AOPC: an Adaptive Optimized Proportional Controller for AQM*

Jianxin WANG, Liang RONG
School of Information Science and Engineering, Central South University, China, 410083
jxwang@mail.csu.edu.cn, csulrong@gmail.com

Abstract

In this paper, we propose a novel AQM algorithm based on the optimized second-order system model for the first time. The algorithm is called Adaptive Optimized Proportional Controller (AOPC). The system design of AOPC is based on classical TCP/AQM interconnection system model, which has a strong relation to the network load level. Through introducing network load estimator in the TCP/AQM interconnection system model, AOPC is capable to detach from the number of TCP sessions N and insensitive to varying network conditions. Meanwhile, AOPC surmounts the parameter configuration difficulties experienced by many AQM algorithms. The parameter tuning rule is compliance with the optimized second-order system model which results fast convergence rate. The performances of AOPC are compared to REM, PI, PID, PIP, and LRED using NS2 simulations. From the experiments and analysis, we conclude that AOPC outperforms other AQM algorithms that it obtains stable queue evolution, achieves the fastest convergence rate to equilibrium point in time-varying network conditions than other algorithms, and fulfills perfect tradeoff between link utilization and queueing delay.

1. Introduction

Design a scalable Active Queue Management (AQM) scheme to co-operate with TCP end-to-end congestion control has been a very interesting research area [1]. In existing AQM schemes, link congestion is estimated through queue length, traffic input rate, packet loss ratio, buffer overflow and emptiness, or a combination of these congestion indicators. Queue length (or average queue length) is widely used in RED [2] and most of its variants [3-5]. The performance of RED has been evaluated by extensive simulations and theoretical analysis. Nearly all of the studies demonstrate that RED is inherent deficient in parameter settings. BLUE [6] adjusts its marking (or dropping) probability upon buffer overflow and link idle events. The traffic input rate is also used in some AQM schemes such as AVQ [7] to make the input rate match the link output rate. Some schemes, for example, REM [8] and PFED [9], use queue length and input rate simultaneously to estimate congestion level. Recently, Hollot *et al*. linearize the dynamic model of TCP behavior [10] about the operation point and obtain a second-order feedback control system; meanwhile a PI controller is designed to regulate the TCP/AQM interconnection system [11, 12]. TCP/AQM interconnection system gives a model for network researchers to design an AQM controller to regulate the system. Based on the model, other schemes, such as PID [13], PIP [14], and LRED [15], are proposed to eliminate the drawbacks existed in PI controller.

In this paper, we have designed a robust AQM scheme, called Adaptive Optimized Proportional Controller (AOPC), to improve the stability and responsiveness. AOPC periodically measures the packet loss ratio and uses the measured packet loss ratio to compute the tuning factor of the control parameter. With this tuning factor, AOPC scheme is capable to adjust the control parameter adaptively. The performance of AOPC scheme is evaluated through extensive simulations under various network configurations. Compared to LRED and existing AQM schemes, such as REM, PI, PID and PIP, AOPC scheme offers more stable control of queue length around the desired queue length, as well as the achievement of high link utilization.

2. Control system model

In this section, we give a description of the TCP/AQM interconnection system model, the optimized second-order system model, and the

*This work is supported the National Natural Science Foundation of China (.90304010) and the Program for New Century Excellent Talents in University (NCET-05-0683)

convergence property of proportional AQM control.

2.1 TCP/AQM interconnection system model

Transient behavior of networks with AQM routers supporting TCP flows was described by a couple of nonlinear ordinary differential equations developed in [10]. These equations are linearized in [11] and the linear TCP/AQM interconnection system can be depicted in Fig.1, where q_0 is the desired queue length, $G_1(s)$ is the AQM controller, $G_2(s)$ is the TCP window-control and queue dynamics.

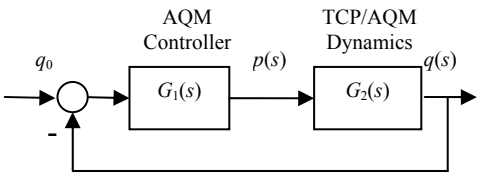


Fig.1. Block diagram of TCP/AQM interconnection system

The objective of the AQM controller is to regulate the queue length to the desired value q_0 by marking (dropping) packets with probability p as a function of measured queue length deviation between instantaneous and desired value. The transfer function of $G_2(s)$ is:

$$G_2(s) = \frac{K_m}{(T_1 s + 1)(T_2 s + 1)}, \tag{1}$$

where

$$K_m = \frac{(RC)^3}{4N^2}, \tag{2a}$$

$$T_1 = \frac{R^2 C}{2N}, \tag{2b}$$

$$T_2 = R. \tag{2c}$$

with

$N \equiv$ load factor (number of TCP sessions)

$R \equiv$ round trip time (RTT)

$C \equiv$ link capacity

Due to the modeling inaccuracies, which are listed in [14], a parameter tuning structure should be provided to correct the simple system model. Moreover, the parameter tuning structure should be insensitive to the drift of system parameters as well.

2.2 Optimized second-order system model

Consider the closed-loop transfer function of the second-order system:

$$G(s) = \frac{K}{\tau^2 s^2 + 2\zeta\tau s + 1}, \tag{3}$$

where K is the static sensitivity, τ is the time constant, and ζ is the damping factor.

The damping factor ζ is vital to the performance of the second-order system [19]. If ζ is a bit large (or small), the settling time becomes very long, which is disadvantage to system control. In engineering, the second-order system is classified into under damping, critical damping, over damping system corresponding to $\zeta<1$, $\zeta=1$, and $\zeta>1$. The second-order system is called optimized system when $\zeta=0.707$. For $\zeta<1$, the step response of the second-order system described by (3) is

$$y(t) = K\left(1 - \frac{1}{\sqrt{1-\zeta^2}} e^{-\zeta\frac{t}{\tau}} \sin\left(\frac{\sqrt{1-\zeta^2}}{\tau} t + \arctan\frac{\sqrt{1-\zeta^2}}{\zeta}\right)\right). \tag{4}$$

From control standpoint, when $t\to\infty$, $y(t)\to 1$, the steady-state error e_{ss} goes to zero. Assume $K\le 1$ is always satisfied in proportional AQM control (In section 2.3, we will see this assumption is consistent valid). Let $y(\infty)=1$, then the steady-state error e_{ss} and the settling time t_s with admissible error set to be 2%, satisfied with $|y(t_s) - y(\infty)| = 0.02 y(\infty)$, are obtained as follows:

$$e_{ss} = 1 - K, \tag{5a}$$

$$t_s \approx \frac{4 - \ln\sqrt{1-\zeta^2}}{\zeta}\tau. \tag{5b}$$

2.3 Convergence property of proportional AQM control

Denote θ to be the control parameter for a proportional AQM controller. Then the closed-loop transfer function of the TCP/AQM control system is:

$$G(s) = \frac{\theta K_m}{T_1 T_2 s^2 + (T_1 + T_2)s + \theta K_m + 1} = \frac{K(\theta)}{\tau^2(\theta)s^2 + 2\zeta(\theta)\tau(\theta)s + 1} \tag{6}$$

In (6), the static sensitivity K, time constant τ, and damping factor ζ are calculated as follows:

$$K(\theta) = \frac{\theta K_m}{\theta K_m + 1}, \tag{7a}$$

$$\tau(\theta) = \sqrt{\frac{T_1 T_2}{\theta K_m + 1}}, \tag{7b}$$

$$\zeta(\theta) = \frac{T_1 + T_2}{2}\sqrt{\frac{1}{(\theta K_m + 1)T_1 T_2}}. \tag{7c}$$

Assume the TCP/AQM interconnection system is an under damping system. We attempt to determine the relation between settling time t_s and control parameter θ.

Lemma 1: If the second-order system is an under damping system, and $\forall\theta\in R^+$, $K(\theta)<1$ is always satisfied, where $K(\theta)$ is defined as given in (7a), then

the settling time t_s of the second-order system in (5b) is a decreasing function of θ.

Proof: From (7a), (7b) and (7c), we obtain

$$\theta K_m + 1 = \frac{1}{1-K}, \quad \frac{\tau\zeta}{T_1+T_2} = \frac{1}{2(\theta K_m+1)} = \frac{1-K}{2}.$$

Then, write the derivative of function t_s about θ as:

$$\frac{\mathrm{d}t_s}{\mathrm{d}\theta} = \frac{\partial t_s}{\partial \tau}\frac{\mathrm{d}\tau}{\mathrm{d}\theta} + \frac{\partial t_s}{\partial \zeta}\frac{\mathrm{d}\zeta}{\mathrm{d}\theta},$$

where

$$\frac{\partial t_s}{\partial \tau} = \frac{4 - \ln\sqrt{1-\zeta^2}}{\zeta},$$

$$\frac{\mathrm{d}\tau}{\mathrm{d}\theta} = -\frac{T_1T_2K_m}{2(\theta K_m+1)\sqrt{T_1T_2(\theta K_m+1)}} = -\frac{K_m\tau^2\zeta}{T_1+T_2} = -\frac{K_m(1-K)}{2}\tau$$

$$\frac{\partial t_s}{\partial \zeta} = \left(\frac{\zeta^2 + (1-\zeta^2)\ln\sqrt{1-\zeta^2}}{(1-\zeta^2)\zeta^2} - \frac{4}{\zeta^2}\right)\tau,$$

$$\frac{\mathrm{d}\zeta}{\mathrm{d}\theta} = -\frac{(T_1+T_2)K_m}{4(\theta K_m+1)\sqrt{T_1T_2(\theta K_m+1)}} = -\frac{K_m\zeta(1-K)}{2}.$$

hence,

$$\frac{\mathrm{d}t_s}{\mathrm{d}\theta} = \frac{\partial t_s}{\partial \tau}\frac{\mathrm{d}\tau}{\mathrm{d}\theta} + \frac{\partial t_s}{\partial \zeta}\frac{\mathrm{d}\zeta}{\mathrm{d}\theta} = -\frac{K_m\left[\zeta^2 + 2(1-\zeta^2)\ln\sqrt{1-\zeta^2}\right](1-K)}{2\zeta(1-\zeta^2)}\tau$$

From (7a), we can see that $K<1$ is consistent valid. Because the system is an under damping system, the damping factor ζ satisfies $0<\zeta<1$. It is clear that $\forall\zeta\in(0,1),\ \zeta^2 + 2(1-\zeta^2)\ln\sqrt{1-\zeta^2} > 0$. So we can get $\frac{\mathrm{d}t_s}{\mathrm{d}\theta} < 0$ and t_s is a decreasing function of θ. □

3. The AOPC scheme

3.1 AOPC description

AQM schemes need to maintain closed-loop performance in face of varying network conditions. These conditions include variations in the number of TCP sessions N and TCP average round trip time R, and the introduction of short-lived flows into the queue. Due to the lifetime of a TCP session remains an unknown to a network router and nonidentical TCP sessions have various lifetimes, the number of TCP sessions varies in a large range. Therefore, it is hard to count the number of TCP sessions directly. However, in the TCP/AQM interconnection system model, the system closed-loop performance has a strong relation to these unknown network state variables.

The motivation of AOPC is to detach the correlation between control parameter and network state variables so as to provide an efficient and flexible mechanism for queue management. The AOPC scheme employs proportional AQM control to calculate packet drop probability. It measures packet loss ratio in a large time scale and updates packet drop probability in a small time scale upon each packet arrival, like LRED. Different from LRED, AOPC tunes its control parameter adaptively according to the packet loss ratio measured in a large time scale to surmount the drawbacks in LRED. Therefore, the packet drop probability in AOPC is as follows:

$$p = \overline{l(k)} + \gamma(q - q_0), \tag{8}$$

where γ is a variable parameter suitable to current network conditions, $\overline{l(k)}$ is the measured packet loss ratio.

AOPC estimates the number of TCP sessions N based on the TCP throughput formula [16], which takes the stable packet drop probability p_0 as an input variable. The key assumption in our design is that the measured packet loss ratio $\overline{l(k)}$ can be used to approximate the stable packet drop probability p_0, that is, $\overline{l(k)} \approx p_0$. Hence, the number of TCP sessions can be estimated after the latest measured packet loss ratio is obtained. The TCP/AOPC interconnection control system is depicted in Fig. 2. AOPC has two components: 1) Network Load Estimator. It estimates the average number of TCP sessions at the end of the latest packet loss ratio measurement period so as to detach the correlation between the control parameter and the number of TCP sessions N; 2) Parameter Optimization Module. It optimizes the TCP/AOPC interconnection system based on the optimized second-order system model.

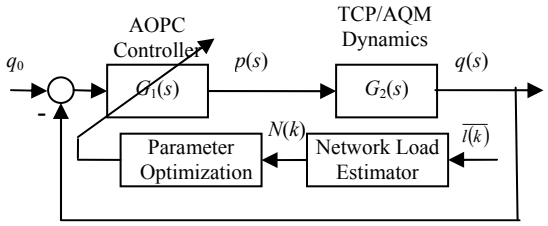


Fig.2. Block diagram of TCP/AOPC interconnection system

(1) Network Load Estimator

A single TCP flow, which experiences packet drop probability p_0, attains the throughput roughly as follows [16]:

$$x = \frac{1}{R}\sqrt{\frac{2}{3p_0}}, \tag{9}$$

where R is the round trip time of the TCP flow.

Now consider a link shared by N flows. Let $y=\Sigma x_i$ (i=1, ..., N) be the total sending rate. Suppose the link has the service rate (link capacity) C and the buffer is large enough to keep the link fully utilized. Clearly the

total sending rate is larger than the service rate, *i.e.* $y>C$, so the drop probability p_0 satisfies

$$(1-p_0)y = C \,. \tag{10}$$

Then, from (9) and (10), the drop probability p_0 is the solution to

$$\sum_{i=1}^{N} \frac{1}{R_i}\sqrt{\frac{2}{3p_0}}(1-p_0) = C \,. \tag{11}$$

Denoting

$$\frac{1}{R_{eq}} = \frac{1}{N}\sum_{i=1}^{N}\frac{1}{R_i}, \tag{12}$$

where R_{eq} is the harmonic mean of the individual round trip times of the flows. In [11], the R_{eq} is interpreted as the equivalent round trip time of the flows, which can be viewed equivalently to R in TCP/AQM model described by Fig.1. Then the system behaves in the mean as a system with N flows each having an identical equivalent round trip time that is R_{eq}.

From (11) and (12), we obtain

$$N = R_{eq}Cf(p_0), \tag{13}$$

where

$$f(p_0) = \frac{\sqrt{3p_0/2}}{1-p_0}, \tag{14}$$

here, we name $f(p_0)$ as a tuning factor for calculating packet drop probability. According to previous assumption that the measured packet loss ratio $\overline{l(k)}$ can be used to approximate the stable packet drop probability p_0, we can rewrite (14) as follows:

$$f(p_0) = f\left(\overline{l(k)}\right) = \frac{\sqrt{3\overline{l(k)}/2}}{1-\overline{l(k)}}. \tag{15}$$

Equation (13) illustrates that the number of TCP sessions N is relative to the harmonic mean R_{eq} of the RTTs yet not to individual RTTs. If the harmonic mean value R_{eq} is a known variable and varies slightly around a stable value, we can estimate the number of TCP sessions N by ignoring the time variant property of R_{eq}. Recent Internet measurements [17] report that roughly 75%~90% of flows have RTTs less than 200ms and the average RTT is distributed around 180ms [18]. These related researches are more likely to suggest an alternate way to improving TCP performance and AQM design.

(2) Parameter Optimization Module

By approximating the stable packet drop probability p_0 as the latest measured packet loss ratio $\overline{l(k)}$, we obtain the following formulas from (2a), (2b), (2c), (13) and (15):

$$K_m = \frac{R_{eq}C}{4f^2\left(\overline{l(k)}\right)}, \tag{16a}$$

$$T_1 = \frac{R_{eq}}{2f\left(\overline{l(k)}\right)}, \tag{16b}$$

$$T_2 = R_{eq}. \tag{16c}$$

Thus, the TCP and queue dynamics transfer function $G_2(s)$ is largely simplified.

```
/* Initialization */
arrPktNum=0;      dropPktNum=0;
allArrNum=0;      allDropNum=0;
index=0; mw=0.1; mp=1.0; M=4;
rtt=0.18; γ=0.0001;
/* LossRatioMeasure()-Called periodically every mp
   seconds */
1   dropNum[index]=dropPktNum;
2   arrNum[index]=arrPktNum;
3   arrPktNum=0;
4   index++;
5   if (index==M) index=0;
6   for (i=0; i<M; i++) {
      /* calculate all dropped and arrived packet number in
         latest M measurement periods */
7       allDropNum+=dropNum[i];
8       allArrNum+=arrNum[i];
9   }
    /* calculate packet loss ratio, see as equation (11) */
10  lossRatioTemp=allDropNum/allArrNum;
11  lossRatio=lossRatioTemp*(1-mw) + lossRatio*mw;
12  allDropNum=0; /* reset the counter for all drops */
13  allArrNum=0;   /* reset the counter for all arrivals */
14  Call UpdateGamma() procedure;
/* UpdateGamma()-Called after packet loss ratio
   measurement finished each time */
1   calculate tuning factor f according to equation (15);
2   update γ according to equation (17);
/* Enque()-Called at each packet arrival */
1   arrPktNum++;   /* count the arrival packet */
2   p=lossRatio+γ*(q-q0);   /* update drop probability */
3   p=max(0, min(1, p));
4   random=uniformRandom(0, 1);
5   if (buffer is full) {
    /* drop packet due to buffer overflow */
6       Drop the packet;
7       dropPktNum++;   /* count the dropped packet */
8   } else if (random>p) { Enqueue the packet;
9   } else {
10      Drop the packet;
11      dropPktNum++;   /* count the dropped packet */
12  }
```

Fig.3. Pseudo code of AOPC

Now, the design rule for designing a stabilizing proportional controller to stabilize the TCP/AOPC interconnection system is given in theorem 1.

Theorem 1: Denoting γ to be the control parameter of AOPC. If

$$\gamma < \frac{f\left(\overline{l(k)}\right)\left|1+4f^2\left(\overline{l(k)}\right)\right|}{R_{eq}C}, \tag{17}$$

the linear feedback control system in Fig.2 using $G_1(s)=\gamma$ is asymptotic stable and the system is an optimized system.

Proof: From (16a), (16b) and (16c), we can obtain

$$\gamma = \frac{f\left(\overline{l(k)}\right)\left[1+4f^2\left(\overline{l(k)}\right)\right]}{R_{eq}C} = \frac{T_1^2+T_2^2}{2K_mT_1T_2}. \quad (18)$$

Replacing θ in (7c) with γ in (18), we obtain

$$\zeta = \frac{T_1+T_2}{2}\sqrt{\frac{1}{(\gamma K_m+1)T_1T_2}} = 0.707.$$

According to the optimized second-order system model discussed in section II, when ζ=0.707, the system is an optimized second-order system, and it is asymptotic stable. So, TCP/AOPC interconnection system is asymptotic stable and the system is an optimized system. □

Compared to the stability condition for LRED (see [15], Theorem 2), AOPC scheme maintains the closed-loop performance in face of varying network conditions. Meanwhile, AOPC simplifies the tuning method and makes it be scalable to be deployed in Internet routers. The pseudo code of AOPC scheme is illustrated in Fig.3. For AOPC implementation, we can track packet departure to obtain link service rate C. From (17), we can see that the harmonic mean of individual RTTs, R_{eq}, contributes a very small weight in control parameter γ. For example, assume the current measured packet loss ratio $\overline{l(k)} = 0.01$, the link capacity is 2500packets/s, when R_{eq}=0.2s, the required AOPC parameter γ is $2.625(10)^{-4}$; while, when R_{eq}=0.15s, the required AOPC parameter γ is $3.5(10)^{-4}$. The example shows that the control parameter γ is insensitive to RTT variation in real network condition.

3.2 Analysis of dynamic performance index

In this section, we will explain why AOPC is more responsive than LRED on the basis of second-order system analysis.

Since both AOPC and LRED are proportional controllers, let θ_a and θ_l denoting control parameter for AOPC and LRED respectively. Then,

$$\theta_a = \gamma = \frac{f\left(\overline{l(k)}\right)\left(1+4f^2\left(\overline{l(k)}\right)\right)}{R_{eq}C} \quad \text{and} \quad \theta_l = \beta\sqrt{\overline{l(k)}}.$$

Ordinarily, the packet loss ratio is very small (close to zero), *i.e.* $\overline{l(k)} << 1$. Thus

$$f\left(\overline{l(k)}\right) = \frac{\sqrt{3\overline{l(k)}/2}}{1-\overline{l(k)}} \approx \sqrt{3\overline{l(k)}/2}.$$

Then we can obtain

$$\frac{\theta_a}{\theta_l} = \frac{\sqrt{\frac{3}{2}}\left(1+6\overline{l(k)}\right)}{\beta R_{eq}C}. \quad (19)$$

Considering the following network conditions: R_{eq}=0.18s, C=2500packets/s, $\overline{l(k)} = 0.01$. According to [15], β=0.001. Substituting these values to (19), we obtain $\frac{\theta_a}{\theta_l} = 2.885$, that is $\theta_a > \theta_l$. Generally, in most network situations, $\theta_a > \theta_l$ is valid.

Suppose TCP/LRED remains an under damping system. Call back for Lemma 1, we derive that the settling time of AOPC is smaller than that of LRED; therefore, AOPC convergences to stable state faster than LRED.

4. Performance evaluation

In this section, we investigate the performance of AOPC through NS simulations. We also compare its performances with existing AQM schemes, in particular, REM, PI, PID, PIP, and LRED. The settings of the parameters for various AQM schemes are based on their authors' recommendations.

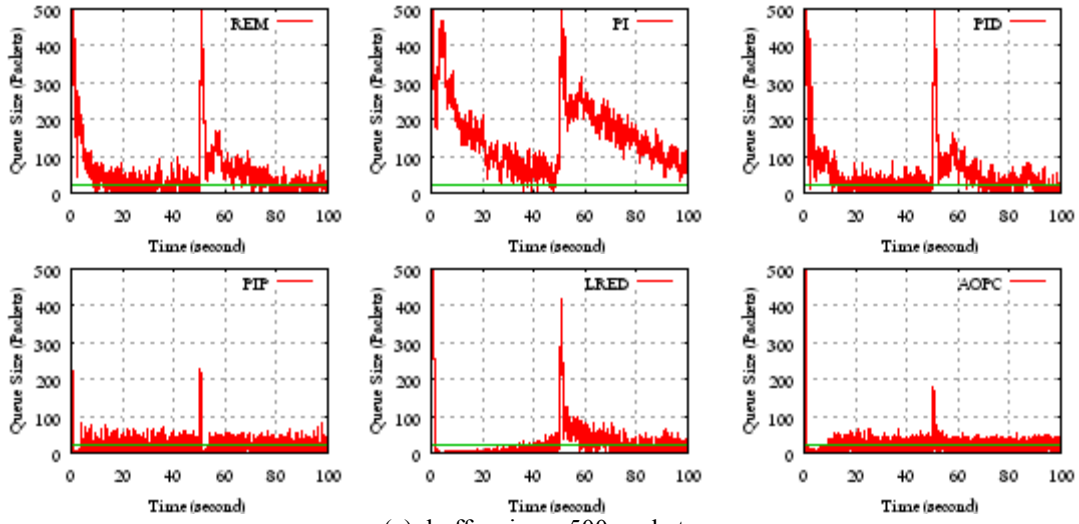
The network topology for simulation is the commonly used dumbbell topology with a bottleneck link capacity of 10 Mbps and a mean packet size of 500 bytes. Round trip propagation delays are uniformly distributed over the range [60, 220]ms.

The desired queue length is set to be 20 packets. To imitate real network situations, we adopt four ordinary traffic types, *i.e.*, infinite FTP flows and burst HTTP flows based on TCP Reno, CBR flows and exponential ON/OFF flows based on UDP. Among them, FTP flows always have data to send during simulation runtime. In contrast to long-lived FTP flows, HTTP flows are short-lived with an average page size of 1000B and an average request interval of 1s. The packet size of CBR flows is 500B and the sending interval is 0.08s. The burst and idle times of the ON/OFF service model are 2s and 1s respectively, and the sending rate during "ON" duration is 64Kbps. The total simulation last for 100s. Unless otherwise specified, the buffer size in bottleneck router is 200 packets.

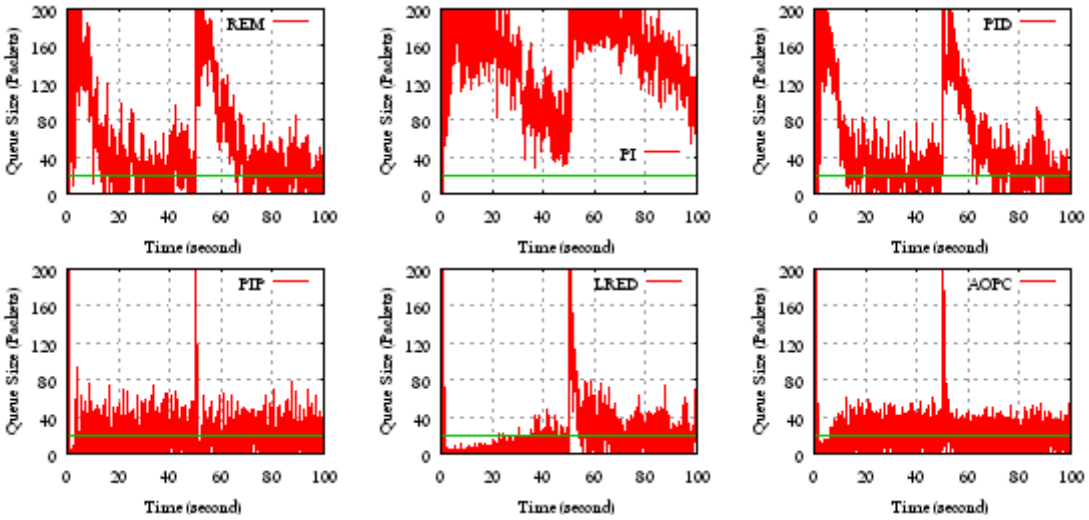
A. Experiment 1: Long-lived FTP flows only

In this experiment, the stability and responsiveness of the AQM schemes are investigated under two extreme cases: 1) buffer size with 500 packets, 2) buffer size with 200 packets. In both cases, the number of FTP flows is 100 at the beginning and 100 additional FTP flows arrival at the link 50 seconds later. The queue lengths for the six algorithms with two different buffer sizes are depicted in Fig.4 (a) and Fig.4 (b) respectively.

It can be seen that, the queue length of REM, PI and PID algorithms can fall sharply in the first case; however, in the second case, their queue length hit the buffer top for a longer time. Besides, the queue evolution of REM bears much resemblance to that of



(a): buffer size = 500 packets



(b): buffer size = 200 packets

Fig.4. Exp 1: Evolution of the queue length with only FTP flows under two different buffer sizes

PID, which results from that REM is a PID-type controller in essence. In both the cases, PIP, LRED, and AOPC achieve shorter response times and better stabilities than REM, PI, and PID. Moreover, PIP, LRED, and AOPC are independent of buffer sizes, but LRED is too aggressive that it makes the buffer empty for a long time. AOPC has less overshoots and smaller queue deviations. On the contrary, the queue length of PIP and LRED oscillate in a larger range.

B. Experiment 2: Adding CBR flows

In this experiment, we use a mixture of FTP and CBR flows and remove all time varying dynamics. The number of FTP flows and the number of CBR flows are 150 and 40 respectively. The queue evolutions are plotted in Fig.5.

The simulation results show that PIP, LRED and AOPC outperform REM, PI and PID in terms of queue dynamics. Moreover, AOPC has a better queue stability than PIP and LRED with whose queue length has a smaller oscillation.

C. Experiment 3: Adding HTTP flows and ON-OFF flows

At last, we consider a more realistic, highly dynamic scenario. The traffic is a mixture of FTP, HTTP, CBR, and ON-OFF flows. The queue evolutions are depicted in Fig.6. The simulation results show that the queue lengths of REM, PID and LRED oscillate along with the dynamics of load levels. The

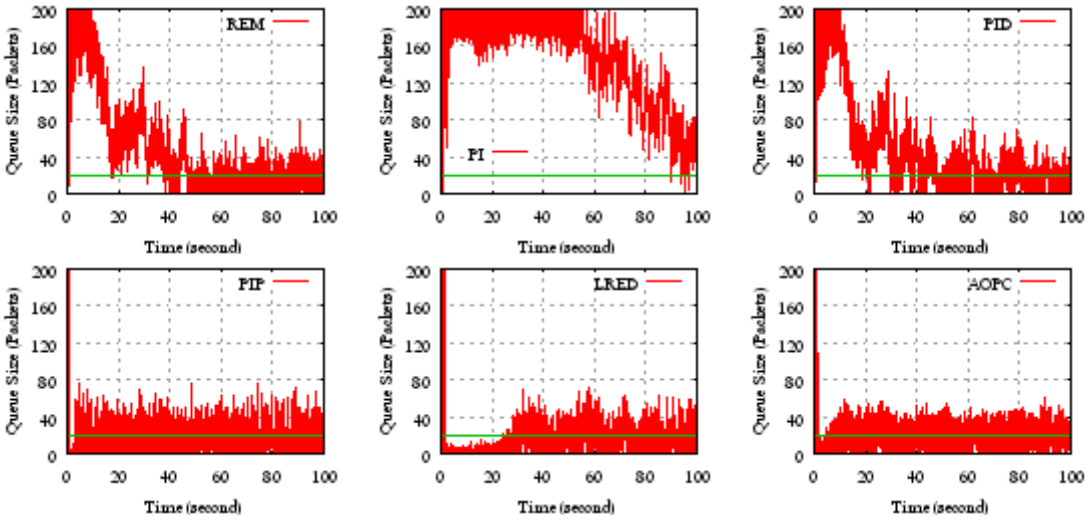


Fig.5. Exp2: Evolution of queue length with a mixture of FTP flows and CBR flows

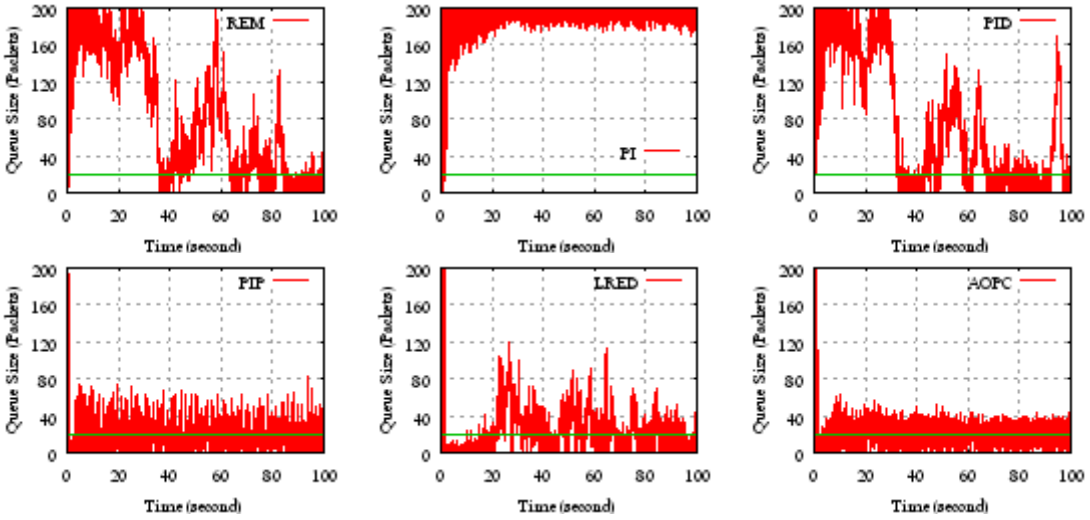


Fig.6. Exp3: Evolution of queue length with hybrid flows

queue length of PI never falls down in the whole simulation runtime. On the other hand, the queue changes in PIP and AOPC are small. Moreover, AOPC maintains more stable control of queue evolution.

In previous simulations, the queue lengths with AOPC are very low. So it is reasonable to doubt the link is under utilized. Here, we continue to consider the hybrid traffic situation, consists of FTP, HTTP, CBR and ON-OFF flows. The link utilizations versus end-to-end delay are illustrated in Fig.7. Since the queue length of PI hit the buffer top in whole simulation runtime, the performance of PI is not considered here and after. AOPC, which is located in the left-upper corner in Fig.7, maintains perfect tradeoff between link utilization and end-to-end delay.

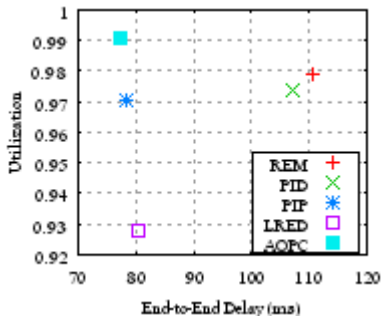


Fig.7. Exp3: Link utilization vs. end-to-end delay under hybrid flows

Additionally, we also compare the standard queue

deviations and averages queue length except PI. Fig.8 depicts the standard queue deviations versus averages queue length, where PIP is covered by AOPC. Here, we can observe that AOPC and PIP obtains low queue deviation. Combine Fig.7 and Fig.8, we can draw the conclusion that small queue oscillations not only indicate low delay jitter but also a guarantee of high link utilization. All of the simulations in the previous sections demonstrate that AOPC can restrain queue oscillations, which means AOPC can achieve high link utilization.

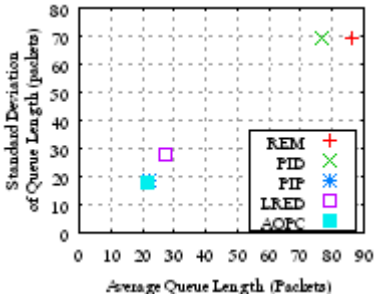


Fig.8. Exp3: Standard queue deviation vs. average queue length under hybrid flows

5. Conclusion

In this paper we present a novel AQM scheme called AOPC, which can be implemented easily and works efficiently. AOPC employs the optimized second-order system model for the first time to adjust control parameter adaptively. The performance of AOPC is evaluated by simulations and compared with REM, PI, PID, PIP, and LRED. The performance analysis and simulation results show that AOPC is superior to all the compared AQM algorithms. Through tuning control parameter dynamically, the stability and responsiveness are greatly improved. Moreover, the sensitivity to system parameter variations is also alleviated. By applying the optimized second-order system model to system design, AOPC can keep the queue length near the desired queue length with small oscillations under widely various traffic conditions. Our approach to AQM is simple and straightforward. More sophisticated controllers can be designed using other advanced control theory.

References

[1] B. Braden, D. Clark, and J.Crowcroft. Recommendations on queue management and congestion avoidance in the Internet. IETF RFC 2309, April 1998.

[2] S. Floyd, V. Jacobson. Random early detection gateways for congestion avoidance, *IEEE/ACM Transactions on Networking*, vol. 1, no. 4, pp. 397-413, August 1993.

[3] S. Floyd. Recommendations on using the gentle "variant of RED". *http://www.aciri.org/floyd/gentle.html*, March 2000.

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

[5] T. Ott, T. Lakshman, L. Wong. SRED: Stabilized RED. In *Proceedings of IEEE INFOCOM*, pp. 1346-1355, New York, March 1999.

[6] W. Feng, D. D. Kandlur, and D. Saha. The blue active queue management algorithms. *IEEE/ACM Transactions on Networking*, vol. 10, no. 4, pp. 513-528, August 2002.

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

[8] S. Athuraliya, S. Low, and V. Li. REM: active queue management. *IEEE Network Magazine*, vol. 15, pp. 48-53, May 2001.

[9] Wenyu Gao, Jianxin Wang, and Jianer Chen. PFED: a prediction-based fair active queue management algorithm. In *Proceedings of IEEE ICPP*, pp. 485-491, Los Alamitos, June 2005.

[10] V. Misra, Wei-Bo Gong, and Don Towsley. Fluid-based analysis of a network of AQM routers supporting TCP flows with an application to RED. In *Proceedings of ACM SIGCOMM*, pp. 151-160, Stockholm, Sweden, August 2000.

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

[12] C. Hollot, V. Misra, and D. Towsley. On designing improved controllers for AQM routers supporting TCP flows. In *Proceedings of IEEE INFOCOM*, pp. 1726-1734, Anchorage, Alaska, USA, April 2001.

[13] Fengyuan Ren, Fubao Wang, and Yong Ren. PID controller for active queue management. *Journal of Electronics and Information Technology*, China, vol. 25, no. 1, 2003.

[14] Heying. Zhang, Baohong Liu, and Wenhua Dou. Design of a robust active queue management algorithm based on feedback compensation. In *Proceedings of ACM SIGCOMM*. pp. 265-276, Kalsruhe, 2003.

[15] C. Wang, B. Li, and Y. Thomas Hou. LRED: A robust active queue management scheme based on packet loss ratio. In *Proceedings of IEEE INFOCOM*, pp. 1-12, Hongkong, 2004.

[16] J. Padhye, V. Firoiu, and D. Towsley. Modeling TCP throughput: A simple model and its empirical validation. In *Proceedings of ACM SIGCOMM*, pp. 304-314, Vancouver, August 1998.

[17] H. Jiang, C. Dovrolis. Passive estimation of TCP round-trip times. *ACM Computer Communications Review*, vol. 32, no. 3, pp. 75-88, September 2001.

[18] Srinivas Shakkottai, R. Srikant, and Nevil Brownlee. The RTT distribution of TCP flows in the Internet and its impact on TCP-based flow control. *http://www.caida.org /outreach /papers/2004/tr-2004-02/tr-2004-02.pdf*, 2004.

[19] Qi Wu. Automatic control theory. *Tsinghua University Press*, China, 1990.