한국경영과학회/대한산업공학회 2003 춘계공동학술대회
2003년 5월 16일-17일 한동대학교(포항)

PAQM: an Adaptive and Proactive Queue Management for end-to-end TCP Congestion Control

류 승완
한국전자통신연구원 이동통신연구소 이동서비스연구부
전화:042-860-1505, FAX:042-860-5119, email: rush@etri.re.kr

Abstract
In this paper, we introduce and analyze a feedback control model of TCP/AQM dynamics. Then, we propose the *Pro-active Queue Management (PAQM)* mechanism, which can provide proactive congestion avoidance and control using an adaptive congestion indicator and a control function for wide range of traffic environments. The PAQM stabilizes the queue length around a desired level while giving smooth and low packet loss rates independent of the traffic load level under a wide range of traffic environments. The PAQM outperforms other AQM algorithms such as Random Early Detection (RED) [1] and PI-controller [2].

1. Introduction

Active queue management (AQM) is a group of FIFO-based queue management mechanisms in a router to support end-to-end congestion control in the Internet. Two main functions are used in AQM: one is the congestion indicator and the other is the congestion control function. Until Random Early Detection (RED) was proposed by the Internet Engineering Task Force (IETF) for deployment [3], FIFO-based tail drop (TD) was the only AQM mechanism used in the network. The TD mechanism uses the instantaneous queue length as a congestion indicator, and drops packets when the buffer becomes full. Although simple and easy to implement, TD has two well-known drawbacks, the lock-out and full queue phenomena [3].

RED enhanced the two functions by introducing queue length averaging and probabilistic early packet dropping. In particular, RED uses an exponentially-weighted moving average (EWMA) queue length not only to detect incipient congestion but also to smooth the busrty incoming traffic and its resulting transient congestion. Following RED [1], many AQM-based extensions such as SRED [4], Adaptive-RED [5], PI-controller [2], etc, have been proposed. However, many AQM proposals have shown severe problems with the (average) queue length as a congestion indicator [6, 2, 7].

One important drawback of currently proposed AQM algorithms is that their congestion detection and control functions depend only on the current queue status or the history of the queue status (e.g., average queue length). Hence the congestion detection and control in these algorithms are *reactive* to current or past congestion not *proactive* to incipient congestion. To address this problem, it is necessary for an AQM mechanism to have a more efficient congestion indicator and control function. To avoid or control congestion *proactively* before it becomes a problem, both the congestion indicator and control function of an AQM should be adaptive to changes in the traffic environment such as the amount of traffic, the fluctuation of traffic load, the nature of traffic.

In this paper, we propose a *proactive queue management (PAQM)* mechanism which can detect incipient congestion as well as control congestion effectively and proactively. The goals of PAQM are to control congestion proactively, to make the queue length agree with a desired level and to give smooth and low packet loss rates. To achieve these goals, having an efficient congestion prediction and control capability is important. We use a classical proportional- integral-derivative (PID) feedback control method in designing PAQM not only to have the anticipatory congestion detection and control capability but also to achieve a long-term control performance such as acceptable queue length behavior (or equivalently delay), acceptable packet loss rates or high link utilization.

This paper is organized as follows. In the next section, we review and analyze some TCP/AQM dynamics including RED [1] and PI-controller [2] using feedback control system modelling and analysis. Then we propose our Pro-Active Queue Management (PAQM) algorithm including the PID-control based discretized implementation of PAQM method. Then we examine the performance of PAQM via simulation using ns-2 [8], and compare the performance to other AQM algorithms. Finally, we summarize our study and suggest directions for future study.

2. Control Theoretic Design of AQM

In this section, we first introduce feedback

control modelling of the TCP/AQM dynamics [2], and analyze RED and PI-controller in terms of feedback control. Then, we proposed PAQM as a pro-active queue management algorithm.

2.1. Feedback control model of TCP/AQM dynamics

TCP congestion control dynamics with an AQM can be modelled as a feedback control system (Figure 1). In [7], a system of nonlinear differential equations for TCP/AQM dynamics was developed using fluid-flow analysis while ignoring the TCP time-out and slow-start mechanisms. Then in [9], a linearized TCP/AQM dynamic model was developed and analyzed especially for TCP/RED dynamics in terms of control theory. There, the forward-path transfer function (FPTF) of the plant, $P(s) = P_{tcp}(s) \cdot P_{Queue}(s)$, was given by

$$P(s) = (((R_0C^2)/(2N^2))/(s+2N/(R_0^2C))) \cdot ((N/R_0)/(s+1/R_0)) \quad (1)$$

where N is a load factor (number of TCP connections), R_0 is a round trip time, and C is the link capacity.

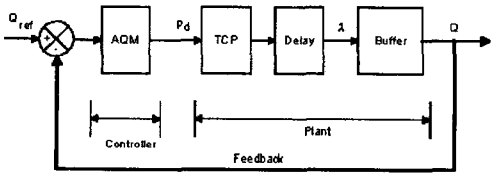

Figure 1 Feedback control modelling of TCP congestion control with AQM algorithm.

Since $P(s)$ has two non-zero poles, it is a type 0 system [10] so that there always exists constant steady-state error to the step-function input [10]. Since, $lim_{s\to 0}P(s) = (R_0C)^3/(4N^2)$ the steady-state error of the FPTF is $e_{ss} = R/(1+(R_0C)^3/(4N^2))$, where R is the reference input to eliminate this constant steady-state error.

2.2 RED and PI-controller

With RED [1], a link maintains the EWMA queue length, $Q_{avg} = (1-w_Q)Q_{avg} + w_Q Q$, where Q is the current queue length and w_Q is a weight parameter, $0 \le w_Q \le 1$. When Q_{avg} is less than the minimum threshold ($\min_{th}$), no packets are dropped. When it exceeds the maximum threshold ($\max_{th}$), all incoming packets are dropped. When it is in between, a packet is dropped with probability p_d that is an increasing function of Q_{avg}, i.e., $p_d = max_p(Q_{avg} - min_{th})/(max_{th} - min_{th})$, where $\max_p$ is a maximum value of p_d. RED attempts to eliminate the steady-state error by introducing the EWMA error terms as an integral (I)-control to the open-loop transfer function, $P(s)$. However, since RED introduces a range of reference input value, i.e., $[min_{th}, max_{th}]$, rather than unique reference input for the I-control, the TCP/RED model shows oscillatory system dynamics. Moreover a very small weight factor value ($w_Q = 1/512 = 0.002$) to the current queue length in calculation of the EWMA queue length is one of the main reasons of queue length oscillation and the bias against bursty sources [11, 12]. In addition, the small value of w_Q brings an effect of large integral time in I-control, and may accompany large overshoot [10]. As a result, RED shows oscillatory queue length dynamics and gives poor control performance under a wide range of traffic environments.

The Proportional-Integral (PI)-controller has been proposed in [2] to overcome these drawbacks of RED with the introduction of a constant desired queue length (Q_{ref}). PI-controller has been designed based on (1) not only to improve responsiveness of the TCP/AQM dynamics but also to stabilize the router queue length around Q_{ref}. The latter can be achieved by means of I-control, while the former can be achieved by means of proportional (P)-control using the instantaneous queue length rather than using the EWMA queue length.

2.3. Limitations of currently proposed AQM algorithms

Since each TCP source controls its sending rate through window size adjustment, the aggregate input traffic load (the offered load) at time $t \ge 0$, λ_t, is proportional to the total window size of all connections, $W \propto \lambda_t(R + Q_t/C)$, where R is the average propagation delay of all connections, Q_t/C is the queueing delay at a router, C is the output link capacity and Q_t is the current queue length. Because of limited buffer size and output link capacity, the *carried traffic load*, λ'_t will be a fraction of offered

traffic load that is not dropped at a router, i.e., $\lambda_t' = \lambda_t(1-p_d)$, where, p_d is the packet drop probability.

In a time-slotted model where time is divided into small time slots and the queue size, Q_t, and total amount of queued input traffic, λ_t', is calculated at the end of each time slot, Q_t is a function of λ_t', i.e, $Q_t = (\lambda_t' - C)\Delta t + Q_{t-1}$, where Δt is the unit length of a time slot. However, the incipient congestion will be a function of the queue length of the next time slot, Q_{t+1}, not a function of Q_t. Therefore, to detect *incipient* congestion *proactively* not *current* congestion *reactively*, p_d should be an increasing function of Q_{t+1} (or λ_{t+1} equivalently). Unfortunately, most AQM algorithms such as RED [1], or PI-controller [2] use only the past traffic history such as Q_t (or the average queue length $\overline{Q}$) as a congestion indicator. As a result, these AQM algorithms are unable to detect incipient congestion adaptively to the traffic load variations.

3. PAQM: A Pro-Active Queue Management

3.1. Proportional-Integral-Derivative control

The proportional (P), integral (I) and derivative (D) feedback in the PID control is based on the *past (I)*, *current (P)* and *future (D)* control error [13]. Thus, PID control is able to regulate the queue length (Q_t) around the desired level (Q_{ref}) and to provide fast speed of response simultaneously with acceptable stability and damping. A generic PID control equation is

$$u(t) = K_P e(t) + K_I \int e(\tau)d\tau + K_D \frac{d}{dt} e(t) \quad (2)$$

where $u(t)$ is control signal at time $t \geq 0$, K_P is a proportional gain, K_I is a integral gain, and K_D is a derivative time. The corresponding discrete PID control equation obtained from (2) for a small time interval, T_s, is [14].

$$u(k) = K_P\, e(k) + \frac{T_s}{T'_I} \sum_{i=0}^{k-1} e(i) + K_P \frac{K'_D}{T_s} (e(k) - e(k-1)), \quad (3)$$

where $u(k)$ is a control signal at a sampling time, $k = t/T_s = 0, 1, \cdots$, $K'_D = K_D/K_P$, $T'_I = K_P K_I$, and $e(k)$ is a sampled error term at time k.

3.2. PAQM

We propose PAQM to overcome the reactive congestion control problems of existing AQM algorithms. PAQM is designed to detect the incipient congestion as well as the current congestion by adopting an adaptive congestion indicator and a control function. For proactive congestion control, PAQM enhanced the control function with the introduction of a predictive traffic measure for the adaptive congestion indication. The goals of PAQM are to avoid and control congestion proactively by anticipating incipient congestion, to stabilize the queue length at a router around a desired queue length (Q_{ref}) and to provide acceptable bounded queueing delay, higher link utilization, etc.

To achieve these goals, PAQM is designed by approximating the direct digital design of PID feedback control [15]. First, a discrete PID control is considered. Then, to resolve two conflicting design goals, the *responsiveness* (the short-term control performance) and the *stability* (the long-term control performance), the discretized PID control is divided into two parts, a proportional-integral (PI) control part for stability and a derivative (D) control part for responsiveness. Thus, PAQM takes advantages of using both the capability of the PI control for elimination of the steady state error and the anticipatory capability of D-control for prediction and reduction of upcoming error, i.e., incipient congestion. A simple block diagram of PAQM is shown in Figure 2.

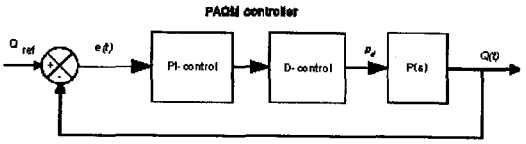

Figure 2 A simple block diagram of PAQM as a PID-controller.

For the digital implementation of PAQM, the traffic history of two recent time slots is used for two reasons: 1) the dynamic changes of network traffic with time and 2) the recent observation of Poisson-like network traffic behavior under heavy traffic loads[1)][16]. To stabilize the queue length to the desired level, Q_{ref}, a velocity PID-control [14] is used in digital implementation of PAQM. The rationale for using a velocity PID-control is that the velocity PID-control

updates the control signal recursively by summing the current rate of change of the control signal, $\Delta u(k) = u(k) - u(k-1)$, and the previous control term, $u(k-1)$. Also, a velocity PID-control can provide better control performance from its anticipatory control nature.

3.2.1 PI-control

The first part of PAQM controller is the *proportional-integral (PI)-control*. A discrete PI-control equation is obtained from (3) when $K'_D = 0$ [14].

$$u(k) = K_P\ e(k) + (T_s/T'_I)\sum_{i=k-2}^{k-1} e(i)\ .$$

Then, the velocity control implementation of this PI-control is $u(k) = u(k-1) + \Delta u(k)$. Since the integral time, T'_I, is assumed to be consisted of two timeslots ($T'_I = 2T_s$) and $e(k-1) + e(k-3) = Q_{k-1} + Q_{k-3} + 2Q_{ref}$, the instantaneous PI-control signal, $\Delta u(k)$, is

$$\Delta u(k) = K_P(e(k) - (\frac{Q_{k-1} + Q_{k-3}}{2} - Q_{ref})) \quad (4)$$

Then (4) is modified to use at most two recent queue lengths rather than using Q_{k-3} for a discretized velocity PI control. If both Q_k and Q_{k-1} are greater than Q_{ref} (or less than Q_{ref}), the average surplus (or slack) amount of traffic is calculated using a *Trapezoidal-Integral rule* [10]. Then the packet drop probability p_d is adjusted proportional to the amount of surplus or slack traffic. Finally, the PI control equation is

$$p_d(k) = p_d(k-1) + \alpha\ \frac{Q_k + Q_{k-1}}{2} - Q_{ref}\ ,$$

where α is a control gain. For better scaling of PI-control implementation, the current amount of surplus/slack traffic, $(Q_k + Q_{k-1})/2 - Q_{ref}$, is normalized by Q_{ref}. This value represents the fraction of the amount of current surplus/slack traffic to the desired queue level. Thus, the normalized PI control equation is

$$p_d(k) = p_d(k-1) + \alpha\ \frac{Q_k + Q_{k-1}}{2} - Q_{ref}\ /\ Q_{ref}, \quad (5)$$

3.2.2 D-control

The second part of the PAQM controller is the *derivative (D)-control*. A discrete D-control equation is obtained from (3) when $K_P = 0$ and $K'_I = 1/T'_I = 0$.

$$u(k) = (K_P K'_D/T_s)(e(k) - e(k-1)).$$

Then, similar to PI-control part, a discretized velocity D-control is $u(k) = u(k-1) + \Delta u(k)$, and the instantaneous control signal, $\Delta u(k)$, is

$$\Delta u(k) = \frac{K_P K'_D}{T_s}(e(k) - 2e(k-1) + e(k-2)),$$

where K_P and K'_D are a proportional gain and a derivative time respectively. From the relation between a differential equation and a difference equation,

$$\frac{d^2 e(t)}{dt^2} \approx \frac{e(k) - 2e(k-1) + e(k-2)}{(T_s)^2},\ \text{and}$$

$$\Delta u(k) = K_P K'_D(T_s \frac{d^2}{dt^2} e(t)).$$

$T_s \frac{d^2}{dt^2} e(t)$ represents the predicted amount of an acceleration of changes on $e(t)$ in time T_s ahead, i.e., $e(t + T_s) = e(t) + K_P K'_D(T_s \frac{d^2}{dt^2} e(t))$. Then the tendency (acceleration) of traffic is obtained from the discretized implementation of a velocity D-control using traffic history as

$$\Delta u(k) = (K_P/2)(Q_k - 2Q_{k-1} + Q_{k-2}). \quad (6)$$

In order to obtain the acceleration term, $\frac{d^2}{dt^2} e(t)$, first the current traffic status, S_k, is set from the surplus or slack amount of traffic, $\gamma_k = Q_k - Q_{k-1}$ via

$$S_k = \begin{cases} 1, & \gamma_k \geq 0 \\ -1, & otherwise \end{cases} \quad (7)$$

Then the traffic tendency of previous two time-slots, $S = S_k * S_{k-1}$, is obtained. Finally, the estimated surplus or slack amount of traffic for the next time slot, $\hat{\gamma}_{k+1}$, is obtained using S, γ_k and γ_{k-1} via

$$\hat{\gamma}_{k+1} = \begin{cases} \gamma_k * (\gamma_k/\gamma_{k-1}), & S = 1 \\ \gamma_k, & otherwise \end{cases} \quad (8)$$

If $S = 1$,

$$T_s \frac{d^2}{dt^2} e(t) \approx \Delta u(k) = \frac{Q_k - 2Q_{k-1} + Q_{k-2}}{T_s}$$

$$= (\gamma_k / T_s)^* (\gamma_k / \gamma_{k-1}).$$

If $S \neq 1$, the predicted error for the next time slot in PAQM controller is the same as the linear extrapolating method [17]. The predicted queue length for the next time slot is $\hat{Q}_{k+1} = Q_k + \hat{\gamma}_{k+1}$ and the corresponding predicted error is $\hat{e}(k+1) = \hat{Q}_{k+1} - Q_{ref}$. The D-control equation is

$$p_d(k) = p_d(k-1) + \alpha(\hat{Q}_{k+1} - Q_{ref}). \quad (9)$$

For better scaling of digital implementation of PAQM, $\hat{e}(k+1)$ is normalized by the surplus buffer capacity, $B - Q_{ref}$. Since the normalization factor, $B - Q_{ref}$, represents the buffer capacity for absorbing the transient surplus bursty traffic without losses, the normalized value represents the ratio of the amount of surplus/slack traffic to the surplus buffer capacity. The normalized D-control equation is

$$p_d(k) = p_d(k-1) + \alpha \frac{\hat{Q}_{k+1} - Q_{ref}}{B - Q_{ref}}, \quad (10)$$

where α is a control gain.

4. Simulation study

In this section, we examine the control performance of the PAQM algorithm with the other AQM algorithms such as RED [1] and PI controller [2]. These are done via simulation over a wide range of traffic environments using the ns-2 simulator [8].

4.1. Simulation setup

We use a simple bottleneck network topology with two routers, nc0 and nc1, and 9 TCP/Reno sources and 9 logically connected destinations. These 9 pairs of TCP/Reno sources and destinations are connected to nc0 and nc1 respectively. The link between nc0 and nc1 is assumed to have 30Mbps of link speed and 10ms of propagation delay. Since all TCP connections are connected to the routers, nc0 and nc1, with link speed of 50Mbps, the link between nc0 and nc1 is the only bottleneck. Each logically connected pairs between TCP source (srci) and the destination (desti), $i = 1, \cdots, 9$, are connected with propagation delays from 40ms to 200ms. We consider two types of traffic flows: *elephant* long-lived FTP flows and *mice* short-lived flows. To obtain realistic TCP flows n FTP flows and $2n$ mice flows are connected to each (srci) so that 66% of TCP flows are mice flows[2]. The network configuration is shown in Figure 3. Each packet is assumed to have an average size of 1000 bytes. All sources and destinations are assumed to use TD queue management with sufficient buffer capacity. The buffer (B) at the bottleneck link uses an AQM algorithm and has capacity of 800 packets which is twice the bandwidth-delay product (BDP)[3].

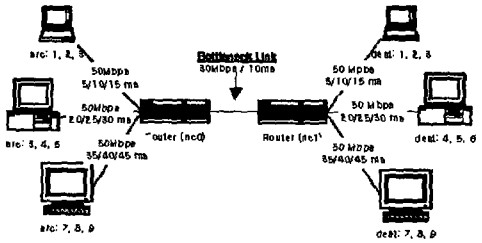

Figure 3. Simulation network topology

We compare PAQM with RED [1] and PI-controller [2] under packet drop mode. In RED, recommended parameter values [18] are used. In the PI-controller, we use the parameters, a, b and T, used in [2].

In *PAQM*, the sampling time interval (T_s) must be selected so that the buffer does not experience overflow or underflow. The amount of packets that arrives in a T_s is $\lambda^* T_s$, where λ is the packet arrival rate. Consider a design requirement that the queue length should not be increased/decreased by the amount of Q_{ref} in a T_s. Then, T_s should be less than Q_{ref}/C to prevent the buffer from underflow. On the other hand, the maximum number of queued packets in a T_s, $\lambda(1-p_d)^* T_s$, must be less than the buffer capacity (B) to prevent the buffer from overflow, where p_d is the packet drop probability. Thus, T_s must be selected so that the buffer does not experience overflow. For example, if the maximum packet queueing rate is assumed to be twice the link capacity, i.e., $\lambda(1-p_d) = 2^* C = 7500$ packets/second, in this network configuration, the maximum surplus traffic will be $3750^* T_s$ packets. Therefore, to satisfy the design requirement, T_s should be less than $Q_{ref}/C = 200/3750 = 53.3$ ms. In this study, we use $T_s = 50$ ms as a T_s for PAQM.

The amount of unit adjustment (i.e., unit increase or decrease) of p_d (or increase or

decrease of queued packets) in a T_s is a function of α. Thus, the value of the control gain is proportional to the buffer capacity (B). However, α should be determined by a trade-off between two conflicting design goals of an AQM, i.e., the responsiveness (the short-term design goal) and the stability (the long-term design goal) to produce satisfactory control performance. We select a fractional value of the buffer size, $B^* 10^{-6} = 8.0^* 10^{-4}$, as a control gain of PAQM empirically via extensive simulations.

A summary of parameters setting of each AQM algorithms is shown in Table 1.

RED	$w_Q = 0.002, max_p = 0.1, min_{th} = 70,$ $max_{th} = 200$
PI-C	$a = 1.822^* 10^{-5}, b = 1.816^* 10^{-5},$ $Q_{ref} = 200, f_s = 160Hz (T_s = 6.25ms)$
PAQM	$\alpha = 8.0^* 10^{-4}, T_s = 50ms, Q_{ref} = 200$

Table 1 Parameters setting of AQM algorithms

4.2. Performance metrics

4.2.1 The queue length

The control performance of an AQM can be evaluated by inspecting whether the queue length is regulated around a desired value or not. We use the instantaneous queue length as a performance metric for control performance. In particular, for the steady-state control performance, we use the *quadratic average of control deviation (QACD)* [14] defined as

$$S_e = \sqrt{(1/(N+1))\sum_{i=0}^{N}(Q_{ref} - Q_i)^2}$$

where Q_{ref} is the desired queue length, Q_i is the i^{th} sampled queue length, $i = 0, \cdots, N$, and N is the number of sampling intervals.

4.2.2 The packet loss rate

Since one of goals of an AQM algorithm is to remove bias against bursty sources, maintaining stable and low packet loss rate is important. High and bursty packet loss involves many packet losses at about the same time. If these packets belong to different flows, these flows experience losses about at the same time, and then may experience global synchronization as a result. On the other hand, if these packets belong to bursty sources, there exists bias against bursty sources. In general, bias against bursty sources and the global synchronization can be eliminated effectively by achieving stable and low packet loss rate over time. In addition, to achieve higher throughput (or goodput), or to accommodate more traffic, maintaining low average packet loss rate is important.

4.3 Control performance of AQM algorithms

We examine the control performance of AQM algorithms, PAQM, PI-controller and RED, in terms of the queue length and the packet loss rate under two different traffic load levels, i.e., $n = 7$ (or 189 flows) and $n = 14$ (or 378 flows).

4.3.1 The queue length dynamics

Figure 4 shows the queue length dynamics of PAQM, PI-controller and RED respectively with 189 flows (left) and 378 flows (right). PAQM (top) shows good control performance under two different load levels by regulating the queue length around $Q_{ref} = 200$ packets. PI-controller (middle) fails to maintain the queue length around Q_{ref}. Instead the queue length stays below Q_{ref} most of time under two different traffic load levels. Thus, PI-controller behaves like a tail-drop (TD) with buffer size of $Q_{ref}$with severe fluctuation. RED maintains the (average) queue length between $min_{th} = 70$ and $max_{th} = 200$ effectively under 189 flows (light traffic). However, RED behaves like TD with buffer size of Q_{ref} under 378 flows similar to PI-controller.

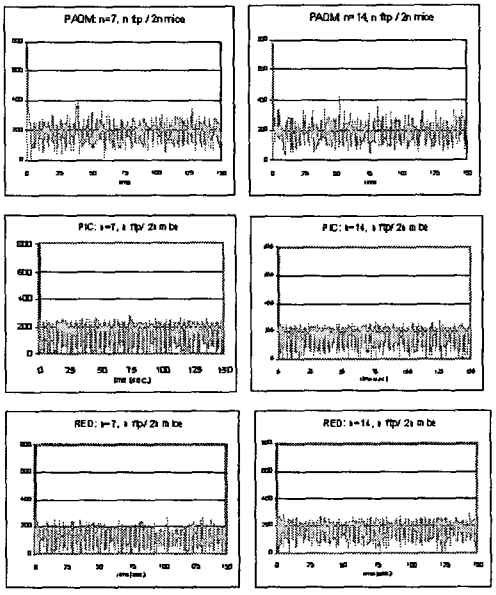

Figure 4 The queue lengths of PAQM (top), PI-controller (middle) and RED (bottom) under light (left) and medium (right) load levels.

Figure 5 shows the steady-state control performance of PAQM and PI-controller in terms of *QACD* under different traffic load levels. Since only PAQM and PI-controller maintain unique desired queue length, Q_{ref}, we compare QACD of these two AQMs. PAQM shows better and more robust steady-state control performance than PI-controller for all cases of traffic load levels.

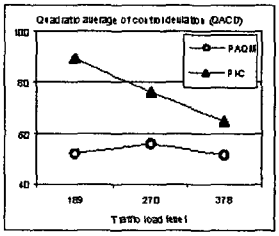

Figure 5 The QACD of AQM algorithms for different traffic load levels.

4.3.2 The packet loss rate

Figure 6 shows the packet loss rate of PAQM, PI-controller and RED over time respectively under the same traffic load levels used in figure 4. PAQM shows stable and low packet loss rate over time, and thus can remove bias against bursty sources effectively. PI-controller and RED show fluctuating high packet loss rate over time under 378 flows. Under 189 flows, only PI-controller shows high packet losses over time.

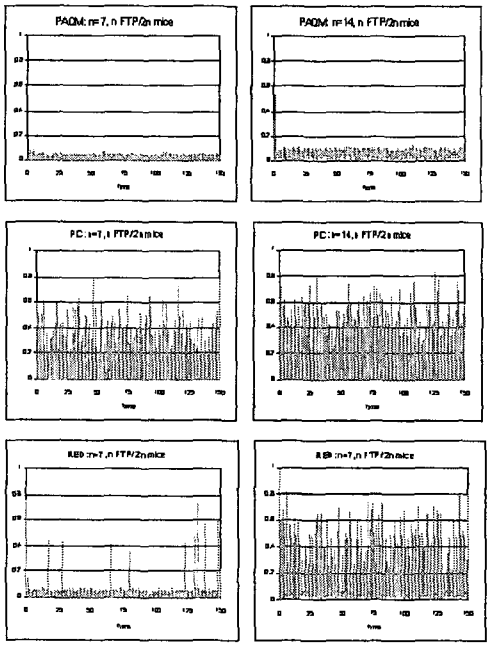

Figure 6 The packet loss rate of PAQM (top), PI-controller (middle) and RED (bottom) with 189 flows (left) and 378 flows (right).

Figure 7 shows the average packet loss probability and link utilization of each AQM algorithms under different traffic load levels. PAQM shows significantly lower average packet loss probability than other AQM algorithms for all cases of traffic load levels. The link utilization of PAQM is higher than PI-controller, and almost the same to RED for all cases of traffic load levels.

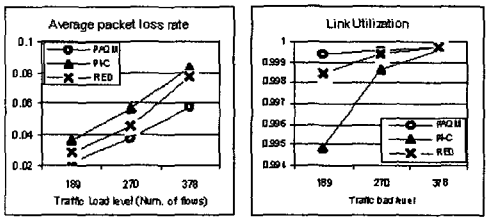

Figure 7 The average packet loss probabilities and the link utilization of AQM algorithms under several different load levels.

5. Conclusion

In this paper, we designed a pro-active queue management, PAQM, using classical PID feedback control concept. PAQM can give fast response and proactive control to the changing traffic load using anticipatory traffic prediction with introduction of a derivative (D) control method. PAQM also can achieve satisfactory steady-state control performance such as stable queue length with introduction of proportional-integral (PI) control method. As a result, PAQM shows robust congestion control performance under various traffic load levels in realistic TCP traffic environment. PAQM outperforms other AQM algorithms such as RED and PI-controller in terms of the queue length dynamics and packet loss rate. In practical implementation, PAQM has comparably little computational overhead. In RED, the EWMA queue length and the packet drop probability are calculated at every packet arrival while maintaining several parameters such as w_Q, $\min_{th}$, $\max_{th}$, $\max_p$ etc. In contrast, PAQM can be easily implemented with a less sampling frequency (20 Hz) compared to the link speed (i.e., 3750 Hz) implementation of RED and 160 Hz of PI-controller while maintaining fewer parameters than RED. Therefore, the computational complexity and overhead at a router can be reduced significantly with PAQM.

There are several further study subjects. First, to examine the impact of the queue length sampling frequency (f_s) on the performance of PAQM, we hope to find the relationship between

f_s and the offered traffic load. We are studying to find the optimal control gain, α, and the stability margin through control-theoretic modelling and analysis of the PAQM algorithm. Finally, we will examine the adaptability and robustness of PAQM under dynamic traffic situations such as various network topologies, time varying traffic factors and different round-trip time (RTT). In general, the control performance of an AQM algorithm is affected by several network components such as the buffer size (B), the link capacity (C), the traffic factor (e.g., traffic load and mixture) and the round-trip time (RTT). Since the buffer size and the link capacity are fixed when an AQM is installed in a router, these parameters are *static* (i.e., time invariant) factors. In contrast, the traffic factors and the RTT are *dynamic* (i.e., time-varying) factors because they are changing dynamically over time. Thus, it is important for an AQM algorithm to have adaptive and robust control performance to the dynamic factors.

References

[1] Floyd, s. and Jacobson, V., "Random early detection gateways for congestion avoidance", *IEEE/ACM Transactions on Networking*, Vol.1, No.4(1993), pp397-413.

[2] Hollot, C.V., Misra, V., Towsley, D., and Gong, W., "On designing improved controllers for AQM routers supporting TCP flows", *Proceedings of INFOCOM*, April 2001, pp1726-1734.

[3] Braden, B., et al., "Recommendations on Queue Management and Congestion Avoidance in the Internet", *IETF RFC2309*, 1998.

[4] Ott, T. J., Lakshman, T. V., Wong, L, "SRED: Stabilized RED", *Proceedings of INFOCOM*, March 1999, pp1346-1355.

[5] Feng, W., Kandlar, D., Saha, D., and Shin, K., "A Self-configuring RED gateway", *Proc. of INFOCOM*, March 1999, pp1320-1328.

[6] Christiansen, M., Jeffey, K., Ott, D., and Smith, D., "Tuning RED for web traffic", *IEEE/ACM Transactions on Networking*, Vol.9, No.3(2001), pp249-264.

[7] Misra, V., Gong, W., and Towsley, D., "Fluid-based analysis of a network of AQM routers supporting TCP flows with an application to RED", *Proc. of ACM SIGCOMM*, September 2000.

[8] McCanne, M. and Floyd, S., "Network simulator - ns (version 2)", http://www.isi.edu/nsnam/ns, 1996.

[9] Hollot, C.V., Misra, V., Towsley, D., and Gong, W., "A Control theoretic analysis of RED", *Proceedings of INFOCOM*, April 2001, pp1510-1519.

[10] Kuo, B.C., *Automatic Control Systems*, John Wiley & Sons, Inc., seventh edition, 1995.

[11] Misra, A., Ott, T., and Baras, J., "Effect of Exponential Averaging on the Variability of a RED Queue", *Proceedings of ICC*, June 2001, pp1817-1823.

[12] Ziegler, T., "On averaging for Active Queue Management Congestion Avoidance", *Proceedings of ISCC*, July 2002.

[13] Astrom, K. and Hagglund, T., "The Future of PID Control", *Control Engineering Practice*, Vol.9, 2001, pp1163-1175.

[14] Isermann, R., *Digital Control Systems Volumn I: Fundamentals, Deterministic Control*", Springer-Verlag, Second revised edition, 1989.

[15] Franklin, G., Powell, J. and Workman, M., *Digital control of dynamic systems*, Addison-Wesley, third edition, 1998.

[16] Cao, J., Cleveland, S., Lin, D., and Sun, D.X., "On the Nonstationarity of Internet traffic", *Proceedings of ACM SIGMETRICS*, 2001, pp102-112.

[17] Astrom, K. and Hagglund, T., *PID-controller: Theory, Design and Tuning*, Instrument Society of America, second edition, 1995.

[18] Floyd, S., "Notes on testing RED implementation", http://www.iciri.org/floyd/papers/redtesting, 1996.

1) As the traffic load and types of the traffic sources increases, the long-range dependence of the network traffic decreases to independence. Moreover, because of the statistical multiplexing of the traffic sources, at higher traffic rate the packet inter-arrival processes are expected to behave like Poisson with independent service times while at lower rate there exists long-range dependence [16].

2) It has been reported that 90% of Internet traffic is TCP traffic, and 50-70% of TCP traffic is short-lived web-like mice traffic [6].

3) Since the average propagation delay is 120ms, BDP is 450 packets (120ms*30Mbps) and 2*BDP=900 packets. However, to compare AQM algorithms under the same network configuration used in [2], we set the buffer size 800 packets.