LRED: A Robust Active Queue Management Scheme Based on Packet Loss Ratio

Chonggang Wang[1] Bin Li[2] Y. Thomas Hou[3] Kazem Sohraby[4] Yu Lin[1]
[1]Beijing University of Posts and Telecommunications, Beijing, P. R. China
[2]China Motion Telecom International Ltd., Hong Kong, P. R. China
[3]Virginia Tech, the Bradley Dept. of Electrical and Computer Engineering, Blacksburg, VA, USA
[4]University of Arkansas, Computer Science and Computer Engineering Dept., Fayetteville, AR, USA

Abstract—**Active queue management (AQM) is an effective method to enhance congestion control, and to achieve tradeoff between link utilization and delay. The *de facto* standard, Random Early Detection (RED), and most of its variants use queue length as a congestion indicator to trigger packet dropping. Despite their simplicity, these approaches suffer from unstable behaviors, as revealed in many simulation studies. Some of them introduce adaptive parameter settings to improve stability; however, there is still no real analytical stability model for them. Recent schemes, such as Proportional-Integral (PI), use both queue length and traffic input rate as congestion indicators; effective stability model and practical design rules built on the TCP control model and abstracted AQM model reveal that such schemes enhance the stability of a system. Nevertheless, compared to RED and its variants, the response time in these schemes often increases, especially within a highly dynamic network with heavy traffic load, which would cause continuous buffer overflow or buffer emptiness.**

In this paper, we propose an AQM scheme with fast response time, yet good robustness. The scheme, called Loss Ratio based RED (LRED), measures the latest packet loss ratio, and uses it as a complement to queue length in order to dynamically adjust packet drop probability. Employing the closed-form relationship between packet loss ratio and the number of TCP flows, this scheme is responsive even if the number of TCP flows varies significantly. We also provide the design rules for this scheme based on the well-known TCP control model. This scheme's performance is examined under various network configurations, and compared to existing AQM schemes, including PI, Random Exponentially Marking (REM), and Adaptive Virtual Queue (AVQ). Our simulation results show that, with comparable complexity, this scheme has short response time, better robustness, and more desirable tradeoff than PI, REM, and AQV, especially under highly dynamic network and heavy traffic load.

Keywords—**Active Queue Management, Packet Loss Ratio, Quality of Service.**

I. INTRODUCTION

Buffer management plays an important role in congestion control [1][2]. However, the two main objectives of buffer management, high link utilization and low packet queuing delay, are often in conflict with each other. Given that most end-nodes employ the responsive TCP window-based Additive Increase and Multiplicative Decrease (AIMD) control scheme, a small buffer can guarantee low queuing delay, but it often suffers from packet loss and low link utilization, and vice versa. Therefore, the key design problem here is to find a good tradeoff between these variables.

The traditional scheme for buffer management is First-In-First-Out (FIFO) Tail-Drop, which drops packets only if there are buffer overflows. This passive behavior with longer queuing delay often causes correlation among packet drops, resulting in the well-known "TCP synchronization" problem [3] and, consequently, low link utilization. The long queuing delay of Tail-Drop also makes it inefficient for delay-sensitive real-time applications.

To mitigate such problems, AQM [3] has been introduced in recent years. The basic philosophy of AQM is to trigger packet dropping (or marking, if explicit congestion notification (ECN) [4] is enabled) before buffer overflows, and when drop probability is proportional to the degree of congestion. In existing AQM schemes, link congestion is estimated through queue length, input rate, events of buffer overflow and emptiness, or a combination of these factors. Queue length (or average queue length) is widely used in RED [5] and most of its variants [6][7][8], where packet drop probability is often linearly proportional to queue length. S-RED [8] and BLUE [9] trigger to adjust the packet drop probability when packet buffer is overflow or empty. The traffic input rate is also used in some AQM schemes such as AVQ [10] to make them more adaptable to instantaneous traffic variance and to achieve desired link utility. Some recent AQM schemes, for example PI [11][12], REM [13], and SFC [14], jointly use queue length and traffic input rate.

In this paper, we argue that the packet loss ratio is an important index in designing a buffer management mechanism. For example, an increasing packet loss ratio is a clear indication that severe congestion occurs, and that aggressive packet dropping is needed. On the other hand, a decrease of packet loss ratio can serve as a signal that congestion is receding and consequently, that packet drop action can change from aggressive to moderate. Therefore, it is possible to use the packet loss ratio to design more adaptive and robust AQM schemes.

In this paper, we propose a packet loss ratio-based RED (LRED) scheme for robust active queue management. In order to regulate queue length to some expected value, two principles in LRED to calculate packet drop probability are: 1) the mismatch of queue length means deviation from stable

C. Wang's work was supported by in part by grants from Research Grants Council (RGC) under contracts AoE/E-01/99 and HKUST 6192/02E, a grant from NSFC/RGC under the contract N_HKUST502/02, and a grant from Microsoft Research under the contract MCCL02/03.EG01.

status and the necessity of updating the packet drop probability; 2) large packet loss ratio implies overload, indicating that aggressive packet drop is needed. According to the first principle, LRED uses instantaneous queue mismatch as an input variable to calculate the required packet drop probability each time packets arrive. Calculated packet drop probability linearly increases with queue mismatch. According to the second principle, when there is a large packet loss ratio, LRED will dynamically increase the packet drop probability. In a summary, LRED uses the instantaneous queue length to calculate the packet drop probability upon the arrival of each new packet, *i.e.*, over short time scales, while dynamically adjusting the packet drop probability according to the measured packet loss ratio over relatively large time scales. Such a combination enables fast response time and high robustness.

The performance of our LRED scheme is evaluated through extensive simulations under various network configurations. Compared to existing AQM schemes, such as S-RED, BLUE, ARED, and AVQ, our LRED scheme offers more stable control of queue length around the expected value, as well as the achievement of high link utility. Compared to new AQM schemes such as PI and REM, our scheme has much faster response time and better robustness.

The remainder of this paper is organized as follows. Section II reviews the important TCP control model and the characteristic-equation method used in AVQ [10]. We also discuss basic properties of the combined system of TCP and AQM control, which serves as the basis for the design and analysis of the proposed LRED scheme. Section III presents LRED scheme and its stability analysis. In Section IV, we study the performance of LRED through extensive simulations and compare it to other existing AQM schemes. Some related research is given in Section V. Section VI concludes this paper.

II. Model of Combined TCP/AQM System

In this section, we describe the general properties of the combined TCP and AQM control system, based on the fluid TCP model [11] as well as the characteristic-equation method [10]. They serve as the basis for designing a stable AQM scheme, and for the analysis of our proposed LRED scheme.

A. Existing Model

We use a 3-tuple (N, C, τ) to represent the network status, where N is the number of TCP flow number, C is the link capacity, and τ is the round trip time, which is assumed to be fixed in the system. The system equation about the queue length (q) and the TCP congestion window (w) can be approximately constructed as follows [11].

$$\dot{w} = f(w, p) = \frac{1}{\tau} - \frac{w(t)w(t-\tau)}{\eta\tau} p(t-\tau), \tag{1}$$

$$\dot{q} = g(w, p) = \frac{N}{\tau} w(t) - C, \tag{2}$$

where p is the packet drop probability, and $\eta > 1$. Let $f(w,p)=0$ and $g(w,p)=0$. The congestion window w_0 and the packet drop probability p_0 in the steady-state can be calculated as:

$$w_0 = \frac{\tau C}{N}, \quad p_0 = \frac{\eta}{w_0^2} = \frac{\eta N^2}{\tau^2 C^2} < 1. \tag{3}$$

When $\eta = 1.5$, the steady-state throughput of a single TCP flow $\frac{C}{N} = \frac{\sqrt{2/3}}{\tau\sqrt{p_0}}$ is consistent with the TCP throughput equations in [15].

From Eqs. (1) and (2), it is clear that the combined system of TCP and AQM is non-linear. Let $\delta w = w - w_0$ and $\delta p = p - p_0$ be the mismatches for TCP congestion window and packet drop probability, respectively. Eqs. (1) and (2) can be locally linearized around a stable point (w_0, p_0) by assuming $w(t) \approx w(t-\tau)$ [11]:

$$\dot{\delta w} = \frac{\partial f}{\partial w}\delta w + \frac{\partial f}{\partial p}\delta p = -[K_{11}\delta w + K_{12}\delta p(t-\tau)], \tag{4}$$

$$\dot{\delta q} = \frac{\partial g}{\partial w}\delta w + \frac{\partial g}{\partial p}\delta p = K_{21}\delta w, \tag{5}$$

where $K_{11} = \frac{2N}{\tau^2 C}$, $K_{12} = \frac{\tau C^2}{\eta N^2}$, and $K_{21} = \frac{N}{\tau}$.

Applying Laplace transform to Eqs. (4) and (5), we have the system equations for the TCP control mechanism, as follows:

$$sW(s) = -[K_{11}W(s) + K_{12}P(s)e^{-s\tau}]. \tag{6}$$

$$sQ(s) = K_{21}W(s). \tag{7}$$

B. General Properties of Proportional AQM Control

In Eqs. (6) and (7), there are three unknown variables: $W(s)$, $P(s)$, and $Q(s)$. Hence, it is necessary to find one more equation to solve this problem. This can be achieved by using the AQM as a bridge for $P(s)$ and $Q(s)$. Consider an AQM control mechanism that calculates the suitable p according to the instantaneous queue length q: $q \to p$. As will be shown later, the proposed LRED is a proportional controller (see Eq. (25)). We only consider proportional AQM control here to construct the analysis basis for LRED. The control equation of the proportional controller can be generally formulated as[1]:

$$\delta p = H_c \delta q. \tag{8}$$

The corresponding system equation can be written as:

[1] Here, $\delta q = q - q_0$, and q_0 is the expected queue length under stable status. The proposed LRED scheme solely uses queue length to calculate packet drop probability. For schemes that use both queue length and input rate, it is still possible to extend the technique in this section to analyze them.

$$P(s) = H_c Q(s), \tag{9}$$

where $H_c > 0$ is a constant for AQM scheme.

From Eqs. (6)-(9), we can obtain the system characteristic equation as:

$$s^2 + K_{11}s + K_c H_c e^{-s\tau} = 0, \tag{10}$$

where $K_c = K_{12}K_{21} = \frac{C^2}{\eta N}$.

Solving Eq. (10), the stability of the combined system can be analyzed by examining whether its root, $s = \sigma + j\omega$, lies in the half-complex plane. We now discuss some properties about Eq. (10) before presenting its stability analysis.

If the root of Eq. (10) strictly lies in the left half-complex plane, the combined system, defined by the network parameters (N,C,τ) and AQM control parameter H_c, is stable. Hence, given (N,C,τ), we can choose an H_c, such that root s strictly lies in the half complex plane. The system is thus stable. On the other hand, given the control parameter H_c, only some of the system (N,C,τ) can be stably controlled. Also note that the root of Eq. (10) continuously changes in the complex plane when any one of the parameters N, C, τ, or H_c changes continuously.

When $\tau = 0$, the root of Eq. (10) can be calculated as:

$$s = s_0 = \frac{-K_{11} \pm \sqrt{K_{11}^2 - 4K_c H_c}}{2}. \tag{11}$$

It can be easily proved that root s_0 strictly lies in the left half-complex plane irrespective of $K_{11}^2 \geq 4K_c H_c$ or $K_{11}^2 < 4K_c H_c$. Therefore, the system with zero delay is stable.

When $\tau > 0$ and letting $s = \sigma + j\omega$, the root of Eq. (10) can be calculated as follows

$$\sigma^2 - \omega^2 + K_{11}\sigma + K_c H_c e^{-\sigma\tau}\cos(\tau\omega) = 0. \tag{12}$$

$$2\sigma\omega + K_{11}\omega - K_c H_c e^{-\sigma\tau}\sin(\tau\omega) = 0. \tag{13}$$

Since $\tau > 0$, the imaginary part (ω) of root s is nonzero, that is, $\tau > 0 \Rightarrow \omega \neq 0$. Otherwise, Eq. (12) will be invalid.

Based on the above discussion, we can conclude that when τ increases from $\tau = 0$, the root s of Eq. (10) will start at s_0, and then cross the imaginary axis for the first time at $\tau = \hat{\tau}$. As a result, when $0 \leq \tau < \hat{\tau}$, the root s strictly lies in the left half-complex plane, and the system of interest is thus stable. Therefore, the problem is how to find the value of $\hat{\tau}$ that makes root s meet the imaginary axis the first time.

We now analyze the stability of Eq. (10). First, we consider the absolute imaginary root ($s = j\omega$) with $\omega > 0$ (The case of $\omega < 0$ is similar, because the roots on the imaginary axis are complementary.). When $\tau > 0$, Eq. (10) can be re-written as:

$$T(s) = \frac{e^{-s\tau}K_c H_c}{s(s + K_{11})} = -1. \tag{14}$$

The conditions on magnitude and angles must be guaranteed as:

$$|T(j\omega)| = 1, \ \angle T(j\omega) = (2k+1)\pi, \quad k = 0, \pm 1, \pm 2.$$

and ω can be calculated as:

$$\omega(N,C,\tau,H_c) = \omega = \sqrt{\nu(N,C,\tau,H_c)/2}. \tag{15}$$

$$\nu(N,C,\tau,H_c) = \sqrt{K_{11}^4 + 4K_c^2 H_c^2} - K_{11}^2. \tag{16}$$

$$\tau\omega + \arctan(\frac{\omega}{K_{11}}) + \frac{\pi}{2} = (2k+1)\pi, \quad k = 0,1,2. \tag{17}$$

Since K_{11} is a decreasing function of τ, and K_c is independent of τ, we have that $\omega(N,C,\tau,H_c)$ is an increasing function of τ if H_c is independent of or an increasing function of τ. Therefore, solving Eqs. (10) and (11), the smallest $\tau = \hat{\tau}$ gives the first time that a root meets the imaginary axis. Then, for all $\tau < \hat{\tau}$, the system is locally asymptotically stable. Now the problem is to determine the value of k that yields the smallest τ, which can be solved through the following Lemma.

Lemma 1: *When k=0, Eqs. (15)-(17) yield the smallest value of τ and ω, if H_c is independent of or an increasing function of τ. Eq. (17) can be simplified to:*

$$\tau\omega + \arctan(\frac{\omega}{K_{11}}) = \frac{\pi}{2}. \tag{18}$$

Proof: The proof is based on contradiction. Assume that $k \to (\tau_k, \omega_k)$, $k > 0$, $\tau_k > 0$, and $\omega_k > 0$. According to Eq. (17), we have:

$$\tau_k\omega_k - \tau_0\omega_0 = 2k\pi + [\arctan(\frac{\omega_0}{K_{11}}) - \arctan(\frac{\omega_k}{K_{11}})]$$
$$> 2k\pi - \pi/2 > 0.$$

Assuming $\tau_k \leq \tau_0$, we have $\omega_k \leq \omega_0$ according to Eqs. (15) and (16) if H_c is independent of or an increasing function of τ. Since $\tau_k \leq \tau_0$ and $\omega_k \leq \omega_0$, then $\tau_k\omega_k \leq \tau_0\omega_0$, but this is in contradiction to the above inequality. So the assumption that $\tau_k \leq \tau_0$ does not hold and $\tau_k > \tau_0$. □

Lemma 2: *Given network parameters (N,C) and AQM control parameter H_c. Assume that $\hat{\tau}$ satisfies*

$$\hat{\tau}\omega + \arctan(\frac{\omega}{K_{II}}) = \frac{\pi}{2}, \quad \hat{\tau} > 0, \tag{19}$$

where ω is the solution to Eqs. (15) and (16). If function $y(N,C,\tau,H_c)$ in Eq. (16) is an increasing function of τ, then the system is stable for all $\tau < \hat{\tau}$, and $\hat{\tau}$ is unique.

The proof for Lemma 2 follows from Lemma 1 and is thus omitted here to conserve space.

Lemma 3: *Given network parameters (C,τ) and AQM control parameter H_c. Assume that $\hat{N}$ satisfies*

$$\tau\omega + \arctan(\frac{\omega}{K_{II}}) = \frac{\pi}{2}, \quad \hat{N} > 0, \tag{20}$$

where ω is the solution to Eqs. (15) and (16). If function $y(N,C,\tau,H_c)$ in Eq. (16) is an increasing function of τ and a decreasing function of N, then the system is stable for $N > \hat{N}$.

Proof: Let $\tau(N)$ be the solution to Eqs. (10) and (12) with $N > \hat{N}$. If $\tau < \tau(N)$, from Lemma 2, the system is stable for all $N > \hat{N}$. Since $y(N,C,\tau,H_c)$ in Eq. (10) is a decreasing function of N, we have:

$$\omega(N) = \omega(N,C,\tau,H_c) < \omega(\hat{N},C,\tau,H_c) = \omega(\hat{N}).$$

Moreover, K_{II} is an increasing function of N. Hence, we have:

$$\tau\omega(N) + \arctan(\frac{\omega(N)}{K_{II}}) < \tau\omega(\hat{N}) + \arctan(\frac{\omega(\hat{N})}{K_{II}}) = \frac{\pi}{2}.$$

To guarantee $\tau(N)\omega(N) + \arctan[\omega(N)/K_{II}] = \frac{\pi}{2}$, we have $\tau(N) > \tau$. Therefore, for all $N > \hat{N}$, the system is stable. □

Lemma 4: *Given network parameters (N,C,τ), and assume that H_c^* satisfies:*

$$\tau\omega + \arctan(\frac{\omega}{K_{II}}) = \frac{\pi}{2}, \quad H_c^* > 0, \tag{21}$$

where ω is as given in Eqs. (15) and (16). If function $y(N,C,\tau,H_c)$ in Eq. (16) is an increasing function of both τ and H_c, then the system is stable for all $H_c < H_c^*$.

The proof follows the same token as that for Lemma 3.

Theorem 1: *Let the network parameters be $(\hat{N},\hat{C},\hat{\tau})$, and assume that H_c^* satisfies*

$$\hat{\tau}\omega + \arctan(\frac{\omega}{K_{II}}) = \frac{\pi}{2}, \quad H_c^* > 0, \tag{22}$$

where ω is defined as given in Eqs. (15) and (16). If function $y(N,C,\tau,H_c)$ in Eq. (16) is an increasing function of τ, a decreasing function of N, and an increasing function of H_c, then the system is stable for $H_c < H_c^*$, $N \geq \hat{N}$, and $\tau \leq \hat{\tau}$.

The proof for Theorem 1 follows from Lemmas 2 to 4.

III. The LRED Scheme

A. Description

As in most existing AQM schemes, LRED also chooses to drop packets from the tail of the queue. To estimate the degree of link-congestion, two indices are employed: packet loss ratio and queue length. The packet loss ratio is used in large time-scale to make the scheme more adaptive and robust, while the queue length is used in small time-scale to make the scheme more responsive in regulating the length to an expected value q_0. In the large time-scale, the basic rule is to guarantee that the average packet drop probability is as close as possible to the packet loss ratio. The design rules in small time-scale include: 1) when queue length is equal to q_0, the packet drop probability will be equal to the packet loss ratio; 2) when queue length is larger (or smaller) than q_0, the packet drop probability will be also larger (or smaller) than the packet loss ratio.

LRED periodically measures the packet loss ratio, which is then set as the desired stable packet drop probability p_0. To achieve a stable measurement and yet be adaptive to dynamic network conditions, LRED calculates the loss ratio periodically for every small period (*mp*), which should be bigger than *RTT*. During each occurrence of a packet loss ratio calculation, LRED will check the packet loss status for the M latest measurement periods, as illustrated in Fig. 1.

Let $l(k)$ be the packet loss ratio during the latest M measurement periods, *i.e.*, the ratio of the number of dropped packets to the number of total arrival packets during the latest M measurement periods. If packets are of different sizes, $l(k)$ can be calculated as the ratio of the total dropped bytes against the total arrival bytes. The measured packet loss ratio $\overline{l(k)}$ is calculated at the end of each measurement period as (see Fig. 1):

$$\overline{l(k)} = \overline{l(k-1)} * mw + (1 - mw) * l(k), \tag{23}$$

$$l(k) = \frac{\sum_{i=0}^{M-1} N_d(k-i)}{\sum_{i=0}^{M-1} N_a(k-i)}, \tag{24}$$

where *mw* is the measured weight factor, which is set to a small value in order to track the current loss ratio, $N_d(k)$ is the number of packets dropped in the *k-th* period, and $N_a(k)$ is the number of packets arrived in the *k-th* period.

After calculating the packet loss ratio, the task is to find some methods to calculate packet drop probability using instantaneous queue length as an input variable, according to the design rules for small time scale, as listed above. These methods should provide that average packet drop probability is as close as possible to the measured packet loss ratio $\overline{l(k)}$, when queue length oscillates about q_0, according to the design rules for large time-scale, as listed above. A simple method is to use the linear function, *i.e.*, $p = \overline{l(k)} + \alpha(q - q_0)$. However, it is often difficult to choose a perfect value of parameter α. For example, if $\overline{l(k)}$ is somewhat large and α is set too small, the packet drop probability for small queue length q would still be some large value, resulting in low link utility. To avoid this, in LRED, we let α increase with the measured loss ratio. Packet drop probability is then calculated as follows:

$$p = \overline{l(k)} + \beta\sqrt{\overline{l(k)}}(q - q_0), \tag{25}$$

where parameter $\beta > 0$ is a pre-configured constant.

In Eq. (25), when the queue length q is equal to the expected value q_0, packet drop probability p is equal to loss ratio $\overline{l(k)}$. On the contrary, p will be larger (or smaller) than $\overline{l(k)}$ if q is larger (or smaller) than q_0. In addition, when $\overline{l(k)}$ is a bit large (or small), p will increase (or decrease) with a large (or small) slope so as to guarantee that the packet drop probability in small queue length is much smaller.

In summary, LRED measures packet loss ratio in the large time-scale, and update packet drop probability in small time-scale at each packet arrival.

B. Stability Analysis

We now derive the design rules of parameter β, and investigate the stability of LRED. The key assumption in our analysis is that the measured packet loss ratio can be used to approximate the stable packet drop probability p_0, *i.e.*, $\overline{l(k)} \approx p_0$. Hence, we can rewrite Eq. (25) as:

$$p = p_0 + \beta\sqrt{p_0}(q - q_0), \tag{26}$$

where $p_0 = \eta N^2/(\tau^2 C^2)$ (see Eq. (3))

From Eq. (26), we can get:

$$\delta p = \beta\sqrt{p_0}\delta q. \tag{27}$$

Thus, the system transfer function in LRED can be written as:

$$P(s) = \beta\sqrt{p_0}Q(s). \tag{28}$$

Then, the constant H_c of LRED is:

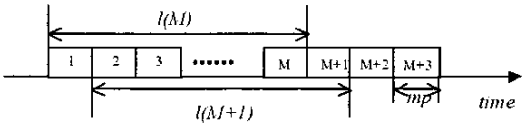


Fig. 1. Packet loss ratio measurement in LRED

```
/*Initialization */
arrPktNum=0;        dropPktNum=0;
allArrNum=0;        allDropNum=0;
index=0;  mw=0.1;  mp=1.0;  M=4;     β = 0.001;
/*LossRatioMeasure()-Called periodically every mp seconds*/
1     dropNum[index]=dropPktNum;
2     arrNum[index]=arrPktNum;
3     arrPktNum=0;
4     dropPktNum=0;
5     index++;
6     if(index==M) index=0;
7     for(i=0; i<M; i++) allDropNum+= dropNum[i];
8     for(i=0; i<M; i++) allArrNum+= arrNum[i];
9     lossRatioTemp= allDropNum/allArrNum;
10    lossRatio=lossratio*mw+ lossRatioTemp*(1-mw);
11    allDropNum=0;
12    allArrNum=0;
/* Enqueue()-Called at each packet arrival*/
1     arrPktNum++;
2     p=lossRatio+ β√lossRatio (q − q0) ;
3     p=max(0, min(1, p));
4     random=uniformRandom(0, 1);
5     if(buffer is full) {
6       Drop the packet;
7       dropPktNum++;
8     } else if(random>p) { Enqueue the packet;
9     } else {
10      Drop the packet;
11      dropPktNum++;
12    }
```

Fig. 2. The LRED scheme

$$H_c = P(s)/Q(s) = \beta\sqrt{p_0}. \tag{29}$$

Substituting the constant H_c of LRED into Eq. (15) and (16), we have the following Lemma.

Lemma 5: *In LRED, the function* $y(N,C,\tau,H_c)$ *in Eq. (15) is a decreasing function of* N, *and an increasing function of* β. *Moreover, if* $\beta < \frac{\sqrt{2\eta}(2N)^2}{\tau^3 C^3}$, *it is an increasing function of* τ.

Proof: For LRED, we have $P(s) = \beta\sqrt{p_0}Q(s)$, and hence $H_c = \beta\sqrt{p_0}$, and $p_0 = \frac{\eta N^2}{\tau^2 C^2}$. As a result, $y(N,C,\tau,H_c)$ in Eq. (15) can be calculated as:

$$y(N,C,\tau,H_c) = \sqrt{\left(\frac{2N}{\tau^2 C}\right)^4 + \frac{4\beta^2 C^2}{\eta\tau^2}} - \left(\frac{2N}{\tau^2 C}\right)^2. \tag{30}$$

It can be easily obtained from Eq. (30) that $N \uparrow \Rightarrow y \downarrow$, and $\beta \uparrow \Rightarrow y \uparrow$. Therefore, $y(N, C, \tau, H_c)$ is a decreasing function of N, and an increasing function of β.

Now we consider τ. First, write the derivative of function y about τ as follows.

$$\frac{\partial y}{\partial \tau} = \frac{-\frac{8(2N)^4}{\tau^9 C^4} - \frac{8\beta^2 C^2}{\eta \tau^3}}{2\sqrt{(\frac{2N}{\tau^2 C})^4 + \frac{4\beta^2 C^2}{\eta \tau^2}}} + \frac{4(2N)^2}{\tau^5 C^2} = \frac{-f_1(\tau)}{f_2(\tau)} + f_3(\tau).$$

To make y be an increasing function of τ, it must satisfy $\frac{\partial y}{\partial \tau} > 0$, or equivalently $[f_2(\tau) * f_3(\tau)]^2 - f_1^2(\tau) > 0$, which can be written as:

$$[f_2(\tau) * f_3(\tau)]^2 - f_1^2(\tau) = \frac{64\beta^2[2\eta(2N)^4 - \beta^2\tau^6 C^6]}{\eta^2 \tau^{12} C^2} > 0.$$

Therefore, if $\beta^2\tau^6 C^6 - 2\eta(2N)^4 < 0$ or $\beta < \frac{\sqrt{2\eta}(2N)^2}{\tau^3 C^3}$, then $f_2(\tau) * f_3(\tau) > f_1(\tau)$ and $\frac{\partial y}{\partial \tau} > 0$. As such, $y(N, C, \tau, H_c)$ is an increasing function of τ when $\beta < \frac{\sqrt{2\eta}(2N)^2}{\tau^3 C^3}$. □

Theorem 2: *Given network parameters* $(\hat{N}, \hat{C}, \hat{\tau})$, *and assume that* $\hat{\beta}$ *satisfies*

$$\hat{\tau}\omega + \arctan(\frac{\omega}{K_{11}}) = \frac{\pi}{2}, \quad \hat{\beta} > 0. \qquad (31)$$

where ω *is defined in Eqs. (15) and (16), and* $H_c = \beta\sqrt{p_0}$ *in LRED. If* $\beta < \hat{\beta} = \min(\beta, \frac{\sqrt{2\eta}(2\hat{N})^4}{\hat{\tau}^3\hat{C}^3})$, *the system remains stable for every values of* $N \geq \hat{N}$ *and* $\tau \leq \hat{\tau}$.

The proof follows from Theorem 1 and Lemma 4.

Given the bounds of network parameters, the practical value of β can then be calculated to guarantee the stability of the system based on Theorem 2. For example, consider a network in which mean packet size = 500 bytes, $\hat{C}$ =2500 packet/second (or equivalently, 10Mbps), $\hat{N}$ =300, and $\hat{\tau}$ =0.35 seconds. According to Theorem 2, the required AQM parameter $\hat{\beta}$ is 0.001. This system is also stable for all $N \geq \hat{N}$ and $\tau \leq \hat{\tau}$, if $\beta \leq \hat{\beta}$.

C. Analysis of Response Time

Since stability and response time are often in conflict with each other in system performance, existing schemes such as PI [12] and REM [13] have tried to find a tradeoff between them. If network parameters (especially N and RTT) are known *a priori*, these schemes can adjust their control constant to obtain the best response property while guaranteeing stability. However, in a dynamic network, it is difficult to obtain these parameters precisely. Therefore, such schemes often use a conservative design that first guarantees stability; the resultant response time would be very long. For example, the default parameter for PI (as used in the *ns*-2 simulation [16]) is based on some small value for N and large value for RTT. When N increases and/or RTT decreases, it will yield a long response time while still maintaining stability.

In this subsection, we present an analysis for the lower bound of response time for PI, REM and out LRED. Through comparison, we show that LRED achieves lower response time under dynamic network status.

Consider the following scenario: at time t=0, N TCP flows become active simultaneously. Assume N is large enough to make the buffer size (Q) fully filled before the system converges to a steady state with packet drop probability p_0, and expected queue length q_0.

Let us first investigate the PI scheme, which periodically updates packet drop probability with the sample frequency f_{PI} (Hz) as:

$$p(k) = p(k-1) + a[q(k) - q_0] - b[q(k-1) - q_0], \qquad (32)$$

where $a > 0$ and $b > 0$ are two constants, as specified in [12]. Before reaching the steady-state, we can assume that $q(k) \approx Q$, and $p(0)$=0. Then $p(k)$ can be also calculated from Eq. (32) as:

$$p(k) \approx k(a-b)(Q - q_0). \qquad (33)$$

Denote $p(k_0) = p_0$. The lower bound of the response time of PI (RT_{PI}) can be calculated as:

$$RT_{PI} \geq \frac{k_0}{f_{PI}} = \frac{p_0}{(Q - q_0)(a-b) f_{PI}}. \qquad (34)$$

In REM, packet drop probability is also periodically updated with a sample frequency f_{REM} (Hz), as follows [13]:

$$p(k) = 1 - \phi^{-u(k)}, \qquad (35)$$

$$u(k) = u(k-1) + \gamma[q(k) - (1-\alpha)q(k-1) - \alpha q_0)], \qquad (36)$$

$$u(k) = \max(0, u(k)), \qquad (37)$$

where $\phi > 1$, $\gamma > 0$, and $\alpha > 0$ are three constants, and the optimal values have been derived in [13]. Before reaching the

steady-state, we can also assume that $q(k) \approx Q$, $u(0)=0$, and $u(k)<<1$. Then $p(k)$ can be calculated from Eqs. (35)-(37), as follows:

$$u(k) \approx k\gamma\alpha(Q - q_0). \tag{38}$$

$$\begin{aligned} p(k) &= 1-\phi^{-u(k)} = 1-\sum_{i=0}^{i=\infty}\frac{[-u(k)\ln\phi]^i}{i!} \\ &\approx u(k)\ln\phi = k\gamma\alpha(Q-q_0)\ln\phi. \end{aligned} \tag{39}$$

Let $p(k_0) = p_0$, then the low bound of the response time for REM (RT_{REM}) can be calculated as:

$$RT_{REM} \geq \frac{k_0}{f_{REM}} = \frac{p_0}{f_{REM}\gamma\alpha(Q-q_0)\ln\phi}. \tag{40}$$

From Eqs. (34) and (40), we can see that the response time of PI or REM is dependent on their control constant parameters, buffer size Q, expected queue length q_0, and desired stable packet drop probability p_0 (which is an increasing function of TCP flow N, and a decreasing function of round-trip time RTT and link capacity C, according to the TCP throughput model [15]). Under heavy congestion or with large stable drop probability p_0, PI and REM suffer from a long response time. In addition, when the buffer size Q is small, the responsiveness of PI and REM will become worse as well.

In our LRED scheme, packet drop probability mainly depends on the measured packet loss ratio. Since the packet loss ratio is close to the stable packet drop probability p_0 in the steady state, the response time of LRED is heavily influenced by the period (mp) to measure packet loss ratio. As a result, LRED can somewhat decouple the response time and packet drop probability, thereby making its response time almost independent of network status. When network conditions are changed dynamically, LRED can still quickly regulate the system to the new steady state. On the contrary, in PI and REM, there would be a much longer response time before arriving at a new steady state.

Nevertheless, when traffic load is very light (for example, N is small, and RTT is large), there are very few packet losses, and hence more rounds of measurement are required to obtain the best stable packet loss ratio. In this case, LRED would experience a slightly longer response time, as compared to PI and REM. In summary, LRED is most suitable for heavily congested networks, though its performance is comparable to PI and REM in a network of light traffic load. In the next section, we shall further validate this finding through simulations.

IV. Simulation Results

In this section, we investigate the performance of LRED performance through *ns-2* [16] simulations. We also compare its performance with existing AQM schemes, in particular, PI [12] and REM [13]. For LRED, the parameters for loss ratio measurement (Eqs. (23) and (24)) are configured as: mw=0.1, mp=1.0s, and M=4. According to Theorem 2, β is set to 0.001. We also study LERD performance under the small value of expected queue length q_0, and compare it with AVQ [10], since it can effectively regulate queue length to a lower level, thereby achieving expected link utilization simultaneously. For PI and REM, we use the default parameters in *ns-2*[2].

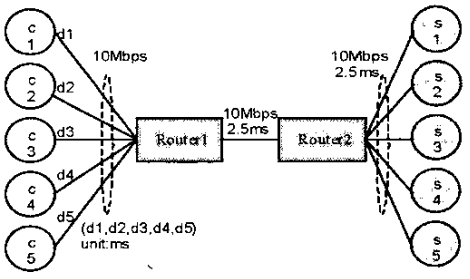


Fig. 3. Network topology

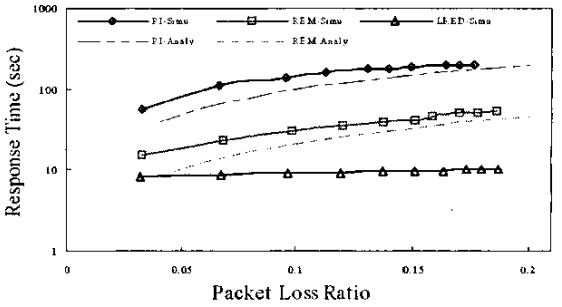


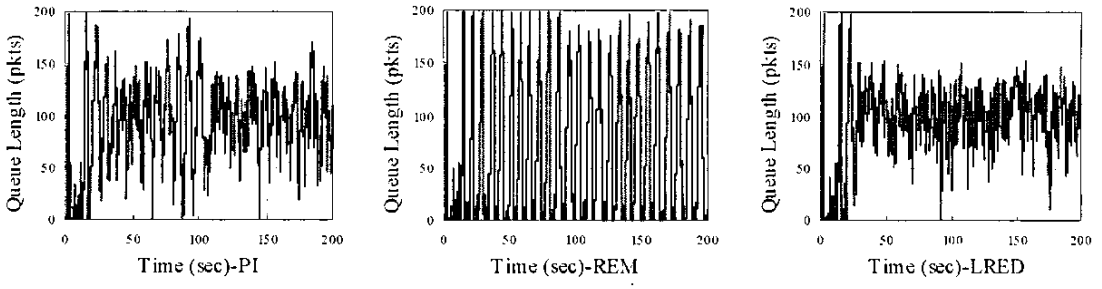
Fig. 4. Exp1: Response time for each AQM scheme[3]

The network topology for simulation is the commonly used dumb-bell topology (see Fig. 3): there is a single congestion link from router 1 to router 2, and the capacity of each link is 10Mbps. The link delay between Client $c(i)$ and router 1 is labeled as a 5-tuple $(d_1, d_2, d_3, d_4, d_5)$ with a unit of millisecond (ms). All flows are uniformly distributed among pairs of Client $c(i)$ and Server $s(i)$. The packet size is 500 bytes, and the buffer size of each router is 200 packets. We run each simulation for 200 seconds, which is long enough to observe transient as well as steady-state behaviors of the AQM schemes.

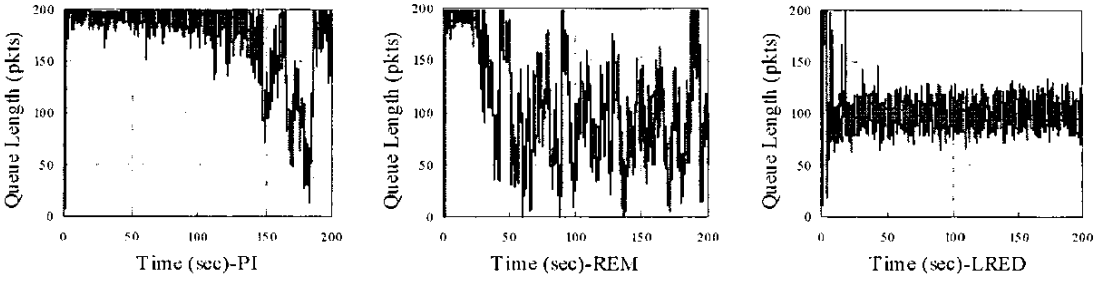
In this simulation, we focus on the following key performance metrics: goodput (excluding packet retransmissions), instantaneous queue length, average queue length, average absolute queue deviation, and packet loss ratio. The average queue length is defined as the arithmetic mean value of instantaneous queue length. The average queue deviation is defined as the absolute deviation between instantaneous queue length and its mean value.

[2] In PI, a= 0.00001822, b= 0.00001816, sampling frequency is 170Hz. In REM, $\phi = 1.001$, $\alpha = 0.1$, $\gamma = 0.001$, sampling interval is 2ms.

[3] In the last section, we obtained the lower response time bound for PI and REM. As for LRED, we only deduced that its response time is dependent on the measurement period, and presented no analysis curve.



(a): light congestion ($p_0 \approx 0.0025$)



(b): heavy congestion($p_0 \approx 0.165$)

Fig. 5. Exp2: Queue length under two extreme cases

A. Homogenous Traffic: Long-lived TCP flows only

Experiment 1: Varying congestion degree to observe response time

In this experiment, all TCP flows are persistent. The experiment's purpose is to compare the response time (RT) of the AQM schemes, which is calculated using the following approximate method:

if $AveQ(t,t+4) \le 0.1q_0$ and $AveQDev(t,t+4) \le 0.3q_0$
then $RT=t+2$

where $AveQ(t,t+4)$ and $AveQDev(t,t+4)$ are the respective average queue length and average absolute queue deviation of q_0 during [t, t+4] s.

In this experiment, the expected queue length q_0 is set to 100 packets. The link delay between Clients and Router 1 (d_1,d_2,d_3,d_4,d_5) is configured as (10, 50, 100, 150, 200). The total number of flows, N, varies from 100 to 1000, to imitate different congestion degrees.

The results are presented in Fig. 4, where the two dotted lines are the analyzed lower bound of response time for PI and REM, respectively, calculated according to Eqs. (23) and (26). The response time of LRED analyzed in the last section is only coarsely proportional to the measurement period (mp), so there is no analyzing curve for LRED in Fig. 4. It can be seen that when the packet loss ratio increases (a result of an increase in the number of TCP flows), the response time of PI or REM increases. Although there are some mismatches as compared to the analytical lower bound, the simulation results do show the trend that the response time of PI or REM is an increasing function of packet loss ratio. On the contrary, LRED is nearly irrespective of packet loss ratio; its response time mainly depends on the measurement period, as analyzed in the previous section.

Experiment 2: Stability under extreme conditions

In this experiment, the stability of the AQM schemes are investigated under two extreme cases: 1) light congestion with a small number of TCP flows N and large RTT, 2) heavy congestion with a large N and small RTT. In the first case, N=80 and $d_1 = d_2 = d_3 = d_4 = d_5 = 250ms$. In the second case, N=80 and $d_1 = d_2 = d_3 = d_4 = d_5 = 10ms$. Other parameters are the same as those in experiment 1. Fig. 5 presents the instantaneous queue length of each AQM scheme under the two cases. It can be seen that, under heavy congestion, LRED achieves a shorter response time and better stability than PI and REM. Under light congestion, the queue length of REM is oscillated and nearly out of control. On the contrary, LRED and PI can still stably regulate the queue. In this case, with large RTT, LRED has a slightly increased response time, because TCP needs more rounds to reach a stable value W_0 of its congestion window ($cwnd$) at this time.

In summary, LRED shows better stability and quick response under either light congestion or heavy congestion.

B. Non-Homogenous Traffic: Hybrid flows

Experiment 3: Adding unresponsive UDP flows

In this experiment, we investigate the influence from unresponsive UDP flows. Besides the 100 persistent TCP flows, we introduce 100 UDP flows arriving in the interval [50s, 150s]. Each UDP is an ON/OFF flow; the duration of ON and OFF state is exponentially distributed with mean

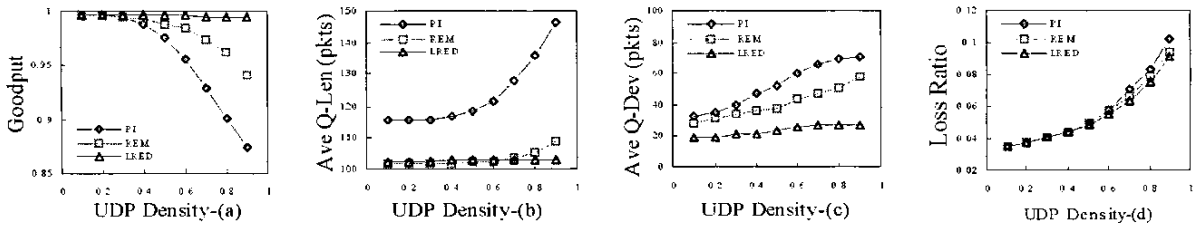
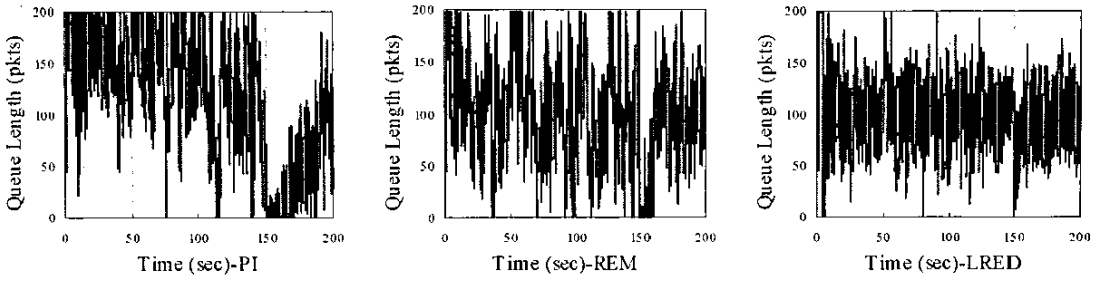
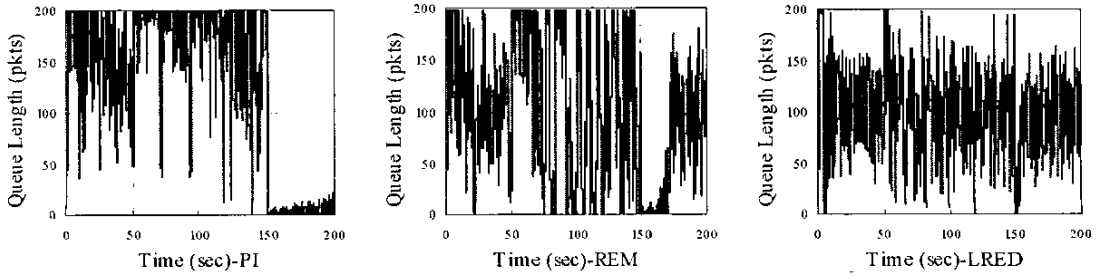


Fig. 6. Exp3: Goodput, average queue length and deviation, and packet loss ratio for each AQM scheme



(a): UDP traffic density (ρ) is equal to 0.5



(b) UDP traffic density (ρ) is equal to 0.9

Fig. 7. Exp3: Queue length for each AQM scheme

value 1.0s. The normalized density of the total UDP flow ρ is from 0.1 to 0.9. In the ON state, each UDP flow has a bit rate $r = \rho \times 10M/100$. The expected queue length and link delay between client and Router1 are: $q_0 = 100$ and $(d_1, d_2, d_3, d_4, d_5) = (10, 50, 100, 150, 200)$.

Fig. 6 presents results of average queue length, average absolute queue deviation, goodput, and packet loss ratio, as functions of the UDP traffic density. The goodput is calculated as the sum of the goodputs of the persistent TCP flows and the UDP flows.

It can be seen that LRED outperforms PI and REM in terms of these measures, especially when UDP traffic density is higher. REM also achieves better performance than PI in this experiment; however, REM is stable with more restricted network conditions than PI, as shown in experiment 2.

We show the instantaneous queue length for each AQM scheme in Fig. 7. It can be seen that: 1) The buffer under PI and REM will be overflowed (or empty) for a long time when UDP flow arrives after 50s (or stops after 150s), especially if ρ is a large value (for example 0.9), because of the slow response in PI and REM. This results in low goodput and high loss ratio, as shown in Fig. 6. On the contrary, LRED has a much shorter response time; hence there is only a sudden short-term increase (or decrease) at time 50s (or 150s). 2) When ρ increases, LRED can still regulate queue length to the expected value with much smaller deviation or overshoot than PI and REM. In summary, LRED is effective in overcoming the disturbance introduced by the unresponsive UDP flows.

Experiment 4: Adding short-lived TCP flows

Besides unresponsive UDP flows, short-lived TCP flows can also influence the control effect of AQM schemes. In this experiment, we introduce short-lived TCP flows, which arrive in intervals [50s, 150s] according to a Poisson process. The mean arrival rate λ varies from 10/s to 100/s. The length of each short-lived TCP flow is uniformly distributed in [1.0s, 2.0s]. Other parameters are the same as those in experiment 3.

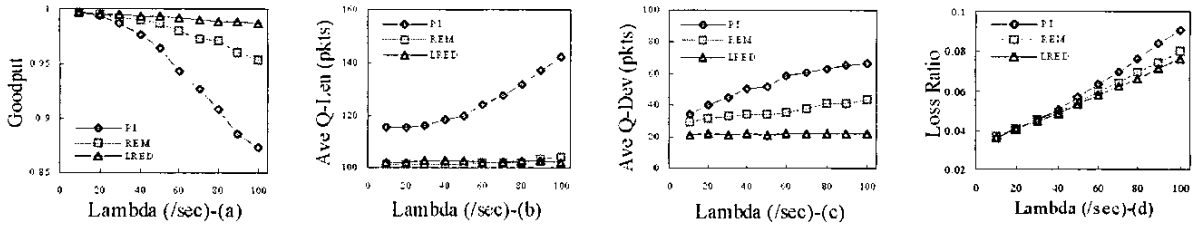
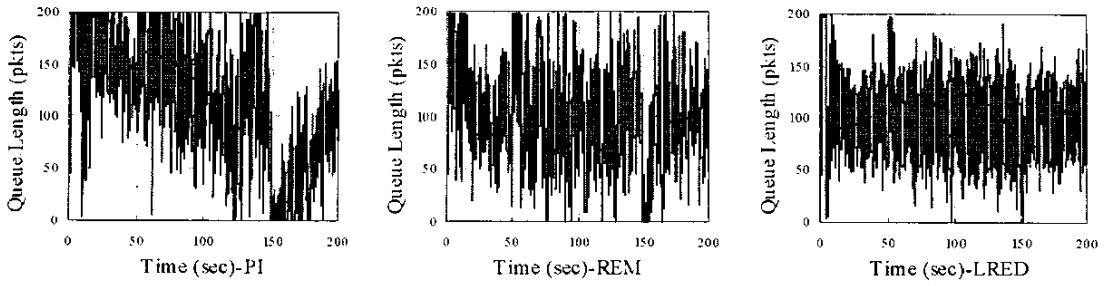
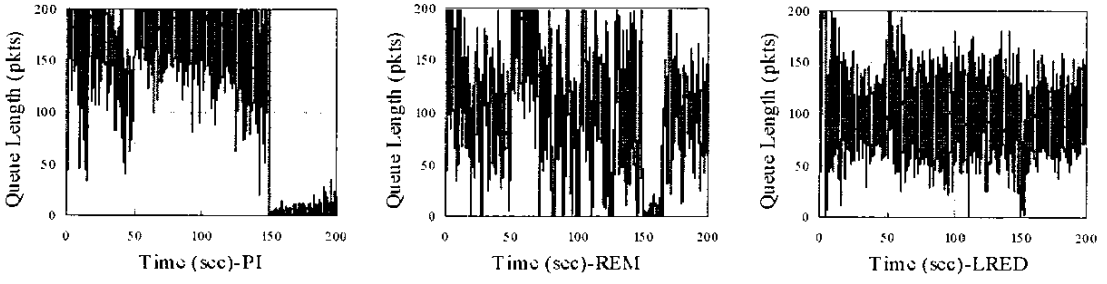


Fig. 8. Exp4: Goodput, average queue length and deviation, and packet loss ratio for each AQM scheme



(a): short-lived TCP arrival rate (λ) is equal to 30



(b) short-lived TCP arrival rate (λ) is equal to 100

Fig. 9. Exp4: Queue length for each AQM scheme

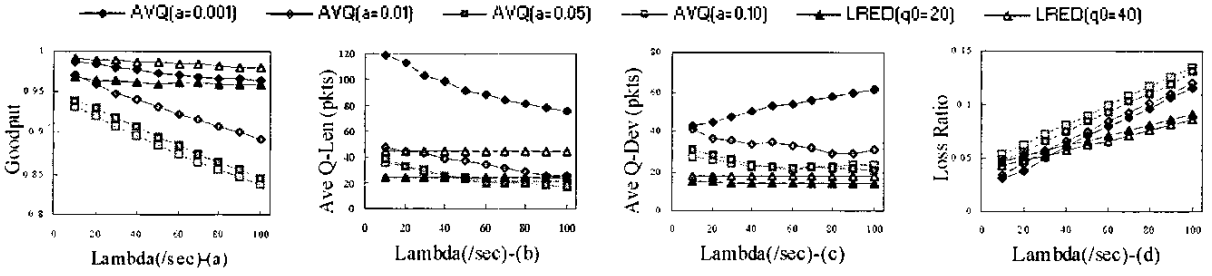


Fig. 10. Exp4: Comparison between LRED and AVQ

The results of this experiment are presented in Figs. 8 and 9. The goodput is the sum of the goodputs from TCP flows. Similar to experiment 3, LRED outperforms PI and REM in terms of average queue length, average absolute queue deviation, goodput, and packet loss ratio (see Fig. 8). Fig. 9 shows the instantaneous queue length for PI, REM, and LRED for λ =30 and 100. From this figure, it can be seen that the buffer in PI or REM is overflowed (or empty) for a long time when short-lived TCP flow arrives after 50s (or stops after 150s), especially if λ is of large value (*e.g.*, 100), because of the slow response in PI and REM. This results in low goodput and high loss ratio. On the contrary, LRED has a much shorter response time, and there is only a sudden short-term increase (or decrease) at time 50s (or 150s) in the case of LRED. When λ increases, LRED can still regulate queue length to the expected value with much smaller deviation or

overshoot than PI and REM. Therefore, LRED can effectively overcome the disturbance resulting from short-lived TCP flow, thereby achieving better stability and robustness.

Finally, we compare LRED with AVQ, which is known to be effective in regulating the queue length and achieving high link utilization [8]. In AVQ, the desired utility is set at $\gamma = 1$, and $\alpha = 0.005, 0.01, 0.05, 0.10$. To make a fair comparison, we configure q_0 (in LRED) to small values (20 and 40) to match the average queue length under AVQ. The results are presented in Fig. 10. It can be observed that LERD can achieve lower queue length and deviation, higher goodput, and lower packet loss ratio than AVQ, through choosing a certain expected queue length. In fact LRED still achieves good performance and stable control, even when using a small queue length.

In summary, LRED has a much smaller response time than PI and REM. When the network is extremely congested, LRED still shows good stability for a variety of network environments. In addition, LRED is effective in overcoming the interference from unresponsive UDP flows and/or short-lived TCP flows. It also exhibits good performance even when the expected queue length q_0 is set at a small value.

V. Related Work

There have been many proposals on AQM. The *de facto* standard, RED, uses average queue length to calculate the packet drop probability and then to regulate the queue length. When the average queue length is higher than a pre-configured threshold (min_{th}), RED begins to drop newly-arrival packets with a probability proportional to average queue length and with a slope of max_p. Despite its simplicity, it is difficult to optimally configure the parameters of RED. Therefore, many variants of RED have been proposed to adaptively configure the parameters. For example S-RED [7] and ARED [8] propose adaptive methods to adjust max_p, respectively, using events of buffer overflow and emptiness, and event of queue increasing. However, these approaches introduce additional parameters that need to be configured again.

BLUE [9] is another type of adaptive scheme. It adaptively calculates packet drop probability based on only two events: buffer overflows and buffer emptiness. When the buffer is overflowed (or empty), it increases (or decreases) packet drop probability by δ_1 (or δ_2). However it is hard for BLUE to control the queue length to an expected value.

AVQ [10] uses only input rate $x(t)$ to control packet drop and to achieve expected link utility γ, while keeping low queue length. Packet drop probability is basically proportional to the mismatch between input rate and expected link utility γ. Through maintaining a virtual queue, AVQ deterministically drops packets upon the arrival of a new packet, realizing the same effect of probabilistic packet drop. As shown in [10], AVQ can achieve low average queue length and high link utility.

Recently, some improved schemes jointly use queue length and input rate to achieve better stability. An example is PI [11][12], which tries to regulate queue length to the expected value using queue mismatch and its integral. The integral of queue length mismatch is factually related to the input rate mismatch. Packet marking/drop probability in PI is periodically and iteratively updated according to Eq. (21). PI provides design rules through control-theory analysis to choose its parameter value. However, when the network parameters are unknown *a priori*, PI can only use conservative design to guarantee stability, yielding long response time, as shown in our simulation.

REM [13] also tries to control the queue length to the expected value. It uses the linear combination of queue mismatch and input rate mismatch to calculate marking/drop probability. In REM, input rate mismatch is equivalently simplified to queue variance between two continuous samplings. Like PI, packet marking/drop probability in REM is also periodically and iteratively updated according to Eq. (24). As shown in experiment 2, REM is stable for a more narrow variety of network environments than PI and LRED, although it has a quicker response than PI.

SFC [14] also uses queue mismatch and input rate mismatch as a congestion index, trying to regulate queue length to the expected value. Packet marking/drop probability in SFC is updated upon arrival of a new packet as:

$$p(k) = k_1[q(k) - q_0] + a.[x(k) - C], \tag{41}$$

$$p(k) = \max\{0, \min[1, p(k)]\}, \tag{42}$$

where the $x(k)$ is the measured traffic input rate at the *k-th* packet arrival time, and $k_1 > 0$ and $a > 0$ are two constants. Through solving the system characteristic polynomial, SFC gives design rules of choosing k_1 and a, under the assumption that $\delta p = p$ (Eq. 15 in [14]). Such an assumption (also made in PI [12]) might be problematic and ineffective under heavy congestion with a large value of p_0, and will cause slow response, as shown in our simulations.

If network parameters remain unchanged, we can assume that $\overline{l(k+1)} \approx \overline{l(k)}$. Then Eq. (25) (for packet drop probability calculation in LRED) can be simplified as:

$$p(i+1) = p(i) + \beta\sqrt{\overline{l(k)}}[q(i+1) - q(i)]. \tag{43}$$

Though Eq. (43) is somewhat similar to Eqs. (32) and (33), there are important differences between LRED and PI (and REM). Specifically, in PI, the packet dropping probability is always iteratively computed. If the stable probability p_0 is of a large value (lower *RTT*, large TCP flow number), PI needs a longer time to make the drop probability p converge to the p_0, as shown in section IV. On the contrary, since LRED uses packet loss ratio to adjust packet drop probability (Eq. (25)), its convergence rate and response time are nearly independent of the stable value p_0, and primarily influenced by the measurement period. As shown in simulations, after

only several measurement periods, LRED can make the drop probability p converge to the p_0.

When network parameters are dynamic, the required stable probability p_0 will be changed correspondingly. However, LRED can adapt itself quickly to a new stable state through measuring packet loss ratio. On the contrary, PI and LRED often need a longer time to converge to the new stable state, which will cause large queue deviation and lower throughput, as shown in our simulations.

VI. Conclusions

In this paper, we proposed a new AQM scheme, LRED, which incorporates packet loss ratio (in addition to queue length) for congestion estimation. Under LRED, packet drop probability is updated under multiple time-scale. At packet level, LRED uses instantaneous queue mismatch to update packet drop probability upon arrival of new packets. On the larger time-scale, LRED adjusts the packet drop probability using the measured packet loss ratio. We analyzed stability and response time of LRED. We also conducted extensive simulations to evaluate its performance, and compared it with existing AQM schemes such as PI, REM, and AVQ. Our results showed that LRED has a shorter response time than PI and REM, especially under heavy congestion scenarios. More importantly, LRED achieves better stability and robustness than PI and REM under dynamic environments, where the number of TCP flows N and RTT varies significantly, or where there are many short-lived flows or unresponsive UDP flows. Finally, LRED can effectively control the queue length to an expected value. It also achieves a better tradeoff between goodput and queue length than the well-known AVQ scheme.

References

[1] R. Gurin and V. Peris, "Quality-of-service in packet networks: basic mechanisms and directions," *Computer Networks*, vol. 31, no. 3, pp. 169-179, Feb. 1999.

[2] Y.T. Hou, D. Wu, B. Li, T. Hamada, I. Ahmad, and H. J. Chao, "A differentiated services architecture for multimedia streaming in next generation Internet," *Computer Networks*, vol. 32, no. 2, pp. 185-209, Feb. 2000.

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

[4] S. Floyd, "TCP and explicit congestion notification," *ACM Computer Communication Review*, vol. 24, pp. 10-23, Oct. 1994.

[5] S. Floyd and V. Jacobson, "Random early detection gateways for congestion avoidance," *IEEE/ACM Trans. on Networking*, vol. 1, no. 4, pp. 397-413, Aug. 1993.

[6] S. Floyd, "Recommendation on using the 'gentle_' variant of RED," http://www.icir.org/floyd/red/gentle.html, Mar. 2000.

[7] W. Feng, D. D. Kandlur, D. Saha, and K. G. Shin, "A self-configuring RED Gateway," In *Proceedings of IEEE INFOCOMM*, vol. 3, pp. 1320-1328, New York, NY, 1999.

[8] S. Floyd, R. Gummadi, and S. Shenker, "Adaptive RED: An algorithm for increasing the robustness of RED's active queue management," *http://www.icir.org/floyd/papers/adaptiveRed.pdf*, Aug. 2001.

[9] W. Feng, D. D. Kandlur, D. Saha, and K. G. Shin, "The blue active queue management algorithms," *IEEE/ACM Trans. on Networking*, vol. 10, no. 4, pp. 513-528, Aug. 2002.

[10] S. Kunniyur and R. Srikant, "Analysis and design of an adaptive virtual queue (AVQ) algorithm for active queue management," In *Proceedings of ACM SIGCOMM*, pp. 123-134, San Diego, CA, Aug. 2001.

[11] C. Hollot, V. Misra, D. Towsley, and W. Gong, "A control theoretic analysis of RED," In *Proceedings of IEEE INFOCOM*, vol. 3, pp. 1510-1519, Anchorage, Alaska, 2001.

[12] C. Hollot, V. Misra, D. Towsley, and W. Gong, "Analysis and design of controllers for AQM routers supporting TCP flows," *IEEE Trans. on Automatic Control*, vol. 47, pp. 945-959, Jun. 2002.

[13] S. Athuraliya, S. Low, V. Li, and Q. Yin, "REM: Active queue management," *IEEE Network Magazine*, vol. 15, pp. 48-53, May 2001.

[14] Y. Gao and J. C. Hou, "A state feedback control approach to stabilizing queues for ECN-enabled TCP flows," In *Proceedings of IEEE INFOCOM*, vol. 3, pp. 2301-2311, San Francisco, CA, 2003.

[15] J. Padhye, V.Firoiu, D. Towsley, and J. Kurose, "Modeling TCP throughput: A simple model and its empirical validation", In *Proceedings of ACM SIGCOMM*, pp. 303-314, Vancouver, CA, Aug. 1998.

[16] UCN/LBL/VINT, Network simulator-ns2, *http://www-mash.cs.berkeley.edu/ns* .