A ROBUST WINDOW-BASED ACTIVE QUEUE MANAGEMENT BASED ON H_∞ CONTROL

M. MON'EMI, M.S.
Control Engineering
Islamic Azad University, Neiriz Branch
Neiriz, I. R. of Iran
email: monemi_mahdi@yahoo.com

S. SA'EEDI, M.S.
Control Engineering
Tarbiat Modares University
Tehran, I. R. of Iran
email: saeedi@modares.ac.ir

Abstract - In order to make the best utilization of a link in computer networks, Active Queue Management (AQM) algorithms are generally used in a variety of methods. In this paper, a systematic control strategy based on self-scheduled H∞, considering the dynamics of the window-based TCP flow, is proposed. Then, the robustness of the designed controller to disturbance and parameter fluctuations of the plant is shown via simulation results.

Keywords - AQM, Congestion Control, ATM, TCP, Robust H_∞, LMI, Window-Based.

INTRODUCTION

As the data traffic is vastly increasing in computer networks, there are two major methods for providing a good end-to-end quality of service (QOS) for customers. The first one is increasing the overall bandwidth available in the network, which is not always economical or practical, and the other one is Traffic Engineering (TE). Active Queue Management would probably be one of the most common methods for Traffic Engineering in which we try to avoid the congestion of the incoming data in the bottleneck switches and routers by providing some sort of feedback information for data transmitters so that they could adjust their sending rates in a way to minimize congestion in the buffers of bottleneck routers.

Generally, two approaches are used for congestion control in communication networks. The first one which is called Window-Based control is mostly used in TCP/IP networks. In this method, as congestion increases, bottleneck switches or routes mark/drop packets using algorithms such as Random Early Detection (RED) and senders step down their windows size to reduce congestion in bottleneck queues. The second method, known as Rate-Based control, is mostly used in ATM and MPLS networks. In this method, transmitting rates of the senders are directly determined by the bottleneck switches. In this paper, we are concerned with Window-Based congestion control for TCP based data traffics. AQM tries to provide QOS in two ways: One is to reduce the average length of queue in routers and decrease the end-to-end delay experienced by packets, and the other is to provide a good utilization of links by reducing packet loss caused when the queues overflow. AQM

is especially important for Internet traffic flow and TCP-based protocols to provide an end-to-end congestion control. RED [3] was originally proposed to control the queue length by intelligent marking or dropping packets. It is the only recommended candidate for achieving AQM in RFC 2309. In [3,7] leading researches have proposed implementation of RED as a good choice for AQM in IP routers, but there are some important problems associated with RED, which sometimes make it a poor candidate. First, tuning of RED parameters for a given network is usually a hard job and there is no specific straightforward method to adjust the RED parameters for the best utilization of the network. The second problem is that once RED tuned, the performance is very sensitive to traffic and parameter variations, and many studies have shown that RED is not a robust method for congestion control and would sometimes lead to instability and large oscillations. One important reason that RED is not the optimum solution for congestion control is that it works on some probabilistic heuristic functions which do not directly take the dynamics of the traffic into consideration. Although numerous variants of RED have been proposed such as SRED [8], FRED [6], Balanced RED [1], etc., all works using a heuristic function in the same way. One other approach for AQM, proposed recently, is to find first a mathematical model of the traffic been studied and then according to that model with the cost of some more complication, compared to pure RED algorithm, design and implement the controller. While finding an exact mathematical model for the large-scaled Internet network dynamics is a hard job, in [7] a nonlinear dynamic model for TCP flow control was developed and tested. Based on this model several control schemes have been recommended in [4,5,7]. The controllers developed in [5,7] though successfully removed some limitations of RED, have also some shortcomings in the real Internet traffic mainly due to not considering the variations of model parameters. In [4] this shortcoming was to some extent covered by designing a robust Sliding Mode Variable Structure (SMVS) Control, but the switching surface and, thus, the controller was best designed at particular fixed-parameters and, hence, would not work optimally for all parameter values. In order to make for this shortage, a self-scheduled H_∞ control for the parameter-varying system of TCP flow dynamics is proposed.

TCP FLUID-FLOW MODEL

In [7] a dynamic model of TCP behavior was developed using fluid flow and stochastic differential equation analysis, the accuracy of this model was then shown by simulation results. The model is a coupled nonlinear differential equation as follows:

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

Where:

W: average TCP window size (packets);

q: average queue length;

$R(t) = \frac{q(t)}{C} + Tp$: Round-trip time (secs);

T_p: the fixed propagation delay;

C: link capacity (packets/sec);

N: load factor (number of TCP sessions);

p: probability of packet mark/drop.

The first equation in (1) describes the control dynamic of the TCP window; the 1/R term shows the windows additive increase due to round-trip delay while the W/2 term, models the windows multiplicative decrease by packet marking probability p and the second equation simply models the rate of bottleneck queue length changes as the difference of arrival rate and capacity of the link.

This nonlinear and time-varying model was then linearized about an operating point by small gain linearization. Considering the time-varying characteristic of parameters it can be shown that the linearized model would be as Figure 1:

Where:

$$G(s) = \frac{K(\theta(t))}{[s + P_1(\theta(t))][s + P_2(\theta(t))]} \tag{2}$$

$$P_1(\theta) = \frac{1}{R(t)} \quad P_2(\theta) = \frac{2N(t)}{R^2(t)C(t)} \quad K(\theta) = \frac{C(t)^2}{2N(t)} \tag{3}$$

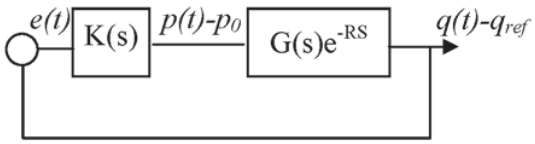

Figure 1

G(s) would be considered as the plant and *K(s)* is the controller to assign *p(t)* in such a way that *q(t)* well tracks q_0 specified by the operator or designer, θ is the vector [*R C N*]. Details of linearization are presented in [7]. This block diagram induces a systematic design strategy for congestion control considering the dynamics of the plant rather than just empirical and stochastic methods used in RED. In this model, variation of plant parameters are well illustrated in *K(t)*, *T1(t)* and *T2(t)*. Parameter variation is something very common in large-scaled networks like Internet. While equation (1) presents a reasonable model for the Internet traffic, it is by no means a complete model. For example it ignores the timeout caused by lacking enough duplicated ACKs in services such as HTTP or Telnet.

Besides, UDP packets are not considered in this model.

SELF-SCHEDULED H_∞ CONTROL DESIGN

In [5,7] the controller *K(s)* was designed by considering all time-varying parameters constant values. This is not the case in practice, for example consider number of TCP sessions, N(t), in the bottle-neck router changes in a range *1~250* which can be the case for real data traffics and consider it as the only variant, then the pole *1/T2* will change by a factor of *250* and the dynamics of the plant is greatly changed. Even in [4] although the Sliding Mode Variable Structure is partially a robust method, variation of parameters was not directly taken into consideration for designing the switching surface and control signal, in other words once *K(s)* designed, the structure of the control signal computation is the same for all parameter values. It is much better to propose a controller which self-tunes as parameters of the plant are varying. Note that $\theta(t)$ can be sensed and measured in real time since the values of *R(t)*, *N(t)* and *C(t)* are all available at the bottle-neck router, subsequently the control strategy can exploit the available measurement of θ to increase performance. To proceed, noting that *K(t)* is measurable, first rearrange the block diagram of Figure 1 as Figure 2.

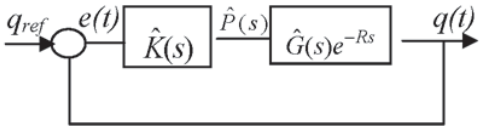

Figure 2

Where:

$$\hat{G}(s) = \frac{1}{K(\theta)} G(s) = \frac{1}{[s + P_1(\theta)][s + P_2(\theta)]} \qquad (4)$$
$$\hat{p}(s) = K(\theta) p(s)$$

Now converting the transfer function of the plant to controllable canonical state space equations we would have:

$$\dot{X} = A(\theta(t))X + B(\theta(t))U + W(\theta(t))$$
$$Y = C(\theta(t))X + D(\theta(t))U \qquad (5)$$

$$A(\theta)=\begin{bmatrix} 0 & 1 \\ -a_1(\theta) & -a_2(\theta) \end{bmatrix} \quad B(\theta)=\begin{bmatrix} 0 \\ 1 \end{bmatrix}$$
$$C(\theta)=\begin{bmatrix} 1 & 0 \end{bmatrix} \quad D(\theta)=0$$
$$W=\begin{bmatrix} 0 \\ \frac{d^2}{dt^2} q_{ref} + (P_1(\theta)+P_2(\theta)) \frac{d}{dt} q_{ref} + P_1(\theta) P_2(\theta) q_{ref} \end{bmatrix}$$

$$a_1(\theta) = P_1(\theta)P2(\theta) = \frac{2N(t)}{R^3(t)C(t)}$$
$$a_2(\theta) = P_1(\theta) + P_2(\theta) = \frac{1}{R(t)} + \frac{2N(t)}{R^2(t)C(t)} \quad (6)$$

Plants of the form (5) are called linear parameter-varying (LPV) plants, which can be viewed as linear time–invariant (LTI) plants subjected to time-varying parametric uncertainty $\theta(t)$.A customary approach for designing controllers for such plants is to divide the space into areas of small parameter variation where the plant is regarded as LTI, design an LTI controller for each region and use some sort of gain scheduling or interpolation to construct the overall control law. The shortcomings of this method are described in [2], but probably the most important deficiency of this approach is that these gain-scheduled controllers are not even guaranteed to stabilize except in the case of slowly varying parameters. A better approach which best matches our problem characteristics is to use self-scheduled H∞ control using Linear Matrix Inequalities (LMI), proposed in [2].

Now consider that the time-varying parameter θ varies in a polytope Θ of vertices θ1, θ2,…,θr.

That is:

$\theta \varepsilon \Theta := Co\{\theta_1, \theta_2, \ldots, \theta_r\}$

In this case, the plant matrix P (θ) ranges in a polytope of vertices $P_1, P_2, \ldots, P_r$, That is:

$$\begin{pmatrix} A(\theta) & B(\theta) \\ C(\theta) & D(\theta) \end{pmatrix} \in P = \left\{ \begin{pmatrix} A_i & B_i \\ C_i & D_i \end{pmatrix} := \begin{pmatrix} A(w_i) & B(w_i) \\ C(w_i) & D(w_i) \end{pmatrix}, i=,1,\ldots r, \right\}$$

Note that mostly, specially in our very problem, P can be related to θ which means that $a_i(t)$ can be found such that expressing $\theta(t)$ as:

$\theta(t) = \sum_{i=1}^{r} \alpha_i(t)\, \theta_i$ θ_i : constant vertices $\sum_{i=1}^{r} \alpha_i = 1$ would result:

$$P = \begin{pmatrix} A(\theta) & B(\theta) \\ C(\theta) & D(\theta) \end{pmatrix} := \sum_{i=1}^{r} \alpha_i \begin{pmatrix} A(\theta_i) & B(\theta_i) \\ C(\theta_i) & D(\theta_i) \end{pmatrix}$$

Theorem: For system (5) with two conditions:

1- θ can be measured real-time

2- P is affinely related to θ,

Given the positive constant γ, if there exists some $(n+k)x(n+k)$, positive definite matrix X_{cl} and kth order LTI controllers:

$$\Omega_i = \begin{pmatrix} A_{Ki} & B_{Ki} \\ C_{Ki} & D_{Ki} \end{pmatrix}$$

can be found such that the following linear matrix inequalities hold:

$$B[A_{cl}(\theta_i), B_c(\theta_i), C_{cl}(\theta_i), D_{cl}(\theta_i)](X, r) \tag{7}$$
$$= \begin{bmatrix} A_{cl}^T X + X A_{cl} & X B_{cl} & C_{cl}^T \\ B_{cl}^T X & -\gamma I & D_{cl}^T \\ C_{cl} & D_{cl} & -\gamma I \end{bmatrix} < 0 \quad i = 1, \ldots r$$

then:

$$\|G\|_\infty = \left\| D_{cl} + C_{cl}(sI - A_{cl})^{-1} B_{cl} \right\|_\infty < \gamma \tag{8}$$

where cl denotes the closed loop and γ is the performance bound to be minimized. The overall controller state space can be shown to be:

$$\Omega = \sum_{i=1}^{r} \alpha_i(t)\, \Omega_i = \sum_{i=1}^{r} \alpha_i(t) \begin{pmatrix} A_{Ki} & B_{Ki} \\ C_{Ki} & D_{Ki} \end{pmatrix} \tag{9}$$

A detailed description and proof of this theorem is presented in [2].

Now returning to our problem, $\theta(t)$ has two elements $a_1(t)$ and $a_2(t)$, so θ is varying in a planar surface. Consider the following parameter variations that might be common in practice:

N(t): 1 ~ 300

Tp: 0.02 sec

q0: 0 ~ 300 packets

C(t): 1250~7500 (packets/sec)

Regarding (6) we have:

$a_{1min} < a_1 < a_{1max}$ $\qquad$ $a_{2min} < a_2 < a_{2max}$

where:

$a_{1min} = 0.09$ $\qquad$ $a_{1max} = 60000$

$a_{2min} = 3.9$ $\qquad$ $a_{2max} = 1250$

$$\theta_1 = \begin{bmatrix} a_{1min} \\ a_{2min} \end{bmatrix} \qquad \theta_2 = \begin{bmatrix} a_{1max} \\ a_{2min} \end{bmatrix}$$

$$\theta_3 = \begin{bmatrix} a_{1min} \\ a_{2max} \end{bmatrix} \qquad \theta_3 = \begin{bmatrix} a_{1max} \\ a_{2max} \end{bmatrix}$$

so θ(t) varies in a polytope Θ of four vertices.

To calculate Ω_i *(i=1,2,3,4)*, first the LFT model of the plant is driven. In order to

attain a better performance range (γ), two filters W_u and W_q are considered as in Figure 3. W_u was selected as a first-order high-pass filter so that the control signal does not have sharp changes; W_q was selected as first-order low-pass filter. There is also a constraint to be met and that is:

$$0 < p(t) = \frac{1}{K(\theta)} \hat{p}(t) < 1 \tag{10}$$

Using matlabs LMI toolbox to solve the four LMIs in (7) and after a little try and error for tuning filters W_u and W_q to gain an acceptable γ and also meet constraint (10), Ω_i (i=1, 2, 3, 4) has been computed on the vertices of the planar surface with:

$$\gamma=0.08 \qquad W_u = \frac{0.1s}{s+400000} \qquad W_q = \frac{2}{1+0.1s}$$

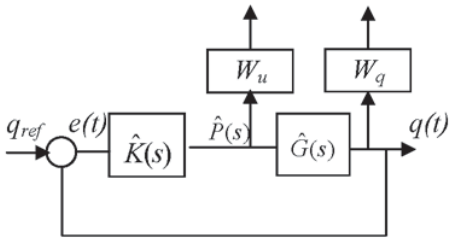

Figure 3

In order to obtain the overall control law given by (9) an affine relation between P and θ needs to be found, $a_i(t)$ can be computed as follows:

$$a1=xy \qquad a2=(1-x)y$$
$$a3=(1-y)x \qquad a4=(1-x)(1-y) \tag{11}$$

where:

$$x(t) = \frac{a_{1max} - a_1(t)}{a_{1max} - a_{1min}} \qquad y(t) = \frac{a_{2max} - a_2(t)}{a_{2max} - a_{2min}(t)} \tag{12}$$

It can be easily seen that a_i in (12) are convex coordinates, since they satisfy $0<=a_i<=1$ and

$$\sum_{i=1}^{r} \alpha_i = 1$$

So the overall control scheme would be as follows:

Initialize:

t=0, $\Omega_i = \begin{pmatrix} A_{Ki} & B_{Ki} \\ C_{Ki} & D_{Ki} \end{pmatrix}$ $i = 1,...,4$

At each epoch of packet arrival

Compute $a_i(t_k)$ i=1,2,3,4 from (11)

$$\Omega = \sum_{i=1}^{4} a_i(t_k)\Omega_i$$

Measure q-q_0 and compute $\hat{p}(t_k)$

If $\hat{p}(t_k) > K$

$\hat{p}(t_k) = K$ $\quad (p(t_k) = 1)$

Else if $\hat{p}(t_k) < 0$

$\hat{p}(t_k) = 0$ $\quad (p(t_k) = 0)$

End

To evaluate the self-scheduled H_∞ controller performance, the parameters are moved along the trajectory shown in Figure 4.

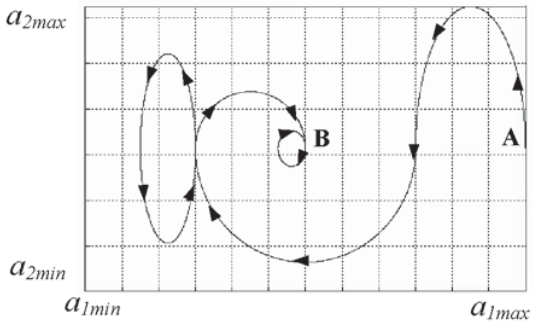

Figure 4

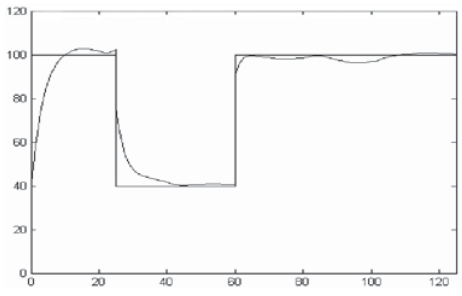

Figure 5

Suppose at *t=0* the parameters are located at point **A**, during the simulation, parameters are varying along the trajectory till they reach point **B** at $t=120\ sec_s$.

The reference is set to *100* packets at *t=0* and then at *t=22sec*, it is set to 40 packets and at *t=60sec* again returned to *100* packets. From Figure 5. it is seen that although the parameters are moving dramatically in a wide range according to trajectory of Figure 4, the buffer length has a good track of the reference input.

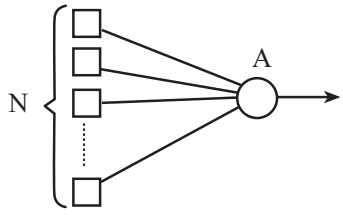

Figure 6

Now to show the effectiveness and performance of the self-scheduled H_∞ controller, we evaluate it for a more realistic scenario. The network topology is shown in Figure 6.

The combination of N Sources (N=20) will connect TCP sessions at random times during *60* seconds for 6 times and each connection consists of 2 short-lived burst sessions, which last just for 2 seconds. The capacity of node A equals *30Mbps* and the delay between all sources and bottleneck router is *20msec*. Maximum buffer size of A equals to 300 packets and the default size of the packet is 500 bytes. Let the reference buffer size be 200 packets. The simulation result is illustrated in Figures 7 and 8.

It is seen that in spite of the random nature of establishing and ending TCP connections on the bottleneck router, which causes the plant dynamics to change very fast, the controller soon tunes itself for the new dynamics.

Self-tuning is important for perfect AQM in practice since the nature of data traffics in communication networks are mostly random and no specific and deterministic algorithm can be found for them, especially bottle-neck routers involved with passing Internet data traffic have no deterministic arrival pattern since the number of sessions established for customers connecting to an Internet Service Provider (ISP) varies in a dramatically random fashion.

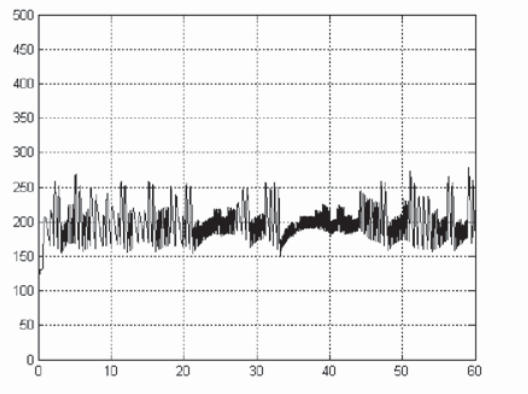

Figure 7

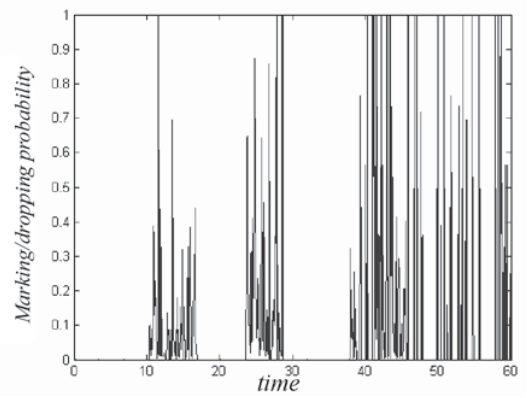

Figure 8

CONCLUSIONS

AQM would be an effective mechanism for congestion control and it provides a good end-to-end QOS, especially it seems to be a necessity for the health of huge and complex networks such as Internet. It was shown that by considering the real mathematical model of the nature of data traffic, the problem of managing AQM is converted to designing a controller, using classical and systematic control techniques. A self-scheduled H_∞ controller for providing QOS and improving utilization was developed. The controller was designed to provide buffer management considering the linearized model of TCP connection dynamics for bottleneck routers, it was very robust to the change of dynamics of the plant caused by random nature of data traffic and was claimed to remove some of the shortcomings

of RED algorithm.

REFERENCES

[1] Anjum, F. and Tassiulas, L., "Balanced-RED: An Algorithm to Achieve Fairness in Internet." Available: at <http://www.isr.umd.edu/CSHCN>.

[2] Apkarian, P. and Gahinet, P., "Self-Scheduled H Control of Linear Parameter-Varying Systems: a Design Example." Automatica, Vol. 31, No. 9, pp. 1251-1261, 1995.

[3] Christiansen, M., Jeffay, K., Ott, D. and Smith, F. D., "Tuning RED for Web Traffic." *ACM SIGCOMM 2000*, March 2000.

[4] Fengyuan, R., Chuang, L., Xunhe, Y., Xiuming, S. and Fubao, W., "A Robust Active Queue Management Algorithm Based on Sliding Mode Variable Structure Control." *IEEE*, 2002.

[5] Hollot, C., Misra, V., Towsley, D. and Gong, W. B., "On Designing Improved Controllers for AQM Routers Supporting TCP Folws." *Proc. INFOCOMM*, 2001.

[6] Lin, D. and Morris, R., "Dynamics of Random Early Detection." *Proc. SIGCOMM,* 1997.

[7] Misra, V., Gong, W.B and Towsley, D., "Fluid-Based Analysis of a Network of AQM Routers Supporting TCP Flows with an Application to RED." *Proc. ACM/SIGCOMM*, 2000.

[8] Teunis J. Ott, Lakshman, T.V. and Lary H. Wong., "SRED: Stabilized RED." *Proc. INFOCOMM 99*, March 99.