a probability p which is updated periodically using

$$p(k+1) = p(k) + a(Q(k+1) - Q_{opt}) - b(Q(k) - Q_{opt}), \quad (22)$$

where $a > 0$ and $b > 0$ are constants chosen according to the design rules given in [12].

The scalable control scheme proposed in [21] uses the link's price $p_l(t)$ as the congestion index and marks packets with probability $1 - \phi^{-p_l(t)}, \phi > 1$. The link then updates its price $p_l(t)$ using the aggregate input rate $y_l(t)$ according to:

$$\dot{p}_l(t) = \begin{cases} \frac{y_l - c_l}{c_l} & \text{if } p_l(t) > 0; \\ \max\{0, \frac{y_l - c_l}{c_l}\} & \text{if } p_l(t) = 0 \end{cases} \quad (23)$$

in which c_l is the *virtual capacity* that is strictly less than the actual link capacity. The source will set its sending rate as an exponential function of aggregate price $q_i(t)$, i.e. $x_i(t) = x_{max,i} e^{-\frac{\alpha_i q_i(t)}{M_i r_i}}$. To utilize this scheme, the current TCP congestion control and avoidance scheme has to be changed.

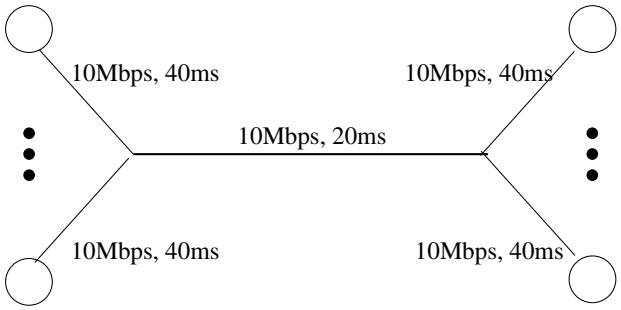


Fig. 6. The single bottleneck simulation topology.

The work that comes closest to ours is SRED and PI controller, as both of them share the same objective of stabilizing the instantaneous queue. Hence we will conduct a comprehensive performance study between SRED, PI and SFC in Section VII. On the other hand, as a side effect of using both the deviations of the queue length and the incoming rate from the operating point to determine the packet dropping/marking probability, SFC also decouples the congestion measure and the performance measure. Hence, we will also compare SFC against AVQ (which is reported to give the best performance in the second category) in Section VII.

VII. SIMULATION RESULTS

We have implemented our scheme along with RED [8], SRED [20], PI [12] and AVQ [16] in ns-2 [4], and conducted a simulation study to validate the performance of proposed design and compare its performance against the other schemes. The performance study was conducted with respect to queue stability, packet loss rate, link utilization, and parameter robustness.

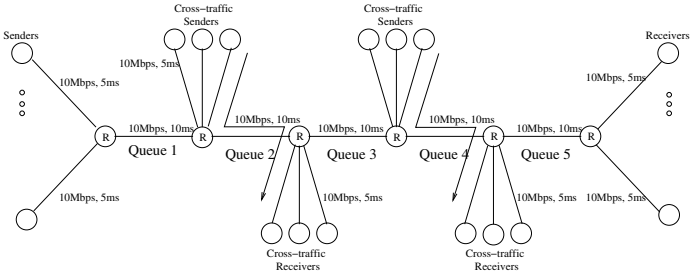


Fig. 7. The multiple bottleneck simulation topology.

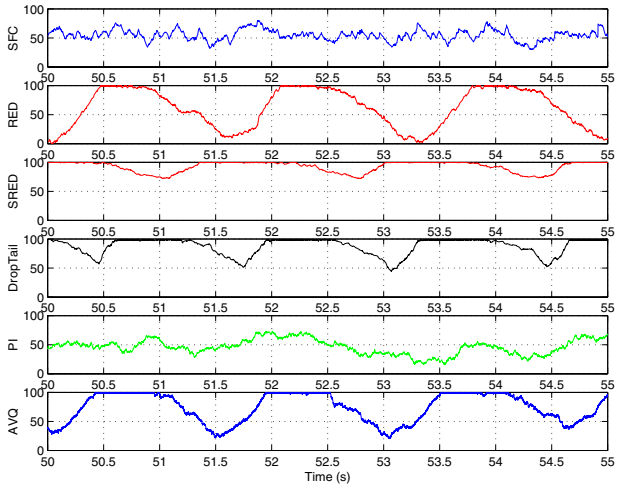


Fig. 8. Performance comparison with respect to instantaneous queue length among different schemes.

We examine the behavior of these schemes under a variety of network topologies and traffic sources. In particular, we have considered the network topologies with a single bottleneck link of various RTTs (Fig. 6) and network topology with multiple bottlenecks (Fig. 7). The link bandwidth and propagation delay used in the simulation are, unless otherwise specified, given in Fig. 6 and Fig. 7. The average packet size is 1000 byte and buffer size on each link is 100 packets. The traffic sources we use include long-termed TCP connections and short-termed TCP connections, both of which support ECN but do not enable the delayed ACK option. The number of connections varies from 100 to 1000. The target queue length is set to 50 packets.

The setting of parameters in the various AQM schemes is as follows. The parameters of RED are set as recommended in http://www.aciri.org/floyd/REDparameters.txt, and those of SRED are chosen as recommended in [20] (i.e., $M = 1000$, $\alpha = 1/M = 0.001$, and $p_{max} = 0.15$). The desirable utilization, γ, of AVQ is set to 0.98, and the damping factor, α, is determined in compliance with Theorem 1 in [16] to ensure system stability ($\alpha = 0.15$). The parameters of our scheme are $k_2 = 0.2$ and $k_1 = 0.0005$ as calculated in the previous example. Each data point is the result averaged over 20 simulation runs. In spite of numerous system parameters involved, the results are found to be quite robust in the sense that the conclusion drawn from the performance curves for a representative set of parameter values (reported below) is valid over a wide range of parameter values.

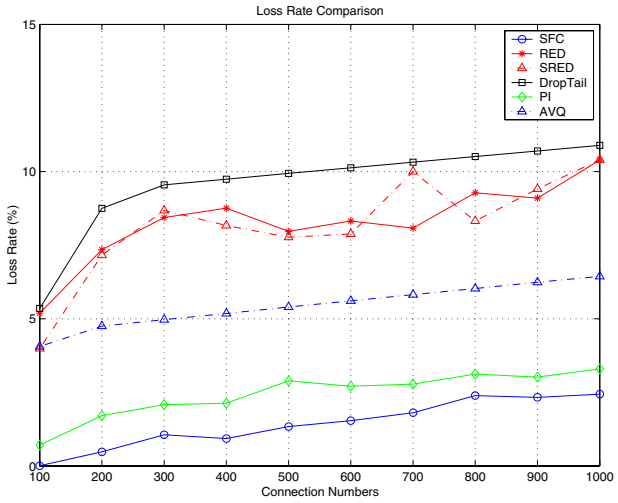


Fig. 9. Performance comparison with respect to packet loss rate among different schemes.

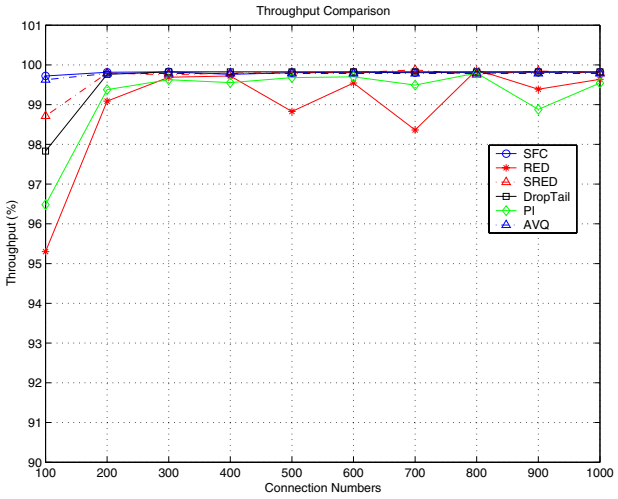


Fig. 10. Performance comparison with respect to link utilization among different schemes.

A. Performance Comparison Under the Single Bottleneck Topology

In this set of experiments, we compare SFC against the other schemes in the single bottleneck topology (Fig. 6). Totally k TCP connections are established over a single bottleneck link of capacity 10 Mbps, where k varies from 100 to 1000. Fig. 8 gives the instantaneous queue length in the cases that 200 TCP connections are established and continuously transmit packets. As shown in the figure, the instantaneous queue lengths under SFC and PI fluctuate around the target level, while the queue length under the other schemes are either always full or oscillates between empty and full.

Figs. 9–10 depict, respectively, the packet loss ratio and the goodput attained by all receivers. SFC outperforms than the other schemes with respect to packet loss rate (by reducing

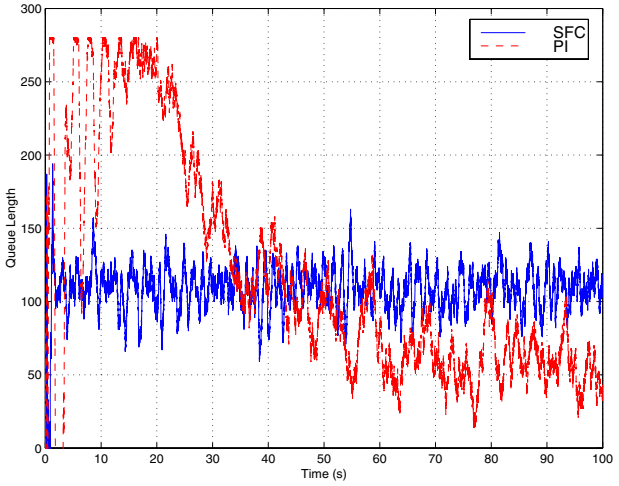


Fig. 11. Performance comparison (in terms of the time taken for the queue to stabilize) between PI and SFC.

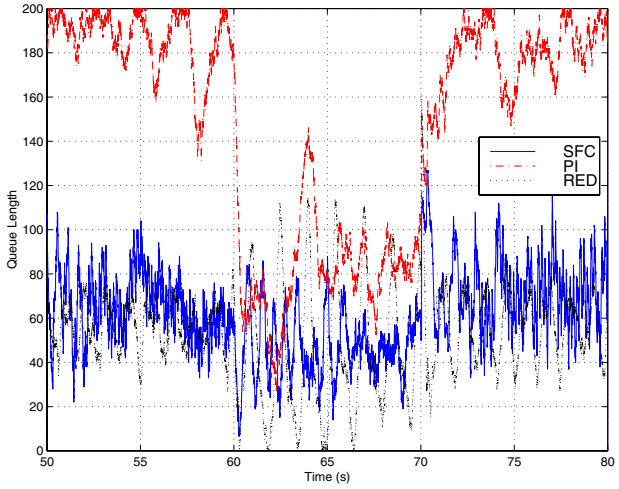


Fig. 12. Performance comparison (in terms of instantaneous queue length) under dynamic traffic changes.

as much as 50% packet losses), because it keeps the queue size at the desirable level. As a result, buffer overflow seldom occurs and less packets are dropped. As the PI controller also attempts to keep the queue length at a target level, it also achieves good performance. Although SRED shares the same objective of stabilizing the queue at a desirable level, it incurs much higher packet losses. This is attributed to the fact that SRED always attempts to keep the queue full (as shown in Fig. 8). On the other hand, as shown in Fig. 10, AVQ, SRED, and SFC (almost) fully utilizes the bandwidth of the bottleneck link. This is because the queue under SFC and AVQ is seldom empty, while SRED always keeps the queue full (which in turns leads to high packet losses). Overall, SFC strikes a balance between reducing packet losses and queuing delay, and utilizing link bandwidth.

B. System Response

In this set of experiments, we set the number of connections to 100, the target queue length to 100, the buffer size to 300 packets, and compare the time it takes for the queue to stabilize at the desirable level under the PI controller and SFC. As shown in Fig. 11, SFC stabilizes the queue much faster than the PI controller. Furthermore, the former incurs very low overshoot. The fact that the PI controller incurs slower response and larger overshoot is attributed to its integral part in the controller.

C. Performance Comparison Under Dynamic Traffic Changes

In this set of experiments, we compare RED, PI, and SFC in terms of their responses to dynamic traffic changes in the single bottleneck topology (Fig. 6). 200 TCP connections (with bulk data transfer) are established over a single bottleneck link of capacity 10 Mbps. 50 TCP connections stop their transmitting at the 60th second, and resume at the 70th second again. Fig. 12 depicts the instantaneous queue length under PI controller, RED and SFC. The instantaneous queue length under PI and RED fluctuates significantly between the 60th and 70th seconds, while that under SFC is less susceptible to dynamic traffic changes. The reason why RED does not respond well to dynamic traffic changes is because the packet marking/dropping probability of RED is a linear function of the average queue length, and averaged queue length responds slowly to instantaneous queue length changes and even more slowly to the incoming rate change. The integral part of the PI controller makes it response slowly to dramatic traffic changes. In contrast, the marking/dropping probability in SFC is a linear combination of the instantaneous queue length and the incoming rate, and hence can adapt to traffic changes. Overall, if taken queue length at a router as AQM system's output, SFC controller behaves sort of like a PD controller. That is the main reason why it responses to queue length and incoming traffic rate changes quickly.

D. Robustness w.r.t. RTT and # Connection Changes

In this set of experiment, we test the robustness of the system parameters chosen in SFC with respect to different values of RTT and different # of TCP connections. The simulation setup is the same as in the first experiment, except that the number of connections varies from 200 to 600 and the RTT value varies from 200ms to 800ms. Figs. 13 and 14 depict, respectively, the link utilization (i.e., the goodput attained by all receivers) and the packet loss ratio for different values of RTTs and different numbers of TCP connections. It is clear that although the system parameters chosen in SFC are for the case of $\tau_{max} = 600ms$ and $N_{min} = 300$, the controller still achieves high link utilization and low packet loss ratio, regardless of the RTT and connection number changes.

E. Performance Comparison under the Multiple Bottleneck Topology

Although SFC is designed for the case of homogeneous TCP connections sharing a single bottleneck, we have conducted simulation to evaluate its multiple in the bottleneck topology (Fig. 7). As shown in Fig. 7, there are 5 queues among which

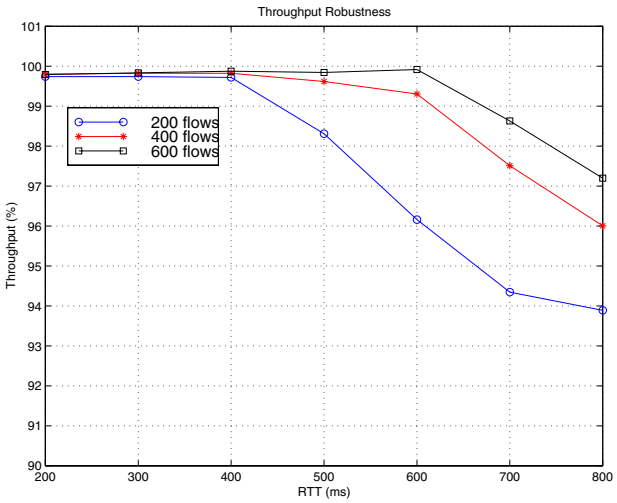


Fig. 13. Robustness of system parameters chosen in SFC (link utilization with respect to different values of RTTs and # of connections).

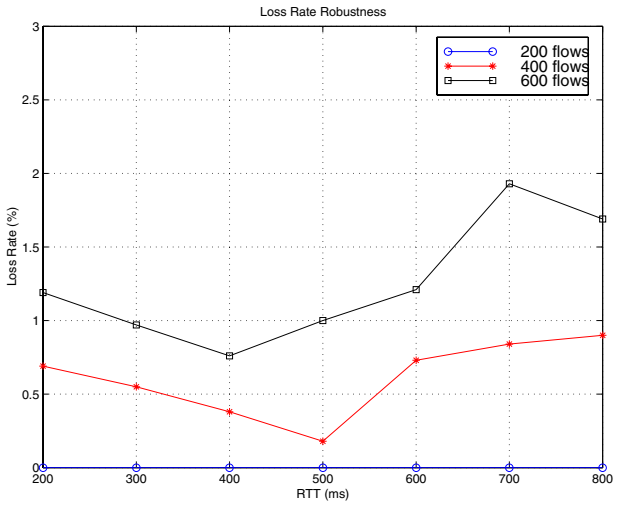


Fig. 14. Robustness of system parameters chosen in SFC (packet loss ratio with respect to different values of RTTs and # of connections).

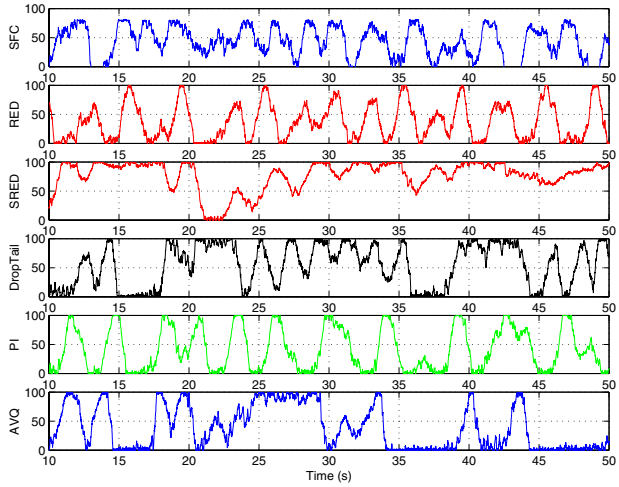


Fig. 15. Instantaneous queue length at queue 2 under different schemes in the multiple bottleneck topology.

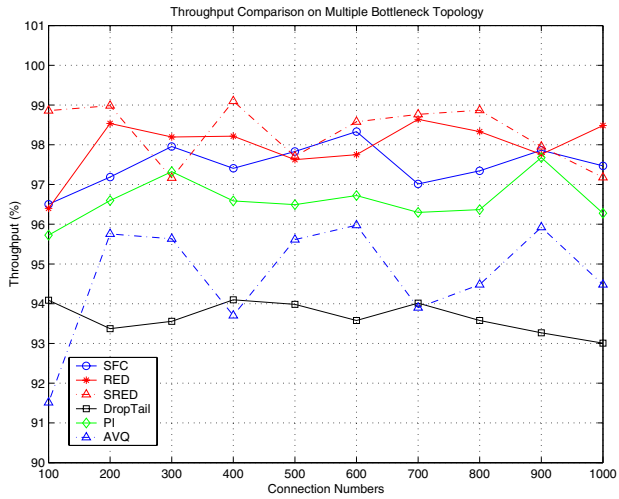


Fig. 16. Link utilization at queue 4 under different schemes in the multiple bottleneck topology.

queue 2 and queue 4 are shared by cross traffic of some other TCP connections. Again we establish k TCP connections with senders at the left hand side and receivers at the right hand side, where k varies from 100 to 1000. The cross traffic is composed of TCP connections as well, and the number of TCP connections in each cross traffic bundle is set to $0.2k$.

The simulation results show that the queue length at queue 5 is always 0 or 1, suggesting that the link is not a bottleneck link. The other four queues exhibit similar trends as far as the performance comparison is concerned. Hence, we arbitrarily choose to depict the instantaneous queue lengths of queue 2 in Fig. 15 and the packet loss ratio and the link utilization of queue 4 in Figs. 16–17, respectively. As shown in Fig. 15, although the capability of SFC to stabilize the instantaneous queue length degrades in the multiple bottleneck topology, its queue length still oscillates around the desirable level (except that the level of oscillation is larger than that in the single bottleneck topology). The is perhaps due to the fact that every router attempts to control its queue level locally, so that the interaction among them becomes complicated.

The link utilization achieved under SFC is the highest, as the controller attempts to keep the queue length stabilized and responds quickly to the queue length and incoming rate changes. The packet loss ratio under SFC is the second smallest and that under PI is the smallest. The reason why SRED does not perform as well in terms of packet loss ratio is because SRED has the tendency to keep the queue (close to) full, and hence packet losses occur as a result of buffer overflow.

We also evaluate and compare the performance of our scheme with other schemes' in the presence of short-lived TCP flows and non ECN-Enabled TCP flows, but due to the space limitation, they are not presented in this paper.

VIII. Conclusion

In this paper, we have developed an analytical TCP model that takes into account of several issues that were ignored in

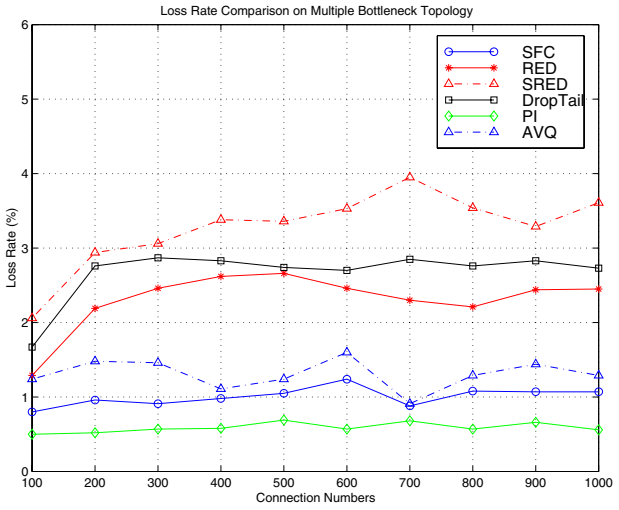


Fig. 17. Packet loss ratio at queue 4 under different schemes in the multiple bottleneck topology.

the other existing models (such as those in [15], [19]), i.e., (i) the congestion window is not *gradually* decreased at the rate of $\frac{w^2p}{2}$, but *suddenly* halved upon receipt of congestion indication and (ii) the congestion window is halved at most once during one RTT. We also include the delayed ACK option in the model. We show that this enhanced model more realistically characterizes the TCP dynamics and that under this model the change in the congestion window size is more significant in response to packet losses. The latter cautions us for designing an appropriate AQM controller. Based on this model, we then design, with the use of state feedback control theory, an AQM controller, called *SFC*, to stabilize the queue at a router. The performance of the new controller is shown via *ns* simulation to outperforms the other schemes in terms of fluctuation in the queue length, link utilization, and packet loss ratio. In particular, SFC can reduce packet loss by more than 50% as compared to the other schemes and yet achieve full link utilization. As compared to PI controller proposed in [12], SFC achieves 10% more link utilization.

We have identified several research avenues. First, it has been shown that the long range dependency characteristics of the Internet traffic can be exploited to better design AQM [9]. Along this research avenue, we will use exercise predictive control by combining traffic prediction and TCP/AQM models to stabilize queue length. Also, we will exploit non-linear control and adaptive control to avoid linearization of the model and enable the controller to be on-line adaptive to system parameter changes.

ACKNOWLEDGMENTS

We would like to thank many colleagues and anonymous reviewers for their constructive criticism and helpful suggestions for improving the overall quality of this paper.

REFERENCES

[1] F. M. Anjum, and L. Tassiulas. Fair Bandwidth Sharing Among Adaptive and Non-Adaptive Flows in the Internet. *Proceedings of IEEE INFOCOM'99*, New York, USA, March 1999.

[2] S. Athuraliya, S. H. Low, V. H. Li, and Q. Yin. REM: Active Queue Management. *IEEE Network Magazine*, Vol. 15, No. 3, May/June 2001.

[3] M. Christiansen, K. Jeffay, D. Ott and F. D. Smith. Tuning RED for web traffic. *Proceedings of ACM SIGCOMM'00*. Stockholm, Sweden, September 2000.

[4] UCB, LBNL, VINT Network Simulator. *http://www-mash.cs.berkeley.edu/ns/*.

[5] K. Claffy, G. Miller and K. Thompson. The Nature of the Beast: Recent Traffic Measurements from An Internet Backbone. In *Proc. of INET'98*

[6] W. Feng, D. Kandlur, and K. Shin. Stochastic Fair Blue: A Queue Management Algorithm for Enforcing Fairness. *Proceedings of IEEE INFOCOM'01*, Alaska, USA, April 2001.

[7] W. Feng, K. Shin, D. Kandlur, and D. Saha. A Self-Configuring RED Gateway. *Proceedings of IEEE INFOCOM'99*, New York, USA, March 1999.

[8] S. Floyd and V. Jacobson. Random Early Detection Gateways for Congestion Avoidance. *IEEE/ACM Transactions on Networking*, Vol.1, No.4, August 1993.

[9] Y. Gao, G. He, and J. Hou. On Leveraging Traffic Predictability in Active Queue Management. *Proceedings of IEEE INFOCOM'02*, New York, USA, June 2002.

[10] Y. Gao and J. Hou. A State Feedback Control Approach to Stabilizing queues for ECN-Enabled TCP Connections. Technical report, Department of Computer Science, University of Illinois at Urbana Champaign. Available at *http://www.cs.uiuc.edu/ yuangao*.

[11] E. Hashem. Analysis of Random Drop for Gateway Congestion Control. MIT Technical Report, 1990.

[12] C. Hollot, V. Misra, D. Towsley, and W. Gong. On Designing Improved Controllers for AQM Routers Supporting TCP Flows. *Proc. IEEE INFOCOM'01*, Alaska, USA, April 2001.

[13] V. Jacobson. Congestion Avoidance and Control. *ACM Computer Communication Reviews*, Vol. 14, 1988

[14] V. Jacobson. Presentations to the IETF Performance and Congestion Control Working Group. August 1989.

[15] F. Kelly. Mathematical Modeling of the Internet. *Mathematics Unlimited - 2001 and Beyond (Editors B. Engquist and W. Schmid)*. Springer-Verlag, Berlin, 2001.

[16] S. Kunniyur and R. Srikant. Analysis and Design of an Adaptive Virtual Queue (AVQ) Algorithm for Active Queue Management. *Proc. of ACM SIGCOMM 2001*, San Diego, USA, August 2001.

[17] D. Lin, and R. Morris. Dynamics of Random Early Detection. *Proceedings of ACM SIGCOM'97*, September 1997.

[18] A. Mankin. Random Drop Congestion Control. *Proc. of ACM SIGCOMM*, September 1990, pp. 1-7.

[19] V. Misra, W. Gong, and D. Towsley. A Fluid-based Analysis of a Network of AQM Routers Supporting TCP Flows with an Application to RED. *Proceedings of ACM SIGCOMM'00*. Stockholm, Sweden, September 2000.

[20] T.J. Ott, T. V. Lakshman, and L.H. Wong. SRED: Stabilized RED. *Proceedings of IEEE INFOCOM'99*, New York, USA, March 1999.

[21] F. Paganini, J. C. Doyle, and S. H. Low. Scalable Laws for Stable Network Congestion Control. *Proceedings of Conference on Decision and Control*, Florida, USA, December 2001.

[22] K. K. Ramakrishnan, S. Floyd, and D. Black. The Addition of Explicit Congestion Notification (ECN) to IP. RFC-3168, IETF, September 2001.

[23] P. Ranjan, E. H. Abed, and R. La. Nonlinear Instabilities in TCP-RED. *Proceedings of IEEE INFOCOM'02*, New York, USA, June 2002.

[24] G. Silva, A. Datta, and S. P. Bhattacharyya. PI Stabilization of First-order Systems with Time Delay. *Automatica*, December, 2001.

[25] A. Veras, and M. Boda. The Chaotic Nature of TCP Congestion Control. *Proceedings of IEEE INFOCOM 2000*, Tel-Aviv, Israel, March 2000.