A Robust Nonlinear PI Controller for Improving AQM Performance

XiaoLin Chang*, Jogesh K. Muppala*, Jen-te Yu*

*Dept. of Computer Science, Hong Kong University of Science and Technology, Clear Water Bay, Kowloon, Hong Kong
Email: {changxl,muppala}@cs.ust.hk

Abstract—In this paper a simple robust Proportional-Integral (R-PI) controller is proposed for Active Queue Management (AQM). We assume that TCP/AQM dynamics can be described by the linearized TCP/AQM model [7]. R-PI aims to address the tradeoff between responsiveness and stability and the tradeoff between responsiveness and high link utilization over a large range of structured and unstructured uncertainties. This controller achieves these goals by varying its control parameters according to the system state. We show that the closed-loop system is asymptotically stable as long as the control parameters are time-invariant and varying in a range. Extensive simulation results demonstrate the robust ability of R-PI compared with some other AQM mechanisms in the literature.

Keywords- Asymptotically Stable; Robust; Adaptive control; Popov criterion

I. INTRODUCTION

Active Queue Management (AQM) mechanisms are employed in the Internet by the routers to provide good congestion control and provide support for Quality of Service (QoS) [1]. Random Early Detection (RED)[2] was among the first AQM to be introduced into the Internet routers. Since then several enhancements [3] to RED and new AQM mechanisms [4], which are based on heuristic techniques, have been proposed. Recently, some controllers for AQM based on feedback control theory have been developed, such as Integral (I) controller in [5], Proportional-Derivative (PD) controller in [6], PI controller in [7] and PID controller in [9]. These static P/I/D controllers can achieve satisfactory transient response (including small rise time, small settling time and small overshoot), and small steady-state error simultaneously in some situations. However, they are unable to satisfy these performance specifications under varying network conditions. Specifically, they do not address the tradeoff between responsiveness and stability.

The authors in [10] and [11] propose self-tuning schemes to improve the ability of these P/I/D controllers in the varying networks. These schemes use on-line algorithms to estimate the network parameters, which are used as real values to tune the control parameters in the controllers. Such adaptive schemes significantly improve the system performance. However, they may be time-consuming and bring in additional considerations. Then the experimental evaluation of R-PI, and some design considerations are given.

This work described in this paper has been supported by the Research Grants Council of Hong Kong SAR, China (Project No. DAG02/03.EG20)

IEEE Communications Society 0-7803-8533-0/04/$20.00 (c) 2004 IEEE
\[
\begin{align*}
\delta \dot{W}(t) &= -\delta W(t) + \frac{2N_q(t)}{C(t)R_t(t)} \cdot \delta p(t) + \frac{C^2(t)R_t(t)}{2N_q(t)}, \\
\delta \dot{q}(t) &= \delta W(t) + \frac{N_q(t)}{R_0(t)} - \delta q(t)>
\end{align*}
\] (1)

where
\(C(t)\) bottleneck link capacity (packets/sec)
\(N(t)\) number of adaptive connections at time \(t\)
\(R(t)\) round-trip delay at time \(t\) (second)
\(p(t)\) dropping/markig probability at time \(t, \in [0,1]\)
\(q(t)\) instantaneous queue length at time \(t\) (packets)
\(q_{tr}\) target queue size at time \(t\) (packets)
\(\delta t\) queue buffer size (packets)
\(W(t)\) window size of a TCP connection at time \(t\) (packets)
\((W_0(t), q_{tr}(t), p_0(t), C_0(t))\) system operating points in steady state
\(\delta w = W - W_0\)
\(\delta q = q - q_0\)
\(\delta p = p - p_0\)

\[
P(s) = \frac{K}{a_1s^2 + a_2s + a_3}
\] (2)

where \(K = C^2R_0^3 / (2N_0)^3\), \(a_1 = R_0C / 2N_0\), \(a_2 = R_0 + R_0C / 2N_0\), \(a_3 = 1\)

B. R-PI controller

As mentioned previously, the main disadvantage of employing a static controller is the difficulty in addressing the tradeoff between responsiveness and stability in the varying network. That is, the static controller may exhibit either a fast response with large queue fluctuations or a slow response with small queue fluctuations. Both these two results could result in

\[
\varphi(t) = \begin{cases} 1 + k_0 \left(\frac{2}{1 + \exp\left(-\frac{|e(t)|}{k_2}\right)}\right) - 1.0 & \text{if } k_i < |e(t)| < \max\{2* q_{tr}, q_{lim} - q_{tr}\} \\
1.0 & \text{otherwise}
\end{cases}
\] (3)

When R-PI is applied, the formula of computing marking probability is shown in Eq. (4).

\[
p(t) = k_i \left[\varphi(t)e(t)\right] + k_1 \int_0^t \left[\varphi(t)e(t)\right] dt
\] (4)

C. Stability analysis

We give the following proposition to specify the condition under which the nonlinear system in Figure 2. is stable. This condition gives the guidelines for selecting \(k_0\).

**Proposition:** Assume that a static PI controller has been designed to stabilize the system in Fig. 1 and \(\varphi(t)\) is defined as given in Eq. (3). Then, there must exist a positive value \(k_{max}\) such that as long as \(k_0 + 1 < k_{max}\), the system in Figure 2. is be asymptotically stable.

**Proof:** We define \(L(s) = F(s)P(s)\exp(-sR)\). We borrow the method of selecting \(k_0\) and \(k_i\) in [8] to stabilize the system in Figure 1. Thus, we obtain the real and imaginary parts of \(L(s)\) as shown in Eq. (5).

\[
\begin{align*}
\text{Re} L(j\omega) &= Kk_1 \left[\sin(\omega R) + \omega R \cos(\omega R)\right] \\
&- \omega (1 + \omega^2R^2) \\
\text{Im} L(j\omega) &= Kk_1 \left[\cos(\omega R) - \omega R \sin(\omega R)\right] \\
&- \omega (1 + \omega^2R^2)
\end{align*}
\] (5)
Eq. (5) shows that $a_kml(j\omega) \rightarrow K_k\phi_0$ when $\omega \rightarrow 0$. So $L(s)$ is stable. Assuming $\beta = \arctan(\alpha R)$, then we obtain

$$
\Re(L(j\omega)) = \frac{KK_k \sin(\beta + \alpha R)}{\omega^2 + \omega R^2 + R^2}
$$

$$
\Im(L(j\omega)) = \frac{-KK_k \cos(\beta + \alpha R)}{\sqrt{1 + \omega^2 R^2}}
$$

(6)

Given $(C,R,N)$, $Re(L(j\omega))$ and $a_kml(j\omega)$ must be bounded when $\omega$ is varying from 0 to infinite. This means that in the plane $(Re(L(j\omega)),a_kml(j\omega))$ there must exist a line intersecting the negative horizontal axis to make the Popov plot to the right of this line for all $\omega \geq 0$. We define $-1/k_{\max}$ as a value less than the intersection point of the curve and the horizontal axis. This means that, for any stable linear system in Fig. 1, there exists a real number $\eta$ such that $\Re[\imath(1+j\eta)\phi(\omega)] + 1/k_{\max} > 0$ for all $\omega \geq 0$. Let $k_{\eta} = 1<k_{\max}$. Thus, $\phi(t)$ satisfies the sector condition [13]. According to the Popov stability criterion, the system consisting of Eq. (1) and Eq. (2) is asymptotically stable.

As a numerical illustration consider the case of $(N_0=60, C_0=15Mbps, R_0=0.246ms)$ used in [8]. Figure 3. gives the Popov plot of $L(j\omega)$ and a line $y=3*(x+1/k_{\max})$, where $x=\Re(L(j\omega))$, $y = a_kml(j\omega)$ and $k_{\max}$ is 8.0. The intersection point of the curve and the horizontal axis is -0.11394. That means that if the system in Fig. 1. has been stable in the vicinity around the equilibrium point $(N_0=60, C_0=15Mbps, R_0=0.246ms)$, the system in Fig. 2. with Eq. (3) is asymptotically stable in this vicinity as long as $k_{\eta}$, $k_1$, $k_2$ are positive constants and $k_{\eta}+1<k_{\max}$. In the following simulations, we choose $k_{\eta}=7$, $k_1=100$, $k_2=50$.

![Figure 3. Popov plot of ReL(j\omega) versus a_kml(j\omega)](image)

**D. Design considerations**

This subsection provides several design considerations.

1. How to accommodate the possible traffic burst and traffic noise. Some AQM mechanisms like [2], [5], and [6] use $\phi(t)$ to adapt $p(t)$ in order to filter the traffic burst and noise. The problems with using $\phi(t)$ have been pointed out by some others like [8]. For example, $\phi(t)$ may not be able to reflect the impact of changes caused by control decisions in a timely manner and may lead to control in the wrong direction. Our simulation results in the next section also show this. Actually, those AQM mechanisms using $q(t)$ to adapt $p(t)$ periodically can accommodate some traffic burst and noise. The process of periodically computing $p(t)$ is a process of averaging $p(t)$. Compared with directly averaging $q(t)$, averaging $p(t)$ can accommodate some burst and noise without compromising system performance.

2. How to address the tradeoff between responsiveness and stability and the tradeoff between responsiveness and high link utilization. Note that (i) the premise of R-PI is that a static PI controller that stabilizes the system in Figure 1. has been developed; (ii) the network stability is a kind of quasi-stability. Thus, when $|e(t)|$ is small, $\phi(t)$ is set to 1.0 in order to maintain the performance with the static PI controller. We use $k_1$ to define the range in which the static PI is active. When $|e(t)|$ is large, $\phi(t)$ amplifies $|v(t)|$ substantially to generate a large corrective action to drive $q(t)$ to $q_{ref}$ quickly.

However, we observe that it is possible that noise or burst may cause $|e(t)|$ above $k_1$ temporarily. In these situations $\phi(t)$ should produce small gain. We use $k_1$ and $k_2$ to define the range of small increase. It is known that there is a time delay before the end hosts know the occurrence of incipient congestion. That means that some more undesirable packets are sent out by the sources after incipient congestion has occurred. This may cause the unnecessary increase in $p(t)$. This problem exists in PI controller [8]. In order to prevent from worsening in R-PI, when $|e(t)|$ is above a level $\phi(t)$ is set back to 1.0. In the following simulations, we set this level as $\phi_{max}(2*\eta_{ref}, q_{max} - q_{ref})$. In addition, we use $|e(t)|$ rather than $e(t)$ in Eq. (3) in order to compensate the overshoot in marking probability to some degree, especially when the traffic load is suddenly decreased.

**III. SIMULATION RESULTS**

In this section, we carry out simulations with ns-2 [14] simulator to evaluate R-PI controller. We compare it with ARED [3], PD [6], PI [8], and P1 [12], among which ARED and PD use $\phi(t)$ to adapt $p(t)$, PI, R-PI and PI use $q(t)$. By comparing with PI and PD, we aim to examine the improvement with R-PI in fast transient response while maintaining small steady-state error. By comparing with P1, we aim to examine the ability of R-PI in addressing the tradeoff between fast response and high link utilization. In order to investigate the advantage of directly controlling $q(t)$ over controlling $\phi(t)$, we do simulations with ARED and a variant of PD which uses $q(t)$ to adapt $p(t)$. In the following, $PD_a$ represents PD controller using $\phi(t)$ as control input; $PD_c$ represents PD controller using $q(t)$ as control input.

**TABLE I.** lists some notations and the default parameter settings in the corresponding mechanisms. We set these control parameters according to the authors’ recommendation. Considering the difficulty in setting parameters, we do simulations using the network scenarios in [6] and [8], respectively. Figure 4. is the network topology. Unless otherwise specified, in all the simulations, the adaptive sources use TCP/Reno; the initial window size of each TCP connection is set to 100 packets; all the adaptive connections and the routers are ECN-enabled. Performance is evaluated using the instantaneous queue length $q(t)$.

**A. Constant TCP flows + burst HTTP + burst UDP**

This simulation aims to investigate the ability of each scheme in the network with constant large number of long-lived TCP flows and with burst HTTP+UDP traffic as noise. We use one of the network scenarios in [6]. The bottleneck link capacity is 45Mbps, the packet size is 1000 bytes, $q_{ref}$
=400packets,  \( q_{lim}=1125 \)packets. So in ARED, \( w_p=0.00017776 \), \( \min_a=200 \)packets, \( \max_a=600 \)packets. In P'I, \( q_{thresh}=650 \), \( q_{ref}=250 \)packets. There are 1000 TCP sources (connections), 400 HTTP sessions (connections) and 500 UDP sources. Each TCP source generates an infinite FTP bulk data transfer with different RTTs uniformly distributed in (50, 150) ms. The packet size is 1000bytes. The RTTs of HTTP connections are uniformly distributed in (50, 300) ms. Each session has 250 pages and each page has one object. The page interval in a session follows an Exponential Distribution with mean of 1.0s. The RTTs of the UDP flows are uniformly distributed in (30, 150) ms. Each HTTP session has 400 pages and the burst times have mean of 0.5 s. The packet size is set to 210 bytes, and the sending rate during on-time is 50kbps. The mean packet size at the router is 500 bytes. The simulation lasts 450s. All the UDP and long-lived TCP connections start randomly in the first second. The HTTP connections start in sequence. The first starts at the 0\( ^{th} \) second. The next starts after 0.1s, and so on. Other settings are as default. Figure 5. is queue length variation. The dark curves represent \( q(t) \) variation and the light curves represent \( \overline{q}(t) \) variation.

### TABLE I.

<table>
<thead>
<tr>
<th>Controller</th>
<th>Parameters</th>
</tr>
</thead>
<tbody>
<tr>
<td>PI [8]</td>
<td>( a=1.816<em>10^{-6} ), ( b=1.822</em>10^{-6} ), ( T_s=1/160 ) s</td>
</tr>
<tr>
<td>R-PI</td>
<td>( a=1.816<em>10^{-6} ), ( b=1.822</em>10^{-6} ), ( k_0=7.0 ), ( k_1=100.0 ), ( k_2=50 ), ( T_s=1/160 ) s</td>
</tr>
<tr>
<td>P'I [12]</td>
<td>( a=1.816<em>10^{-6} ), ( b=1.822</em>10^{-6} ), ( T_s=1/160 ) s, ( \alpha_0=\beta_0=2.0 )</td>
</tr>
<tr>
<td>PD_a [6]</td>
<td>( L=q_{ref}*0.8 ), ( k_0=0.0002 ), ( k_2=0.08 ), filter_gain = 0.0025, ( T_s=1/100 ) s</td>
</tr>
<tr>
<td>PD_c [6]</td>
<td>( \min_a=q_{ref}*0.5 ); ( \max_a=q_{ref}*1.5 ), ( w_p=1-\exp(-1/C) ), all other parameters are set to the default values in NS2</td>
</tr>
</tbody>
</table>

| ARED [3]   |  |

\[ S_1 \rightarrow 10 \text{Mbps} \rightarrow \text{C} \rightarrow 20 \text{Mbps} \rightarrow \text{D}_1 \rightarrow \text{E} \rightarrow \text{D}_n \rightarrow \text{S}_2 \]

Figure 4. Network topology

B. Large variation in traffic load

This experiment aims to examine the performance of each scheme when the number of long-lived TCP flows is varying in (200, 1000) in the network, where there exist disturbances caused by 200 HTTP and 200 UDP burst connections. Network settings and control parameter settings in each scheme are same as in Simulation A. Each TCP source generates an infinite FTP bulk data transfer with different RTTs uniformly distributed in (160, 240) ms. The packet size is 1000bytes. The RTTs of HTTP connections are uniformly distributed in (50, 300) ms. Each HTTP session has 400 pages and the page interval in a session follows an Exponential Distribution with mean of 1.0s. The RTTs of the UDP flows are uniformly distributed in (30, 150) ms. Each UDP source follows an exponential ON/OFF traffic model, both the idle and the burst times have mean of 0.5 s. The packet size is set to 210 bytes, and the sending rate during on-time is 50kbps. The mean packet size at the router is 500 bytes. The simulation lasts 450s. All the UDP and long-lived TCP connections start randomly in the first second. The HTTP connections start in sequence. The first starts at the 0\( ^{th} \) second. The next starts after 0.0005s, and so on. For FTP flows, 200 flows are started at 0\( ^{th} \) second, and increased in steps of 200 flows at each 50 second interval until \( t=250 \)th second. From \( t=250 \)th second, the number of flows is reduced by 200 at each 50 second interval until \( t=400 \)th second. Figure 6. is queue length variation. The dark curves represent \( q(t) \) variation and the light curves represent \( \overline{q}(t) \) variation.

C. Summary and discussion

The above simulation results show that (1) ARED and PD have a fast transient response than PI. However both of them do not address the conflict between responsiveness and stability in all the above network scenarios. (2) When PD controller is applied, using \( \overline{q}(t) \) does not produce better performance than using \( q(t) \). (3) PI controller can lead to a stable system in all the network scenarios but it has a slow transient response. (4) P'I controller has a fast transient response. It can reach the steady state but it does not solve the conflict between responsiveness and high link utilization. In the above two groups of experiments, the performance with PI' is worse than that of PI, especially in terms of link utilization. Due to space limit, we do not show the results of link utilization with each scheme. (5) In all the simulations, R-PI controller performs the best among the five mechanisms. It addresses the conflicts existing in PD and P'I controllers and makes great improvement of the ability of PI over a wide range of network dynamics.
IV. CONCLUSION

In this paper we present a simple robust Proportional-Integral (R-PI) controller. The varying gain enables the controller to adapt its response according to the system state. By using Popov Stability Criterion, we analyze the asymptotic stability of the closed-loop system. Some features demonstrate its simplicity. Extensive simulation results display its robust ability in addressing (1) the tradeoff between responsiveness and stability; (2) the tradeoff between responsiveness and high link utilization over a wide range of network uncertainties.

REFERENCES


