Comparison of PI Controllers Designed for the Delay Model of TCP / AQM Networks

Hakkı Ulaş Ünal[a,*], Daniel Melchor-Aguilar[b,1], Deniz Üstebay[c], Silviu-Iulian Niculescu[d], Hitay Özbay[e]

[a] *Department of Electrical and Electronics Engineering, Anadolu University, 26555, Eskişehir, Turkey*
[b] *Division of Applied Mathematics, IPICyT, San Luis Potosi, SLP, México*
[c] *Department of Electrical and Computer Engineering, McGill University, Montreal, Quebec, Canada*
[d] *Laboratoire des Signaux et Systèmes, CNRS-SUPELEC, 3 rue Joliot Curie, 91190, Gif-Sur-Yvette, France*
[e] *Department of Electrical and Electronics Engineering, Bilkent University, 06800, Ankara, Turkey*

Abstract

One of the major problems of communication networks is congestion. In order to address this problem in TCP/IP networks, *Active Queue Management* (AQM) scheme is recommended. AQM aims to minimize the congestion by regulating the average queue size at the routers. To improve upon AQM, recently, several feedback control approaches were proposed. Among these approaches, PI controllers are gaining attention because of their simplicity and ease of implementation. In this paper, by utilizing the fluid-flow model of TCP networks, we study the PI controllers designed for TCP/AQM. We compare these controllers by first analyzing their robustness and fragility. Then, we implement these controllers in ns-2 platform and conduct simulation experiments to compare their performances in terms of queue length. Taken together, our results provide a guideline for choosing a PI controller for AQM given specific performance requirements.

Keywords: Active Queue Management, Delay systems, PI Controller, Network Congestion, TCP/IP.

1. Introduction

Routers in a network transmit incoming packets to the destinations over links which have finite bandwidth. Thus, links can get congested if the amount of incoming packets exceeds the link capacity. When there are congested links in a network, the buffers of the routers might overflow and, consequently, new incoming packets might be lost. To address the congestion problem in TCP/IP networks, queue management and scheduling algorithms are required at the routers. The traditional queue management technique at a router, known as *tail drop*, sets a maximum queue length in terms of packets and accepts packets for the queue until it overflows, then drops subsequent incoming packets until the queue decreases. Tail drop has some drawbacks such as flow synchronization, link

*Corresponding Author
Email addresses: `huunal@anadolu.edu.tr` (Hakkı Ulaş Ünal), `dmelchor@ipicyt.edu.mx` (Daniel Melchor-Aguilar), `deniz.ustebay@mail.mcgill.ca` (Deniz Üstebay), `Silviu.Niculescu@lss.supelec.fr` (Silviu-Iulian Niculescu), `hitay@bilkent.edu.tr` (Hitay Özbay)
[1] Currently on a sabbatical leave in the Mechatronic Section, CINVESTAV-IPN, 07360, México D.F., México

under-utilization, and long end-to-end delay [1]. In order to overcome these drawbacks, *Active Queue Management* (AQM) scheme is recommended in [1]. The well-known AQM scheme is *Random Early Detection* (RED), which drops packets with a probability that depends on the average queue length. Since RED drops packets by detecting the congestion, it significantly improves the link utilization compared to tail drop scheme. In addition, the flow-synchronization is eliminated and the effects of burst traffic are attenuated [2]. However, tuning RED parameters is a difficult task; if these parameters are not chosen carefully, the performance of RED can degrade, and, the system may become unstable. The stability of RED is investigated in [3] and [4] by studying the maximum value of the packet marking probability that does not cause instability. As shown in [3], TCP/RED system becomes unstable if the round-trip delay and link capacity increase significantly, and/or the number of TCP sessions decreases drastically.

In order to obtain better performance compared to RED, by using the linearized fluid-flow model of TCP proposed in [5], several feedback control based advanced AQM controllers are proposed in the literature e.g., [6, 7, 8, 9] and references therein. In [6], an $\mathcal{H}^\infty$ AQM controller was constructed by solving two-block $\mathcal{H}^\infty$ minimization problem to regulate the queue length against the variations of the plant parameters. By using the μ-synthesis approach in [7], an $\mathcal{H}^\infty$ AQM controller was designed considering delay-free part. In [8], by designing a robust observer, an $\mathcal{H}^\infty$ state feedback controller was designed to solve the same problem. In [9], an $\mathcal{H}^\infty$ state feedback was designed in order to solve the problem considering also the disturbances on the available bandwidth. However, it appears that these proposed controllers are not easy to implement in real networks due to their computational complexities.

In [10, 11], a PI AQM controller design was proposed by using the small-gain theorem. It was shown there that PI controllers provide good responses in achieving AQM performance requirements. Based on this and their ease of implementation in real networks, several PI AQM controller designs have been proposed following those works, see for instance, [12, 13, 14, 15]. Note that, from the practical implementation of a controller, it is required to keep the stability of the closed-loop system under round-off errors during implementation. A controller for which the closed-loop system can be destabilized by small perturbations in the controller coefficients is said to be *fragile* (see, e.g., [16]). There are only a few studies addressing the fragility problem of AQM controllers. To the best of the authors' knowledge, the first study was performed in [14], where a method to compute the largest available intervals for the PI controllers parameters have been developed. Recently in [15], using the complete characterization of the set of all stabilizing PI controllers and its corresponding geometric properties, a new method for tuning the parameters of PI AQM controllers has been proposed. Such an approach allows us to design a PI controller stabilizing the network against perturbations in the network parameters. In addition, since the approach gives a simple procedure to determine the controller coefficients providing the maximum parametric stability margin in the controller's gains space, the designed PI controller stabilizes the network also against the perturbations on the coefficients of the controllers.

Although PI controllers are widely used in many control applications including complicated systems (see e.g., [17]), there does not exist a generally accepted tuning methodology. In addition, determining the PI parameters is a difficult task for many applications [18]. As pointed out in several surveys (see [18] and references therein), a high percentage of PI controllers have poor performances in many applications, due to bad controller tuning. Many tuning methods do not consider some restrictions such as unmodelled dynamics, non-linearities, and presence of delay. Another reason for the performance degradation of the PI-controllers is the uncertainties in the controller components due to the aging problem. This means that fragility of the controller should be taken into account.

In this paper, we compare several PI controllers designed for TCP/AQM considering some performance requirements with arising problems in practice such as fragility and robustness. Some of these PI controllers are currently available for TCP/AQM given in [11, 15, 14] and the other PI controllers are designed in the paper by utilizing the approaches in [19, 20, 21, 22, 23]. The designed PI controllers are based on considering the transfer function of the linearized model of TCP as an integrating system or a second order system with delay. In order to compare the robustness and fragility of the controllers, the stability region of all stabilizing PI AQM controllers for the considered network presented in [15] is utilized. For a performance comparison, the controllers are implemented in ns-2[2] and validated under different realistic scenarios considering various performance metrics.

It is worth mentioning that there also are propositions of PD and PID controllers for AQM schemes, see for instance, [25, 26, 27, 28]. However, to the best of the authors' knowledge, the boundary of the stability region in the controller's parameters space of such controllers is not completely known and, therefore, an appropriate comparison of robustness and fragility issues can not be made as we performed here for PI AQM controllers.

The remainder of the paper is organized as follows. The mathematical model of the TCP fluid-flow model is given in Section 2. Various PI controller design methods for AQM schemes are summarized in Section 3. Section 4 provides a theoretical analysis of these PI controllers as well as simulation results comparing their performance in ns-2 platform. Concluding remarks are presented in Section 5.

2. Mathematical model of the TCP flows

In this section, we present the dynamical fluid-flow model developed by [11] for describing the behaviour of TCP/AQM networks. This model considers a network of N homogeneous TCP-controlled sources and a single router. The average values of the key network variables are modelled by the following coupled and time-delayed non-linear differ-

[2]ns-2 is a discrete event simulator that captures the stochastic and non-linear nature of the network dynamics [24].

ential equations:

$$\dot{W}(t) = \frac{1}{R(t)} - \frac{W(t)}{2}\frac{W(t-R(t))}{R(t-R(t))}p(t-R(t))$$
$$\dot{q}(t) = \begin{cases} -C + \frac{N(t)}{R(t)}W(t), & q > 0 \\ \max\{0, -C + \frac{N(t)}{R(t)}W(t)\}, & q = 0 \end{cases}, \quad (1)$$

where $W(t)$ is the average TCP window size (packets), $N(t)$ is the number of TCP sessions, $R(t) = \frac{q(t)}{C} + T_o$ is the round-trip time delay (s), $q(t)$ is the average queue length (packets), C is the link capacity (packets/s), T_o is the propagation delay (s), and $p(t)$ is the probability of packet marking. Since the equations in (1) are non-linear, the transfer function for (1) can be obtained by making a linearization around their equilibrium points. In order to obtain the transfer function of (1), let $N(t) = N_o$, $C = C_o$, $W(t) = \delta_W(t) + W_o$, $q(t) = \delta_q(t) + q_o$, and $p(t) = \delta_p(t) + p_o$, where W_o, q_o, p_o are the equilibrium points determined by the nominal values. Then the transfer function from δ_p to δ_q can be obtained as in [11]:

$$G_{pq}(s) = \frac{R_o C_o K}{(R_o s + \frac{1}{K})(R_o s + 1)} e^{-R_o s}, \quad (2)$$

where $K = \frac{R_o C_o}{2N_o}$, $R_o = T_o + \frac{q_o}{C_o}$. Therefore, by (2), it is possible to construct a closed-loop feedback system by designing PI controllers using various approaches, which are summarized in Section 3, for TCP/AQM model.

3. PI controller design approaches for the delay model of TCP/AQM

In this section, several PI controller design approaches are summarized for the delay model of TCP/AQM. The stabilizing PI controllers are designed to provide a packet marking probability function as AQM strategy for regulating the average queue length at a desired operation point. Each PI controller has the structure

$$K_{pi}(s) = K_p + \frac{K_i}{s},$$

where K_p and K_i correspond to the proportional and integral gains, respectively.

3.1. PI controller design by Ziegler-Nichols approach

Ziegler-Nichols approach is an empirical PID tuning method, which is based on the following steps:

- Set $K_i = 0$. Stabilize the feedback system for a step reference q_o with a very small gain K_p.
- Gradually increase K_p until output of the controller starts to oscillate. Then, record the gain K_p as K and oscillation period as T.

Then, the PI controller parameters are determined as $K_p = 0.45K$ and $K_i = \frac{K_p}{1.2}T$ [19].

3.2. PI controller design by Panda et al. ([20])

In this approach, PI controller design is presented for first order systems with time-delay considering the robustness by using the *Internal Model Control* (IMC) with Padé approximation. In order to design such a PI controller for AQM scheme, the approximation of the plant is obtained as

$$G_{pq}(s) \approx \frac{R_o C_o K^2}{\tau_m s + 1} e^{-D_m s},$$

where $\tau_m := \left(0.828 + \frac{0.812}{K} + 0.172 R_o K e^{-\frac{6.9}{K}}\right)$ and $D_m := 1.116\frac{R_o K}{K + 1.208} + R_o$. Then, the controller parameters are determined as $K_p = \frac{2\tau_m + D_m}{2R_o C_o K^2 \lambda}$ and $K_i = \frac{1}{R_o C_o K^2 \lambda}$, where $\lambda = \max\{\tau_m, 1.7R_o\}$ (see also [29] for the details of the choice of λ).

3.3. *PI controller design by Hollot et al. ([11])*

This PI controller design is proposed to stabilize the feedback system with plant (2) against the high-frequency TCP parasitic. It is shown that the designed controller also stabilizes the system against the larger TCP sessions and smaller link capacity and round-trip time delay compared to nominal values, therefore, the resulting controller is robust. In this design method, the zero of PI controller is chosen to coincide with the corner frequency of the TCP window dynamic. Hence, if $L(s) := K_{pi}(s)G_{pq}(s)$ and $K_p = \frac{K_i}{z}$, then z is chosen as $z = \frac{1}{R_oK}$. Therefore, the phase of the open-loop system depends on the TCP queue dynamics, and the round-trip time delay. In order to meet the crossover condition, i.e. $|L(jw_g)| = 1$, K_i is chosen as $K_i = w_g z \left|\frac{1 + jR_ow_g}{C_oK}\right|$. Then, the phase of the open-loop transfer function can be written as

$$\angle L(jw_g) := -90^\circ - \frac{180}{\pi}\beta - \arctan\beta,$$

where $\beta := w_gR_o$. Therefore, to design a stabilizing PI controller, β should satisfy $\angle L(jw_g) - 180^\circ > 0$. Hence, once w_g is chosen for design purposes, i.e. large bandwidth for a fast response, the stabilizing PI controller can be obtained provided that β satisfies $\angle L(jw_g) - 180^\circ > 0$. Note that, large bandwidth requires larger β, which decreases the phase margin of the open-loop system, hence, deteriorates the system performance.

3.4. *PI controller design by Melchor-Aguilar, Niculescu ([15])*

In this approach, first the set of all robustly stabilizing PI controllers for the linearized model is determined. Then, by utilizing this set, a tuning methodology is presented to determine a non-fragile PI AQM controller. In order to design such a controller, let us introduce

$$\sigma(t) := \int_0^t (q(\nu) - q_o)d\nu. \tag{3}$$

Then, by linearizing the augmented system (1)-(3) with the control law $\delta_p(t) = K_p\delta_q(t) + K_i(\sigma(t) - \frac{1}{K_i}(p_o - K_pq_o))$, around the equilibrium points, it can be shown that the closed-loop system is exponentially stable if and only if

$$f(s) = s^3 + \frac{1}{R_o}\left(1 + \frac{N_o}{R_oC_o}\right)s^2 + \frac{2N_o}{R_o^3C_o}s + \left[\frac{N_o}{R_o^2C_o}s^2 + \frac{C_o^2}{2N_o}(K_ps + K_i)\right]e^{-R_os},$$

has no zeros with non-negative real parts [30]. Following [15], the set of all robustly stabilizing PI controllers for the linearized model of TCP can be described as

$$K_p(\omega) = \frac{2N_o}{C_o^2}\left[\left(\omega^2 - \frac{2N_o}{R_o^3C_o}\right)\cos(\omega R_o) + \frac{\omega}{R_o}\left(1 + \frac{N_o}{R_oC_o}\right)\sin(\omega R_o)\right] \tag{4}$$

$$K_i(\omega) = \frac{2N_o\omega}{C_o^2}\left[\frac{\omega}{R_o}\left(1 + \frac{N_o}{R_oC_o}\right)\cos(\omega R_o) + \left(\frac{2N_o}{R_o^3C_o} - \omega^2\right)\sin(\omega R_o) + \frac{N_o\omega}{R_o^2C_o}\right] \tag{5}$$

where $w \in [\bar{w}, w^*]$. Here, $\bar{w}$ and w^* are respectively the solution of

$$\tan(\omega R_o) = \frac{\frac{2N_o}{R_o^3C_o} - \omega^2}{\frac{\omega}{R_o}(1 + \frac{N_o}{R_oC_o})},$$

and

$$\frac{N_o}{R_o^2C_o\omega} = \frac{R_o\omega\sin(\omega R_o) - \cos(\omega R_o)}{R_o\omega(1 + \cos(\omega R_o)) + 2\sin(\omega R_o)},$$

where $w \in \left(0, \frac{\pi}{2R_o}\right)$. Then, the stability region of all PI controllers for TCP/AQM is determined by the coordinate axes $K_p = 0$ and $K_i = 0$ and the curve defined by (4) and (5). Now, once the stability region is obtained, in order to determine the non-fragile controller, the nominal controller parameters are chosen to put the largest circle in this region, where the radius of this circle represents the maximum l_2 parametric stability margin in the controller's gain space.

3.5. PI controller design by Poulin, Pomerleau ([21])

This approach is proposed for the integrating systems with time-delay. It is based on limiting the maximum peak-resonance ($\mathrm{M_r}$) of the closed-loop transfer function to minimize the integral time of the absolute error (ITAE) due to the output step disturbance. In order to achieve this, the controller parameters are adjusted such that the transfer function of the open-loop system at the frequency where maximum phase occurs is tangent to the ellipse in the Nichols chart specified by the desired $\mathrm{M_r}$ of the closed-loop system. To design such a PI controller for AQM, the considered plant has structure

$$G_{pq}(s) \approx \frac{C_o K}{s(R_o s + 1)} e^{-R_o s}. \tag{6}$$

Note that this approximation is a well approximation of (1) if $K \gg 1$ [12]. Following [21], the PI parameters are chosen as $K_p = \frac{A_{max}}{2C_o K}\sqrt{\frac{R_o+2T_i}{T_i^2 R_o + 2R_o^2 T_i}}$, $K_i = \frac{K_p}{T_i}$, where $T_i = \frac{32R_o}{(2\phi_{max}+\pi)^2}$, $\phi_{max} = \arccos\left(\frac{\sqrt{10^{0.1\mathrm{M_r}}-1}}{10^{0.05\mathrm{M_r}}}\right) - \pi$, $A_{max} = \frac{10^{0.05\mathrm{M_r}}}{\sqrt{10^{0.1\mathrm{M_r}}-1}}$. The optimal M_r values, which satisfy the design criterion, are plotted in Figure 2 of [21] with respect to plant parameters. By utilizing that plot and considering (6), M_r is chosen 4.25.

3.6. PI controller design by Üstebay, Özbay ([14])

This approach is based on the work of [31] and aims to design a resilient controller in the sense of [32] for integrating systems with delay. In [31], firstly, the necessary and sufficient conditions for the stability of the closed-loop system are presented by utilizing the coprime factorizations of the considered plant and K_{pi}. Then, by using the small-gain theorem, the allowable intervals for K_p and K_i that ensure the stability of the closed-loop system are determined. To design a resilient PI controller for AQM, in [14], the transfer function in (6) is considered by assuming K in (2) as $K \gg 1$. Then, it is shown that the optimal K_p, which maximizes the interval of allowable K_i, is found as $\frac{N_o}{2R_o^2 C_o^2}$. By the optimal K_p, the maximum value of the interval of allowable K_i is found as $\frac{N_o}{16 R_o^3 C_o^2}$. Then, to design a resilient controller, K_p is chosen as $\frac{N_o}{2R_o^2 C_o^2}$ and K_i is chosen as $\frac{N_o}{32 R_o^3 C_o^2}$, which is the midpoint of the allowable interval for K_i.

3.7. PI controller design by Wang, Shao ([22])

In this method, the PI parameters are adjusted to minimize the integral error under a constraint such that the Nyquist curve of the open loop transfer function is tangent to a line parallel to the imaginary axis with a distance ensuring the stability margins. If $f(K_p, K_i, \omega) := \mathrm{Re}\left(K_{pi}(j\omega) G_{pq}(j\omega)\right)$, where $\mathrm{Re}(z)$ represents the real part of the complex number z, then, the constraint can be defined as

$$f(K_p, K_i, \omega) = -\frac{1}{\lambda} \text{ with } \frac{\partial f(K_p, K_i, \omega)}{\partial \omega} = 0, \tag{7}$$

where $\lambda \in [1.5, 2.5]$ for reasonable stability margins. Since the integral error is inversely proportional to K_i [33], the controller parameters are obtained to maximize K_i while satisfying (7). If we consider AQM problem, since $G_{pq}(j\omega)$ can be written as $G_{pq}(j\omega) = \alpha(\omega) + j\beta(\omega)$, where

$$\alpha(\omega) = \frac{R_oC_oK}{n(\omega)}\left[\left(\frac{1}{K} - R_o^2\omega^2\right)\cos(R_o\omega) - \omega R_o\left(1 + \frac{1}{K}\right)\sin(R_o\omega)\right]$$

$$\beta(\omega) = \frac{-R_oC_oK}{n(\omega)}\left[\left(\frac{1}{K} - R_o^2\omega^2\right)\sin(R_o\omega) + \omega R_o\left(1 + \frac{1}{K}\right)\sin(R_o\omega)\right],$$

$n(\omega) = \left(\frac{1}{K} - R_o^2\omega^2\right)^2 + \left(\omega R_o + \frac{R_o\omega}{K}\right)^2$, then, the resulting controller parameters are obtained as

$$K_p = \frac{1}{\lambda \frac{d\alpha(\omega)}{d\omega}_{|\omega=\omega_0}}\left(\frac{1}{\beta(\omega_0)}\frac{d\beta(\omega)}{d\omega}_{|\omega=\omega_0} - \frac{1}{\omega_0}\right)$$

and $K_i = -\frac{w_0}{\lambda\beta(w_0)}$, where ω_0 satisfying $\alpha(\omega_0) = 0$ and λ is chosen 2, which is the midpoint of the interval for reasonable stability margins.

3.8. PI controller design by Skogestad ([23])

The PI controller design by this approach is based on two steps. In the first step, the original system is approximated to a first order system with delay. Since the delay term may limit the performance of the controller, an approximation technique, called "half rule", is recommended to reduce the conservativeness. Then, considering the system, obtained by "half rule", direct synthesis technique is used to provide the desired closed-loop system as a first order system with the same delay of the considered system. Since the resulting controller becomes "Smith Predictor", due to the direct synthesis technique and existence of delay in the desired response, Taylor series approximation is used to obtain a PI controller. In order to design a PI controller for AQM scheme, the first order approximation of (2) by "half rule" is obtained as

$$G_{pq}(s) \approx \frac{R_oC_oK^2}{(K + \frac{1}{2})R_os + 1}e^{-\frac{3}{2}R_os}.$$

Then, by using Skogestad-IMC settings, the controller parameters are obtained as

$$\begin{aligned} K_p &= \frac{1}{C_oK^2}\frac{K + 1/2}{\tau_c + 3R_o/2} \\ K_i &= \frac{K_p}{\min\{KR_o + R_o/2, 4(\tau_c + 3R_o/2)\}}, \end{aligned}$$

where τ_c is the time constant of the desired closed-loop response. For a fast response, good disturbance rejection and moderate robustness margins, $\tau_c = 3R_o/2$ is recommended in [23].

4. Comparison of the PI controllers

In this section, we compare the designed controllers in the sense of fragility, robustness, and performance issues. The fragility and robustness properties of the controllers are compared using the stability region obtained by the approach of [15]. To validate and compare the performance issues of the controllers, we implement the controllers in ns-2 and conduct simulations in different scenarios. Throughout the section, PI^{ZN}, PI^{PYH}, PI^{H}, PI^{MN}, PI^{PP}, PI^{UO}, PI^{WS}, and PI^{S} correspond to the controller designed by the approach of Ziegler-Nichols, [20], [11], [15], [21], [14], [22], and [23], respectively. For the sake of clarity, the PI controllers are designed for the same network parameters as in [11], i.e. $N_o = 60$, $C_o = 3750$ packets/s, and $R_o = 0.246$ s. The corresponding proportional and integral gain values of each of the designed controller are given in Table 1.

Table 1: PI Controller Parameters

Controllers	$(K_p, K_i) \times 10^{-5}$
$\mathrm{PI}^{\mathrm{ZN}}$	(8.3745, 11.375)
$\mathrm{PI}^{\mathrm{PYH}}$	(11.245, 8.5981)
PI^{H}	(1.8182, 0.9612)
$\mathrm{PI}^{\mathrm{MN}}$	(9.1044, 6.8)
$\mathrm{PI}^{\mathrm{PP}}$	(3.7925, 1.5987)
$\mathrm{PI}^{\mathrm{UO}}$	(3.5243, 0.8953)
$\mathrm{PI}^{\mathrm{WS}}$	(4.1633, 2.0146)
PI^{S}	(5.0046, 2.4841)

Table 2: Fragility and robustness metrics of the controllers

	$\rho \times 10^{-5}$	κ_p	κ_i
$\mathrm{PI}^{\mathrm{ZN}}$	3.2949	1.7314	1.3447
$\mathrm{PI}^{\mathrm{PYH}}$	4.1421	1.3994	1.7630
PI^{H}	0.9612	9.8983	8.0615
$\mathrm{PI}^{\mathrm{MN}}$	6.7411	1.7988	2.2796
$\mathrm{PI}^{\mathrm{PP}}$	1.5987	4.7057	6.8670
$\mathrm{PI}^{\mathrm{UO}}$	0.8953	5.1114	11.819
$\mathrm{PI}^{\mathrm{WS}}$	2.0146	4.2622	5.7095
PI^{S}	2.4841	3.5221	5.0670

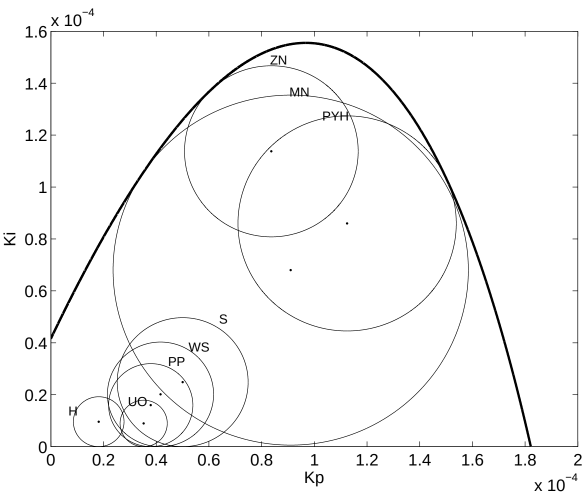


Figure 1: Fragility comparison of the designed controllers

4.1. Fragility and Robustness comparisons

The stability region of all stabilizing PI controllers for the considered network parameters is shown in Fig. 1. The region is determined by the coordinate axes $K_p = 0$, $K_i = 0$ and the curve defined by (4) and (5). As seen in Fig. 1, the designed controllers by each of the approaches belong to the stability region as expected.

In order to compare the fragility of each one of the designed controllers, let us define the following metric borrowed from [15]:

$$\rho = \min\{K_p, K_i, \hat{\rho}\}, \tag{8}$$

where $\hat{\rho}$ is the minimum distance from (K_p, K_i) of each controller given in Table 1 to the boundary of the stability region computed by the approach of [15]. By (8), we get a circle with center at (K_p, K_i) and radius ρ. Such a circle is the largest one inside the stability region that can be obtained for each of the designed controller's gains (K_p, K_i), see Fig. 1. Thus, a large ρ yields a less fragile controller while a small ρ leads to a more fragile controller. Hence, the controllers designed by [11] and [14] are more fragile compared to the other controllers, as their ρ values given in Table 2 are small. As seen in Table 2 and also shown in Fig. 1, controllers PI^{MN}, PI^{PYH}, PI^{ZN}, PI^{S}, and PI^{WS} may not suffer fragility problem compared to the rest of the designed controllers. PI^{PYH} is designed without taking into account the fragility issue, however, its distance to the boundary is close to the distance of PI^{MN}, which is the optimally non-fragile controller in the sense that it provides the greatest l_2 parametric margin.

For the robustness issue, we can compare the controllers in the sense of how much each of their parameters can be increased (with fixing the other one) without violating the stability. This issue is related to the classical gain margin problem. Therefore, let us define κ_i (κ_p), which is the maximum gain such that $\kappa_i K_i$ ($\kappa_p K_p$) does not destabilize the system with fixing the nominal value of K_p (K_i). Clearly, in view of Fig. 1, in this case, a good choice would be to take a small nominal K_i (respectively K_p). Performance constraint should determine the lower bounds for the nominal parameters. Note that κ_p or κi values for PI^H, PI^{UO}, and to some extent PI^{PP} are larger compared to the corresponding values obtained with other controllers as presented in Table 2. So, these controllers are preferable vis-a-vis gain margin considerations.

4.2. Performance comparisons

For AQM, performance objectives include efficient queue utilization, low jitter, low packet dropping, and robustness with respect to varying network parameters. Now, we compare the performance of the PI controllers by implementing them in ns-2 considering different scenarios. The parameters of the PI controllers, given in Table 1, are obtained in the s domain. However, for the implementation of these controllers in ns-2, each controller is converted to the z domain by a sampling frequency chosen as 15 times of its open-loop bandwidth frequency [10].

For the simulations, we consider a dumbbell network given in Fig. 2. In the first 4 scenarios, the sources are TCP/Reno connections generating FTP flows, and in the last scenario, the sources generate UDP, HTTP and FTP flows. The capacity of the

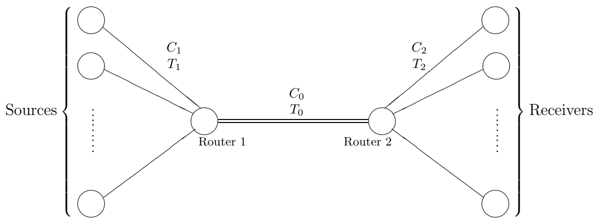


Figure 2: Network topology

bottleneck is denoted by C_0 and the propagation delay between the routers is denoted by T_o. The pair (C_1, T_1) represents the capacity of the links and the propagation delays between the sources and the first router. The pair (C_2, T_2) represents the capacity of the links and the propagation delays between the second router and the sinks. In simulations, q_0 is taken as 200 packets with 400 packets buffer sizes for each of the router, the average packet size is taken as 500 Bytes, and the simulation duration, T_{total}, is 200 s.

In order to evaluate the performance of the designed PI controllers, we introduce five metrics related to the above performance objectives. The first metric is the RMS percentage error of the queue length with respect to the desired queue length q_0:

$$RMS_{err} = \left(\frac{1}{M} \sum_{i=1}^{M} \left(\frac{q(i) - q_0}{q_0} \right)^2 \right)^{1/2},$$

where M is the number of total samples generated by ns-2, $q(i)$ is the queue length at instant i. Note that since the buffer size is 400 packets, oscillation of $q(t)$ around 400, i.e., hitting $q(t)$ to the buffer limit, implies the existence of congestion and packet dropping due to the saturation. Then, we can define the second metric as

$$\Omega := T_s/T_{\text{total}} \times 10^3,$$

where T_s is the total length of the time-intervals of $q(t)$ oscillating in the interval around 400, let us choose this interval as $[399, 400]$. Here, T_s can be thought of as the total time interval for buffer overflow, i.e. saturation, and packet dropping, hence, the controller which produces small Ω should be preferred. Note, since packet dropping may happen due to the larger overshoots, Ω does not give alone the complete packet loss. The link utilization is related with the time how long queue is efficiently used (i.e. the buffer is not empty) during the network traffic, hence, it is the function of total time intervals where $q(t) \neq 0$. Therefore, let T_z be the total duration of the time when $q(t)$ drops to 0. Since there will be no packet at the router during the time intervals lie in T_z, the link utilization can be defined as

$$U := utilization = \frac{T_{\text{total}} - T_z}{T_{\text{total}}}. \tag{9}$$

Then, by (9), $C_u := (1-U) \times 10^3 = \frac{T_z}{T_{\text{total}}} \times 10^3$ can be defined as the third metric. Since T_z corresponds to the total duration of the link underutilized, the controller providing U closest to 1 (or C_u closest to 0) satisfies better link utilization compared to the other controllers. Now, let us define $Loss_r$ as the ratio of the number of lost packets to the total number of sent packets by all the sources, then, we can define another metric as

$$P_{Loss} = Loss_r \times 10^4.$$

Since minimization of the packet loss is one of the AQM performance objectives, the controller which provides small P_{Loss} should be preferred. Another metric is related to the response speed of the controllers. We define this metric, called R_t, as the required time for $q(t)$ reaches 90% of the desired value but by discarding the time interval where $q(t)$ saturates the buffer capacity. In order to discuss jitter properties of the designed PI controllers, let us define

$$R_v(t_i) := \frac{q(t_{i+1}) - q(t_i)}{t_{i+1} - t_i} \frac{1}{R_o} \times 10^3, \tag{10}$$

where $q(t_i)$ is the queue length at discrete time t_i generated by ns-2 in the interval (R_t, T_{total}). By the definition in (10), $R_v(t_i)$ can be considered as a *relative delay variation* at time t_i.

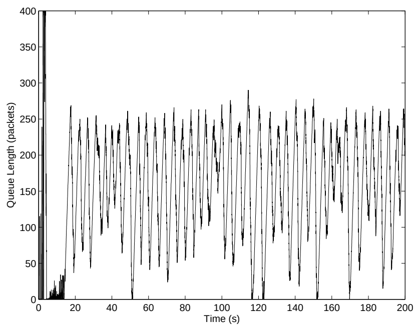
Case 1: In this scenario, we consider the nominal response of the designed controllers. For this reason, the parameters of the network in the simulations are chosen as $N_o = 60$ FTP flows, $C_0 = C_1 = C_2 = 15$ Mbps, $T_0 = 192.7$ ms, $T_1 = T_2 = 40$ ms. The performance analysis of the designed controllers are given in Table 3. As shown in Table 3, PI^{UO} produces the smallest RMS error, while PI^{ZN} produces the greatest RMS error compared to the other ones. Most of the controllers have the same link utilization performance, however, PI^{WS} has the best one. Smallest packet dropping happens by PI^{MN} and PI^{ZN}. Table 3 demon-

Table 3: Performance analysis of the PI controllers for Case 1

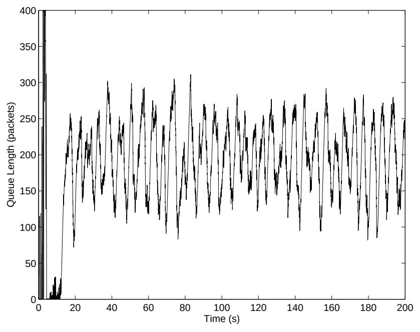
	RMS_{err}	Ω	C_u	P_{Loss}	R_t
PI^{ZN}	0.4999	2.50	1.3520	3.2004	16.33
PI^{WS}	0.4183	3.1344	0.8467	4.4172	14.62
PI^{PYH}	0.4719	2.8725	1.3575	3.8865	17.59
PI^{H}	0.4756	3.0517	0.8818	4.5030	14.30
PI^{MN}	0.4557	2.4992	1.5114	3.1786	15.94
PI^{PP}	0.4256	3.1288	0.8895	4.4806	13.71
PI^{UO}	0.4091	3.1288	0.8895	4.4797	13.71
PI^{S}	0.4693	3.2408	1.0319	5.3128	17.69

strates that PI^S and PI^{PYH} have slower response than the other ones, while PI^{UO} and PI^{PP} have faster response. In order to compare the controllers which provide low jitter, by using (10), maximum (max_{R_v}), minimum (min_{R_v}), and average (ave_{R_v}) values of $\{R_v(t_i)\}_{t_i \in (R_t, T_{total})}$ for each of the designed PI controllers are presented in Table 4. As seen in Table 4, the controllers PI^{UO}, PI^{PP}, and PI^{WS} provide low jitter compared to other controllers. On the other hand, the controllers PI^S and PI^H result in large delay variations. As shown in Table 3, the controllers, which result in low jitter, provide small RMS error with high link utilization and the controllers, which result in large jitter, result in more packet dropping. The simulation results are presented in Figures 3 and 4. As shown in all figures, the designed PI controllers regulate the queue length at the routers. Figures 3(a) and 3(d) demonstrate that PI^{ZN} makes large undershoots, whereas PI^H makes large overshoots. The controllers PI^{WS} and PI^{UO}, as seen in Fig. 3(b) and Fig. 4(c), result in small oscillations around the desired queue length and they have better steady-state responses. Note that, by considering Tables 3 and 4, the simulation results confirm that the controllers, which provide small queue oscillations around desired queue length, indicate low jitter, small RMS error, and also high link utilization.

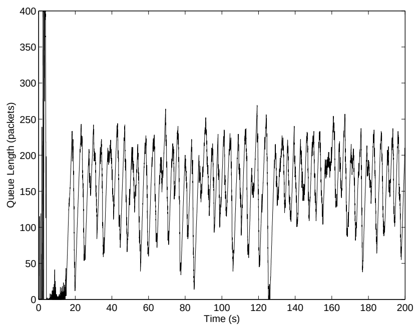
Case 2: In this scenario, we consider the robustness property of the designed controllers. It has been shown in [15], by using geometric properties of the boundary of the stability region, that a stabilizing PI controller designed for network parameters (N_o, C_o, R_o) also stabilizes a network with param-



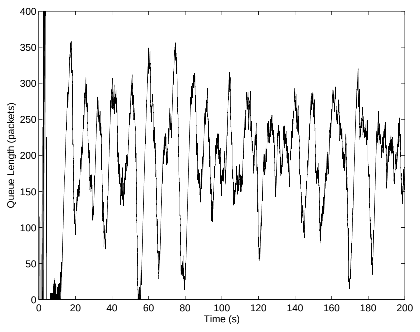
(a)



(b)

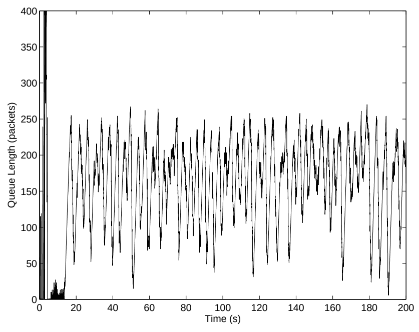


(c)

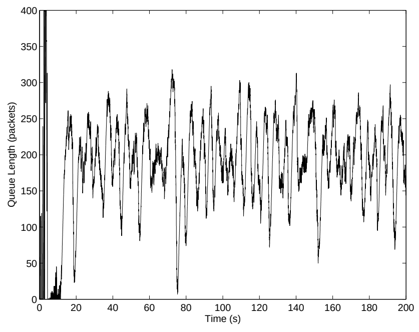


(d)

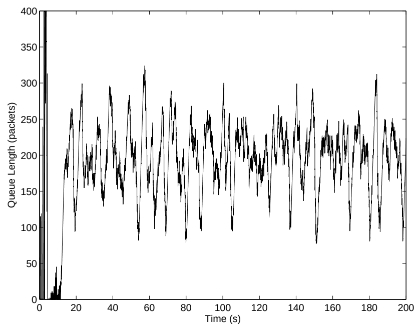
Figure 3: Simulation results of a)PI^{ZN}, b)PI^{WS}, c)PI^{PYH} d)PI^H for Case 1



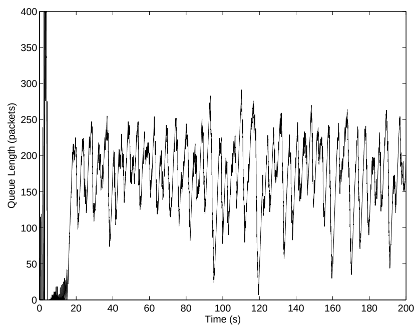
(a)



(b)



(c)



(d)

Figure 4: Simulation results of a)PI^{MN}, b)PI^{PP}, c)PI^{UO} and d)PI^{S} for Case 1

Table 4: Relative delay variation for Case 1

	max_{R_v}	min_{R_v}	ave_{R_v}
PI$^{\mathrm{ZN}}$	13.548	-8.129	-1.9279
PI$^{\mathrm{PYH}}$	13.548	-8.129	-2.0260
PI$^{\mathrm{H}}$	20.322	-8.129	-1.9124
PI$^{\mathrm{MN}}$	13.548	-8.129	-1.9536
PI$^{\mathrm{PP}}$	8.129	-8.129	-1.8971
PI$^{\mathrm{UO}}$	8.129	-8.129	-1.8873
PI$^{\mathrm{WS}}$	8.129	-8.129	-1.9025
PI$^{\mathrm{S}}$	40.645	-8.129	-2.0656

eters $(\widetilde{N}_o, \widetilde{C}_o, \widetilde{R}_o)$, where $\widetilde{N}_o \geq N_o$, $\widetilde{C}_o \leq C_o$ and $\widetilde{R}_o \leq R_o$. Therefore, the number of FTP flows are taken 200, link capacity at all the links are taken as 10 Mbps and the propagation delay between routers is taken as $T_o = 43$ ms. The other simulation parameters are kept as in Case 1. The performance analysis of the controllers are given in Table 5. As seen in Table 5, the majority of the controllers provide less RMS error compared to Case 1, therefore, they regulate the queue length at the routers. In addition, compared to Case 1, the controllers provide better link utilization, however, more packets are dropped. One of the reasons for this result is the fact that the number of loads is taken more than 3 times of Case 1, hence, $q(t)$ oscillates around the upper limit of buffer for a long time as seen by comparing the second metric in Tables 3 and 5, therefore, more packet-dropping happens. In addition, oscillation of $q(t)$ around its upper limit for a long period implies that $q(t)$ becomes zero only for a short time compared to Case 1, hence, the link utilization is improved. From Table 5, PI^{H} and PI^{UO} yield larger RMS errors, PI^{ZN}, PI^{PYH}, and PI^{MN}

Table 5: Performance analysis of the PI controllers for Case 2

	$\mathrm{RMS_{err}}$	Ω	C_u	$\mathrm{P_{Loss}}$	$\mathrm{R_t}$
$\mathrm{PI^{ZN}}$	0.2222	5.1938	0.1752	5.7922	14.85
$\mathrm{PI^{WS}}$	0.3553	13.214	0.1157	8.9505	62.58
$\mathrm{PI^{PYH}}$	0.2259	4.5065	0.1043	5.452	14.91
$\mathrm{PI^{H}}$	0.5134	35.186	0.1268	16.855	127.3
$\mathrm{PI^{MN}}$	0.2343	5.1727	0.1267	5.7779	20.96
$\mathrm{PI^{PP}}$	0.3908	16.078	0.1209	9.9564	62.82
$\mathrm{PI^{UO}}$	0.5170	27.738	0.0926	14.147	114.9
$\mathrm{PI^{S}}$	0.3284	11.662	0.1542	8.2289	43.38

provide smaller RMS errors. The best link utilization is provided by PI^{UO}, and the rest of the controllers have similar levels of utilization. PI^{H} and PI^{UO} yield more packet dropping than the others, while PI^{PYH}, PI^{MN} and PI^{ZN} provide relatively small packet dropping. The last column of Table 5 shows that PI^{H} and PI^{UO} have slower response, while PI^{ZN}, PI^{PYH}, PI^{MN} have faster response. As discussed above, PI^{UO}, which provides the best link utilization, and PI^{H} yield large RMS error due to the fact that they saturate for a long time as shown by the second metric in Table 5. However, such a long saturation duration results in slower response and the controllers provide small oscillations around the desired queue length. Hence, these controllers result in low jitter compared to other controllers.

Case 3: In this scenario, we aim to evaluate the response of the controllers for a large nominal plant gain. The number of FTP flows is 45, link capacities C_0, C_1 and C_2 are 18 Mbps and the propagation delay between the routers is set to $T_o = 350$ ms. The rest of the simulation parameters are kept as in Case 1. The performance analysis of the controllers are given in Table 6. As shown by the table, performances of all the controllers are deteriorated, they result in larger RMS error and worse link utilization compared to the previous scenarios. In addition, since the controllers yield $q(t)$ to become zero frequently due to the worse link utilization, as seen in Table 6, fewer packets are dropped compared to Cases 1 and 2. As seen from Table 6, most of the controllers produce the same RMS error. Among these controllers, PI^{H}, PI^{PP}, PI^{UO}, and PI^{WS} result in the smaller RMS error, PI^{ZN} and PI^{MN} result in the larger RMS error and worse link utilization, while PI^{S} and PI^{H} provide the better link utilization. Most of the controllers yield the same packet dropping, however, PI^{PYH} yields the minimum packet dropping, PI^{WS}, PI^{PP}, and PI^{UO} result in larger packet droppings. The response time of the most of the controllers are close to each other, however, PI^{MN} is the controller which has a slowest response, while PI^{H} and PI^{PYH} have faster response. Since the controllers yield worse link utilization with larger RMS error, they have worse performance in the sense of jitter compared to the previous cases.

Case 4: In this scenario, the gain of the nominal plant is larger than the one in Case 3. The number of FTP flows are taken 30, link capacities C_0, C_1 and C_2 are taken 18 Mbps and the propagation delay between the routers is taken as $T_o = 537.6$ ms. The other simulation parameters are kept as in Case 1. As seen by the performance analysis of the controllers given in Table 7, the controllers have worse performances in the sense of RMS error but better performance in the sense of lost packet ra-

Table 6: Performance analysis of the PI controllers for Case 3

	$\mathrm{RMS}_{\mathrm{err}}$	Ω	C_u	$\mathrm{P}_{\mathrm{Loss}}$	R_{t}
$\mathrm{PI}^{\mathrm{ZN}}$	0.7188	1.7703	11.056	3.2220	47.89
$\mathrm{PI}^{\mathrm{WS}}$	0.6153	1.9390	5.5956	3.6808	44.88
$\mathrm{PI}^{\mathrm{PYH}}$	0.6700	0.9952	6.9587	1.8312	41.58
PI^{H}	0.5482	1.8281	3.9471	3.3795	41.21
$\mathrm{PI}^{\mathrm{MN}}$	0.7153	1.7703	11.079	3.2234	50.40
$\mathrm{PI}^{\mathrm{PP}}$	0.6085	1.9390	5.8341	3.6749	43.89
$\mathrm{PI}^{\mathrm{UO}}$	0.6272	1.9390	6.0446	3.6673	47.25
PI^{S}	0.6550	1.7957	3.8808	3.2332	44.32

Table 7: Performance analysis of the PI controllers for Case 4

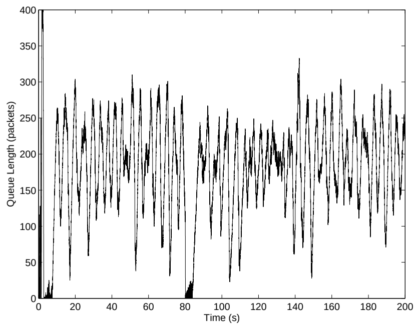
	$\mathrm{RMS}_{\mathrm{err}}$	Ω	C_u	$\mathrm{P}_{\mathrm{Loss}}$	R_{t}
$\mathrm{PI}^{\mathrm{ZN}}$	0.8341	0.8967	25.51	1.9321	115.1
$\mathrm{PI}^{\mathrm{WS}}$	0.7610	1.2249	3.1337	2.3703	116.7
$\mathrm{PI}^{\mathrm{PYH}}$	0.8137	0.9014	8.1792	1.7468	124.8
PI^{H}	0.7787	1.6062	1.4446	3.6100	129.1
$\mathrm{PI}^{\mathrm{MN}}$	0.7857	0.8967	9.9103	1.8741	115.1
$\mathrm{PI}^{\mathrm{PP}}$	0.7767	1.2249	4.5427	2.3750	116.7
$\mathrm{PI}^{\mathrm{UO}}$	0.7751	1.2249	2.3877	2.3695	116.7
PI^{S}	0.7713	1.1117	2.9463	2.2004	125.0

tio compared to the previous cases. Additionally, all the controllers, except PI^{ZN} and PI^{PYH}, have better link utilization compared to Case 3. From Table 7, the maximum RMS error is produced by PI^{ZN}, and the other controllers produce RMS errors close to each other. The better link utilization is provided by PI^{H}, which causes more packet dropping. PI^{PYH}, PI^{ZN} and PI^{MN} provide small packet droppings. The last column of Table 7 shows that the controllers have slower response compared to the ones in previous cases. Among the controllers, PI^{H} is the slowest one. Longer response time and worst link utilization can be attributed to the drastic increase in the open-loop gain. As discussed in Case 3, the controllers in this case may result in high jitter compared to the previous cases.

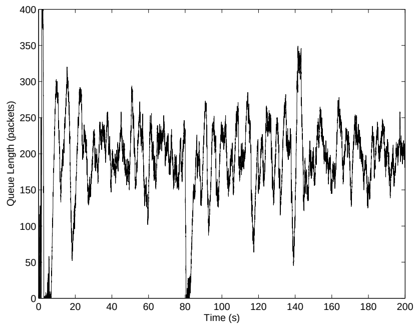
Case 5: We here consider a more realistic traffic scenario. The network sources, link capacity and propagation delay between the routers change dynamically. We consider 180 HTTP sessions (180 clients and 1 server), 60 FTP flows, and 10 UDP flows with a packet size 250 bytes. Therefore, 75% of the traffic consists of short-lived flows, called *web mice*, which make the traffic more realistic [34]. The UDP flows follow an exponential ON/OFF traffic model such that both the idle and burst times have mean of 0.5 ms and the sending rate during the on-time is 0.05 Mbps. The propagation delay of each UDP flow uniformly varies within the interval $[20, 80]$ ms and these flows are active between $t = 50$ s and $t = 150$ s. We introduce dynamic load $N_o(t)$ such that at $t = 80$ s, 30 of the FTP flows drop out and at $t = 140$ s they return. The propagation delay T_o and the link capacity C_0 uniformly vary within the interval $[100, 300]$ ms and $[12, 18]$ Mbps respectively. The rest of the simulation parameters are kept as in Case 1. The performance analysis of the controllers are given in Table 8. As seen from the table, PI^{PP} and PI^{WS} provide small RMS error, while PI^{ZN} provides the largest one. The link utilization performance of most of the controllers are close to each other, however, PI^{H} is the best one and PI^{ZN} is the worst. PI^{H} yields more packet dropping, whereas PI^{PYH} provides less packet dropping. The response time of controllers are close each other, however, PI^{H} has fast response compared to the others, while PI^{MN} has slowest response. The simulation results are presented in Figures 5(a)– 5(d) and Figures 6(a)–6(d). As seen from the figures, when the number of sources drop to 30, at $t = 80$ s, the queue length drops to 0 suddenly and the link is underutilized for a short duration. As seen in Fig. 5(b), PI^{WS} acts faster compared to the other controllers to regulate the queue length due to the change of the number of sources. When these sources return at $t = 140$ s, as seen in Fig. 5(d), for the controller PI^{H}, queue length drops to 0 for some duration compared to other controllers, which means that the link is underutilized. In addition, as seen in Figures 5(c), 5(d) and Figures 6(c), 6(d), PI^{PYH}, PI^{H}, PI^{UO}, and PI^{S} have aggressive responses and larger overshoots. As seen in Fig. 5(d), PI^{H} result in larger overshoots around the desired queue length, which may result in larger jitter. Similarly, PI^{ZN}, which provide largest RMS error and worst link utilization, may also result in large jitter. Since queue length in Fig. 5(b) and Fig. 6(b) does not make large overshoots around its desired level, the controllers PI^{WS} and PI^{PP}, which provide small RMS error with high link utilization, may also provide low jitter. As seen in Fig. 5(a), PI^{ZN} is less robust to unresponsive flows, since the controller results in large undershoots when UDP sources switched on and off.

4.3. Discussions

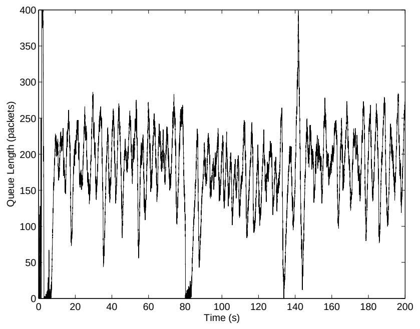
As shown in Figures 3–6, the designed controllers regulate the queue length at the desired level. For large TCP sessions, as in Case 2, the designed controllers still regulate the queue length. However,



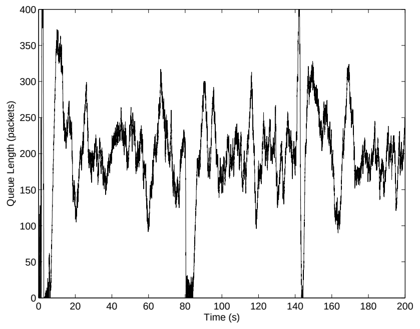
(a)



(b)

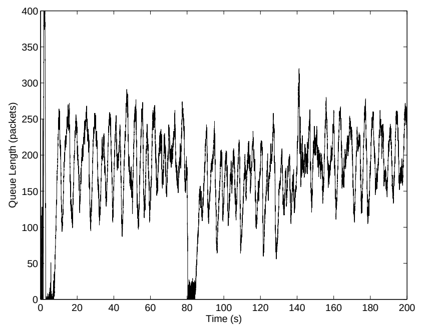


(c)

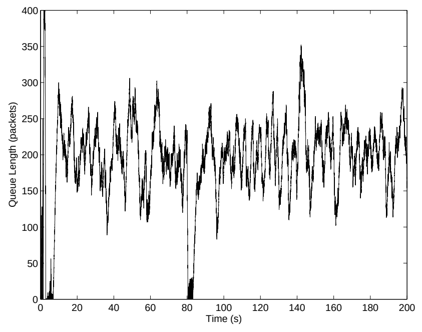


(d)

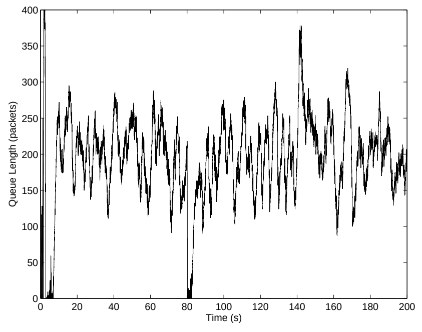
Figure 5: Simulation results of a)PI^{ZN}, b)PI^{WS}, c)PI^{PYH} d)PI^{H} for Case 5



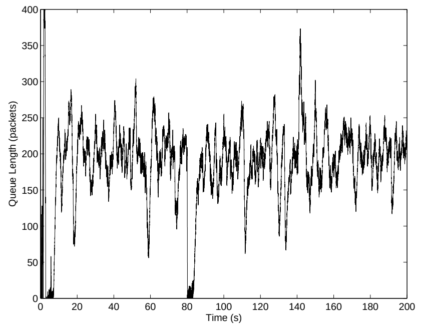
(a)



(b)



(c)



(d)

Figure 6: Simulation results of a)PI^{MN}, b)PI^{PP}, c)PI^{UO} and d)PI^{S} for Case 5

Table 8: Performance analysis of the PI controllers for Case 5

	$\mathrm{RMS_{err}}$	Ω	C_u	$\mathrm{P_{Loss}}$	$\mathrm{R_t}$
$\mathrm{PI^{ZN}}$	0.3758	2.7466	1.4798	5.4678	9.08
$\mathrm{PI^{WS}}$	0.3126	2.5742	0.8874	5.1014	8.22
$\mathrm{PI^{PYH}}$	0.3479	2.4980	1.3401	4.8442	8.54
$\mathrm{PI^{H}}$	0.3597	3.8994	0.7907	6.8847	7.95
$\mathrm{PI^{MN}}$	0.3408	2.6530	0.8956	5.2050	9.15
$\mathrm{PI^{PP}}$	0.3123	2.5644	0.8330	5.0743	8.45
$\mathrm{PI^{UO}}$	0.3212	2.5644	1.0573	5.0645	8.44
$\mathrm{PI^{S}}$	0.3170	2.8621	1.2066	5.5632	8.64

if the nominal plant gain is drastically increased, then, the controllers loose their ability to regulate the queue length. As seen in Case 5, the designed PI AQM controllers achieve the objectives on the queue length in a dynamic traffic scenario and in presence of disturbance factors such as short-lived and UDP flows.

To recap, the least fragile PI controllers are PI^{MN} and PI^{PYH}. In the sense of maximizing κ_p (respectively κ_i) the best PI controllers are PI^{H} (respectively PI^{UO}). Since PI^{PP} is designed to bound the $\mathrm{M_r}$ of the closed-loop transfer function, it bounds the complementary sensitivity function, so it has some robustness property. In general, the controller PI^{ZN} gives the largest RMS error, and, PI^{MN} and PI^{PYH}, provide the smaller packet dropping. In most of the cases studied, PI^{ZN} and PI^{MN} have worst link utilization, while PI^{UO} and PI^{H} have the best link utilization. Considering the response time of the controllers, PI^{ZN} has faster response in most cases. In general, PI^{WS}, PI^{PP}, and PI^{UO} provide low jitter, while PI^{ZN} and PI^{H} result in high jitter. Note that, additional analysis and simulation results for different scenarios are presented in [35].

5. Conclusion

In this paper, a comparison of various PI controllers designed for the delay-model of TCP/AQM is performed. The compared PI controllers are the ones currently available for TCP/AQM proposed in [11, 14, 15] and the designed ones for TCP/AQM in the current paper using the approaches given in [19, 20, 21, 22, 23]. It should be noted that, it is possible to find different PI controller design approaches for TCP/AQM than the presented ones in the paper. However, we believe that the chosen PI controller design approaches are more appropriate considering the linear model of TCP/AQM among the others in the literature. The comparison in the paper is based on the fragility, robustness, and performance issues of the controllers. For a comprehensive and realistic comparison, we implemented the controllers in ns-2 for different traffic scenarios and validated the results considering some AQM performance objectives.

Our analysis showed that the PI controllers of [15] and [20] are less fragile compared to the controllers designed by the approach of Zeigler-Nichols, and [22, 11, 21, 14, 23]. On the other hand, the controllers proposed by [11, 14, 21], and [22] are more robust compared with the rest because their proportional or integral gain margins are relatively large. Additionally we note that the controller designed by the methods of [21] and [11] have also good robustness property in the sense that they limit the sensitivity function. The simulation results indicate that, in general, robust controllers have better link utilization and provide low jitter, while the controllers, which may not suffer "fragility" problem, provide smaller packet dropping.

Note that the approaches of [14] and [21] are for the simplified model of the system $K \gg 1$, which may result in conservativeness. In addition, since the approach of [14] is based on the small-gain theorem, like [11], another conservativeness arises. The controllers proposed by [20, 23] can also be considered conservative because they are designed using a first order approximation of the transfer function. The approach of [15], however, gives the set of all stabilizing controllers, and allows us to measure the parametric stability margins.

In conclusion, we have identified which PI controllers to choose for different performance and robustness metrics in AQM. As in all other application areas, these controllers form a baseline for an initial design; depending on the performance requirements of a particular network, PI controllers can be modified for further improvements.

Acknowledgments

The authors wish to thank the Associate Editor and reviewers for their useful suggestions that helped us to improve the paper. This work is supported in part by the French-Turkish PIA Bosphorus (TUBITAK Grant No. 109E127 and EGIDE Project No. 22974WJ). The work of Hakkı Ulaş Ünal is supported by the project STRT1- 09/33 of the K.U. Leuven Research Council. The work of Hitay Özbay is partially supported by DPT-HAMIT project. The work of Daniel Melchor-Aguilar is supported by CONACYT grant 131587.

References

[1] B. Braden, D. Clark, J. Crowcroft, B. Davie, S. Deering, D. Estrin, S. Floyd, V. Jacobson, G. Minshall, C. Partridge, L. Peterson, K. Ramakrishnan, S. Shenker, J. Wroclawski, L. Zhang, Recommendations on queue management and congestion avoidance in the Internet, RFC 2309.

[2] S. Ryu, C. Rump, C. Qiao, Advances in Internet congestion control, IEEE Communications Surveys and Tutorials 5 (1) (2003) 28–39.

[3] L. Tan, W. Zhang, G. Peng, G. Chen, Stability of TCP/RED systems in AQM routers, IEEE Transactions on Automatic Control 51 (8) (2006) 1393–1398.

[4] W. Zhang, L. Tan, G. Peng, Dynamic queue level control of TCP/RED systems in AQM routers, Computers and Electrical Engineering 35 (1) (2009) 59–70.

[5] V. Misra, W. B. Gong, D. Towsley, Fluid-based analysis of a network of AQM routers supporting TCP flows with an application to RED, in: Proceedings of ACM/SIGCOMM, Stockholm, Sweden, 2000, pp. 151–160.

[6] P. F. Quet, H. Özbay, On the design of AQM supporting TCP flows using robust control theory, IEEE Transactions on Automatic Control 49 (2004) 1031–1036.

[7] Q. Chen, Q. W. W. Yang, Robust controller design for AQM router, IEEE Transactions on Automatic Control 52 (2007) 938–943.

[8] S. Manfredi, M. Bernardo, F. Garofalo, Design, validation and experimental testing of a robust AQM control, Control Engineering Practice 17 (2009) 394–407.

[9] F. Zheng, J. Nelson, An $\mathcal{H}_\infty$ approach to the controller design of AQM routers supporting TCP flows, Automatica 45 (3) (2009) 757–763.

[10] C. V. Hollot, V. Misra, D. Towsley, W. Gong, On designing improved controllers for AQM routers supporting TCP flows, in: Proceedings of IEEE INFOCOM, Alaska, USA, 2001, pp. 1726–1734.

[11] C. V. Hollot, V. Misra, D. Towsley, W. Gong, Analysis and design of controllers for AQM routers supporting TCP flows, IEEE Transactions on Automatic Control 47 (6) (2002) 945–959.

[12] W. Michiels, D. Melchor-Aguilar, S.-I. Niculescu, Stability analysis of some classes of TCP/AQM networks, International Journal of Control 79 (2006) 1136–1144.

[13] D. Melchor-Aguilar, V. Castillo-Tores, Stability analysis of proportional-integral AQM controllers supporting TCP flows, Computacion y Sistemas 10 (2007) 401–414.

[14] D. Üstebay, H. Özbay, Switching resilient PI controllers for active queue management of TCP flows, in: Proceedings of the 2007 IEEE International Conference on Networking, Sensing and Control, London, 2007, pp. 574–578.

[15] D. Melchor-Aguilar, S.-I. Niculescu, Computing non-fragile PI controllers for delay models of TCP/AQM networks, International Journal of Control 82 (2009) 2249–2259.

[16] L. H. Keel, S. P. Bhattacharyya, Robust, fragile, or optimal?, IEEE Transactions on Automatic Control 42 (8) (1997) 1098–1105.

[17] W. Michiels, S. I. Niculescu, Stability and Stabilization of Time-Delay Systems: An Eigenvalue-Based Approach, SIAM, Philadelphia, PA, 2007.

[18] C.-C. Yu, Autotuning of PID controllers: A Relay Feedback Approach, Springer, Germany, 2006.

[19] Q. C. Zhong, Robust Control of Time-Delay Systems, Springer-Verlag, London, UK, 2006.

[20] R. C. Panda, C.-C. Yu, H. P. Huang, PID tuning rules for SOPDT systems: Review and some new results, ISA Transactions 43 (2004) 283–295.

[21] E. Poulin, A. Pomerleau, PID tuning for integrating and unstable processes, Control Theory and Applications, IEE Proceedings 143 (1996) 429–435.

[22] Y.-G. Wang, H.-H. Shao, Optimal tuning for PI controller, Automatica 36 (2000) 147–152.

[23] S. Skogestad, Simple analytic rules for model reduction and PID controller tuning, Journal of Process Control 13 (2003) 291–309.

[24] The network simulator, ns-2 [online], `http://www.isi.edu/nsnam/ns/` (2000).

[25] J. Sun, K.-T. Ko, G. Chen, S. Chan, M. Zukerman, PD-RED: To improve the performance of RED, IEEE Communications Letters 7 (8) (2003) 406–408.

[26] X. Deng, S. Yi, G. Kesidis, C. R. Das, A control theoretic approach for designing adaptive AQM

schemes, in: IEEE Global Telecommunications Conference (GLOBECOM 2003), San Francisco, USA, 2003, pp. 2947–2951.

[27] S. Ryu, C. Rump, C. Qiao, Advances in active queue management (AQM) based TCP congestion control, Telecommunication Systems 25 (2004) 317–351.

[28] D. Üstebay, H. Özbay, N. Gündeş, A new PI and PID control design method for integrating systems with time delays, in: Proceedings of the 6th WSEAS International Conference on Signal Processing, Robotics and Automation, Corfu Island, Greece, 2007, pp. 60–65.

[29] A. O'Dwyer, Handbook of PI and PID Controller Tuning Rules, Imperial College Press, London, 2006.

[30] J. K. Hale, S. M. V. Lunel, Introduction to Functional Differential Equations, Vol. 99 of Applied Mathematics Science, Springer-Verlag, New York, 1993.

[31] H. Özbay, A. N. Gündeş, Resilient PI and PD controller designs for a class of unstable pants with I/O delays, Applied and Computational Mathematics 6 (2007) 18–26.

[32] G. J. Silva, A. Datta, S. Bhattacharyya, PID Controllers for Time-Delay Systems, Birkhäuser, Boston, USA, 2004.

[33] K. J.Âström, T. Hägglund, PID controllers: Theory, Design and Tunning, Instrument Society of America, Research Triangle Park, NC, 1995.

[34] M. Christiansen, K. Jeffay, D. Ott, F. D. Smith, Tuning RED for Web traffic, IEEE/ACM Transactions on Networking 9 (3) (2001) 249–264.

[35] H. U. Ünal, D. Melchor-Aguilar, D. Üstebay, S. I. Niculescu, H. Özbay, Comparing PI controllers for delay models of TCP/AQM networks, in: Proceedings of the 4th IFAC Symposium on Systems, Structure, and Control, Ancona, Italy, 2010, pp. 76–83.