A Load-Adaptive ACK Pacer for TCP Traffic Control

Yishen Sun, C. C. Lee, Randall Berry, A. H. Haddad
Department of Electrical and Computer Engineering
Northwestern University
2145 Sheridan Road, Evanston, IL 60208
{yishen, cclee, rberry, ahaddad}@ece.nwu.edu

Introduction Delivering reliable per-flow Quality of Service in a broadband access network entails a comprehensive congestion control solution at the edge router. The edge router has no direct control over the downstream traffic sources and its egress channel (access link) usually has a smaller bandwidth than its ingress channel. Therefore, to minimize QoS-damaging congestions while maximizing the access link utilization, we leverage on TCP's congestion control mechanism to pace downstream traffic flows at their sources. Some well-known approaches in the area of traffic shaping and/or packet buffering mechanisms include token bucket rate-limiting and random early detection (RED)[1]. This paper presents the idea of using a TCP acknowledgement (ACK) pacer at the edge router to regulate downstream TCP traffic in accordance with the downstream buffer occupancy at the edge router. Although the general idea of holding ACKs has been studied in [2] and [3], our ACK rate allocation scheme is devised by viewing the problem as an adaptive control system with the control target being to guide the buffer occupancy on a desired queue-length trajectory. Some preliminary simulation results are included.

Load-Adaptive ACK Pacer Let *w(t)* denote the congestion window size of the traffic source at time *t*. Denote the traffic sending rate and acknowledgement(ACK) receiving rate as *s(t)* and *a(t)* respectively. The dynamics of a TCP connection can be characterized by the following equations: $s(t) = a(t) + \dot{w}(t)$ (1)

$$\dot{w}(t) = \begin{cases} a(t) & \text{, if the traffic source is in slow start} \\ a(t)/w(t) & \text{, if the traffic source is in congestion avoidance} \\ 0 & \text{, if the traffic source is in saturation} \end{cases} \quad (2)$$

Consider a broadband network in which the edge router connects end user applications to the core internet through upstream (i.e., end user to core network) and downstream access channels. The ACK pacing algorithm proposed here deals primarily with traffic control issues on the downstream. Denote the capacity of the bottleneck link between the edge router and the access network as *C*, and the downstream buffer occupancy at the edge router as *B(t)*. We use subscript *i* to index the connections that share the same bottleneck link, and τ_i to represent the delay between the i^{th} traffic source and the edge router. From Eqs. (1) and (2), we obtain

$$\dot{B}(t) = \sum_{i=1}^{N} (a_i(t-\tau_i) + \dot{w}_i(t-\tau_i)) - C \quad (3)$$

$$\sum_{i=1}^{N} a_i^{(1)}(t-\tau_i) = \tilde{\alpha}(t)(C + \dot{B}(t)) \quad (4)$$

where $\frac{1}{2} \le \tilde{\alpha}(t) \le 1$ is a time-varying parameter depending on the TCP congestion states of active TCP connections. The network employing the ACK pacer can be modeled as a feedback system as shown in Figure 1:

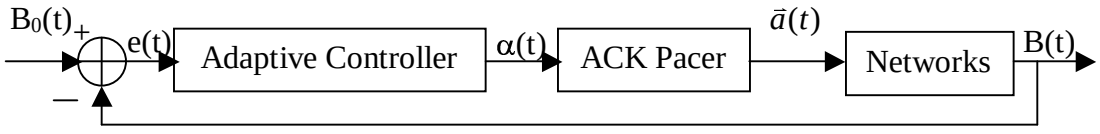

Figure 1 ACK Pacer System Model

In the ACK Pacer system above, we use the downstream queue length *B(t)* to represent the load of the network. The basic control strategy for the ACK pacer is as follows:

1. ACK Pacer is triggered as soon as the queue length exceeds a threshold B_1;
2. ACK Pacer is deactivated when the queue length falls below a threshold B_3, here $B_3 < B_1$;
3. While the ACK Pacer is active, the 'Adaptive Controller' adjusts $\alpha(t)$ to keep track of the time-varying parameter $\tilde{\alpha}(t)$ in (4). In addition, the ACK Pacer allocates the ACK releasing rates $\bar{a}(t) = [a_1(t), \cdots, a_N(t)]$ to the connections based on (3) and (4), in order to achieve a desired *B(t)* trajectory, i.e., $B(t) \approx B_0(t)$, and keep *B(t)* below B_2.

The dynamics of the 'Networks' block is characterized by (1) and (2). The design of $B_0(t)$ is try to slow down the increasing speed of *B(t)*, e.g. $B_0(t) = B_1 + (B_2 - B_1)(1 - e^{-\beta t})$. Various control methods can be applied in the 'Adaptive Controller' block to achieve this goal. One possible implementation is to make the adjustment proportional to the error signal *e(t)*, i.e., $\alpha_{n+1} = \alpha_n + e(t) \cdot \Delta\alpha$. If all the connections are in the same congestion stages, it can be verified that $\alpha(t)$ will converge to $\tilde{\alpha}(t)$ following the above α-updating method.

Simulation results and Conclusion The effectiveness of the ACK pacer is tested preliminarily under FTP traffic, because these long TCP sessions are the ones that needed to be contained the most.

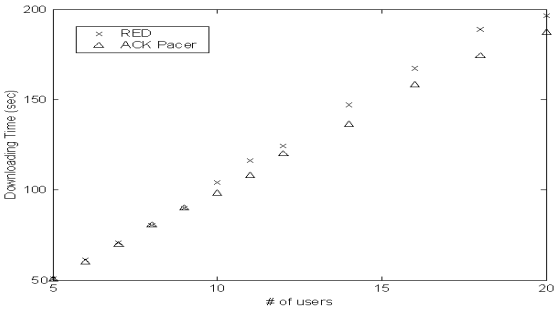

Figure 2 Downloading Time(sec) vs. # of users

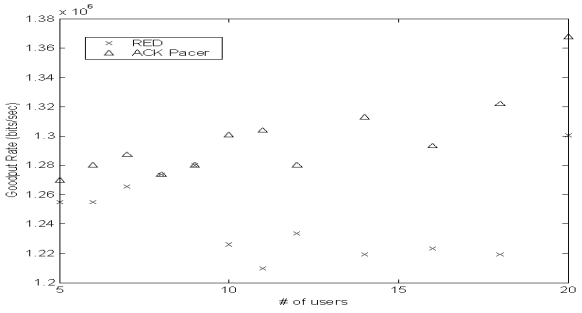

Figure 3 Goodput Rate(bits/sec) vs. # of users

From Figure 2 and Figure 3 it can be seen that ACK Pacer leads to shorter downloading times and higher goodput rates compared to RED.

When compared to the per-flow token buckets in the forward (downstream) path, we may drastically reduce buffer requirement by holding ACKs in the reverse path rather than holding packets in the forward direction. The ACK pacer is also superior to RED because dropping packets causes TCP timeouts and retransmissions, which may result in more congestions and damage fairness. 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 [2]. Most importantly, the presented algorithm adapts to the aggregate traffic fluctuation rather than to individual flow states as in [2] and [3], and thus may be advantageous in terms of implementation. Also using the ACK Pacer doesn't require changing TCP implementation in end-users as with ECN.

References

[1] 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

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

[3] 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