Iranian Journal of Science & Technology, Transaction B, Engineering, Vol. 31, No. B6, pp 663-678
Printed in The Islamic Republic of Iran, 2007
© Shiraz University

ARM-PFC, AN OPTIMIZED AQM CONGESTION CONTROLLER IN TCP/IP NETWORKS[*]

N. BIGDELI[1**] AND M. HAERI[2]

[1] Dept of Electrical Engineering, Faculty of Engineering and Technology, Imam Khomeini International University, Qazvin, I. R. of Iran, Email: bigdeli@ikiu.ac.ir
[2]Advance Control System Lab., Dept. of Electrical Engineering, Sharif University of Technology, Tehran, I. R. of Iran

Abstract– In this paper, a new effective and computationally reduced method for congestion control in high speed dynamic computer networks is introduced. The controller is designed using the well-known predictive functional control (PFC) scheme and an ARMarkov model representation that considers the system delay explicitly. Use of the multi-step-ahead predictive ARMarkov model structure within the PFC results in a simple algebraic control law that does not require recursive model output computation in the so-called prediction horizon performed in the other Model Predictive Controllers (MPC). This combination not only reduces the required computational load, but the accumulative error due to the model uncertainties decrease considerably. Packet-level simulations based on *ns*-2 are provided to show good performance of ARM-PFC in a large variety of topology and traffic mixtures for both queue regulation and resource utilization. Fast response, low queue fluctuations (and consequently low delay and jitter), high link utilization, good disturbance rejection, scalability, and low packet marking probability are other features of the proposed method with respect to the well-known AQM methods such as RED, PI, and REM, which are also simulated for comparison.

Keywords– Active queue management, ARMarkov model, computer networks, congestion control, predictive functional control

1. INTRODUCTION

The current internet is faced with an aggressive number of users that require high quality real time services such as video-conferencing, gaming and multimedia services, a large amount of data transferring, electronic business, and so on. Therefore, the available resources such as the core routers' outlet bandwidth should be allocated between the competing flows through some resource allocation and congestion management algorithms. These algorithms are designed in order to: 1) prevent the congestion occurrence or handle it if it occurs, 2) make proper use of the resources or equivalently achieve high throughput.

As formulated in [1], congestion control is a distributed control problem that must be solved by the cooperation of two groups of network components; the end users (especially data senders), and routers, where the congestion occurs. The algorithms for source data flow control have been implemented via different variants of TCP congestion control methods being TCP Reno, Tahoe, New Reno, Sack, Vegas, Westwood, etc. In these algorithms, the data sending rate is controlled via increasing/decreasing a congestion window size. The window size is determined based on the implicit/explicit feedback from the congestion status in the corresponding paths. In routers, on the other hand, congestion control and signaling it back to the senders has been done by link algorithms. The developed algorithms drop some of the excessive input packets based on the router buffer occupancy or the rate of entering data to the router.

[*]Received by the editors January 7, 2007; final revised form October 7, 2007.
[**]Corresponding author

The former methods are called queue-based or active queue management (AQM) [2] methods and the latter, rate-based approaches [3-4]. Guaranteed delay and its variance, jitter, as well as low packet loss and high TCP throughput are the main objectives of any congestion management algorithm, especially AQM methods and computational load and complexity, one of the most important issues that may restrict the applicability of a congestion management method in a high speed router. Scalability and the ability to cope with network dynamics are the other issues that must be considered in developing a new congestion control method.

Up to now, many different AQM schemes have been introduced in the literature. They can be categorized in various groups such as heuristic methods that are mainly developed by computer scientists; mathematical schemes such as game-theory based algorithms; and control theory-based approaches. The last family is those that employ control-theory methods for system analysis and AQM design. Some examples are the control-theory based models for TCP/AQM networks reported in [1], [5] and [6], and different AQM-controllers such as P, PI, PD, the sliding mode and coefficient diagram method (CDM) ([3] and [7]- [11]) designed based on these models. These controllers, however, are categorized as conventional control methods having some advantages and drawbacks. Their main drawback is that they do not deal with the inherent internet delay and do not contain any prediction capability. These features are important since they enable the controller to adjust the control command based on the desired output behavior in the future time after the delay and so result in better transient and steady-state responses for non-minimum phase systems such as TCP/AQM networks.

Model Predictive Controllers (MPC), as modern control strategies, possess such features as good tracking and disturbance rejection capabilities, while adequately dealing with the system delay. The main bottleneck of applying an MPC method such as DMC or GPC to fast systems like TCP/AQM networks is the large amount of calculation flops that are needed for their matrix operations.

Predictive Functional Control (PFC), as a predictive controller, was first proposed in [12-13] to control dynamically fast systems. It uses a transfer function internal model formulation and the same concepts of prediction and control horizons as other MPC variants. The difference is that in PFC, instead of minimizing a cost function, the control command is derived by making a few future output points equal to their corresponding points in the desired trajectory. With this idea, in PFC, instead of implementing an optimization problem, just a few algebraic equations are solved and this is why the computational complexity is reduced considerably [14].

The ARMarkov model structure, first introduced in [15], combines both parametric and non-parametric representations in the modeling of a system. The model is defined by μ first Markov parameters of the system and some other coefficients, of which all of them can be estimated directly using parameter estimation algorithms [15-17]. The main feature of ARMarkov model representation is that it does not rely on the exact knowledge of the delay and order of the system [18]. This property of ARMarkov representation, in addition to its predictive form, makes it a convenient choice as a predictive model in MPC.

Here, we use the idea of PFC in conjunction with ARMarkov model representation to derive a computationally efficient method for the congestion control problem. To do this, at first the congestion control system is reformulated as an ARMarkov model representation. We modify the current form of the ARMarkov model structure by considering the system delay and calculate the corresponding ARMarkov coefficients for the system. Having the μ-step-ahead predictive model, the need for recursively running the model to get output prediction is eliminated, and therefore, not only the computational load is reduced considerably in comparison with the commonly used ARMAX/ARIMAX structures, but also the accumulative error due to the system uncertainties is also omitted. Afterwards the calculated model is used to formulate PFC as an AQM method for the control of congestion. We call the resulting controller ARM-

PFC. Finally, the performance of ARM-PFC is investigated through simulations. We show, using a wide range of *ns*-2 [19] simulations, that having a rough estimate of the network parameters, ARM-PFC can handle congestion very well in the case of different network dynamics (variations in network topology and traffic consisting of bottleneck link capacity, links transport delays, number of responsive flows and the existence and nature of unresponsive flows such as http and both short-range and long-range dependent UDP traffic). We also investigate the scalability of ARM-PFC as an AQM method through multiple bottleneck network scenarios as well. Besides, three well-known AQM methods, RED, PI, and REM are also simulated for comparison.

The remainder of the paper is organized as follows. In Section 2, the dynamic fluid-flow model of TCP/AQM in the congestion avoidance phase is presented. Section 3 deals with ARM-PFC AQM controller formulation, simulation results are brought in Section 4 and the paper is concluded in Section 5.

2. NONLINEAR TCP/AQM DYNAMICS

We begin our discussion of AQM by first introducing a dynamic model for TCP's congestion-avoidance mode. In recent years several mathematical models have been developed by researchers [1-3, 20] and a variety of control theory-based AQM schemes have been proposed based on these models. In [1], a dynamic model of TCP behavior has been developed using fluid-flow and stochastic differential equation analysis and then modified in [20] to consider the effect of unresponsive flows as well. This model relates the average value of key network variables and is described by two coupled, nonlinear differential equations. Simulation results have demonstrated that this model accurately captures the dynamics of TCP. In this paper, we use a simplified version of this model which ignores the TCP timeout mechanism and is linearized (as in [4]) to design the controller. In the following subsections the developed model is briefly introduced.

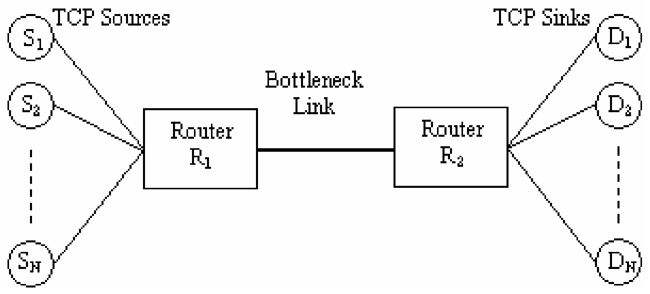

Fig. 1. Dumbbell network topology

a) Fluid-flow model of TCP behavior

Consider a network of N homogeneous long-lived TCP flows sharing a bottleneck link of capacity C (Mbps) as in Fig. 1. Further, let there be some other flows sharing the link that are unresponsive to congestion notification signals. Figure 2 shows the interaction of different parts of such a network, where the long-lived flows, l, are under direct AQM control, while the unresponsive flows are not. The additive increase, multiplicative decrease (AIMD) congestion window adjustment law in TCP can then be modeled as:

$$\frac{dW_l}{dt} = \frac{1}{R(q(t))} - \frac{W_l(t)}{2}\frac{W_l(t-R_0)}{R(q(t-R_0))}p_d(t-R_0) \tag{1}$$

Where, W_l is the congestion window size in packets, $R = q/C + T_p$ is the round trip time, and q, p_d, C and T_p denote the queue length (packets), the packet drop probability, the link capacity (packet/sec) and propagation delay (sec), respectively. The queue length dynamic in router R_1 can be described as:

$$\frac{dq}{dt} = -(C - u(t)) + \frac{N}{R(q(t))} W_l(t) \tag{2}$$

$u(t)$ represents the effect of disturbances due to the unresponsive flows. If we are only interested in the average behavior of the queue on a time scale which is coarser than time scales of $u(t)$'s variations, we can replace $u(t)$ in Eq. (2) by its mean u_0. Then

$$\frac{d\overline{q}}{dt} = -(C - u_0) + \frac{N}{R(\overline{q}(t))} \overline{W_l}(t) \tag{3}$$

In which the effective link bandwidth is diminished from C to $C_{eff} \overset{\Delta}{=} C - u_0$.

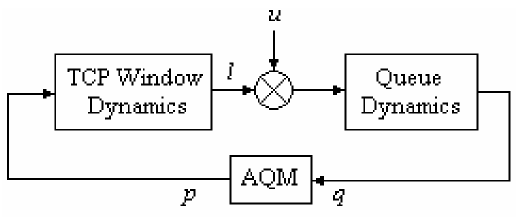

Fig. 2. Block diagram of TCP/AQM network interactions including responsive and unresponsive flows

b) Linearization

The equilibrium point (W_{l0}, p_{d0}, q_0, u_0) for Eqs. (1) and (3) is determined as: $W_{l0}^2 p_{d0} = 2$, $W_{l0} = R_0 C_{eff} / N$ and $R_0 = q_0 / C + T_p$. Linearizing the model on this equilibrium results in:

$$\delta W_l(s) = -P_{win}(s) e^{-sR_0} \delta p_d(s), \quad \delta l(s) = \frac{N}{R_0} \delta W_l(s), \; \delta q(s) = P_{que}(s)(\delta l(s) + \delta u(s)) \tag{4}$$

Where δW_l, δq, δl, and δp_d are the corresponding perturbations. The window and queue transfer functions are:

$$P_{win}(s) = \frac{C_{eff}^2 R_0 / 2N^2}{s + 2N / R_0^2 C_{eff}}; \quad P_{que} = \frac{1}{s + C_{eff} / C R_0} \tag{5}$$

Therefore, the small signal transfer function of the system can be represented by:

$$P_{ol}(s) = \frac{\delta q(s)}{\delta p_d(s)} = \frac{N}{R_0} P_{win}(s) P_{que}(s) \tag{6}$$

3. ARM-PFC AS AN AQM METHOD

In this section, the main idea and the design procedure of ARM-PFC as an AQM controller is investigated. The PFC technique was first developed by Richalet and his colleague in ADERSA ([12-14]) as an MPC variant to work in fast dynamic systems. Similar to other MPC variants, PFC operates based on four principals: a *prediction model* is used to predict the future system outputs; a *desired trajectory,* which is an exponential curve from the current output value to its desired one, is calculated to be followed by the closed loop system; *auto-compensation* of probable mismatches between the model and the real process is done by introducing a disturbance term equal to the process and model output difference in each sampling time; and finally the *control law* is computed from equating the estimated outputs (prediction model output plus the considered disturbance component) and the reference trajectory in a few future points called *coincident points* [21]. With this idea, in PFC, instead of solving an optimization problem, just a few algebraic equations are solved and that is why the computational complexity is reduced considerably. In this paper, ARMarkov representation is used to implement the predictive model and so the matrix calculations generally associated with ARX or ARIMAX model structures are also omitted and this further reduces the calculation complexity. In the following subsections, we develop the ARM-PFC formulation as an AQM method.

a) Predictive model and ARMarkov model representation

The ARMarkov representation for a discrete-time finite-dimensional linear time-invariant SISO system was first introduced in [15]. The ARMarkov model structure is an FIR predictive structure that contains the Markov parameters of the system explicitly. It has been shown in [15-17] that the least-square estimation of the model parameters leads to a statistically consistent estimation which is less sensitive to disturbances in the system and does not need to determine the exact relative degree such as the case, for example, in an ARMAX structure. Consider the discrete-time finite dimensional linear time-invariant SISO system of transfer function $G(z)$ and state-space representation of:

$$\begin{aligned} x(k+1) &= A_x x(k) + B_x u_x(k) \\ y(k) &= C_x x(k) + D_x u_x(k) \end{aligned} \tag{7}$$

Where, $A_x \in R^{n\times n}$, $B_x \in R^{n\times 1}$, $C_x \in R^{1\times n}$, $D_x \in R^{1\times 1}$. Then, the Markov parameters h_j are defined by [22]:

$$h_j \underset{=}{\Delta} \begin{cases} D_x & for\ \ j=0 \\ C_x A_x^{j-1} B_x & for\ j \ge 1 \end{cases} \tag{8}$$

and satisfy $G(z) = \sum_{j=0}^{\infty} h_j z^{-j}$. The n^{th} order ARMarkov time-domain representation for a strictly proper system is given by:

$$y(k) = -\sum_{j=1}^{n} \alpha'_j y(k-\mu-j+1) + \sum_{j=1}^{\mu} h_j u(k-j) + \sum_{j=1}^{n} \beta'_j u(k-\mu-j) \tag{9}$$

The model involves the first μ Markov parameters $h_1,\cdots,h_\mu$. The parameters $\alpha'_1,\cdots,\alpha'_n$ and $\beta'_1,\cdots,\beta'_n$ represent the weights of the previous inputs and outputs effect on the present output. From (9), the μ-step ahead predictor of the system is defined by:

$$y(k+\mu) = -\sum_{j=1}^{n} \alpha'_j y(k-j+1) + \sum_{j=1}^{\mu} h_j u(k+\mu-j) + \sum_{j=1}^{n} \beta'_j u(k-j) \tag{10}$$

In different MPC variants, however, the future inputs are derived by minimizing the differences between the desired and the predicted outputs assuming some requirements on the control command. Therefore, having the corresponding coefficients estimated directly, the predictor in (10) seems to be a convenient form for MPC's. To continue, we formulate the TCP/AQM system using an ARMarkov model representation in which the effect of the systems delay and a special form of PFC control command are considered. Then an explicit statement for the required prediction model parameters is derived.

The block diagram of PFC as an AQM method is depicted in Fig. 3. In order to design the controller, the plant dynamic in (6) is considered as our model. Neglecting the effect of unresponsive flows it can be represented as:

$$P(s) = \frac{N}{R_0} P_{win}(s) P_{que}(s) e^{-sR_0} = \frac{k_t e^{-sR_0}}{(\tau_1 s+1)(\tau_2 s+1)} \tag{11}$$

Where, $k_t = R_0^3 C^3 / 4N^2$, $\tau_1 = R_0$, and $\tau_2 = R_0^2 C / 2N$

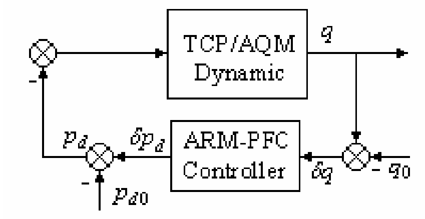

Fig. 3. Block diagram of ARM-PFC as an AQM method

By discretizing (11) with a sampling period of T_s and zero-order-hold transformation method, the equivalent discrete model is of the form [23]:

$$P(z) = \frac{B(z)}{A(z)} = z^{-d} \frac{m_1 z^{-1} + m_2 z^{-2}}{1 + n_1 z^{-1} + n_2 z^{-2}} \tag{12}$$

Where delay d is the integer part of R_0/T_s . Coefficients n_1, n_2, m_1, and m_2 depend on the chosen sampling period T_s, as well as the system parameters, they are:

$$\begin{cases} n_1 = -(\alpha_1 + \alpha_2) \\ n_2 = \alpha_1 \alpha_2 \\ m_1 = k_1(1-\alpha_1) + k_2(1-\alpha_2) \\ m_2 = -[k_1(1-\alpha_1)\alpha_2 + k_2(1-\alpha_2)\alpha_1] \end{cases} \tag{13}$$

With, $\alpha_i = \exp(-T_s/\tau_i)$ for $i=1$ and 2, $k_1 = \tau_1 k_t/(\tau_1 - \tau_2)$ and $k_2 = \tau_2 k_t/(\tau_2 - \tau_1)$.

Now, we try to represent the h – step-ahead predictor of the model in (12) in terms of its current and past outputs (maximally up to degree of system) and the past inputs using first h Markov parameters. We also extend such formulation to include the effect of the transport delay where without loss of generality we consider it as an integer multiple of the sampling time. To do this, first consider the model in (13). The model output, $y_m(k)$, in each sampling time is given by:

$$y_m(k) = y_1(k) + y_2(k) \tag{14}$$

Where, y_1 and y_2 are outputs of the corresponding first-order models in (12). The discrete equivalent of the first-order models are given by:

$$y_i(k) = \alpha_i y_i(k-1) + k_i(1-\alpha_i)u(k-d-1) \tag{15}$$

For $i = 1, 2$ and $\alpha_i = e^{-T_s/\tau_i}$ and the delay d is equal to R_0/T_s . We choose the sampling time as follows [22]:

$$T_s = \gamma\sqrt{\tau_1^2 + \tau_2^2}, \qquad \gamma \in [0.1, 0.25] \tag{16}$$

With this choice, the sampling frequency is 4-10 times that of the equivalent cut-off frequency of the system. By proper choice of γ, one can find T_s such that d becomes an integer number.

One of the differences between PFC and other MPC variants is the type of parameterization used for representing the control signals. Indeed, instead of using the actual control inputs, in PFC, they are considered composed of a priori known functions such as steps, ramps, trapezoids or even more complex functions. The considered components are chosen in such a way that corresponds to the expected features of reference trajectory. For example when the reference input variations are stepwise, the input is parameterized just in terms of its current variation values. If the reference input variations are expected in ramp form, the input is considered as: $\Delta \underset{\rightarrow}{u}^T = [\Delta u_k \quad 0 \quad 0 \quad \cdots] + \beta[1 \quad 1 \quad 1 \quad \cdots]$, where, $\Delta \underset{\rightarrow}{u}$ is the vector of h future command variations and β is the ramp variation rate.

Based on the above discussions, assuming d to be integer and replacing (15) in (14), the h-step-ahead predictor (assuming $u(k+j) = u(k)$ for $j = 1, \cdots, h$ and $h \geq d+1$) is given by:

$$y_m(k+h) = y_1(k+h) + y_2(k+h) = \sum_{i=1}^{2} \alpha_i^{h-d} y_i(k+d) + \sum_{i=1}^{2} k_i(1-\alpha_i^{h-d})u(k) \tag{17}$$

The last right term in (17) includes effects of the $h-d+1$ Markov parameters of the model that has been reduced to this form due to the special form of the command signal in PFC. $y_i(k+d)$ is determined from (15).

$$y_i(k+d) = \alpha_i^d y_i(k) + k_i(1-\alpha_i)u(k) + [\alpha_i^{d-1}u(k-d) + \alpha_i^{d-2}u(k-d+1) + \ldots + u(k-1)] \tag{18}$$

Therefore

$$y_i(k+h) = \alpha_i^h y_i(k) + k_i(1-\alpha_i^{h-d})u(k) + k_i(1-\alpha_i)\sum_{j=1}^{d} \alpha_i^{h-d-1+j} u(k-j) \tag{19}$$

Note that the h-step-ahead predictor in (17) should be implemented only based on the output measurements up to sample time k^{th}. To do this, $y_i(k)$ in (19) is substituted by $y(k)$ using (14) and (15), assuming that $y_m(k-1) = y(k-1)$ and $y_m(k) = y(k)$. This results in;

$$\begin{aligned} y_1(k) &= \frac{\alpha_1}{\alpha_1 - \alpha_2}\left[y(k) - \alpha_2 y(k-1) - [k_1(1-\alpha_1) + k_2(1-\alpha_2)]u(k-d-1)\right] \\ y_2(k) &= \frac{\alpha_2}{\alpha_2 - \alpha_1}\left[y(k) - \alpha_1 y(k-1) - [k_1(1-\alpha_1) + k_2(1-\alpha_2)]u(k-d-1)\right] \end{aligned} \tag{20}$$

By substituting (20) into (17), the final form of the desired ARMarkov representation is derived.

$$\begin{aligned} y(k+h) &= \sum_{j=1}^{2} \beta_j y(k-j+1) + \sum_{i=1}^{2} k_i(1-\alpha_i^{h-d})u(k) + \sum_{i=1}^{2} k_i(1-\alpha_i)\sum_{j=1}^{d} \alpha_i^{h-d-1+j}u(k-j) + \eta u(k-d-1) \\ &= y_{past} + \textstyle\sum_{i=1}^{2} k_i(1-\alpha_i^{h-d})u(k) \end{aligned} \tag{21}$$

y_{past} is the free response part of the predictive model output and is calculated from past $d+1$ inputs and 2 output terms. The coefficients β_j and η are determined from (19) and (20).

b) Reference trajectory and auto-compensation

PFC is designed to make the closed loop system response track a desired reference trajectory which is an exponential curve from the current output value to its desired one [21]. Assume r is the actual set point, then the closed-loop desired trajectory, *w*, is defined by:

$$w(k+i\,|\,k) = r(k) - (r(k) - y(k))\Psi^i \qquad i > 0 \tag{22}$$

$y(k)$ is the current measured output and Ψ is a constant given by:

$$\Psi = e^{-T_s/T_R} \tag{23}$$

with T_R being the desired closed-loop system response time constant.

In an AQM case one encounters a regulatory problem with a set point of q_0. Therefore, the desired trajectory always starts from $y(k) = \delta q(k) = q(k) - q_0$ and ends up at $r \equiv 0$. In PFC, to derive the control command, the estimated output $\hat{y}$ is set equal to the corresponding point in the desired trajectory for a few (usually one or two [21]) future points called *coincident points*, that is:

$$\hat{y}(k+h) = w(k+h) \quad h = l_1, l_2, \cdots \tag{24}$$

The process output estimate $\hat{y}$ at the coincidence horizon h is given by:

$$\hat{y}(k+h) = y_m(k+h) + (y(k) - y_m(k)) \tag{25}$$

The first term in the right hand side, $y_m(k+h)$, stands for the model prediction and the second term represents the disturbance, which is the difference between the process and model outputs (auto-compensation property). $y_m(k+h)$ and $y_m(k)$ are derived from the prediction model described in Subsection *A*. Note, however that to calculate $y_m(k)$, measurements up to (k-1)th are employed, while to estimate $y_m(k+h)$, measurements up to kth are assumed available.

c) Determination of the control law

The control law can be derived by replacing (21) and (22) into (25). For one coincident point at $h > d$, we have;

$$y(k)\Psi^h = y_{past} + \sum_{i=1}^{2} k_i (1 - \alpha_i^{h-d}) u(k) + y(k) - y_m(k) \tag{26}$$

and therefore,

$$u(k) = \frac{y(k)(\Psi^h - 1) - y_{past} + y_m(k)}{\sum_{i=1}^{2} k_i (1 - \alpha_i^{h-d})} \tag{27}$$

In order to implement this controller adaptively, it is best to implement it in an indirect method. For this purpose, the coefficients of the prediction model in Eq. (21) can be easily estimated through common algorithms such as the recursive least square method.

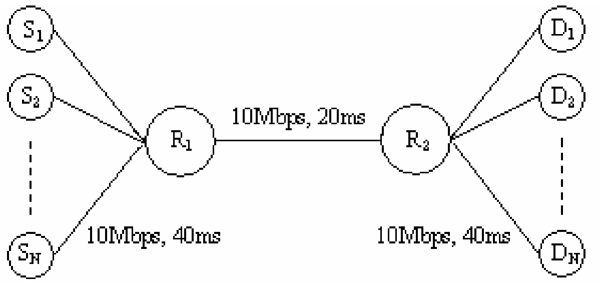

Fig. 4. Dumbbell network topology for *ns*-2 simulations

4. SIMULATION RESULTS

The performance of ARM-PFC has been evaluated in this section through a variety of simulation scenarios in ns-2. The simulation scenarios have been considered the same as the ones employed in [24], plus some additional scenarios. For comparison, RED, REM and PI have been simulated as well. The simulations have been arranged mainly for a single bottleneck dumbbell topology, but multiple bottleneck networks have been simulated as well to show scalability properties of the developed ARM-PFC AQM scheme.

a) Simulation configuration

The dynamic behavior of the designed ARM-PFC AQM controller is simulated under a variety of network topologies and traffic sources through a packet-level *ns*-2 simulator [19]. As the first structure, consider the dumbbell network topology depicted in Fig. 4, where N_p long-lived TCP connections share a single bottleneck link. We assume that the TCP sources always have data to send (long-lived FTP connections). The links between the TCP sources and the router R_1 are 10Mbps links with a mean of 40ms propagation delay, which are the same as those between the TCP sinks and the router R_2. We have considered the propagation delay of different lines slightly different to prevent the occurrence of the synchronization effect. Router R_1 is connected to R_2 through a 10Mbps, 20ms delay link. The maximum buffer size of each router is set to 300 packets (of size 1000 bytes). Meanwhile, we consider the network topology with multiple bottleneck links (Fig. 5), where the maximum buffer size of each router is 200 packets. The bandwidth and the propagation delay of each link are indicated in Fig. 5 and each sender-receiver pair has n_{cross} TCP connections as cross traffic. In both scenarios, TCP Reno is used as the transport agent [24].

The network parameters used in designing the ARM-PFC controller are: $N_p = 100$, $C_p = 1250$, $q_0 = 100$, and $R_0 = 280$ ms, and the controller parameters then become: $d = 5$, $h = d + 2$ and $\Psi = 0.83$ corresponding to a closed loop time constant equal to $4R_0$. This time constant is equivalent to one RTT more than the time required for the drop detection mechanism of TCP through three duplicate acknowledges. For the PI controller we determine the parameters from the procedure described in [8], that is: $a = 6.48\text{e-}007$, and $b = 6.42\text{e-}007$ with a sampling frequency of 150Hz. The parameters of RED are set as recommended in http://www.aciriorg/floyd/RED parameters.txt. For REM, the parameters are set as the *ns*-2 default mentioned in [2]. That is, the parameters are: $\alpha = 0.1$, $\gamma = 0.001$, and $\phi = 1.001$. Besides, in all the following simulations, ECN is enabled for all the simulated AQM strategies.

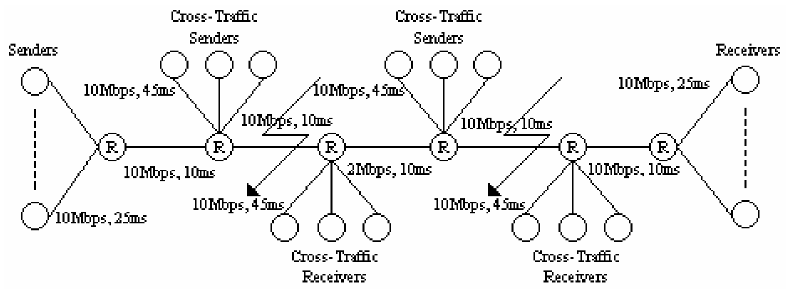

Fig. 5. Network topology with multiple bottlenecks

b) Scenarios of single bottleneck topology

Exp. 1: Comparison of different AQM schemes for unknown network parameters: In this experiment, we choose $N_p = 120$, $C_p = 8$ Mbps and $q_0 = 100$ in Fig. 4, which corresponds to 120 greedy FTP flows sharing the bottleneck link. The system responses for different AQM schemes are depicted in Fig. 6. From these figures, the out-performance of ARM-PFC with respect to others in both the speed of response and the stability of regulated queue size is obvious. Link utilization of the AQM schemes has been calculated, too. There are 0.9957, 0.9754, 0.9955, and 0.9719 of the total link capacity for ARM-PFC, RED, PI and REM, respectively. As seen, ARM-PFC has the best link utilization as well.

We have also tested larger values of h. The test results in deviation from the set-point and larger queue variances (or equivalently larger jitter). Therefore, we continue our simulations with the previous value of h, i.e. $h = d + 2$ in the remainder of the simulations.

Exp. 2: Performance under dynamic traffic changes: In this scenario, we provide some time-varying dynamics and investigate the performance of PFC and other simulated schemes. There are 150 TCP connections at time $t = 0$. At time $t = 40$, 50 of the TCP connections stop transmitting data, and resume again each at time $t = 70$. The queue length and marking probability evolutions are depicted in Fig. 7. Note that while PI and REM performance degrades noticeably under such dynamics, ARM-PFC is very robust against such variations. Besides, although RED does not appear very sensitive in this scenario, its link utilization is degraded under the imposed dynamic. The link utilizations for ARM-PFC, RED, PI and REM are 0.9964, 0.9156, 0.9953, and 0.9694, respectively.

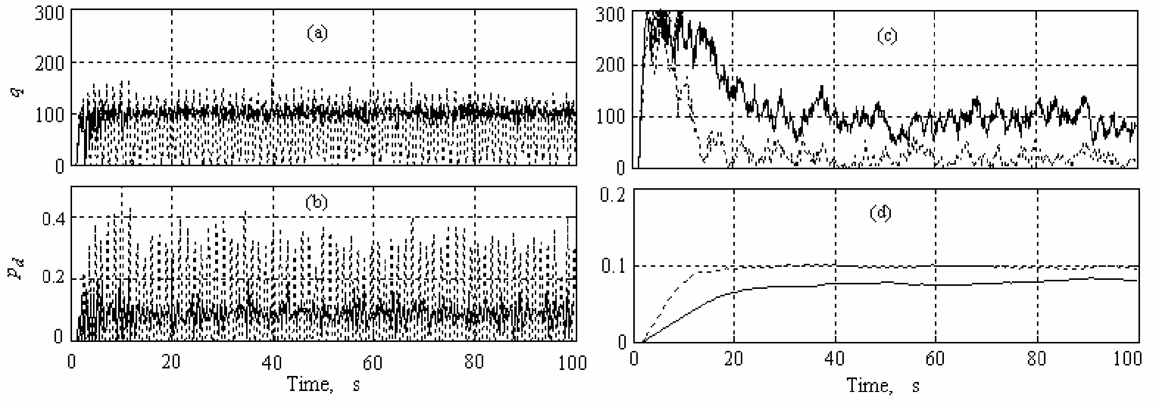

Fig. 6. Queue length and marking probability evolution for Exp. 1, (a) and (b): ARM-PFC (line), RED (dot); (c) and((d): PI (line), REM (dot)

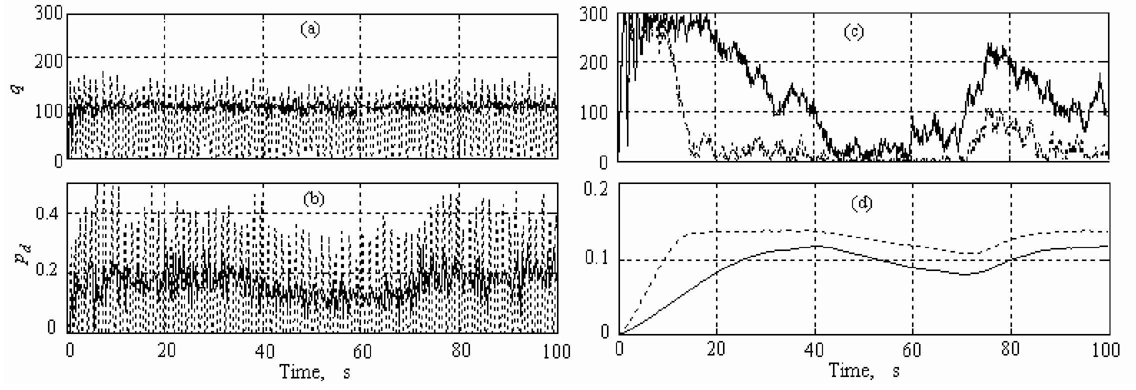

Fig. 7. Queue length and marking probability evolution for Exp. 2, (a) and (b): ARM-PFC (line), RED (dot); (c) and((d): PI (line), REM (dot)

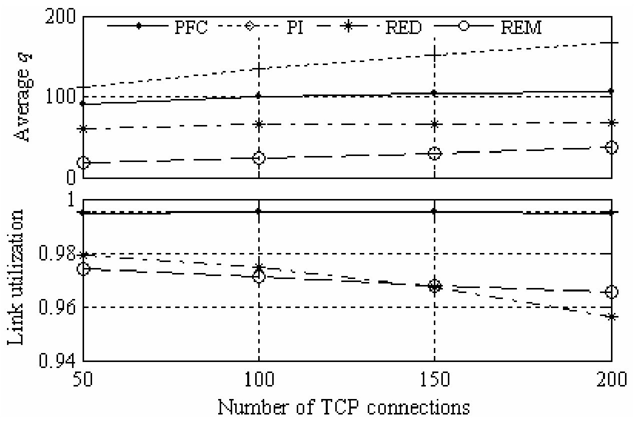

Fig. 8. Average queue and link utilization w.r.t. number of TCP flows (Exp. 3)

Exp. 3: Robustness w.r.t. number of TCP connections: In this experiment, the performance and robustness of ARM-PFC AQM is explored with respect to different TCP loads. The simulations have been done with the same settings as Exp. 1, except that the number of connections varies from 50 to 200. Average queue lengths from 10 to 100s. The corresponding link utilizations have been plotted in Fig. 8 for different AQM schemes. From these graphs, it is observed that PFC can robustly stabilize the queue length around 100 packets, while for PI it blows up when the number of the flows increases. However, RED and REM average queue length show lower sensitivity w.r.t. number of connections. From Fig. 8 it is also obvious that while PI and ARM-PFC keep the link utilization high in spite of load variations, REM and RED link utilization are low and somehow sensitive to load variations. This analysis shows that the control-theory-based AQM schemes, say PI and PFC AQM, have better link utilization by stabilizing the queue length to the desired values.

Exp. 4: Comparison with PI AQM under higher TCP loads: In [8], it has been shown that the PI controller has robustness with respect to the number of connections, although a larger number of connections results in a slower response. In this experiment, we set the number of flows $N_p = 300$ and compare the system response for ARM-PFC and PI schemes. As shown in Fig. 9, the ARM-PFC controller has a very good transient response and very low overshoot with lower packet marking, while the PI response is sluggish and with higher packet loss. Therefore, the high link utilization of PI (discussed in the last experiment) is at the expense of higher loss, sluggish response and large overshoot (and consequently larger RTT) in the case of high loads. Note here that the results of Exps. 3 and 4 approve the results derived for PFC AQM performance in Sec. 3.

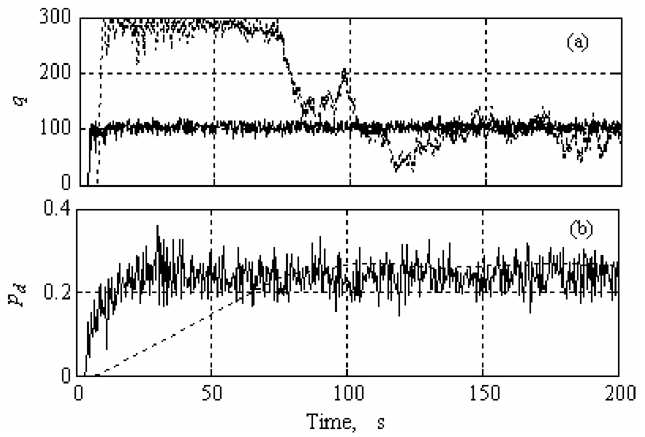

Fig. 9: a) Queue length evolution, b) Marking probability for Exp. 4: ARM-PFC (line), PI (dot)

Exp. 5: Performance under bottleneck link capacity variation: The available bandwidth of bottleneck links may vary due to different reasons such as SLA and QoS requirements [25]. Therefore, in this experiment the performance of ARM-PFC under such variations is investigated. For the same network setting as Exp. 1, at time $t = 40$ the bottleneck link bandwidth has been reduced from 10Mbps to 6Mbps and then has resumed its nominal value 10Mbps at time $t = 70$. The simulation results are shown in Fig. 10. While ARM-PFC shows its ability in effective queue regulation, PI and REM are highly sensitive w.r.t. link capacity variations. RED performs better than PI in this regard, however, its drop rate is increased slightly. The aggregate good put of the AQM schemes has also been measured. There are 124464043, 108863810, 106406600, and 99086200 packets for ARM-PFC, PI, RED and REM, respectively. Again, ARM-PFC shows better performance w.r.t. then other schemes.

Exp. 6: Robustness w.r.t. RTT: In this experiment, we change the propagation delays in the network topology of Fig. 4 and evaluate the robustness of the ARM-PFC controller. We set the propagation delay between the routers to 10ms and the mean of the delays between routers and the end hosts 2ms. This corresponds to a much smaller RTT than the nominal one. The regulated queues and marking probabilities are depicted in Fig. 11. ARM-PFC is still capable of robustly stabilizing the queue length. Note that for this scenario RED performs better, however, this result is a representative of the RED performance sensitivity w.r.t, the network parameters in accordance with [26]. On the other hand, REM tends to mark too many packets and keeps the queue size too small, so the link utilization is lower than other AQMs (0.9672 in comparison with 0.9981 for ARM-PFC). Meanwhile, PI AQM suffers from sluggish transient behavior.

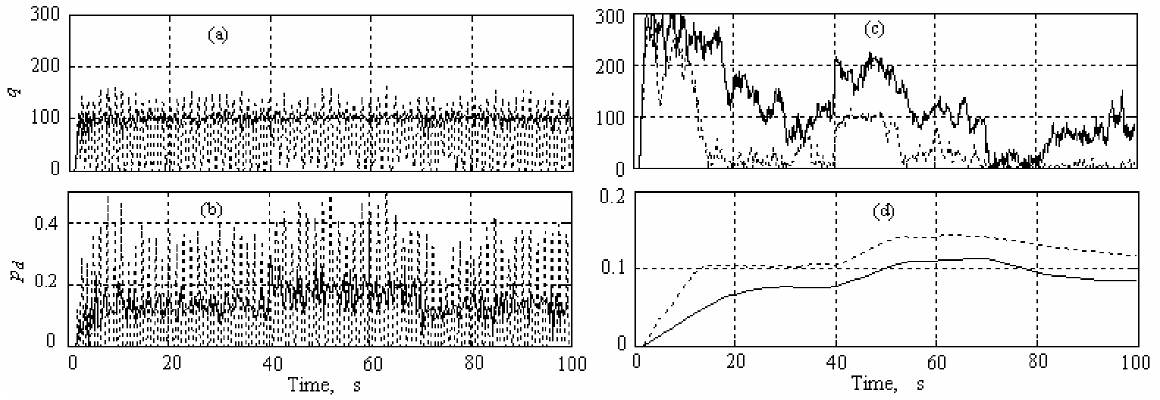

Fig. 10: Queue length and marking probability evolution for Exp. 5, a) and b): ARM-PFC (line), RED (dot); c) and d): PI (line), REM (dot)

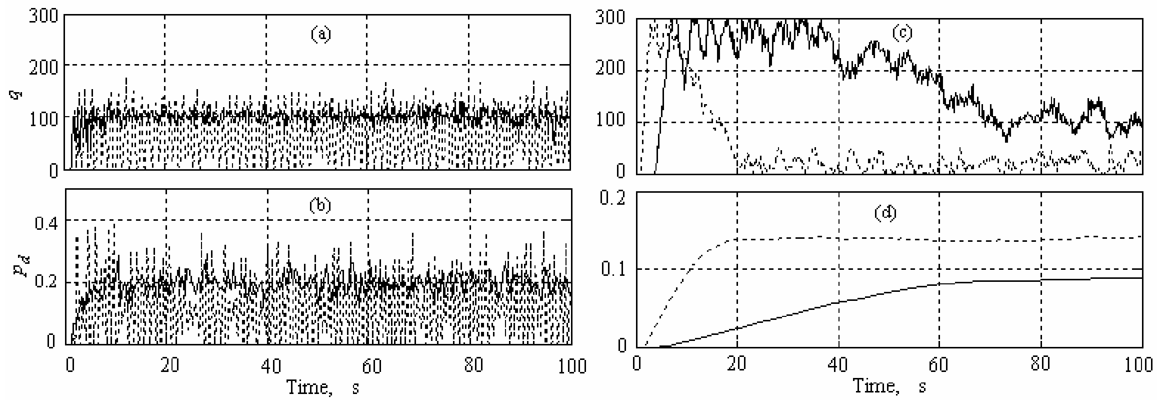

Fig. 11. Queue length and marking probability evolution Exp. 6 for much smaller RTT, a) and b):ARM-PFC (line), RED (dot); (c) and((d): PI (line), REM (dot)

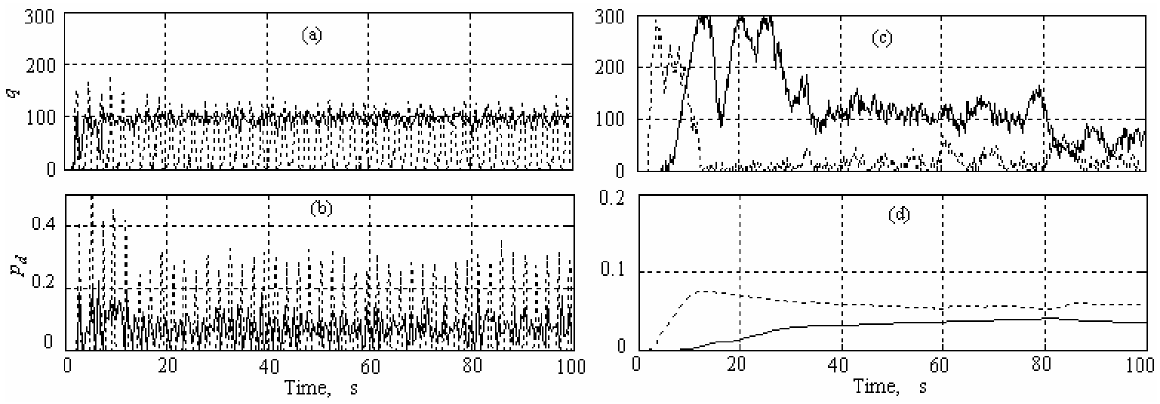

Fig. 12. Queue length and marking probability evolution Exp. 6 for much larger RTT, a) and (b):ARM-PFC (line), RED (dot); (c) and((d): PI (line), REM (dot)

As another part of this experiment, we consider the scenario with a much larger RTT, where we set the propagation delay between the routers to 120ms and the delay between the routers and the end hosts to 20ms. The simulations have been repeated with 100 FTP flows and the results are depicted in Fig. 12. As seen, ARM-PFC outperforms others both in queue regulation and speed of response.

Exp. 7: Performance in presence of short-lived http flows: In our sixth experiment, we consider a richer mix of traffic which contains long-lived TCP flows plus short-lived TCP flows (http flows). The single bottleneck topology of Fig. 4 has been considered for this experiment, while 120 greedy FTP flows and 150 short-lived http flows share the bottleneck link. Each http flow idles after finishing a session. The idle time is an exponential random variable with a mean of 3 seconds. Simulation results are shown in Fig. 13. The out-performance of ARM-PFC with respect to the other schemes is obvious from these figures once again.

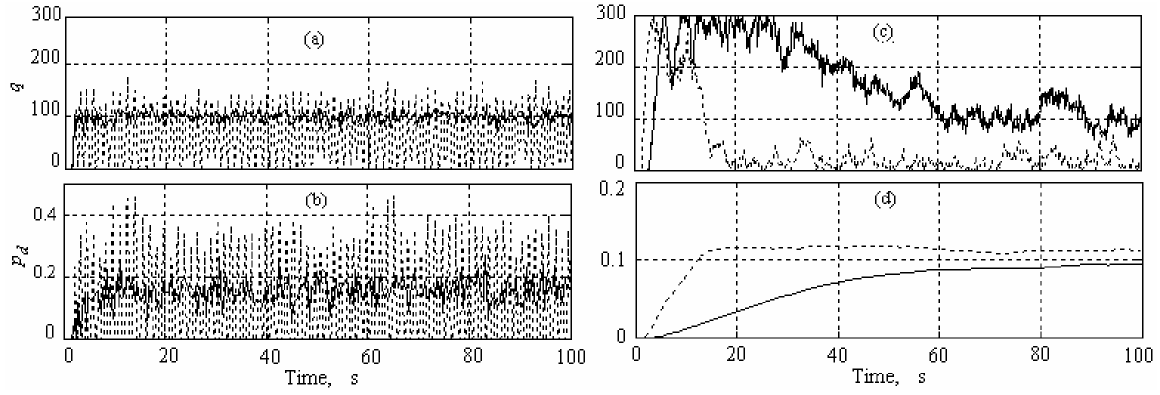

Fig. 13: Queue length and marking probability evolution for Exp. 7, a) and b): ARM-PFC (line), RED (dot); c) and d): PI (line), REM (dot)

Exp. 8: Performance in presence of unresponsive ON/OFF UDP flows: UDP ON/OFF traffic is another component of current internet traffic. In this experiment, the effect of the presence of this type of traffic on networks with ARM-PFC AQM has been compared with other schemes. 120 greedy FTP flows and 150 On/OFF UDP flows have been assumed. The UDP traffic has been generated with an Exponential traffic generator of *ns*-2 with the following parameters: the packet size is set to 210, the burst and idle time are equal to 500ms and the data sending rate has been chosen as 20kbyte/s. The simulation results are depicted in Fig. 14. These figures are also representative of the good ability of ARM-PFC to manage the queue length in the desired manner, even in the presence of such disturbances.

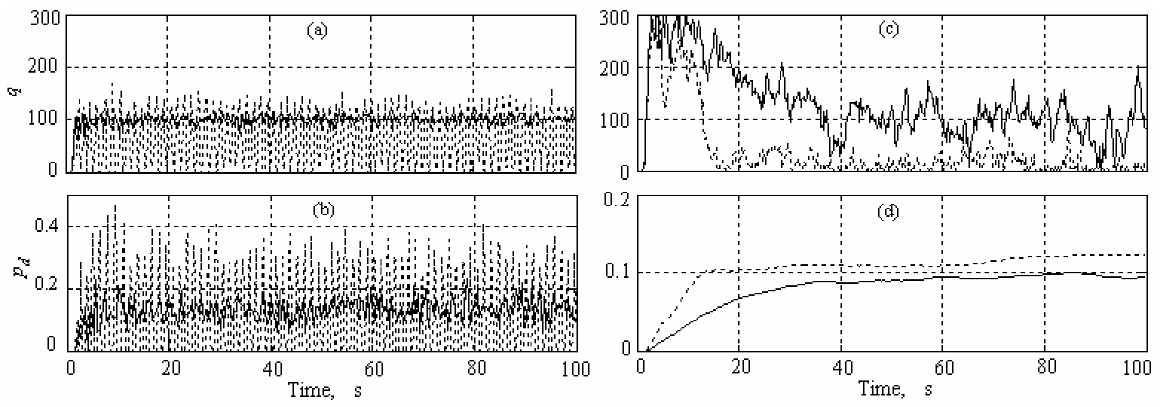

Fig. 14. Queue length and marking probability evolution for Exp. 8, a) and b): ARM-PFC (line), RED (dot); (c) and((d): PI (line), REM (dot)

a) Scenario of multiple bottleneck topology

Using the multiple bottleneck topology in Fig. 5, we study the behaviors of different AQM schemes in the presence of cross traffic to make sense of the scalability of these schemes. We set 300 TCP connections with senders at the left hand side and the receivers at the right hand side, with 50 TCP flows ($n_{cross} = 50$) for each cross traffic sender-receiver pair. The desired queue length has been set to 100 for this part as well. The instantaneous queues of Queues 2 and 4 are depicted in Fig. 15 (Queue 3 is similar to Queue 2) for different AQM schemes. Queues 1 and 5 are almost empty, indicating that these two links are not bottleneck links. Once again, the out-performance of ARM-PFC w.r.t other schemes is noticeable. We have repeated and investigated the simulations for a variety of TCP and cross traffic loads. Simulation results confirm good scalability of PFC in the multiple bottleneck networks.

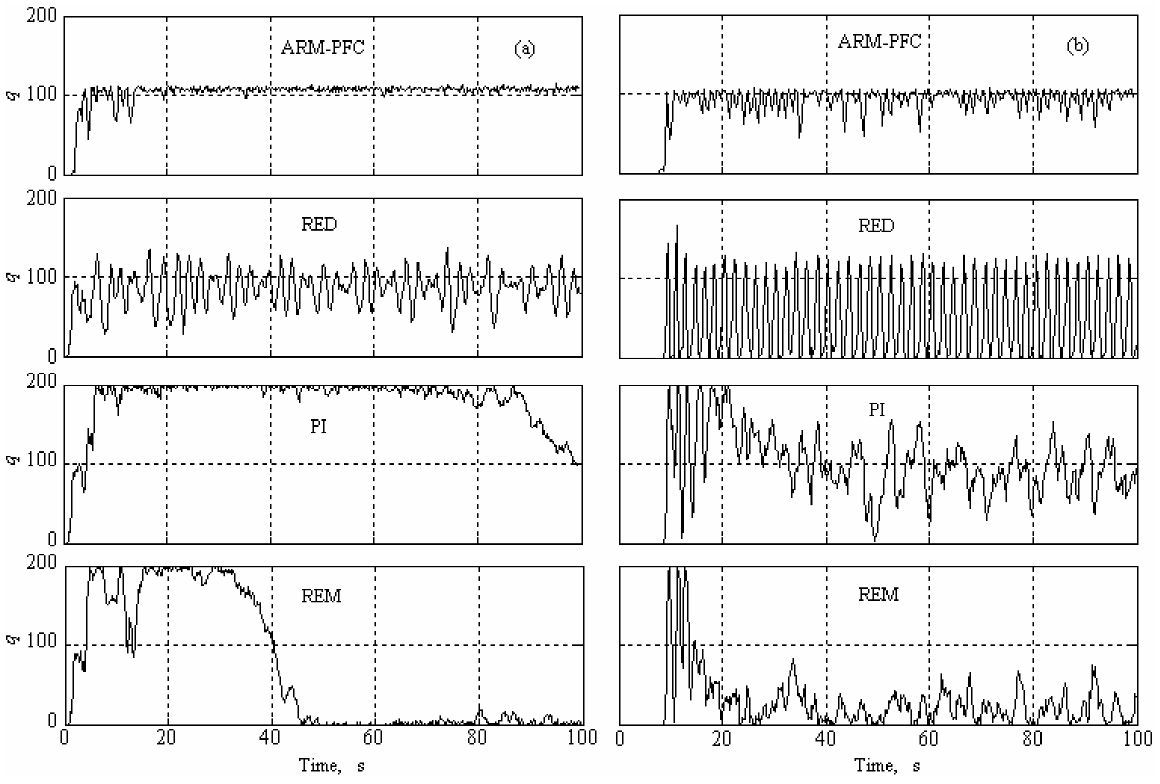

Fig. 15. Queue length evolution for (a) Queue 2 and (b) Queue 4 in the multiple bottleneck scenario

5. CONCLUSION

In this paper, a new predictive functional control called ARM-PFC is proposed as an active queue management system to control the congestion in TCP/IP networks supporting ECN. The developed system benefits from good features of both PFC as its controller and the ARMarkov structure as its prediction model to introduce a computationally optimized and effective congestion control strategy. PFC is a simplified version of generalized predictive control (GPC) which was developed to be efficiently applied in practical situations. The ability of handling system delay and rejecting unknown disturbances as well as its simplicity and low computational load makes PFC an appropriate choice as an AQM method. The proposed method achieves good performance in both queue regulation and compensation of the dynamic variations in a high speed network, having only a rough estimate of the network model. The model structure employed in the proposed PFC is an ARMarkov representation of the linearized fluid-flow model of TCP/AQM networks ([5] and [20]). The ARMarkov model enables the controller to predict the system output without any recursion, and therefore omits most of the computation usually involved in common MPC, and also reduces accumulative errors encountered in the other MPC variants. The compact model representation makes ARM-PFC more useful to be employed in fast dynamic computer networks, as it may be easily implemented in an adaptive form. In this paper, the common ARMarkov representation has been extended to capture the delay effects in a TCP/AQM system. Also a closed-form relation of the predictor has been derived which omits the need to employ some identification procedures, as it is regular in the related literature. For an adaptive scheme, the number of parameters to be identified on-line is decreased in this way, which means lower computational complexity again.

The performance of an ARM-PFC controller as an AQM in dynamic networks is investigated for networks with varying characteristics including network topology (e. g. bottleneck link capacity and links transport delays), and the network traffic (e.g. number of responsive flows and existence and nature of unresponsive flows such as http and both short-range and long-range dependent UDP traffics). The proposed control scheme performs very well in regulating the queue length around its desired value in all the simulated situations for both single and multiple bottleneck topologies. Fast response, low queue fluctuations, and consequently, low delay jitter, high link utilization, scalability and low marking probability are the other features of the proposed method with respect to the well-known AQM methods such as RED and PI, and REM which are also simulated for comparison. A challenging extension of this work is the design and implementation of adaptive ARM-PFC to increase the controller dynamic range, which is the ongoing research of the authors.

***Acknowledgements**:* The authors gratefully acknowledge the financial support of the Iranian Telecommunication Research Center (ITRC) under Grant T500-8210.

REFERENCES

1. Low, S. H. (2003). A duality model of TCP and queue management algorithms. *IEEE/ACM Trans. Networking*, Vol. 11, No. 4, pp. 525-536.
2. Athuraliya, S., Li, V.H., Low, S. H. & Yin, Q. (2001). REM: Active queue management. *IEEE Network*, Vol. 15, No. 3, pp. 48-53.
3. Floyd, S., Gummadi, R. & Shenker, S. (2001). Adaptive RED: An algorithm for increasing the robustness of RED's active queue management. Technical Report, August 2001, Available at http://www.icir.org/floyd/papers.html.
4. Floyd, S. & Jacobson, V. (1993). Random early detection gateways for congestion avoidance. *IEEE/ACM Transactions on Networking*, Vol. 1, pp. 397-413.
5. Misra, V., Gong, W. B. & Towsley, D. (2000). Fluid-based analysis of a network of AQM routers supporting TCP flows with an application to RED. *in Proc. ACM/SIGCOMM*, pp. 151-160.
6. Kelly, F. P., Maulloo, A. & Tan, D. (1998). Rate control in communication networks: shadow prices, proportional fairness and stability. *Journal of the Operational Research Society*, Vol. 49, pp. 237-252.
7. Sun, J., Ko, K. T., Chen, G., Chan, S. & Zukerman, M. (2003). PD-RED: To improve the performance of RED. *IEEE Communications Letters*, Vol. 7, No. 8, pp. 406-408.
8. Hollot, C. V., Misra, V., Towsley, D. & Gong, W. B. (2002). Analysis and design of controllers for AQM routers supporting TCP flows. *IEEE Transactions on Automatic Control*, Vol. 47, pp. 945-959.
9. Fengyuan, R., Chuang, L., Xunhe, Y., Xiuming, S. & Fubao, W. (2002). A robust active queue management based on sliding mode variable structure control. *INFOCOM'02*, pp. 13-20.
10. Ryu, S., Rump, C. & Qiao, C. (2003). A predictive and robust active queue management for internet congestion

control. *in Proc. The Eighth IEEE International Symposium on Computers and Communication (ISCC' 03)*, pp. 1530-1346.

11. Bigdeli, N. & Haeri, M. (2005). Design of a robust AQM strategy for dynamic TCP/AQM networks based on CDM. *in Proc. CCA05*, Toronto, Canada, pp. 716-721.
12. Richalet, J. (1993). *Pratique de la commande prédictive.* Paris: Editions Hermés.
13. Richalet, J. (1993). Industrial applications of model-based predictive control. *Automatica*, Vol. 29, No. 5, pp. 1251-1274.
14. Richalet, J., Abu, E., Arber, C., Kuntze, H. B., Jacubasch, A. & Schill, W. (1997). Predictive functional control. Application to fast and accurate robot. *Tenth IFAC World Congress,* Munich, Germany.
15. Akers, J. C. & Bernstein, D. S. (1997). ARMarkov least-squares identification. *in Proc. The American Control Conference,* Albuquerque, New Mexico, USA, pp. 186-190.
16. Kamrunnahr, M., Huang, B. & Fisherr, D. G. (2000). Estimation of markov parameters and time-delay/interactor matrix. *Chemical Engineering Science*, Vol. 55, No. 17, pp. 3353-3363.
17. Kamrunnahr, M., Fisherr, D. G. & Huang, B. (2002). Model predictive control using an extended ARMarkov model. *Journal of Process Control*, Vol. 12, pp. 123-129.
18. Kamrunnahr, M., Fisher, D. G. & Huang, B. (2004). Performance assessment and robustness analysis using an ARMarkov approach. *Journal of Process Control*, Vol. 14, pp. 915-925.
19. Network Simulator, *ns-2*, http:/www.isi.edu/nsnam/ns/
20. Hollot, C. V., Liu, Y., Misra, V. & Towsley, D. (2003). Unresponsive flows and AQM performance. *IEEE INFOCOM* 2003, pp. 85-95.
21. Soltanzadeh, A. (2005). Choosing proper predictive horizon in MPC design for oscillating systems. MSc thesis. Sharif University of Technology, Tehran, IRAN.
22. Akers, J. C. & Bernstein, D. S. (1997). Time-domain identification using ARMarkov/Toeplitz models. *in Proc. The American Control Conference,* Albuquerque, New Mexico, USA.
23. Astrom, K. J. & Wittenmark, B. (1997). *Computer-controlled systems, theory and design*, Prentice Hall, 1997.
24. Yan, P., Gao, Y. & Ozbay, H. (2005). A variable structure control approach to active queue management for TCP with ECN. *IEEE Transactions on Control System Technology,* Vol. 13, No. 2, pp. 203-215.
25. Zheng, Y., Lu, M. & Feng. Z. (2003). Performance evaluation of adaptive AQM algorithms in a variable bandwidth network. *IEICE Transactions on Communications,* Vol. E86-B, No. 6, pp. 2060-2067.
26. Christiansen, M., Jeffy, K., Ott, D. & Smith, F. D. (2000). Tuning RED for web traffic. in proc. ACM/SIGCOMM, Stockholm, Sweden, pp. 139-159.