Design a Robust Controller for Active Queue Management in Large Delay Networks

Fengyuan Ren Chuang Lin Bo Wei

Abstract—AQM (Active Queue Management) can maintain the smaller queuing delay and higher throughput by the purposefully dropping the packets at the intermediate nodes. Almost all the existed schemes for AQM neglect the impact of large delay on performance. In this study, we firstly verify a fact through simulation experiments, which is the queues controlled by the several popular AQM schemes, including RED, PI controller and REM, appear the dramatic oscillations in large delay networks, which decreases the utilization of the bottleneck link and introduces the avoidable delay jitter. After some appropriate model approximation, we design a robust AQM controller to compensate the delay using the principle of internal mode compensation in control theory. The novel algorithm restrains the negative effect on queue stability caused by large delay. The simulation results show that the integrated performance of the proposed algorithm is obviously superior to that of several well-known schemes when the connections have large delay, at the same time, the buffers keep at small queue length.

Index Terms—*Active queue management, delay, stability, internal mode princile*

I. NTRODUCTION

TCP flow control is the most important mechanism for congestion control in IP networks. Since V. Jacobson invented the end-to-end flow control scheme in 1988, there have been so many enhanced and improved versions, such as Tahoe, Reno, New Reno [1], SACK [2] and Vegas [3] etc, these works merely pay attention on the end system. The recent research begins to explore how to make the intermediate node contribute to congestion avoidance because there is a limit to how much control can be accomplished at end system. It is needed to implement some measures in the intermediate nodes to complement the end system flow control mechanisms. ECN (Explicit Congestion Control) is an effective effort [4]. Active Queue Management, as one class of packet dropping/marking mechanism in the router queue, has been recently proposed to support the end-to-end congestion control in the Internet [5]. It has been a very active research area in the Internet community.

This work was supported in part by NSFC (No. 60273009, No.60218003), the Projects of Development Plan of the State Key Fundamental Research (No. 2003CB314804, G1999032707) and Intel IXA University Research Plan (No. 9077).

Fengyuan Ren is with Computer Science and Technology Department in Tsinghua University, Beijing, China; e-mail: renfy@ csnet1.cs.tsinghua.edu.cn

Chuang Lin is with Computer Science and Technology Department in Tsinghua University, Beijing,100084, China

Wei Bo is with Department of Electrical Engineering, Southern Methodist University, Dallas, TX 75275, U. S. A.

The goals of AQM are to (1) reduce the average length of queue in routers and thereby decrease the end-to-end delay experienced by packets, and (2) ensure the network resources to be used efficiently by reducing the packet loss that occurs when queues overflow. AQM highlights the tradeoff between delay and throughput. By keeping the average queue size small, AQM will have the ability to provide greater capacity to accommodate nature-occurring burst without dropping packets, at the same time, reduce the delays experienced by flow, this is very particularly important for real-time interactive applications. RED [6] was originally proposed to achieve fairness among sources with different burst attributes and to control queue length, which just meets the requirements of AQM, thus RED was recommended as only candidature algorithm for AQM in RFC 2309. However, many subsequent studies verified that RED has two main flaws, namely instability and unfairness under some network environment. In order to work around these problems existed in RED algorithm, numerous variants of RED and novel schemes have been proposed. Among them, Balanced RED [7] and FRED [8] are for fairness. Most of literatures mainly care about the stability. M. Christiansen et al. [9] investigated it with experiment approach, concluded that tuning of RED for stable operation is a difficult job. The theory analysis about RED stability also absorbed much attention. In [10], V. Firoiu and M. Borden modeled the relationship between queue length and dropping probability, illustrated why RED could led queue oscillations; C. Hollot et al [11] deduced a proposition about RED stability based on the linear control theory. Both of them agreed that number of connections, i.e. network load, and capacity of bottleneck link are dominative causes of RED unstable operations. As modifications, SRED [12] uses a list called as *zombie* to estimate the number of active TCP sessions, then updates the packet dropping probability, although it is no longer sensitive to load, the estimation to active sessions is too coarse; In order to adaptively adjust the dropping probability, Self-Configuring Gateway [13] employs the queue length as a prediction to load status, but it seems to be a bit rough. Except for RED variants, some new mechanisms, such as BLUE [14] and GKVQ [15] etc., were also introduced for AQM. In the view of design approach, all above schemes excessively depend on the intuition, are heurist algorithms. The partial simulations and experiments on the special network configuration is only measure to validate the algorithm performance, and the systematic and theoretic analysis and evaluation are neglected. Thus we lack the whole understanding about the algorithm performance and efficiency. Once the problems occur in practice, the remediation was made through

new simulations or experiments. For instance, the original paper about RED presented the impact of the configuration parameter on the performance, and suggested the guidelines towards the appropriate values based on heuristic and simulation. Subsequently, some further investigations found that they were not optimal, and then gave the current suggestions [16], in which w_q (weighted factor) is increased from 0.001 to 0.002 because w_q is too low to timely detect the congestion in relative high speed network; max_p (maximum packet dropping probability) is set to 0.1 instead of to 0.02 because 2% drop probability is not enough to force multiple TCP sources to sufficiently reduce their window sizes, especially in many short lived and burst HTTP sessions. It is likely that the current parameter settings would be inappropriate again when the speed of Internet backbone upgrades to terabit magnitude. It is well known that the intuition and partial experience are not always scientific and reasonable under any conditions. The algorithm designed based on theory analysis and evaluation should be more reliable than one originated from intuition. C. Hollot designed the PI controller for AQM [17] based linear control theory, and S. Kunniyur and R. Srikant analyzed the stability of Adaptive Virtual Queue (AVQ) scheme [18]. Kelly et al. constructed a unified framework [19] for optimizing flow control. The problem of congestion control was formulated as a convex program, with the aggregate source utility being maximized subject to bandwidth constrain. REM [20] is a controller designed in a dual formulation to obtain optimal source rates. These efforts are very valuable for comprehensively understanding the performance of algorithms served for resource management mechanisms in networks. So far, almost all existed implementation algorithms for AQM schemes neglected a fact, which the delay, especially large delay, readily causes an unstable operation in router queue. In this paper, we will pay attention on the impact of large delay on the performance of AQM algorithms, and intend to compensate the inevitable delay including in AQM system using internal mode compensation principle in control theory. The reminder of the paper is organized as follows. In Section II, we evaluate the impact of large delay on the performance of popular AQM schemes through simulation experiments. Section III develops the DC-AQM algorithm, and presents design guidelines. In the next Section, we testify the validity of DC-AQM algorithm, and then compare its performance with that of popular AQM schemes by numerical simulation results. Finally the conclusion is drawn in Section V.

II. AQM STABILITY IN LARGE DELAY NETWORKS

When evaluating the performance in most of literatures about AQM, RTTs were set at several microseconds, which is suspectable. In order to investigate the performance of several typical AQM algorithms in large delay networks, we did simulations on *ns2* platform using the dumbbell topology depicted in Fig.1.

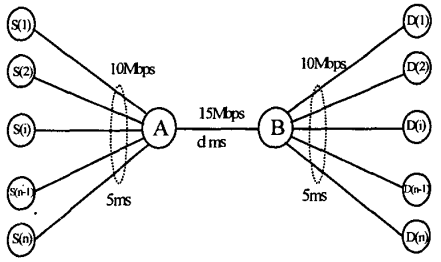

Fig.1 Simulation Network Topology

The only bottleneck link lie between node ***A*** and node ***B***, its capacity is 15Mbps (3750 packets/sec, default packet size is 500bytes), and delay is d ms. The other links have 10Mbps capacity and 5ms delay. All sources are greedy sustained FTP applications. Let d = 30 and 190 respectively represent LAN and WAN. The size of all buffers is 300 packets. Queue ***B*** is Drop Tail. Queue ***A*** controlled by the popular AQM schemes, respectively selecting RED, PI controller and REM during experiments. For RED, the high and low thresholds are 100 packets and 200 packets respectively. The expected queue length was set to 150 packets in others algorithms. The results were presented in Fig 2 to Fig.4. Obviously, all queues controlled by the popular AQM schemes drastically oscillate in large delay networks. On the one spectrum, queue oscillation with great magnitude increases end-to-end jitter; on the other spectrum, empty queue frequently appears, which badly lows the link utilization. These are inconsistent with the objectives of AQM [5]. Table 2 summaries performance and statistic of queue length. Generally, after delay increasing, the bottleneck link utilization decrease.

Observing the Fig.3, although PI controller still keeps the relative high utilization, the queue is unstable. In RED and PI controller, the standard deviation of queue length increases 40, it becomes worse in REM, arrives to 98.01,which is unfavorable to QoS guarantee implementation for special real-time service.

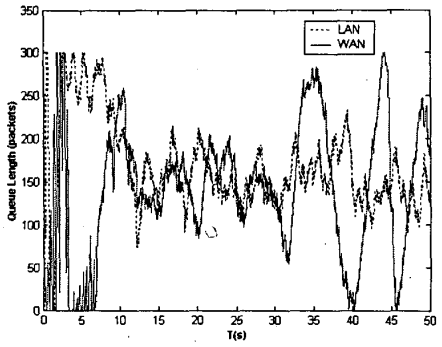

Fig. 2 RED

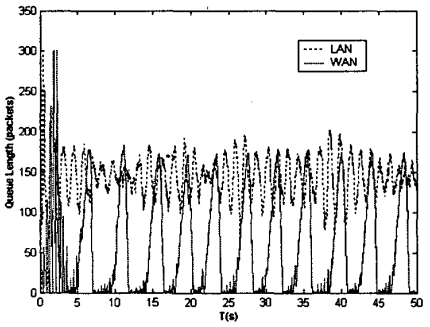

Fig. 3 PI Controller

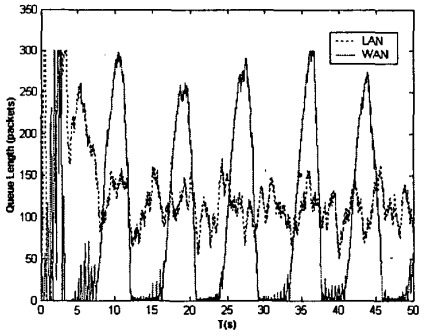

Fig. 4 REM

TABLE I. STATISTICS OF PERFORMANCE

Scheme	RED		PI		REM	
	LAN	WAN	LAN	WAN	LAN	WAN
Queue Length	142.95	72.98	146.52	147.28	109.51	113.94
Standard Deviation	19.89	61.15	20.95	64.62	23.57	98.01
Link Utilization	99.9%	91.6%	99.4%	99.3%	99.3%	90.1%

III. MODEL

A. Flow Control Model

In [21], a non-linear dynamic model for TCP flow control was developed based on fluid-flow theory. The following is a simplified version of that model.

$$\begin{cases} \dfrac{dW(t)}{dt} = \dfrac{1}{R(t)} - \dfrac{W(t)W(t-R(t))}{2R(t)} p(t-R(t)) \\ \dfrac{dq(t)}{dt} = \dfrac{N(t)}{R(t)} W(t) - C(t) \end{cases} \quad (1)$$

Subsequently, C. Hollot et al approximate this nonlinear and time-varying dynamic as a linear constant system by small-signal linearization about an operating point [22]. The Fig.5 illustrates this linearized model. The details about modeling and linearization can be seen in [21] and [22]. In the block diagram, C(s) is the compensator or controller, namely AQM algorithm. G(s) is the plant that we are trying to control. The meanings of parameters presented in Fig.5 are following:

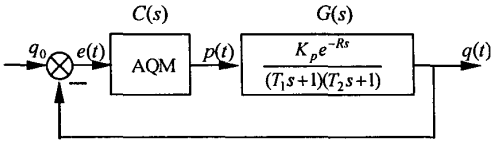

Fig. 5 Block Diagram of AQM System

$$K_p = \frac{(RC)^3}{4N^2} \qquad T_1 = R \qquad T_2 = \frac{R^2 C}{2N}$$

where C denotes link capacity, N denotes number of active sessions, and R is round trip time.

B. First Order Approximation toFlow Control Model

In industry process control, most of plants can be approximately modeled as a first order system with delay. For this model, there are many mature and well-known approaches to design the excellent controller. Thus, we firstly need to fit the system depicted in Fig. 5 to a first order system with delay, namely, to find parameter relationship between system (2) and (3) when they are equivalent.

$$G_p(s) = \frac{K_p e^{-Rs}}{(T_1 s+1)(T_2 s+1)} \quad (2)$$

$$\hat{G}_p(s) = \frac{Ke^{-Ls}}{Ts+1} \quad (3)$$

Calculate the first and second derivatives of (2) and (3) at s=0 point, yield:

$$G_p'(0) = -(R+T_1+T_2)$$

$$G_p''(0) = (R+T_1+T_2)^2 + T_1^2 + T_1^2$$

$$\hat{G}_p'(0) = -(L+T)$$

$$\hat{G}_p''(0) = (L+T)^2 + T^2$$

Let $\hat{G}_p(0) = G_p(0)$, $\hat{G}_p'(0) = G_p'(0)$, $\hat{G}_p''(0) = G_p''(0)$,

We have:

$$T = \sqrt{T_1^2 + T_2^2} \tag{4}$$

$$Kp = K \tag{5}$$

$$L = T_1 + T_2 + R - \sqrt{T_1^2 + T_2^2} \tag{6}$$

Thus, the TCP flow control model can be approximated as a first order system with delay:

$$\hat{G}_p(s) \approx \frac{K_p e^{(-T_1 - T_2 - R + \sqrt{T_1^2 + T_1^2})s}}{\sqrt{T_1^2 + T_2^2}\, s + 1} \tag{7}$$

To verify its correctness, given that a set of typical network configuration parameters, N=60, C=3570 packets/sec (default packet size is 500 bytes), R=400ms, we have:

$$G_p(s) = \frac{2.34 \times 10^5 \times e^{-0.4s}}{(0.4s+1)(5.0s+1)}$$

$$\hat{G}_p(s) = \frac{2.34 \times 10^5 \times e^{-0.784s}}{5.016s+1}$$

IV. ALGORITHM DESIGN

A. Internal Mode Control

The large delay system is always a focus in control theory. The Smith predictor is an effective and prevail delay compensator, but it has a fatal flaw, which is excessively sensitive to model mismatch. For varying-time system and inaccurate model, such as flow control system in network, the Smith predictor is sure to be helpless. As a modification to the Smith predictor, Garcia and Morari put forward Internal Mode Control (IMC) [23]. IMC overcomes the Smith predictor's drawback to depending on the accurate model, and enhances robustness and ability against disturbance, at the same time, the design is more straightforward, in addition to, the controller has very simple structure. Based on the assumed plant model G_m and the given filter f for a reasonable tradeoff between performance and robustness, the IMC design procedure includes two steps.

Firstly, factor the reference model G_m into two parts:

$$G_m = G_{m+} G_{m-} \tag{8}$$

G_{m+} contains all Nonminimum Phase Elements in the plant reference model, that is all Right-Hall-Plane (RHP) zeros and its delays. The factor G_{m-}, meanwhile, is Minimum Phase in invertible; an IMC controller defined as

$$\tilde{G}_{imc} = G_{m-}^{-1} \tag{9}$$

is stable and casual.

The factorization of G_{m+} from G_m is depended upon the objective function chosen. Choosing Integral-Absolute-Error (IAE) optimization, G_{m+} has the form as following:

$$G_{m+} = e^{-\theta s} \prod_i (-\beta_i s + 1) \qquad \mathrm{Re}(\beta_i), \theta > 0$$

Adopting Integral-Square-Error (ISE) optimization criterion, G_{m+} should be:

$$G_{m+} = e^{-\theta s} \prod_i \frac{-s + \beta_i}{s + \beta_i^H} \qquad \mathrm{Re}(\xi_i), \theta > 0$$

where the superscript H denotes complex conjugate.

Secondly, augment $\tilde{G}_{imc}$ with a filter $f(s)$ such that the final IMC controller $G_{imc} = \tilde{G}_{imc} f(s)$ is now, in addition to stable and causal, proper. Although the inclusion of the filter means that we no longer obtain "optimal control", as implied in step 1, the robustness is enhanced. For no offset to step inputs, define $f(s)$ as following:

$$f(s) = 1/(\lambda s + 1) \tag{10}$$

where λ is an adjustable parameter which determines the speed-of-response. Increasing λ increases the closed-loop time constant and slows the speed of response; decreasing λ does the opposite. λ can be adjusted on-line to compensate for plant/model mismatch in design of control system; the higher the value of λ, the higher the robustness the control system.

For AQM system in networks, the feedback controller is more convenient. In fact, there is relationship between IMC controller and feedback controller as shown in Fig.8, where G_p is transfer function of plant, G_m is reference model of plant, G_{imc} is IMC controller, G_c is feedback controller and G_f is transfer function of disturbance. Having designed G_{imc}, its equivalent classical feedback controller G_c can be readily obtained via algebraic transformation, and vice-verse

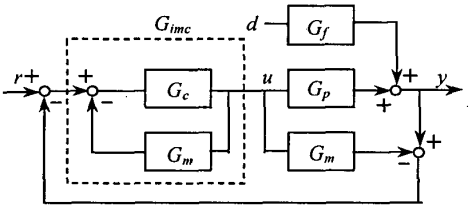

Fig.8 Relationship between IMC controller and feedback controller

$$G_c = \frac{G_{imc}}{1 - G_{imc} G_m} \tag{11}$$

$$G_{imc} = \frac{G_c}{1 + G_c G_m} \tag{12}$$

Thus, we can obtain the feedback controller based on internal compensation principle.

$$G_c = \frac{G_m^{-1} f}{1 - f G_{m+}} \tag{13}$$

B. AQM Controller for Compensating Delay

In section 2, we investigated how the large delay impacts on stability of AQM through simulation experiments, concluded that most of the existed AQM schemes don't take large delay into consideration, so that the performance sharply deteriorates, and is unable to satisfy with the objectives of AQM when they work in large delay networks. In this section, based on internal mode control theory, we will design a robust controller for AQM to compensate possible delay, wish to keep the stable queue and higher link utilization in large delay networks.

We use the first-order with delay model deduced in section 3.2 as the reference model:

$$G_m(s) = \frac{K e^{-Ls}}{Ts + 1} \tag{14}$$

In order to get the simple and realizable controller structure, we use first-order Pade approximation in lieu of the time delay in (14):

$$G_m(s) \approx \frac{K(1 - \frac{L}{2}s)}{(Ts + 1)(1 + \frac{L}{2}s)} \tag{15}$$

Choose the IAE-optimal factorization, yield:

$$G_{m-}(s) = \frac{K}{(Ts + 1)(1 + \frac{L}{2}s)}$$

$$G_{m+}(s) = (1 - \frac{L}{2}s)$$

The first order filter leads to no offset to the expected queue length, namely:

$$f(s) = \frac{1}{(\lambda s + 1)}$$

Then, obtain the corresponding IMC controller:

$$G_{imc}(s) = \frac{(Ts + 1)(\frac{L}{2}s + 1)}{K(\lambda s + 1)} \tag{16}$$

In [24], the impact of parameter λ on performance is discussed in detail. When $\lambda / L \approx 0.8$, the system makes a tradeoff between transient and steady responses. We follow this rule.

According to (11), we solve for the feedback controller to get G_c, which has the structure of classical PID controller:

$$G_c(s) = \frac{P(s)}{E(s)} = K_p + \frac{K_i}{s} + K_d s \tag{17}$$

where

$$K_p = \frac{(2T + L)}{2.6KL}, K_i = \frac{1}{1.3KL}, K_d = \frac{T}{2.6K} \tag{18}$$

So far, the AQM controller with ability to compensate delay is obtained, and named as DC-AQM algorithm. In practice, the sampled queue system needs the discrete form of controller. Substituting a series of sampling time kT for continues time t, replacing integral and differentiation with sum and difference respectively; we get the discrete expression of PID controller:

$$\begin{aligned} \Delta p(k) &= p(k) - p(k-1) \\ &= K_p \left\{ (1 + \frac{T}{T_i} + \frac{T_d}{T}) e(k) - (1 + \frac{2T_d}{T}) e(k-1) + \frac{T_d}{T} e(k-2) \right\} \end{aligned} \tag{19}$$

Assume that the link capacity is 3750 packets /sec, the number of active TCP sessions is 60, the typical RTT is 0.4sec, we solve for the analogy DC-AQM controller to obtain:

$$G_c(s) = 2.268 \times 10^{-5} + 4.193 \times 10^{-6} / s + 8.245 \times 10^{-6} s \tag{20}$$

For the convenience of comparison, we still adopt the sampling frequency defined in [17], i.e. 160Hz, then the sampling time is T=0.00625sec. The output value of DC-AQM controller at kth step is:

$$\begin{aligned} c(k+1) = c(k) &+ 1.34183 \times 10^{-3} e(k) - 2.66093 \times 10^{-3} \\ &\times e(k-1) + 1.31913 \times 10^{-3} e(k-2) \end{aligned} \tag{21}$$

where $e(k) = q(k) - q_0$, $q(k)$ is sampling value of queue length at kth step, q_0 denotes the expected queue length.

Taking the meaning of probability into account, the packet dropping/marking probability is calculated as following:

$$p(k) = \begin{cases} 0 & c(k) < 0 \\ c(k) & 0 \le c(k) \le 1 \\ 1 & 1 < c(k) \end{cases} \tag{22}$$

V. SIMULATION AND EVALUATION

We implemented DC-AQM algorithm in *ns2*, and validated its performance by simulations. The simulation network topology is shown in Fig.1. Given that the propagation delay between node ***A*** and node ***B*** is 190ms, then RTT is at least 400ms. Set the expected queue length to 200 packets. The Fig.9 and Fig.10 demonstrate the evolution of queue length and packet dropping probability respectively, which shows that the DC-AQM algorithm effectively compensates the delay when controlling the node queue. Obviously, for the same configuration settings, the great oscillations occurred in Fig.2 to Fig.4 disappear in Fig.9, which shows that the DC-AQM algorithm effectively compensates the delay when controlling the node queue.

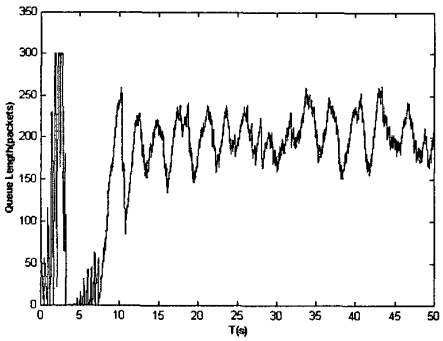

Fig.9 Queue evolution

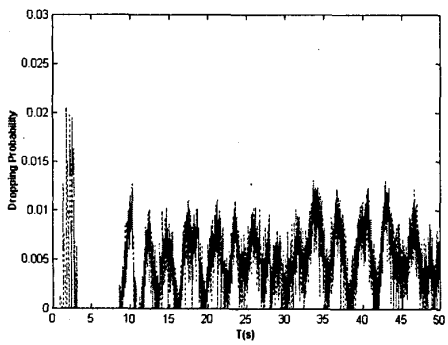

Fig.10 Dropping probability evolution

In order to evaluate the impact on the link utilization of expected queue length, we fixed RTT at 400ms, changed the reference queue length from 5 packets to 150 packets, and plotted the utilization on link AB on the Fig.11. Evidently, for DC-AQM controller, the variance of equilibrium queue length scarcely has impact on link utilization. However, the utilization only reaches 30%~40% in REM scheme when the reference queue length is relative small, and the reduction of expected queue length has more or less negative impact on the utilization in RED and PI schemes.

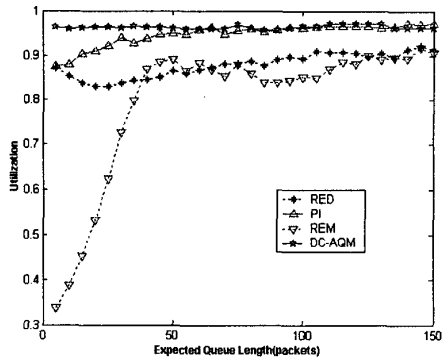

Fig. 11 Link utilization for varying reference queue length for different AQM schemes

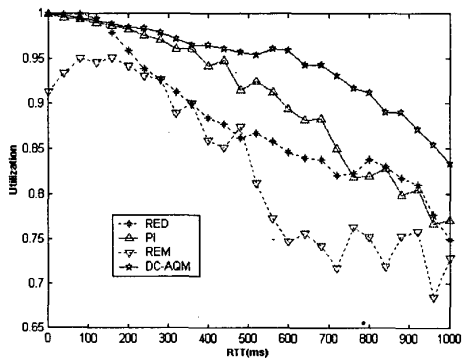

Fig. 12 Link utilization for varying RTT for various AQM schemes

Afterwards, we fixed the expected queue length at 150 packets, varied the propagation delay in RTT from 0ms to 1000ms, and presented the link utilization in Fig.12. Although the variance of RTT has effect on utilization for all AQM schemes, DC-AQM always keeps the higher utilization, the performance of REM severely influenced by variance of RTT, and RED and PI lie between them. Before 700ms, PI has higher utilization than RED

VI. CONCLUSION AND FUTURE WORK

Active queue management has been a very active research area in the Internet community. It intends to achieve the lower waiting delay and higher link utilization through selectively dropping/marking packets at intermediate nodes, and is also an effective end-to-end congestion control mechanism. For most of existed AQM schemes, the impact of large delay on performance wasn't taken into consideration during design them. In this study, we firstly pointed out the invalidity of popular AQM schemes, such as RED, PI controller and REM etc., in large delay networks through simulation experiments. After made an appropriate approximation to the plant model, we

designed a robust AQM controller with ability to compensate delay using internal mode control theory The simulation results show that the new algorithm greatly restrains the oscillations occurred in the existed algorithms, and increases the link utilization. Moreover, we comprehensively compared the performance of DC-AQM algorithm with those of existed schemes for various reference queue length and various RTTs through experiments. The conclusion is that all AQM scheme exhibit the perfect performance for small delay and large expected queue length, but the DC-AQM algorithm is superior to other algorithms for large delay and small reference queue length. Of course, our works still have some limitations, for example, RTT is assumed as known and constant, which is impossible for real network environment because different sessions have various RTTs. However, we can abstract a dominative RTT from various RTTs at intermediate node using estimation theory, just like to estimate the RTT for setting RTO at end system, which is required by TCP flow control procedure. This paper aims at reminding us to pay attention to impact on performance of large delay during designing AQM scheme, at the same time, lay emphasis on the approach to eliminate the negative effect caused by large delay. How to estimate the dominative delay and make an adaptive compensation is a very interesting works and we are currently investigating that.

REFERENCES

[1] Stevens. TCP Slow Start, Congestion Avoidance, Fast Retransmit, and Fast Recovery. RFC 2001, January 1997.

[2] K.Fall, and S. Floyd. Simulation-based Comparisons of Tahoe, Reno, and SACK TCP. Computer Communication Review, V. 26 N. 3, July 1996, pp. 5-21.

[3] L. Brakmo and L. Peterson. TCP Vegas: End-to-End Congestion Avoidance on a Global Internet. IEEE Journal on Selected Areas in Communication, Vol 13, No. 8 (October 1995) pages 1465-1480

[4] S. Floyd. The Addition of Explicit Congestion Notification to IP. http://www.aciri.org/floyd/papers.html

[5] B. Braden et al. Recommendations on Queue Management and Congestion Avoidance in the Internet. RFC2309, April 1998

[6] S. Floyd and V. Jacobson. Random Early Detection Gateways for Congestion Avoidance. IEE/ACM Transactions on networking, August 1993

[7] F. Anjum and L. Tassiulas. Balanced-RED: An Algorithm to Achieve Fairness in Internet. http://www.isr.umd.edu/ CSHCN/

[8] D. Lin and R. Morris. Dynamics of Random Early Detection. Proc. SIGCOMM1997.

[9] M. Christiansen, K. Jeffay, D. Ott, and F.D. Smith, Tuning RED for Web Traffic, ACM SIGCOMM 2000, August 2000

[10] V. Firoiu and M. Borden. A Study of Active Queue Management for Congestion Control. Proc. INFOCOM 2000, March 2000

[11] C. Hollot, V. Misra, D. Towsley and W. B. Gong. A Control Theoretic Analysis of RED. Proc. INFOCOM 2001

[12] Teunis J. Ott, T.V. Lakshman and Larry H. Wong. SRED: Stabilized RED. Proc. INFOCOM'99, March 1999.

[13] W. Feng, D. Kandlur, D. Saha and K. Shin. A Self-Configuring RED Gateway. Proc. INFOCOM'99, March 1999, pp1320-1328

[14] W. Feng, D. Kandlur, D. Saha and K. Shin. Stochastic Fair BLUE: A Queue Management Algorithm for Enforcing Fairness. Proc. INFOCOM 2001

[15] R.J. Gibbens and F.P. Kelly. "Distributed connection acceptance control for a connectionless network." In Proc. of the 16th Intl. Teletraffic Congress, Edinburgh, Scotland, June 1999.

[16] S.Floyd,RED: Discussion of Setting Parameters, http://www.aciri.org/floyd/redparameters.txt

[17] C. Hollot, V. Misra, D. Towsley and W. B. Gong. On Designing Improved Controllers for AQM Routers Supporting TCP Flows. Proc. INFOCOM 2001

[18] Analysis and Design of an Adaptive Virtual Queue (AVQ) Algorithm for Active Queue Management. Proc. ACM/SIGCOMM 2001

[19] F. P. Kelly, A. Maulloo and D. Tan. Rate Control in Communication Networks. Journal of the Operation Research Society, vol.49, pp.237-252, 1998

[20] S. Athuraliya, D.E. Lapsley and S.H.Low. Random Early Marking for Internet

[21] V. Misra, W.B. Gong Congestion Control. Proc. IEEE Globecom 1999. D. Towsley. Fluid-based Analysis of a Network of AQM Routers Supporting TCP Flows with an Application to RED. Proc. ACM/SIGCOMM 2000.

[22] C. Hollot, V. Misra, D. Towsley and W. B. Gong. A Control Theoretic Analysis of RED. Proc. INFOCOM 2001.

[23] Garcia and Morari. Internal Mode Control-1: A unifying Review and Some New Results. Ind. Eng. Chem. Proc. Des. Dev. 1982,21:308-323

[24] Rivera D.E, Morari M., Skogestad S. Internal Mode Control-4: PID Controller Design. Ind. Eng. Chem. Proc. Des. Dev. 1986,25:252-265