On Exploiting Traffic Predictability in Active Queue Management

Yuan Gao,
Department of Electrical Engineering
The Ohio State University
{gaoy}@ee.eng.ohio-state.edu

Guanghui He and Jennifer C. Hou
Department of Computer Science
University of Illinois at Urbana Champaign
{ghe,jhou}@cs.uiuc.edu

Abstract—

Analytical and empirical studies have shown that self-similar traffic can have detrimental impact on network performance including amplified queuing delay and packet loss ratio. On the flip side, the ubiquity of scale-invariant burstiness observed across diverse networking contexts can be exploited to better design resource control algorithms. In this paper, we explore the issue of exploiting traffic predictability to enhance the performance of active queue management (AQM). We show that the correlation structure present in long-range dependent traffic can be detected on-line and used to accurately predict the future traffic. We then design, with the objective of stabling the instantaneous queue length at a desirable level, a LMMSE-based controller, and figure in the prediction results in the calculation of the packet dropping probability. The resulting scheme is termed as predictive AQM (PAQM).

Through analytical reasoning, we show that PAQM is a generalized version of RED with a new dimension of congestion index — the amount of traffic that will arrive in the next few measurement intervals. By stabilizing the queue at a desirable level with consideration of future traffic, PAQM enables the link capacity to be fully utilized, while not incurring excessive packet loss. Through ns-2 simulation, we compare PAQM against existing AQM schemes with respect to different performance criteria. In particular, we show that under most cases PAQM outperforms SRED in stabilizing the instantaneous queue length, and adaptive virtual queue (AVQ) in reducing packet loss ratio and better utilizing the link capacity.

I. Introduction

A number of recent empirical studies of traffic measurements from a variety of working packet networks have convincingly demonstrated that network traffic is self-similar or long-range dependent (LRD) in nature [14], [6], [19], [20]. This implies the existence of concentrated periods of low activity and high activity (i.e., burstiness) at a wide range of time scales. Scale-invariant burstiness introduces new complexities into resource control and QoS provisioning. On one hand, burstiness at coarser time scales induces extended periods of either over-utilization or under-utilization. Packets that arrive in the periods of over-utilization experience long delays and are even dropped due to buffer overflow, while packets that arrive in the periods of under-utilization experience the opposite. The large variation in the end-to-end delay packets experience has an adverse impact on transport of QoS-sensitive traffic such as multimedia traffic. On the other hand, if resource reservation is deployed for QoS provisioning, resources have to be reserved with respect to the *peak* rates over a *wide* range of time scales. This adversely affects resource control and degrades the overall performance.

While the LRD characteristic of network traffic introduces difficulty and complexity into traffic and resource management, it also opens up a new direction for research — the existence of nontrivial correlation structure at larger time scales can be judiciously exploited for better congestion and resource control. How to exploit the abundant correlation structure to improve the performance of active queue management (AQM) is the subject matter of this paper.

Central to the notion of AQM with traffic prediction is prediction of the future traffic based on recent traffic measurements and use of the prediction results to modulate the magnitude of the packet dropping probability at a router. We first design a traffic predictor based on the *linear minimum mean square error* (*LMMSE*) estimation. Through *ns-2* simulation and analytical reasoning, we show that the LMMSE based predictor performs as accurately as the other fractional-model based predictors (e.g. Fractional Brownian Motion (*FBM*) model and the Fractal ARIMA model), and yet can be practically implemented in hardware/software with readily available fast algorithms. Then, we design a simple controller that takes the prediction result (i.e., the amount of traffic to arrive in the next few measurement intervals) as input to determine the packet dropping probability. The objective here is to stabilize the queue length at a desirable level. We choose this objective for two reasons: if the queue length can be controlled (in anticipation of future traffic) to stabilize at certain small value (or an pre-determined trajectory), (i) the delay jitter experienced by packets that traverse the router remain approximately constant (which is an important criterion for transporting QoS-sensitive multimedia applications); and (ii) the capacity of the link can be fully utilized even under pro-active packet dropping/marking. The resulting AQM scheme is termed as *predictive AQM* (*PAQM*).

The controller output of PAQM indicates that RED can be viewed as a special case of PAQM, with the predicted future traffic always fixed at certain value. In other words, PAQM augments the set of congestion indices with a new dimension — the amount of traffic to arrive in the next few measurement intervals (or equivalently the future arrival rate). This also avoids use of the queue length as both the congestion *and* performance indices — which is commonly believed to be the cause that queue-length-based AQM schemes (such as RED) cannot simultaneously achieve high link utilization and low packet loss ratio [3]. By stabilizing, with consideration of future traffic, the queue at a desirable level, PAQM enables the link capacity to be fully utilized, while not incurring excessive packet loss ratio. Through *ns-2* simulation, we show that PAQM can indeed simultaneously achieve both.

The rest of the paper is organized as follows. In Section II,

The work reported in this paper was supported in part by NSF under Grant No. ANI-0082861.

0-7803-7476-2/02/$17.00 (c) 2002 IEEE.

we give an overview of AQM with traffic prediction. Then we delve into the detailed descriptions of PAQM. In particular, we propose in Section III an LMMSE-based predictor, and compare it against FBM and Fractal ARIMA. We also validate the design of the predictor via *ns-2* simulation. In Section IV, we present the design of the controller and derive the expression of packet dropping probability to be used in the next time interval. Following that, we give a taxonomy, and a detailed summary, of existing AQM schemes in Section V, and present the simulation results in Section VI. Finally the paper concludes in Section VII.

II. Overview of AQM with Traffic Prediction

AQM refers to the notion of equipping core routers inside a network with the capability to detect incipient congestion and to explicitly signal traffic sources before congestion actually takes place [5]. AQM differs from the traditional drop tail queue in that in a drop-tail queue packets are dropped when the buffer overflows, while in active queue management schemes, packets may be dropped early before buffer overflow occurs. To illustrate the operations of AQM, we use RED as an example scheme. RED operates by calculating the average queue length, *avg_queue*, upon packet arrival using exponentially weighted moving average. Then *avg_queue* is used as the congestion index. The policy used to detect likelihood of congestion is characterized by two thresholds, min_th and max_th. If *avg_queue* is less than min_th, the router is considered congestion free and no arriving packets are dropped. On the other hand, if *avg_queue* is greater than max_th, all arriving packets are (early) dropped to notify the end hosts of (the likelihood of) congestion. When *avg_queue* is between the two thresholds, the probability of (early) dropping a packet linearly increases with *avg_queue* from 0 to p_{max}, where p_{max} is the pre-determined maximum packet dropping probability.

RED was shown in [9] to prevent global synchronization,accommodate bursty traffic, incur little overheads, and coordinate well with TCP under serious congestion conditions. The performance of RED, however, heavily depends on whether or not the two thresholds are properly tuned. Also, RED was shown to be unfair to individual flows [15], unable to achieve high link utilization and low packet loss ratio simultaneously [8], [3], [13], and exhibit short-term fluctuation in the queue length [17]. In particular, it was shown in [8], [3], [13], [11] that queue length should not be the only parameter to be observed and controlled. Instead, other control variables and control policies should be deployed. In what follows, we propose a novel AQM scheme that takes a new angle and manages the queue in anticipation of future incoming traffic.

A. AQM with Traffic Prediction

The block diagram is shown in Fig. 1. Let $f(t)$ denote a time series representing the amount of traffic (in bytes) that arrives at a router in a sampling interval. The predictor keeps track of the aggregate serie samples,

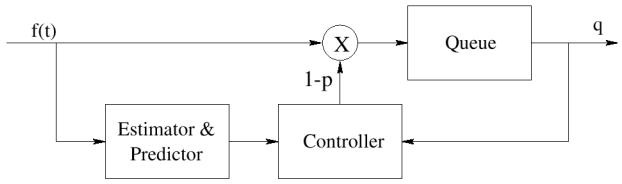


Fig. 1. AQM with traffic prediction.

$f^m(1), f^m(2), \ldots, f^m(n)$, measured in the past n measurement intervals(each of which is of length τ), where $f^m(k)$, $1 \leq k \leq n$, is the aggregate serie samples taken in the $(n+1-k)$th most recent interval, i.e.,

$$f^m(k) = \frac{1}{m} \sum_{i=(k-1)m+1}^{km} f(i), \tag{1}$$

where m is the number of sampling intervals in one measurement interval. Based on these aggregate serie samples, the predictor predicts the amount of traffic, $\widehat{f}(n+1) \stackrel{\triangle}{=} f^m(n+1) \cdot m$, and $\widehat{f}(n+2) \stackrel{\triangle}{=} f^m(n+2) \cdot m$, in the next two intervals. The controller then uses the predicted results to modulate the packet dropping probability, p, to be used in the next time interval, with the objective of stabilizing the instantaneous queue length at a desirable and stable level. As mentioned in Section I, the fact that the amount of future traffic is figured in the calculation of packet dropping probability is equivalent to augmenting the set of congestion indices with a new dimension.

III. Design of the Traffic Predictor

Several issues have to be addressed in designing a good traffic predictor: First, we have to determine the traffic quantity to measure and predict. In this work, we measure and predict the amount of traffic averaged over a measurement interval (Eq. (1)). This is due to the following two reasons: (i) the *LRD* characteristic of the Internet traffic implies that the autocorrelation function of the average traffic obeys the same law as that of the original traffic. Hence, the aggregate time series is still a good representative of the original time series. (ii) The averaging operation in Eq. (1) can be viewed as sampling and smoothing operations. As reported in [18], smoothed and sampled traffic exhibits more predictable behaviors, and hence prediction based on the aggregate traffic is more accurate.

Second, we have to determine which prediction method to use. On one hand, prediction should be grounded on a solid theoretic foundation, but on the other hand, the resulting predictor should be easy to implement, and the parameters that have to be on-line measured/estimated should be kept minimal. Third, we have to determine how far ahead traffic prediction is made, because there exists a trade-off between how far ahead prediction can be made and how significant the prediction error is([18]).

Instead of using existing fractional models for time series with LRD, we propose a predictor based on linear minimum

0-7803-7476-2/02/$17.00 (c) 2002 IEEE.

mean square error (*LMMSE*) estimation. In what follows, we elaborate on the design of the LMMSE based predictor, and compare it against two widely used fractional models: Fractional Brownian Motion (*FBM*) model and the Fractal ARIMA model, in terms of accuracy and ease of implementation. We also comment on how we determine the length, τ, of each measurement interval.

A. LMMSE Predictor

Specifically, given the aggregate series $f^m(k), k = 1, ..., n$, where $f^m(k)$ is given in Eq. (1), we predict the aggregate series sample, $f^m(n+1)$, in the next interval as a weighted sum of the past n average samples:

$$\widehat{f}^m(n+1) = \begin{bmatrix} a_1 & a_2 & ... & a_n \end{bmatrix} \begin{bmatrix} f^m(1) \\ f^m(2) \\ ... \\ f^m(n) \end{bmatrix}, \quad (2)$$

where $a_1, a_2, ..., a_n$ are the *LMMSE* coefficients and can be expressed as

$$\begin{bmatrix} a_1 & a_2 & .. & a_n \end{bmatrix} = \begin{bmatrix} R(n) & R(n-1) & .. & R(1) \end{bmatrix} \times \begin{bmatrix} R(0) & R(1) & .. & R(n-1) \\ R(1) & R(0) & .. & R(n-2) \\ .. & .. & .. & .. \\ R(n-1) & R(n-2) & .. & R(0) \end{bmatrix}^{-1} \quad (3)$$

where $R(n)$ is the covariance function of the time series, and can be estimated (due to the property of asymptotically second-order self-similarity) in practice as

$$R(i) = \frac{1}{n} \sum_{t=i+1}^{n} f^m(t) f^m(t-i), 0 \le i \le n-1, \quad (4)$$

where n is the number of aggregate series samples kept and is a tunable parameter. (In the simulation study, we use $n = 20$, as in all the simulation runs conducted the performance improvement is marginal as n exceeds 20.)

Similarly, to estimate the aggregate series sample, $f^m(n+2)$, in the interval following the next time interval, we take the estimated value of $f^m(n+1)$ as a series sample, and calculate

$$\widehat{f}^m(n+2) = \begin{bmatrix} a_1 & a_2 & ... & a_n \end{bmatrix} \begin{bmatrix} f^m(2) \\ f^m(3) \\ ... \\ \widehat{f}^m(n+1) \end{bmatrix}.$$

The mean square error of the *LMMSE* predictor can be calculated (after a few algebraic operations) as

$$\sigma^2 = \sigma_x^2 - \begin{bmatrix} R(n) & R(n-1) & ... & R(1) \end{bmatrix} \cdot \begin{bmatrix} R(0) & ... & R(n-1) \\ R(1) & ... & R(n-2) \\ ... & ... & ... \\ R(n-1) & ... & R(0) \end{bmatrix}^{-1} \cdot \begin{bmatrix} R(n) \\ R(n-1) \\ ... \\ R(1) \end{bmatrix}. \quad (5)$$

Since the *LMMSE* predictor does not depend on any fractional model, the close form of σ^2 cannot be easily obtained. Instead, we use the following asymptotic result as an approximation:

$$R(\tau) \sim H(2H-1)\tau^{2H-2}. \quad (6)$$

Implementation issues. To practically implement the *LMMSE* predictor, we consider the following three issues:

1. The *LMMSE* predictor is derived under the assumption of zero mean stationary stochastic process. As the stochastic time series on-line measured may not be of zero mean, we subtract the mean value from the original time series, apply the *LMMSE* predictor to estimate the average of the time series in the next interval, and then add back the mean value.

2. We have to determine how far ahead traffic prediction is made. To simplify the implementation, we make prediction-ahea interval same as measurement interval (τ). Choice of τ is a trade-off between the prediction accuracy and computational overhead. Fortunately, the LRD characteristic of the network traffic implies the relatively low decay of the autocorrelation function, and hence the value of τ can be chosen relatively large. In the simulation study, we set the value of τ to be in the range of 0.02 to 0.05 seconds.

3. The operations involved in the calculation of LMMSE coefficients (Eqs. (3) and (4)) are multiplication of time series samples and matrix manipulation, for which fast algorithms exist [1]. Hence they can be readily implemented in hardware. In particular, the author of [1] gave an adaptive algorithm in linear prediction in which the matrix inverse operation in Eq. (4) can be avoided. Succinctly, the algorithm starts with an initial estimate of the coefficient a_0. Each time a new data point, $f^m(n)$, is obtained, the algorithm updates a_{n+1} using the recursive equation:

$$a_{n+1} = a_n + \mu \cdot e(n) \cdot f^m(n), \quad (7)$$

where $e(n)$ is the prediction error and μ is a constant. If $f^m(n)$ is stationary, a_i is shown to converge in the mean to the optimal solution [1]. The interested reader is referred to [1] for a detailed account. Since, the prediction is executed only once per measurement interval (τ), the computational overhead is small, considering the current CPUs' capability.

B. Comparison with Fractional Model-Based Predictors

Several fractional models and their corresponding predictors have been proposed for time series with LRD characteristics, among which the FBM model and the fractal ARIMA model may have received the most attention. The interested reader is referred to the extended version of this paper [10] for a brief summary of the two fractional model-based predictors and several important results that are relevant to the discussion of traffic prediction in Section III-C.

The reason why we use the *LMMSE* predictor instead in predicting incoming traffic at a router is two fold:

Accuracy: The most important criterion in choosing a predictor is the accuracy. We depict the relative mean square errors versus the Hurst parameter H under the LMMSE predictor (Eq. (5)/σ_x^2), the FBM predictor (Eq. (17) in [10]), and the fractal ARIMA predictor (Eq. (22) in [10]) in Fig. 2. When $H \to 1$ the relative error converges to 0 under all three models, suggesting that the more pronounced the LRD characteristic, the better the performance of predictors. Moreover, the three curves are close to one another

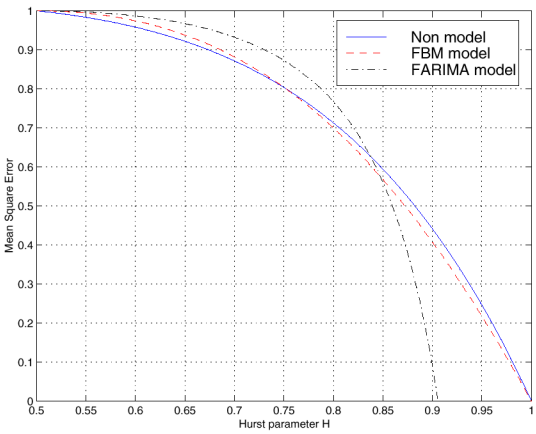


Fig. 2. Comparison of mean square errors among the *FBM*, *FARIMA*, and *LMMSE* predictors.

when $H \leq 0.85$ (beyond which the curve corresponding to the *FARIMA* model based predictor differs notably). Since analysis of real traffic traces indicates that the H parameter rarely exceeds 0.85 [19], from the theoretic perspective, all three predictors are equally well-suited for Internet traffic prediction.

Ease of implementation: To implement the two fractional model based predictors, one has to on-line estimate the H parameter and engage in complicated calculation of weight coefficients. Specifically, one has to estimate the value of H in the *FBM* model and calculate the weight coefficient in the form of

$$\frac{sin(\pi(H-\frac{1}{2}))}{\pi}(-t(T+t))^{-H+\frac{1}{2}}\int_0^a \frac{(\tau(\tau+T))^{H-\frac{1}{2}}}{\tau-t}d\tau.$$

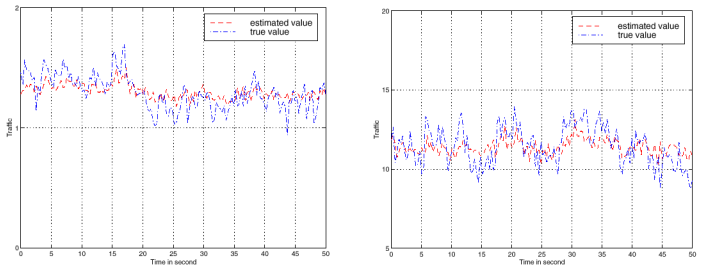
where T is the interval in which the time series samples are gathered in order to predict the time series at time $T+t$. Similarly, one has to estimate the value of $d = H-1/2$ in the fractal *ARIMA* model and calculate the weight coefficient in the form of

$$\beta_{kj} = -\begin{pmatrix} k \\ j \end{pmatrix} \frac{\Gamma(j-d)\Gamma(k-d-j+1)}{\Gamma(-d)\Gamma(k-d+1)},$$

where $\Gamma()$ is the gamma function. (The interested reader is referred to [10] for a detailed account.) As a result, it is more difficult to practically implement model-based predictors in router hardware/software. The *LMMSE* predictor, on the other hand, does not require estimation of such parameters, but instead calculates these parameters (in particular, $R(i)$'s in Eq. (4)) directly from the collected traffic samples. Moreover, as mentioned in Section III-A, there exist fast algorithms that can be readily implemented to perform operations involved in the calculation of LMMSE coefficients (Eqs. (3) and (4)) [1].

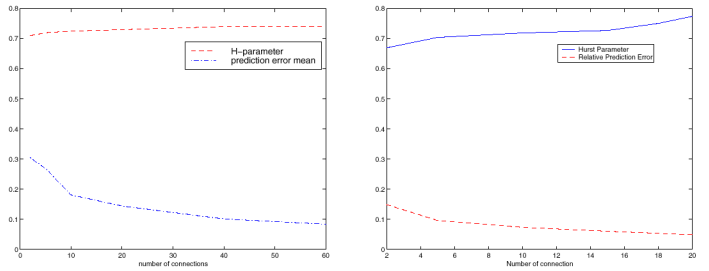
C. Validation of the LMMSE Predictor

To validate the design of the *LMMSE* predictor, we have implemented it in a router in *ns-2* and tested its prediction capability in both single-bottleneck networks and networks of arbitrary topology. We found that the actual traffic and the estimated traffic agree very well. To illustrate this, we depict in Fig. 3 the actual traffic over a bottleneck link and its corresponding *LMMSE* estimate. The single bottleneck has a capacity of 20Mbps and a buffer size of 100 packets (each of 1000 bytes). Totally 60 connections are established over the bottleneck link, with each source generating packets either (i) using the on-off traffic model or (ii) using real network traffic traces. The H parameter estimated under both cases in Fig. 3 are 0.75 ((a)) and 0.70 ((b)), respectively. This confirms the LRD characteristic of the traffic. Moreover, under both cases, the estimated traffic agrees very well with the actual traffic. In the former case, the ratio of the mean estimate error to the actual value is 0.08, while in the latter case, the ratio is approximately 0.15.



(a) on-off model (b) real network traffic traces

Fig. 3. Actual and estimated traffic traces when TCP packets are generated using the on-off model or real network traffic traces in *ns-2*.



(a) on-off model (b) real network traffic traces

Fig. 4. The Hurst parameter and the estimation error versus the number of connections.

To study the effect of the number of connections on the performance, we repeat the above experiment, but vary the number of connections on the bottleneck link. Fig. 4 depicts the Hurst parameter and the estimation error versus the number of connections. As shown in Fig. 4, the traffic observed on the bottleneck link exhibit the LRD characteristic under both cases, regardless of the number of connections. In particular, the estimation error of the *LMMSE* predictor is less than 0.2 when the number of connections exceeds 10 in the former case, and less than 0.1 when the number of connections exceeds 4 in the latter case.

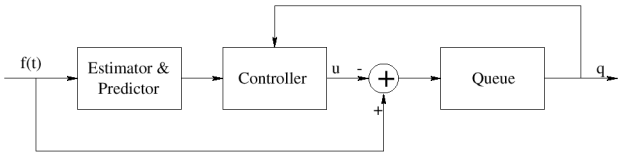


Fig. 5. An alternate block diagram for AQM with traffic prediction.

IV. Design of the Controller

Recall that in Fig. 1 the controller utilizes the prediction results (i.e., the amount of traffic that arrive in the next two intervals) to determine the magnitude of the packet dropping probability. The controller in Fig. 1 is not easy to analyze, as it is a non-linear system. Hence we replace the controller with an alternative, but equivalent one in Fig. 5. In this system, the output, $u(t)$, of the controller is the amount of packets that should be dropped ($0 \le u(t) \le f(t)$, where $f(t)$ is the amount of traffic that arrives in one measurement interval). By setting $p = \frac{u(t)}{f(t)}$, the two systems are equivalent.

Recall that traffic predication is made on a per-interval basis (where the interval is of duration τ). Let $Q(k)$ denote the queue length at the end of the kth interval. Then, the discrete-time function of the queue length is $Q(k+1) = Q(k) + f(k+1) - u(k+1) - C$, where $C = R \times \tau$ is the number of packets transmitted on the outgoing link and R is the capacity of the outgoing link.

Now the objective of the controller is to keep the queue length at an appropriate level or follow an pre-determined, time-variant trajectory. Specifically, let $\widehat{Q}(k)$ and $Q_{opt}(k)$ denote, respectively, the discrete-time functions of the predicted and desirable queue lengths at the end of the kth interval. Then, the objective function we consider is either

$$J = \sum_{i=1}^{M} \left(\widehat{Q}(k+i) - Q_{opt}(k+i) \right)^2, \quad \text{or} \tag{8}$$

$$J = \sum_{i=1}^{M} \left(\widehat{Q}(k+i) - Q_{opt}(k+i) \right)^2 + \sum_{i=1}^{M} b_{i-1} u^2(k+i), \tag{9}$$

where M is the number of time intervals to be predicted ahead and b_i is the weight for the controller output. Note that the second term in Eq. (9) is used to prevent the system from dropping packets excessively. The optimization problem can then be formally stated as

Problem 1: Find $u^* = \arg\min_u J$, where J is expressed in either Eq. (8) or Eq. (9), subject to $0 \le u(k+i) \le \widehat{f}(k+i), 1 \le i \le M$.

We first solve Problem 1 with the objective function of Eq. (8) in the case of $M = 2$ and $Q_{opt}(k) = \overline{Q}, \forall k$, where $\overline{Q}$ is a pre-determined value. To obtain $u^*(k)$ and $u^*(k+1)$, we take the derivatives of J with respect to $u(k+1)$ and with respect to $u(k+2)$, and set them to zero:

$$\frac{\partial J}{\partial u(k+1)} = 0, \quad \text{and} \quad \frac{\partial J}{\partial u(k+2)} = 0.$$

The results of the above equations are (after a few algebraic operations) are

$$\begin{cases} u^*(k+1) = Q(k) + \widehat{f}(k+1) - C - \overline{Q}, \\ u^*(k+2) = \widehat{f}(k+2) - C. \end{cases} \tag{10}$$

Using a similar procedure, we can solve problem 1 with the objective function of Eq. (9) in the case of $M = 2$ and $Q_{opt}(k) = \overline{Q}, \forall k$. The results are

$$\begin{cases} u^*(k+1) = -\frac{(4b_1-1)(Q(k)+\widehat{f}(k+1)-\overline{Q})+2b_1\widehat{f}(k+2)+(1-6b_1)C}{1-4b_1-2b_0+4b_0b_1}, \\ u^*(k+2) = -\frac{2b_0(Q(k)+\widehat{f}(k+1)-\overline{Q})+(2b_0-1)\widehat{f}(k+2)+(1-4b_0)C}{1-4b_1-2b_0+4b_0b_1}. \end{cases} \tag{11}$$

Note that Eq. (11) reduces to Eq. (10) when $b_1 = b_2 = 0$. As $u^*(k+1)$ and $u^*(k+2)$ obtained in Eq. (11) may violate the constraints of $0 \le u(k+1) \le \widehat{f}(k+1)$ and $0 \le u(k+2) \le \widehat{f}(k+2)$, we consider the following four cases:

(C1) If $u^*(k+1) \in [0, \widehat{f}(k+1)]$ and $u^*(k+2) \in [0, \widehat{f}(k+2)]$, then $u(k+1) \leftarrow u^*(k+1)$ and $u(k+2) \leftarrow u^*(k+2)$.

(C2) If $u^*(k+1) \notin [0, \widehat{f}(k+1)]$ and $u^*(k+2) \notin [0, \widehat{f}(k+2)]$, then the optimal solution is one of the four possible pairs, $(0,0)$, $(\widehat{f}(k+1), 0)$, $(0, \widehat{f}(k+2))$ and $(\widehat{f}(k+1), \widehat{f}(k+2))$. One can compute $J(0,0)$, $J(\widehat{f}(k+1), 0)$, $J(0, \widehat{f}(k+2))$ and $J(\widehat{f}(k+1), \widehat{f}(k+2))$ and select the one that gives the smallest value.

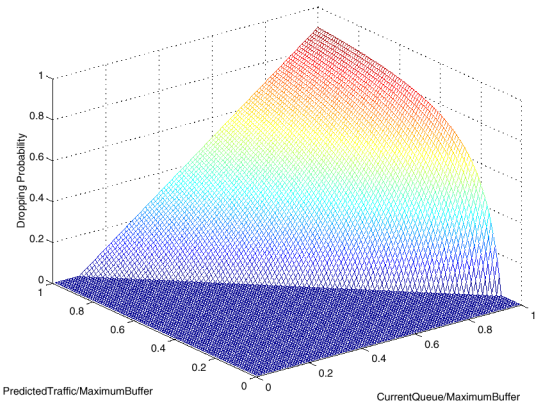
(C3) If $u^*(k+1) \in [0, \widehat{f}(k+1)]$ and $u^*(k+2) \notin [0, \widehat{f}(k+2)]$, then the optimal solution is one of the two possible pairs, $(u^*(k+1), 0)$ and $(u^*(k+1), \widehat{f}(k+2))$. One can compute $J(u^*(k+1), 0)$ and $J(u^*(k+1), \widehat{f}(k+2))$ and determine the optimal controller output accordingly.

(C4) If $u^*(k+1) \notin [0, \widehat{f}(k+1)]$ and $u^*(k+2) \in [0, \widehat{f}(k+2)]$, then the optimal solution is one of the two possible pairs, $(0, u^*(k+2))$ and $(\widehat{f}(k+1), u^*(k+2))$. One can compare $J(0, u^*(k+2))$ and $\mathrm{J}(\widehat{f}(k+1), u^*(k+1))$ and determine the optimal controller output accordingly.

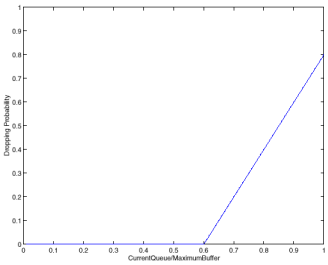
After $u(k+1)$ is determined, we set $p(k+1) = \frac{u(k+1)}{\widehat{f}(k+1)}$ as the packet dropping probability to be used in the next interval. For example, if we use u^* derived in Eq. (10), then

$$p(k+1) = \begin{cases} 0, & Q(k) < C + \overline{Q} - \widehat{f}(k+1), \\ \frac{Q(k)+\widehat{f}(k+1)-C-\overline{Q}}{\widehat{f}(k+1)}, & C + \overline{Q} - \widehat{f}(k+1) < Q(k) \\ & \quad < C + \overline{Q}, \\ 1, & Q(k) > C + \overline{Q}. \end{cases}$$

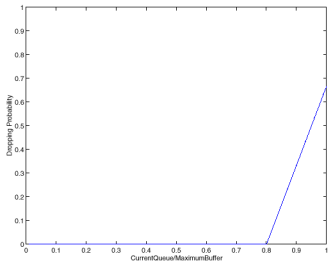
Note that $p(k+1)$ is a linear function of $Q(k)$ when $\widehat{f}(k+1)$ is at a fixed value. Fig. 6 gives the values of $p(k+1)$ as a function of $Q(k)$ and $\widehat{f}(k+1)$, with $Q_{opt} = 0.6 \cdot$ MaximumBufferSize and $R = (0.5 \cdot$ MaximumBufferSize $)/\tau$ (i.e., half of the queue can be emptied in one measurement interval). As shown in Fig. 6 (b)-(c), the projection of $p(k+1)$ onto the plane that corresponds to a fixed value of $\widehat{f}(k+1)$ is a



(a) Dropping probability as a function of current queue size and predicted incoming traffic



(b) $\widehat{f}(k+1) = 0.5 \cdot$ MaxBufSize



(c) $\widehat{f}(k+1) = 0.3 \cdot$ MaxBufSize

Fig. 6. The packet dropping probability, $p(k+1)$, to be used in the next time interval versus the queue length and the estimated traffic. The values of $Q(k)$ and $\widehat{f}(k+1)$ are normalized with respect to the maximum buffer size.

RED-like curve. That is, RED can be viewed as a special case of PAQM with the estimated future incoming traffic fixed at certain value. Another interpretation of Fig. 6 is that a RED queue can be stabilized at a desirable queue length, if it considers the amount of incoming traffic and sets the values of the two thresholds, *min_th* and *max_th*, in compliance with the curves given in Fig. 6.

Fig. 7 gives the packet dropping probability, $p(k+1)$, calculated in an *ns-2* simulation with the same setting as in Fig. 6. Comparing Fig. 7 against Fig. 6, we observe that the packet dropping probability used by a router in the simulation indeed lies on the plane depicted in Fig. 7.

V. Related Work

To alleviate the drawbacks of RED, several AQM schemes have been proposed, e.g., FRED [15], balanced RED (BRED) [2], BLUE [8], stabilized RED (SRED) [17], random exponential marking (REM) [3], PI controller [11], and AVQ [13]. These schemes differ in (i) the performance objectives (in addition to that of notifying end hosts of incipient congestion by dropping/marking packets); (ii) the parameters used as an indicator of congestion; and (iii) the policies used to detect (incipient) congestion and to drop/mark packets. In what follows, we summarize these schemes with respect to the above three aspects.

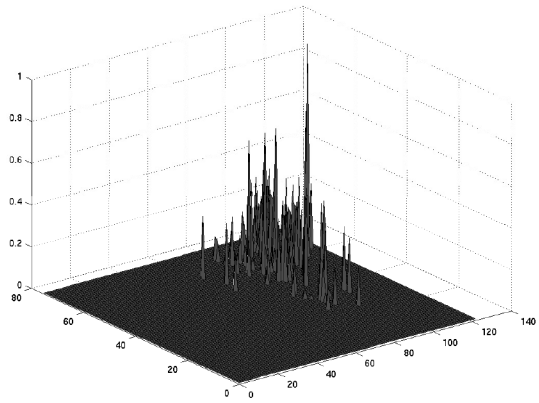


Fig. 7. The packet dropping probability, $p(k+1)$, calculated in an *ns-2* simulation run.

(1) Schemes that aim to achieve per-flow 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 $min_th \leq avg_queue \leq max_th$. 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: $(0, \ell_1)$, (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. In essence, both FRED and BRED aim to improve the fairness of RED at the expense of keeping per-active-flow state information.

(2) Schemes that decouple the congestion index and the performance index. Schemes in this category aim at achieving both high utilization and low packet delay. 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). Conversely, if the link is idle (i.e., the queue is empty), the BLUE gateway decreases p by an amount of *delta* periodically. 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

$$c(k+1) = \max(0, c(k) + \gamma(\alpha(Q(k) - Q_{opt}) + x(k) - R)), \quad (12)$$

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. The price increases if the weighted sum of these mismatches is positive, and decreases 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. Sharing a very similar viewpoint with REM, the PI controller marks each packet with a probability p which is updated periodically using

$$p(k+1) = p(k) + a(Q(k+1) - Q_{opt}) - b(Q(k) - Q_{opt}), \quad (13)$$

where $a > 0$ and $b > 0$ are constants chosen according to the design rules given in [11].

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

$$\frac{d\hat{R}}{dt} = \alpha(\gamma R - x(t)). \quad (14)$$

where γ is the desired utilization. The rationale behind Eq. (14) 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. AVQ is shown to outperform REM and PI in terms of reducing the packet drop rate and average queue length and achieving high utilization.

(3) 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.

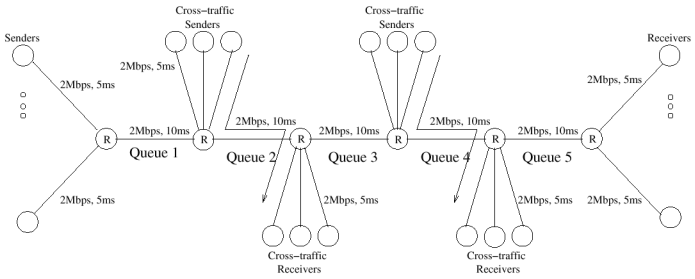


Fig. 8. The multiple bottleneck simulation topology.

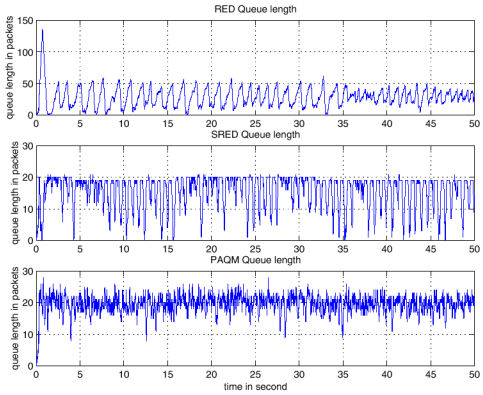
The work that comes closest to ours is SRED, as both share the same objective of stabilizing the instantaneous queue. Hence, we will make comprehensive performance comparisons between SRED and PAQM in Section VI. On the other hand, PAQM also decouples the congestion measure and the performance measure and can be classified into the second category. Hence, we will also compare PAQM against AVQ (which is reported to give the best performance in the second category) in Section VI.

VI. Simulation Results

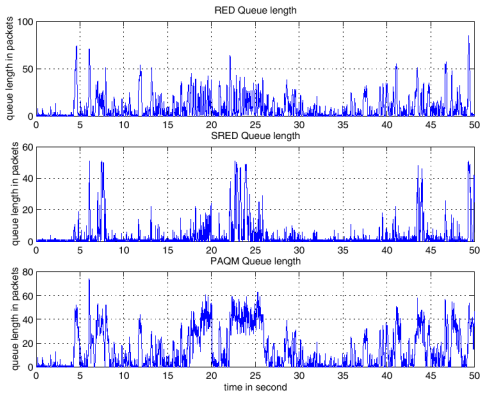
We have implemented PAQM along with RED [9], SRED [17], and AVQ [13] in *ns-2*, and conducted a simulation study to validate the proposed design and compare the performance of PAQM against the other schemes. We examine the behavior of these schemes under a variety of network topologies and traffic sources. In particular, we have considered the network topologies with a single bottleneck link, with multiple bottleneck links (e.g., Fig. 8), as well as of arbitrary topology. The maximum buffer size of each router is set to 100 packets (of size 1000 bytes) under PAQM, RED, and AVQ, and 20 packets under SRED. The reason for choosing a different maximum buffer size under SRED is that through extensive simulation, we find that SRED always attempts to keep the queue close to full. The value is so chosen that the queue length is comparable to those of PAQM and RED. We use an assortment of traffic sources (namely TCP sources that generate packets according to the on-off model or real traffic traces down-loaded from the Internet, and constant bit rate UDP sources). Due to the space limitation, we only report on a small set of the simulations which we believe is the most representative. More simulation results are available in [10].

The parameters used in PAQM are: Q_{opt} is set to 20 packets , and τ is in the range of seconds. The parameters of RED are set as $w_p = 0.002$, $min_th = 5$, $max_th = 15$packets, and those of SRED are chosen as recommended in [17] (i.e., $M = 1000$, $\alpha = 1/M = 0.001$, and $p_{max} = 0.15$). Finally, the desirable utilization, γ, of AVQ is set to 0.98, and the damping factor, α, is determined in compliance with Theorem 1 in [13] to ensure system stability ($\alpha = 0.15$).

0-7803-7476-2/02/$17.00 (c) 2002 IEEE.



(a) on-off model



(b) real traffic trace

Fig. 9. Instantaneous queue length in the single bottleneck network with TCP sources.

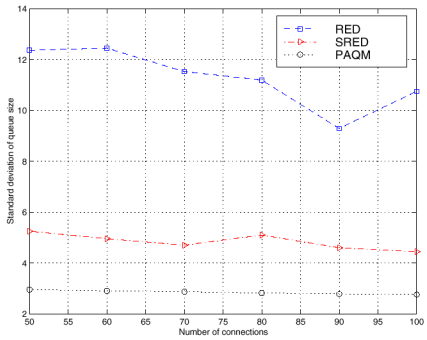
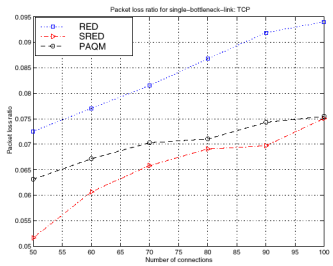
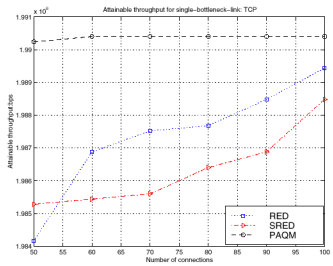


Fig. 10. The standard deviation of the instantaneous queue length in the single bottleneck network.



(a) packet loss ratio



(b) attainable throughput

Fig. 11. The packet loss ratio and attainable throughput in the single bottleneck network.

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.

A. Comparison Between RED, SRED, and PAQM

In this set of experiments we compare PAQM against RED and SRED with respect to instantaneous queue length, packet loss ratio, and total throughput attained by receivers under different network topologies and traffic mixes.

Simulation results under the single bottleneck topology. k TCP connections are established over a single bottleneck link of capacity 20 Mbps. where k varies from 20 to 100. Fig. 9 gives the instantaneous queue length in the cases that 60 TCP connections are established and generate packets using either the on-off model ((a)) or the real traffic traces ((b)). Fig 10 gives the standard deviation of instantaneous queue length with the same setting in Fig. 9 (a) (except that the number of connections varies from 50 to 100). As compared to RED and SRED, PAQM indeed keeps the queue at the desirable level and better stabilizes the queue.

Fig 11 gives the packet loss ratio and the throughput attained by all receivers under the case of TCP sources with the on-off model. PAQM performs better than SRED w. r. t. attainable throughput, but slightly worse than SRED w. r. t. packet loss ratio. This is because SRED always attempts to keep the queue (close to) full, and hence does not proactively drop packets that aggressively. However, keeping the queue full also makes the queue more susceptible to the global synchronization effect. This is evidenced in Fig. 9 (a) that the instantaneous queue length frequently oscillates between empty and full. This also accounts for the fact that SRED does not attain as much throughput as PAQM.

Simulation results under the multiple bottleneck network. We repeat the same experiments in a network with multiple bottlenecks. As shown in Fig. 8, there are 5 queues among which queue 2 and queue 4 are shared with cross traffic of 20 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 20 to 100. The cross traffic is composed of TCP connections as well, and the number of TCP connections in each cross traffic bundle is set to $0.5k$. 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 length of queue 2 , the packet loss ratio of queue 2 and the attainable throughput by all receivers in Fig. 12 and 14, respectively. PAQM outperforms the other two schemes with respect to all three measures. The reason why SRED does not perform as well in terms of packet loss

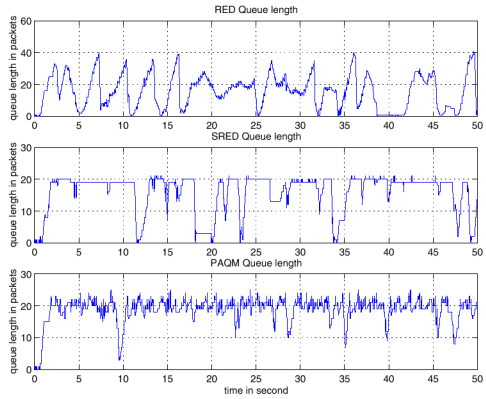


Fig. 12. The instantaneous queue lengths at queue 2 in the multiple bottleneck network.

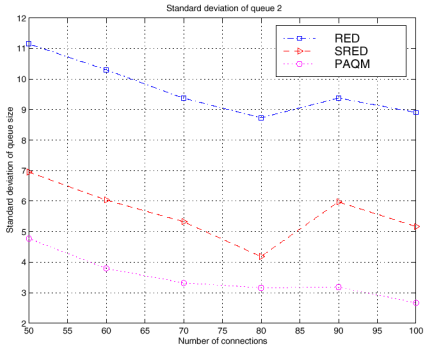
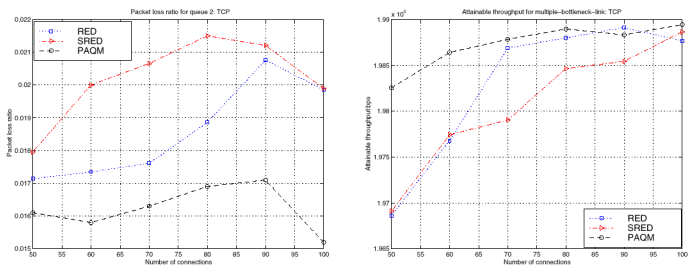


Fig. 13. The standard deviation of the instantaneous queue length at queue 2 in the multiple bottleneck network.

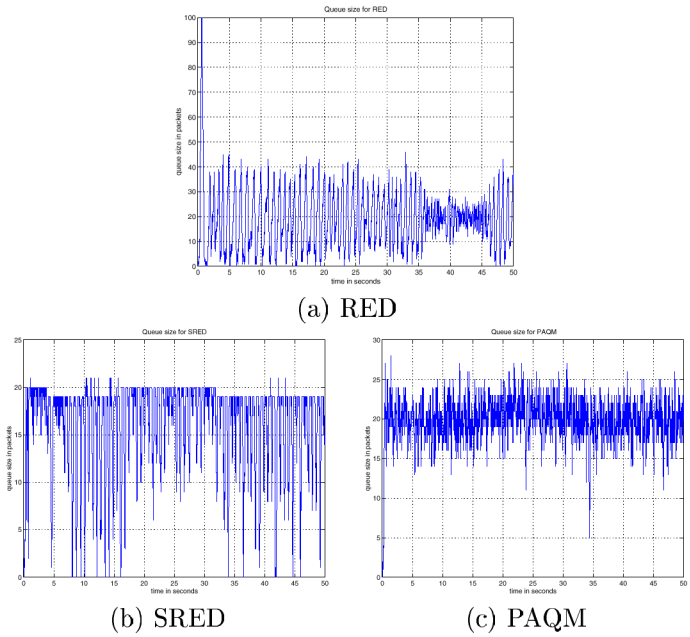
ratio is because as SRED has the tendency to keep the queue (close to) full, it becomes more likely that packet losses occur as a result of buffer overflow, when the link traffic becomes more bursty/congested with cross traffic.

Simulation results in the case of dynamic connection establishment and termination. To test the responsiveness in stabilizing the queue in the case that connections are dynamically established and terminated, we repeat the same experiment in the single bottleneck network except that initially 40 TCP connections commence at time 0, at time 10 another 20 connections are established, and finally at time



(a) packet loss ratio (b) attainable throughput

Fig. 14. The packet loss ratio at queue 2 and attainable throughput in the multiple bottleneck network.



(a) RED

(b) SRED (c) PAQM

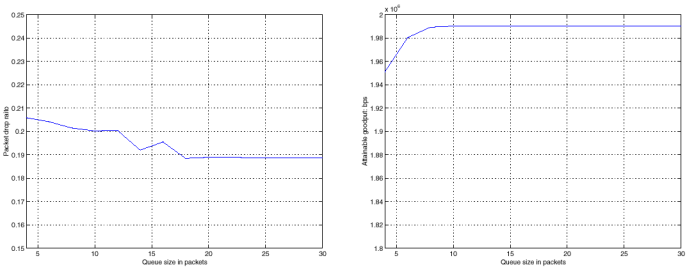
Fig. 15. The instantaneous queue length in the case of dynamic connection establishment and termination.

30, 20 connections are terminated. All the simulation runs last for 50 seconds. Fig. 15 gives the instantaneous queue length under RED, SRED, and PAQM. The queue length under PAQM is always stable around $Q_{opt} = 20$ packets, regardless of change of the number of connections. The queue size under RED, on the other hand, oscillates dramatically almost all the time. SRED again keeps its queue length close to the maximum buffer size, and hence is subject to the global synchronization effect experienced by a drop-tail queue. This is evidenced by the fact that the instantaneous queue length frequently oscillates between empty and full.

B. Comparison between AVQ and PAQM

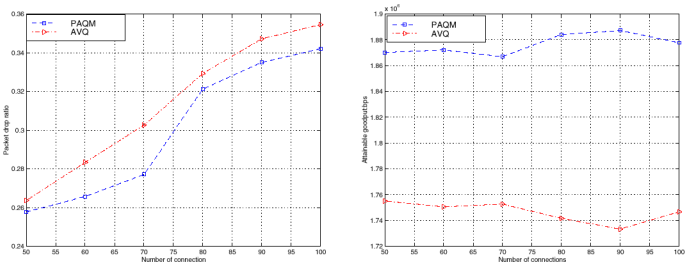
As mentioned in Section V, although PAQM is not targeted to decouple the congestion measure and the performance measure, the fact that it takes into account of the amount of future traffic in the next two measurement intervals does help to achieve this objective. To illustrate this, we repeat the same experiment in the single bottleneck network, but change Q_{opt} from 4 to 30, and measure the packets loss ratio and attainable throughput. Fig. 16 gives the simulation results. As Q_{opt} changes from 4 to 35, the packet loss ratio decrease from 0.206 to 0.19, while the attainable throughput increases from 1.95 Mbps to 1.99 Mbps. This suggests that under PAQM the equilibrium value of the congestion measure is independent of the equilibrium performance measure (e.g., packet loss or attainable throughput).

As PAQM does exhibit the decoupling behavior, we compare it against AVQ — the scheme currently reported in [13] to give the best performance in the second category. Again we repeat the same experiment in the single bottleneck net-



(a) packet loss ratio (b) attainable throughput

Fig. 16. The packet loss ratio and attainable throughput versus Q_{opt}.



(a) packet loss ratio (b) attainable throughput

Fig. 17. Performance comparison between AVQ and PAQM.

work except that we set $Q_{opt} = 2.5$ under PAQM. This is because AVQ usually keeps its average queue length at 2.5 packets under the given simulation setting. By setting $Q_{opt} = 2.5$ packets, a fair comparison can then be made with respect to packet loss ratio and attainable throughput. Fig. 17 gives the simulation results. The average queue length both under AVQ and PAQM is kept around 2.5 packets, but PAQM achieves better performance both in terms of packet loss ratio and attainable throughput. This demonstrates the powerfulness of exploiting LRD and taking into account of future arrival rate in determining the packet dropping probability. We have observed similar trends in simulation runs conducted in the multiple bottleneck network and in the network of arbitrary topology, but due to the space limitation, do not include the results.

VII. Conclusion

We have explored in this paper the issue of exploiting traffic predictability to enhance the performance of AQM. We show that the correlation structure present in long-range dependent traffic can be detected on-line and used to accurately predict the future traffic. We then exploit the prediction results in the calculation of the packet dropping probability. By stabilizing the instantaneous queue length at a desirable level, PAQM enables the link capacity to be fully utilized, while not incurring excessive packet loss ratio. Through *ns-2* simulation, we show that under most cases PAQM outperforms SRED in stabilizing the instantaneous queue length, and AVQ in reducing packet loss ratio and utilizing the link capacity.

It is worth mentioning that PAQM is orthogonal to REM, PI, and AVQ in the sense that the effect of future incoming traffic can be figured in in the calculation of the price function (Eq. (12)) or in the adjustment of the virtual queue capacity (Eq. (14)) to further improve their performance. As PAQM has been shown to be a generalized version of RED with future traffic figured in (Fig. 6), one can expect that REM/PI/AVQ, when coupled with PAQM, also gives a more generalized version expectedly with better performance. This is a subject of future investigation.

References

[1] A. M. Adas. Using Adaptive Linear Prediction to Support Real-Time VBR Video Under RCBR Network Service Model. *IEEE/ACM Transactions on Networking*, Vol.6, No.5, October 1998.

[2] 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.

[3] S. Athuraliya, S. H. Low, V. H. Li, and Q. Yin. REM: Active Queue Management. *IEEE Network Magazine*, Vol. 15, No. 3, pp. 48–53, May/June 2001.

[4] G. E. P. Box and G. M. Jenkins. Time Series Analysis: Forecasting and Control. Holden Day, San Francisco, 1970.

[5] B. Braden *et al.* Recommendations on Queue Management and Congestion Avoidance in the Internet. RFC-2309, IETF, April 1998.

[6] A. Erramilli, O. Narayan, and W. Willinger. Experimental Queuing Analysis with Long-Range Dependent Traffic. *IEEE/ACM Transactions on Networking*, April 1996.

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

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

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

[10] Y. Gao, G. He, and J. Hou. On Leveraging Traffic Predictability in Active Queue Management. Extended version. Available at http://eewww.eng.ohio-state.edu/∼heg.

[11] C. Hollot, V. Misra, D. Towsley, and W. Gong. On Designing Improved Controllers for AQM Routers Supporting TCP Flows. *Proc. IEEE INFOCOM 2001*, April 2000.

[12] J. R. M. Hosking. Fractional Differencing. Biometrika, 68, pp. 165-176, 1981.

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

[14] W. E. Leland, M. S. Taqqu, W. Willinger and D. V. Wilson. On the Self-Similar Nature of Ethernet Traffic (Extended Version). *IEEE/ACM Transactions on Networking*, February 1994.

[15] W. Lin, and R. Morris. Dynamics of Random Early Detection. *Proceedings of ACM SIGCOMM'97*, September 1997.

[16] I. Norros. On the Use of Fractional Brownian Motion in the Theory of Connectionless Networks. *IEEE Journal on selected areas in communications*, Vol. 13, No. 6, August 1995.

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

[18] A. Sang, and S.-q. Li. A Predictability Analysis of Network Traffic. *Proceedings of IEEE INFOCOM'2000*, Tel-Aviv, Israel, March 2000.

[19] W. Willinger, M. S. Taqqu, R. Sherman, and D. V. Wilson. Self-Similarity Through High-Variability: Statistical Analysis of Ethernet LAN Traffic at the Source Level. In *Proc. ACM SIGCOMM'91*, pp. 149–157, 1991.

[20] W. Willinger, V. Paxson, and M. S. Taqqu. Self-Similarity and Heavy Tails: Structural Modeling of Network Traffic. In R. Adler, R. Feldman, and M. S. Taqqu, editors, "A Practical Guide to Heavy Tails: Statistical Techniques and Applications," Birkhauser, Boston, 1998.

0-7803-7476-2/02/$17.00 (c) 2002 IEEE.