An Application of the Control Theoretic Modeling for a Scalable TCP ACK Pacer

Yishen Sun, C. C. Lee, Randall Berry and A. H. Haddad
Department of Electrical and Computer Engineering, Northwestern University
Evanston, IL 60208, U.S.A.
Email: {yishen, cclee, rberry, ahaddad}@ece.northwestern.edu

***Abstract*[1] -- In this paper, we present the design of a TCP ACK pacer which regulates the downstream buffer occupancy at an edge router in the Internet to avoid Quality of Service (QoS) damaging congestion while maximizing the link utilization. This technique has the advantage of requiring no changes in existing TCP implementations. Based on a feedback system representation of the network, a PI-type controller is proposed to determine the aggregate ACK releasing rate according to the buffer occupancy. This approach is scalable in that it requires no per-flow state information. The robustness of the controller is addressed with respect to the system uncertainty such as the traffic dynamics and the system delay. Generalizations of the control objective are discussed as well. A special case of the controller, corresponding to a proportional plus lag compensator is analyzed and used to provide a systematic approach for satisfying a given delay requirement. Numerical examples to illustrate this approach are also given.**

I. INTRODUCTION

Providing differentiated Quality of Service (QoS) is of increasing importance in broadband networks. As sustained congestion at a router may trigger excessive packet dropping and render its QoS capability ineffective, traffic control methods capable of minimizing such damaging congestion occurrences are essential for the router to deliver per-flow QoS. In this paper, we consider the congestion control problem at an edge router that connects an end-user access network to the core Internet. In many cases, an edge router is able to control the upstream traffic flows (from the access network to the core) through the data link layer access control or flow control. However, such techniques cannot be applied to the downstream traffic (from the core to the access network) that originates elsewhere in the Internet. Therefore, congestion is more likely to occur in the downstream direction as traffic bursts arrive at the edge router from the high-speed ingress links and are directed to the lower-speed access links. Consequently, it is desirable to have an effective means of minimizing downstream congestion in the QoS solution of the edge router.

An effective approach to prevent damaging congestion and packet losses while maintaining high bandwidth utilization is to properly manage the downstream traffic flow by traffic shaping and/or packet buffering mechanisms. Some well-known approaches in this area include token bucket rate limiting [1] and Random Early Detection (RED) [2]. The burstiness of Internet data traffic today is caused largely by Transport Control Protocol (TCP) traffic. The well-known TCP congestion control mechanism maintains a dynamic congestion window at the source node where the window size increases whenever an acknowledgement (ACK) is received before its retransmission timer goes off. Therefore, a logical means for shaping TCP traffic flows on the downstream is to pace the delivery of the ACK packets, as appropriate, on the upstream [3], [4]. When the traffic load is low, the shaping may not be necessary; however, the shaping ought to be more aggressive when the edge router experiences an increasing downstream traffic load. This is the rationale behind the ACK pacer presented in this paper.

When compared to the per-flow token bucket algorithm in the forward (downstream) path, ACK pacing has the potential advantage of drastically reducing the buffer requirement at the edge router since holding ACKs typically requires much less buffer space than holding data packets in the forward direction. The advantage of an ACK pacer over approaches such as RED, is that the ACK pacer seeks to proactively regulate TCP traffic at their sources while RED reacts to emerging congestion by dropping packets.[2] The general idea of ACK pacing has been introduced in [3], [4] and [9]. The present algorithm differs from that of [3] and [4], in that it adapts to the aggregate traffic load as represented by the downstream buffer occupancy, rather than to individual flows states, and thus leads to a more scalable implementation. In addition, since the proposed ACK pacer does not need to write into the ACK header, it is capable of controlling encrypted flows and is thus different from the TCP rate controller proposed in [3]. The approach in [9] is also based on the aggregate traffic load, but uses a different type of controller and does

[1] This work was supported by the Motorola Center for Communications at Northwestern University

[2] Packet dropping can be avoided in RED if Explicit Congestion Notification (ECN) is used; however, this requires that the end-user TCP implementation be modified to be ECN capable.

not offer a systematic approach for setting the parameters or analyzing the performance. TCP pacing has been considered in other contexts as well, such as improving the initial transient performance in networks with large bandwidth delay products [12].

Because of the feedback nature of the TCP congestion control mechanism [5], adaptive control methods have the potential to guide the design of an ACK pacing algorithm. Starting from a state-space representation of the congestion window dynamics, we proposed an *α-tracking load-adaptive ACK pacer* in [6]. In this paper, a PI-type ACK pacer is investigated. PI controllers have been used in other networking problems including explicit rate control in ATM networks [7], [11], Active Queue Management (AQM) in TCP/IP networks [8], and QoS adaptation [13]. Compared to the heuristic algorithm presented in [6], the control theoretic modeling of the network presented in this paper is more rigorous so that the stability and robustness of the controller can be mathematically analyzed.

The rest of the paper is organized as follows. The next section describes the network model and the basic control strategy of the ACK pacer. System dynamics are presented in Section III. In Section IV, a PI-type ACK pacer is characterized in both the time domain and the frequency domain based on the feedback control representation of a sampled data system. In addition, sufficient and necessary conditions on the ACK pacer parameters for a stable system are derived. The system's robustness with respect to the aggregate traffic uncertainty and the delay jitter is investigated in the same section, and a generalization of the control objective is discussed as well to include filtered versions of the buffer occupancy. In Section V, a more systematic design methodology for one example of the controller is presented to guarantee a specified delay requirement. Some numerical results are given in Section VI. Finally Section VII summarizes the paper.

II. Network Model

Consider a network where an edge router connects an end-user access network to the core Internet. The ACK pacing algorithm proposed here deals primarily with the traffic control for the downstream traffic (i.e., traffic from the core to the end-user) at the edge router. Assume that the link capacity between the core network and the edge router is sufficiently large so that it does not impose any restrictions on how fast data packets and ACKs can flow between TCP sources and the edge router. On the other hand, the link between the edge router and the access network, denoted as C (bps), imposes constraints on the bandwidth available to TCP receivers, and hence may develop a bottleneck. The objective of the ACK pacer is to maximize TCP "goodput", while controlling the downstream buffer occupancy $B(t)$ (bits) at the edge router to reduce the latency and minimize packet losses. The basic control strategy of the ACK pacer is as follows:

1) Activate the ACK pacer when the queue length $B(t)$ exceeds B_1;
2) Deactivate the ACK pacer if $B(t)$ falls below B_2;
3) When activated, adjust the ACK releasing rate to keep $B(t)$ below a threshold B_{th},

where $0 < B_2 < B_1 < B_{th}$.

The reason for setting the activation and deactivation thresholds for the ACK pacer is to avoid over-controlling. It is obvious that we do not need to hold upstream ACKs at all if the bottleneck link is far from over-loaded. An additional buffer increase should also be provisioned for, before the ACK pacer takes effect. The threshold B_{th}, which is the desired upper bound of $B(t)$, is determined by considering various network characteristics, such as packet latency requirements and any AQM scheme for the downstream buffer. For example, if the router buffer implements a Tail Drop scheme, B_{th} may be set equal to the physical buffer limit; if RED is used for the AQM, B_{th} may be the threshold at which the random early dropping begins.

III. System Dynamics

Consider the case where N unlimited TCP sources are sending data through the edge router. Each of these sources transmits constant sized segments as fast as its congestion window allows and that the receiver advertises a consistent flow control window throughout the TCP session. The transmission rate of such a TCP source is determined by both the number of ACKs received and the congestion window size. We denote the traffic sending rate and the ACK receiving rate of the i^{th} flow at time t as $r_i(t)$ and $a_i(t)$ respectively, both measured in bits per second (bps). Let $w_i(t)$ denote the congestion window size (*cwnd*) of the i^{th} traffic source at time t; this is measured in TCP segments where all segments are assumed to be of the same size. The dynamics of a TCP connection can be approximated by the following equation:

$$r_i(t) = La_i(t) + L_D\dot{w}_i(t), \qquad (1)$$

where L_D (bits) is the downstream data segment length, L_A (bits) is the upstream ACK segment length, and $L = L_D/L_A$.

The TCP congestion control algorithm dynamically adjusts the congestion window according to the network state. In its *slow start* phase, the sender increases the *cwnd* by one segment upon the receipt of each ACK. In its *congestion control* phase, however, the *cwnd* is increased by one for every *cwnd* ACKs received. If the *cwnd* is larger than the amount of data which the receiver is willing to receive in the future, the connection enters its *saturation* phase, and the *cwnd* stops increasing. Therefore the evolution of the TCP congestion window of source i can be summarized as follows:

$$\dot{w}_i(t) = \begin{cases} a_i(t)/L_A & ,if\ in\ slow\ start; \\ a_i(t)/(L_A w_i(t)) & ,if\ in\ congestion\ avoidance; \\ 0 & ,if\ in\ saturation. \end{cases} \quad (2)$$

Neglecting the delay between the source and the edge router, we have

$$\dot{B}(t) = \begin{cases} \sum_{i=1}^{N} r_i(t) - C & ,if\ B(t) > 0 \\ \max\left\{0, \sum_{i=1}^{N} r_i(t) - C\right\} & ,if\ B(t) = 0 \end{cases} . \quad (3)$$

Equations (1) to (3) imply that both source transmission rates and the buffer occupancy can be regulated implicitly by adjusting the ACK releasing rate at the edge router.

IV. PI-TYPE ACK PACER

We consider a sampled-data control model shown in Figure 1 for a system consisting of an ACK pacer and a TCP/IP network where the bottleneck may develop in the downstream access link. The input to the controller is sampled every T seconds, and the discrete time index n corresponds to the continuous time $t=nT$. In particular, $a(n)$ and $a(t)$ (bps) are used to denote the aggregate upstream ACK releasing rate at the output of the ACK pacer in the edge router which is adjusted every T seconds. The control target B_0 (bits) is determined by dropping thresholds of AQM schemes or the latency guarantee of data packets. The error signal $e(t)$ equals $B_0 - B(t)$, and the input to the controller is $e(n)$, the sampled value of $e(t)$.

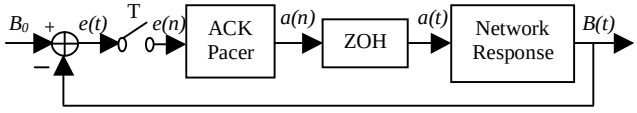

Figure 1 Time Domain System Block Diagram

A. Time Domain Characterization

The time domain response of the ACK pacer, a PI-type controller, is given by

$$a(n) = a(n-1) + K_x[e(n) - e(n-1)] + K_y e(n), \quad (4)$$

where K_x and K_y are gain parameters of the controller.

The function of the zero-order hold (ZOH) is to keep the input value at the same level between two successive adjustments, i.e., $a(t)=a(n)$ for $nT \le t < (n+1)T$.

The network response can be considered to be consisted of two concatenated components. The first component characterizes the behavior of the traffic sources. The second component describes the downstream buffer occupancy fluctuation at the edge router.

Let $r(t)$ (bps) denote the aggregate downstream traffic arrival rate at the edge router at time t. Assume that the delay between the edge router and the traffic source is negligible, then we have $r(t)=r_1(t)+\ldots+r_N(t)$ and $a(t)=a_1(t)+\ldots+a_N(t)$. The dynamic response of traffic sources can be captured by a unitless parameter M, which is the ratio between the aggregate source transmission rate and the aggregate ACK receiving rate, i.e., $r(t)=Ma(t)$. The exact value of M depends on the congestion state of TCP sources. From (1) and (2) it can be obtained that the relation between $r_i(t)$ and $a_i(t)$ is:

$$r_i(t) = \begin{cases} 2La_i(t) & ,if\ in\ slow\ start; \\ (1 + \frac{1}{w_i(t)})La_i(t) & ,if\ in\ congestion\ avoidance; \\ La_i(t) & ,if\ in\ saturation. \end{cases} \quad (5)$$

From (5) it follows that $L \le M \le 2L$. Thus, if all flows are in *slow start*, we have $M=2L$. If all flows are in *congestion avoidance* and have the same window size w, we have $M=(1+1/w)L$. If all flows are in *saturation*, we have $M=L$. Assume that the bottleneck link is overloaded, i.e., the downstream buffer at the edge router is never empty. The buffer occupancy $B(t)$ then evolves as:

$$\dot{B}(t) = r(t) - C. \quad (6)$$

Note that the total delay of the network transmission is usually time varying, and is not accounted for in (4)-(6). In Section IV.E we will discuss the robustness of the system with respect to different delays, and in Section V a more systematic design procedure will be studied to meet certain delay requirements.

B. Frequency Domain Transfer Function

It can be shown that the sampled-data system in Figure 1 is equivalent to the discrete-time feedback loop of Figure 2 in the frequency domain through the zero-order hold equivalence transformation [10].

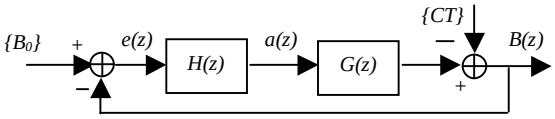

Figure 2 Zero-Order Hold Equivalent System

Here $\{B_0\}$ and $\{C\}$ are the Z-transform of step inputs of magnitudes B_0 and C respectively, and

$$H(z) = \frac{a(z)}{e(z)} = K_x + K_y \frac{1}{1 - z^{-1}}, \quad (7)$$

$$G(z) = MT \frac{z^{-1}}{1 - z^{-1}}. \quad (8)$$

The open-loop gain from $\{B_0\}$ to $B(z)$ is given by

$$G_o(z) = \frac{(X+Y)z - X}{z^2 - 2z + 1}, \quad (9)$$

where $X=K_xMT$ and $Y=K_yMT$. Finally, the close-loop transfer function from $\{B_0\}$ to $B(z)$ is given by

$$G(z) = \frac{(X+Y)z - X}{z^2 + (X+Y-2)z + (1-X)}. \quad (10)$$

C. Stable Region

A sufficient and necessary condition for the system to remain stable is that the denominator polynomial of (10) has no roots on or outside the unit circle. Based on Jury's stability test [10], input-output stability can be guaranteed if

and only if X and Y satisfy following conditions:

$$0 < X < 2, \quad 0 < Y < 4 - 2X \,. \tag{11}$$

Therefore, gain parameters of the PI controller, K_x and K_y, can be selected as follows. First, pick a pair (X, Y) within the stable region. Then, for the given M and T, set

$$K_x = X/(MT), \qquad K_y = Y/(MT). \tag{12}$$

Notice that the stable region of (X, Y) is convex, and is independent of C, the bottleneck link capacity.

D. Robustness Analysis with respect to M

Equation (12) shows that M is directly related to the design of controller gains K_x and K_y. However, M is time varying and has to be estimated in a real system. We propose to use a nominal M to determine values of K_x and K_y instead of estimating it continually. The criterion for choosing a nominal M is that the system should remain stable regardless of the actual value of M. The following lemma asserts that there exists a nominal value of M that satisfies this criterion.

Lemma: Using $M=2L$ as the nominal value can guarantee the system stability under all possible values of M.

The proof is based on the fact that the stable region characterized by Equation (11) is convex, and that the exact value of M is bounded by L and $2L$.

If we have a reliable estimate of the maximum value of M and it is smaller than $2L$, then larger values of K_x and K_y can be used while still guaranteeing the stability. The advantage of larger K_x and K_y is a better transient behavior of $B(t)$. Specifically, as the difference between the nominal and the actual value of M increases, the overshoot and the settle time increase as well.

E. Robustness Analysis with respect to Delay

Previous results are obtained based on assumptions that the ACK pacer adjusts the aggregate ACK releasing rate every T seconds, and that network delays are negligible. However, in a practical environment, these delays may be substantial and ignoring them may result in system instability. Furthermore, the value of the total delay τ is usually time varying. The amount of the delay that can be tolerated in the system in Figure 2 can be predicted using the standard control theory. The maximum tolerable delay τ_{max} for the close-loop system can be found from the phase margin and the crossover frequency of the open-loop gain $G_0(e^{j\omega})$ [11]. Let ω_c (rads/sec) denote the normalized crossover frequency, and PM (rads) is the phase margin, then the maximum delay that can be tolerated is given by

$$\tau_{\max} = \frac{PM}{\omega_c} T \,. \tag{13}$$

From (9) we can see that the value of (PM/ω_c) is completely determined by X and Y. Therefore, (13) implies that increasing T while keeping $X=K_xMT$ and $Y=K_yMT$ unchanged would make the system stability more robust to larger delays. However, since $a(t)$ is adjusted every T seconds, a larger T will slow down the controller's reaction to a steep change in $r(t)$. On the other hand, decreasing T will shorten the time for the ACK pacer to respond to the traffic congestion, but only up to the point where T is still no smaller than the round trip delay RTD between the edge router and traffic sources. Once T gets smaller than the RTD, the adjustment in the ACK releasing rate from the controller no longer effects the buffer occupancy after T seconds, and the RTD becomes the limiting factor for the response time of the system. Thus a trade-off exists between the system responsiveness with respect to the emerging congestion and the robustness with respect to delays when choosing T.

Numerical results show that (PM/ω_c) decreases as X and Y increase, and the maximum tolerable delay for given X and Y is readily calculated using Equation (13). Therefore, for a specified delay requirement, different values of X and Y may be tried until an appropriate choice is found. In Section V, we will consider a more direct design procedure in which the ACK pacer is the concatenation of a proportional and a lag compensator. The controller designed in this way is a special case of the ACK pacer of the form (7), but with a structure that leads to a systematic design to satisfy a given delay requirement.

F. Generalization of the Control Objective

In the previous analysis, the control objective is to keep the instantaneous buffer occupancy $B(t)$ at sample times around a desired value B_0, which is directly related to the QoS delay requirement assuming that a simple first-in-first-out queuing discipline is used at the edge router. We note that a different control objective may be more appropriate if a more sophisticated AQM scheme is used or if a QoS requirement other than the queuing latency is of interest instead. In this section, we consider a generalization of the system to accommodate such variations.

The modified system is shown in Figure 3 where a filtered version of the buffer occupancy is used, and a superscript * is used in the figure to denote the generalized quantities. In the figure we have $G_1(s)=M$ and $G_2(s)=1/s$. Here $G_3(s)$ represents a filter that averages the buffer occupancy between sample data points, and $F(z)$ represents a filter that averages the buffer occupancy across sample data points. For example, $G_3(s)=(1-e^{-sT})/(sT)$ if the average buffer occupancy over T seconds, $B^*(t)$, rather than the instantaneous value $B(t)$ is used to specify the control target. If RED is implemented as the AQM scheme, then the packet dropping is based on a weighted average of the queue size, and a reasonable control objective is to keep the weighted average queue size $B^*(n)=(1-\alpha)B^*(n-1)+\alpha B(n)$ below the threshold at which the random early dropping begins. This can be modeled by $F(z)=\alpha/(1-(1-\alpha)z^{-1})$. For this modified system, the stable region and the robustness analysis with respect to M and/or delays can be obtained in a similar way as in Section IV.B—IV.E.

V. ACK Pacer composed of a Proportional and a Lag Compensator

The discussion in Section IV.E enables us to calculate the maximum tolerable delay τ_{max} for given controller parameters K_x and K_y. In this section, we focus on the selection of K_x and K_y for the given τ_{max}. We investigate a special case of the PI controller in (7), which can be viewed as a combination of a proportional and a lag compensator. This leads to a more systematic approach for designing the controller to meet a specified delay requirement.

A. Proportional Compensator

A simple choice for the ACK pacer $H(z)$ in Figure 2 is to use a proportional controller,

$$a(n) = K_p e(n) + A, \tag{14}$$

where K_p is the proportional gain, and A is a constant to assure that the output link is fully utilized. Hence, we have

$$H(z) = H_P(z) = K_p. \tag{15}$$

The frequency response of the open-loop gain of the system with this proportional compensator is given by

$$G_o^{(P)}(e^{j\omega}) = \frac{K_p MT e^{-j\omega}}{1 - e^{-j\omega}} = \frac{P}{2\sin\frac{\omega}{2}} e^{-j(\frac{\pi+\omega}{2})}, \tag{16}$$

where $P=K_pMT$.

It is clear from examining the close-loop transfer function that P should be between 0 and 2 to make the entire system stable. For a given P with $0<P<2$, the maximum tolerable delay τ_{max} can be calculated from (16) as follows:

$$\tau_{\max} = \frac{PM}{\omega_c} T = (\frac{\pi}{4\arcsin\frac{P}{2}} - \frac{1}{2})T \ . \tag{17}$$

It can be seen from (17) that the maximum tolerable delay τ_{max} decreases monotonically as the proportional gain K_p increases, if M and T remain the same. (17) also suggests an analytical way to determine K_p for fixed M and T to satisfy the robustness requirement with respect to delays. However, the steady-state error of a proportional controller is not zero, and is inversely proportional to the controller gain. Therefore, a lag compensator $H_L(z)$ should be included in $H(z)$ to force the steady state error toward zero.

B. Lag Compensator

In this section, the use of a lag compensator in $H(z)$ is examined. The complete controller to be considered then has the form

$$H(z) = H_P(z)H_L(z) = K_p(\frac{2}{1+\beta})(\frac{z-\beta}{z-1}), \tag{18}$$

where $$H_L(z) = (\frac{2}{1+\beta})(\frac{z-\beta}{z-1}). \tag{19}$$

Note that $H(z)$ can be thought of as a proportional compensator $H_P(z)$ in series with a lag compensator $H_L(z)$ which, due to the normalizing factor $2/(1+\beta)$, has a high frequency gain of one. The frequency response of the open loop gain of the system with both proportional and lag compensators is now given by

$$G_o(e^{j\omega}) = \left((\frac{2}{1+\beta})(\frac{e^{j\omega}-\beta}{e^{j\omega}-1})\right)\left(\frac{P\cdot e^{-j\omega}}{1-e^{-j\omega}}\right). \tag{20}$$

A standard design procedure for $H(z)$ is to first design the proportional compensator, $H_P(z)$, to achieve a basic level of robustness with respect to delays as characterized in (17). Second, if β, the zero of the lag compensator, is properly chosen such that the corner frequency of $H_L(e^{j\omega})$ is sufficiently far below the crossover frequency of $G_0^{(P)}(e^{j\omega})$, then little phase lag will be added at the crossover. This will result in a system with the additional lag compensator, $G_0(e^{j\omega})$, having approximately the same robustness with respect to delays as the system with just the proportional compensator, $G_0^{(P)}(e^{j\omega})$, but having the added effect of pressuring the steady-state queue size towards the control target B_0. Choosing the zero that is one decade below the crossover results in the following value for β:

$$\beta = e^{-(\omega_c/10)}, \quad \omega_c = 2\arcsin\frac{P}{2}. \tag{21}$$

In comparison with the general form of (4), the ACK pacer with a proportional and a lag compensator as in (19) is a special case of (4) with

$$K_x = \beta\gamma, \qquad K_y = (1-\beta)\gamma, \tag{22}$$

where $\gamma=2K_p/(1+\beta)$. Thus, the robustness analysis of the system stability with respect to the uncertainty of M still applies.

Therefore, to design an ACK pacer of the form (18), which guarantees the maximum tolerable delay requirement for given M and T, (17) should be used to select K_p first. Then β is calculated from (21), and K_x and K_y are derived from (22) to complete the time domain representation of the ACK pacer. Finally, the precise value of the maximum tolerable delay τ_{max} should be verified by checking the actual phase margin and the crossover frequency using values of K_x and K_y obtained.

VI. Numerical Results

In this section, we present some numerical results for the system in Figure 1. In the following, L, the ratio of the data segment size and the ACK segment size, is set to be 22.8125, reflecting a typical TCP data segment length of 1460 bytes and a typical ACK segment length of 64 bytes. In the following figures, the x-axis is for the time index n, which corresponds to the time instant nT, and the y-axis is for the normalized buffer occupancy B/B_0.

First we consider the delay robustness of the ACK pacer in (4) with X=0.2378 and Y=0.0701. Three buffer occupancy traces are shown in Figure 4. Trace 1 is for the system in which the additional delay is exactly T seconds.

Trace 2 is for the system in which delay is T seconds before n=20, and increases to $2T$ seconds after that. Trace 3 is for the system in which delay is T seconds before n<20, increases to $3T$ seconds for $20 \le n \le 60$, and returns to T seconds for n>60. Following the analysis in Section IV.E we learn that τ_{max} = 2.2234T seconds for X=0.2378 and Y=0.0701. Therefore, the network is still stable for delays of T or $2T$ seconds, as indicated by trace 1 and 2. However, the stability cannot be achieved for the delay larger than 2.2234T seconds. The instability can be observed in trace 3 for $20 \le n \le 60$. The system is stabilized again after the delay returns back to T seconds for n > 60.

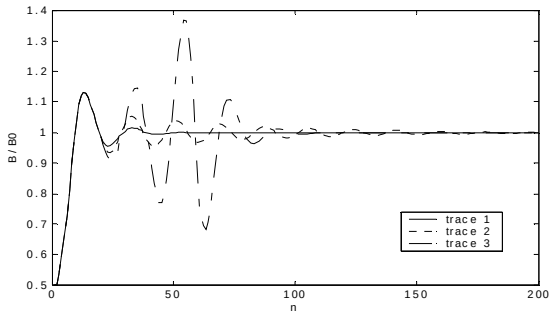

Figure 4 System Response with Additional Delay (X=0.2378, Y=0.0701)

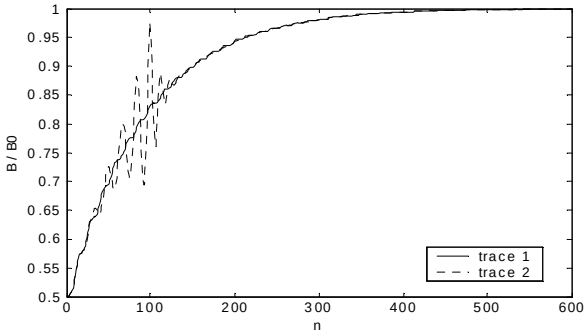

Figure 5 System Response with Additional Delay (X=0.4192, Y=0.0045)

Figure 5 is an example of the proportional and lag compensator in Section V. We consider the design of an ACK pacer that can tolerate delays up to 3T seconds. Assume τ_{max}=3.2T to provide the adequate robustness to delays, then design procedures in Section V yield that X=0.4192 and Y=0.0045. These values result in a system with an actual delay tolerance of τ_{max}=3.1396T. The buffer occupancy trace 1 is for the system in which the additional delay is 3T seconds. Trace 2 is for the system in which delay is 3T seconds before n<20, increases to 4T seconds for $20 \le n \le 100$, and returns to 2T seconds for n>100.

VII. Summary

A control-theoretic TCP ACK pacing technique is presented to regulate the downstream TCP traffic and to avoid the QoS-damaging congestion. A sampled data feedback control system representation is used to govern the design and analysis of a PI controller for the ACK pacer. The selection of controller parameters was investigated with a main focus on the system stability. In addition, the robustness of the system with respect to variations in the aggregate traffic and round trip delays is studied. Numerical results that verify functions and properties of the ACK pacer are also provided.

Reference

[1] A. L.-Garcia, and I. Widjaja, "Communication Networks: Fundamental Concepts and Key Architecture", McGraw-Hill, 2000

[2] S. Floyd, and 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. Karandikar, S. Kalyanaraman, P. Bagal, and B. Packer, "TCP Rate Control", *Computer Communications Review*, vol.30, no.1, pp.45-58, January 2000

[4] P. Narvaez, and K. -Y. Siu, "An Acknowledgement Bucket Scheme for Regulating TCP flow over ATM", *Computer Networks and ISDN Systems*, special issue on ATM traffic management, 1998

[5] S. H. Low, F. Paganini, and J. C. Doyle, "Internet Congestion Control," *IEEE Control Systems Magazine*, pp. 28-43, February 2002

[6] Y. Sun, C. C. Lee, R. Berry and A. H. Haddad, "A Load Adaptive ACK Pacer for TCP Traffic Control", *Proc. of 40th Allerton Conference on Communication, Control, and Computing*, Oct. 2-4, 2002

[7] S. Kubo, T. Ushio, and S. Yamamoto, "PID Rate-Based Control with Propagation Delay—An Application of Robust Stability Analysis", *Internet Conference, November 2001*, Osaka, Japan

[8] C. V. Hollot, V. Misra, D. Towsley, and W. B. Gong, "Analysis and Design of Controllers for AQM Routers Supporting TCP Flows", *Special issue of IEEE Transactions on Automatic Control on "Systems and Control Methods for Communication Networks"*, vol. 47, pp. 945-959, June 2002

[9] J. Aweya, M. Ouellette, and D. Y. Montuno, "A Self-regulating TCP Acknowledgement (ACK) Pacing Scheme", *International Journal of Network Management*, vol. 12, pp. 145-163, 2002

[10] C. L. Philips, and H. T. Nagle, "Digital Control System Analysis and Design", Prentice Hall, 1995

[11] C. E. Rohrs, R. A. Berry, "A Linear Control Approach to Explicit Rate Feedback in ATM Networks", *Proc. IEEE Infocom '97*, pp. 277-82, Kobe Japan, April 7-11, 1997.

[12] V. Visweswaraiah and J. Heidemann, "Improving Restart of Idle TCP Connections," Technical Report USC TR 97-661, University of Southern California, November, 1997.

[13] B. Li and K. Nahrstedt, "A Control Theoretical Model for Quality of Service Adaptations," *Proc. 6th International Workshop on Quality of Service*, pp. 145-153, May 1998.

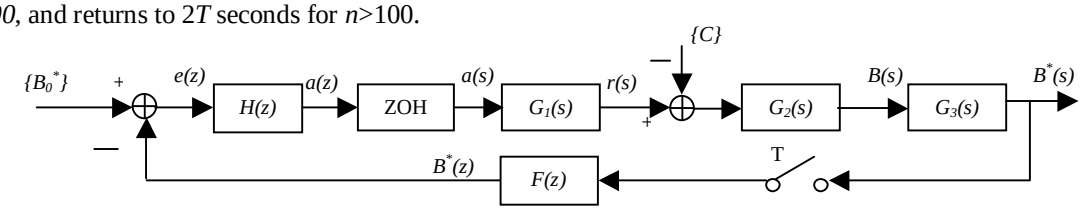

Figure 3 Generalization of ACK Pacer Control Objective