

A NEW PI AND PID CONTROL DESIGN METHOD
AND ITS APPLICATION TO ACTIVE QUEUE
MANAGEMENT OF TCP FLOWS

A THESIS

SUBMITTED TO THE DEPARTMENT OF ELECTRICAL AND

ELECTRONICS ENGINEERING

AND THE INSTITUTE OF ENGINEERING AND SCIENCES

OF BILKENT UNIVERSITY

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS

FOR THE DEGREE OF

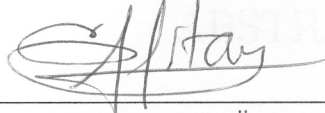
MASTER OF SCIENCE

By

Deniz Üstebay

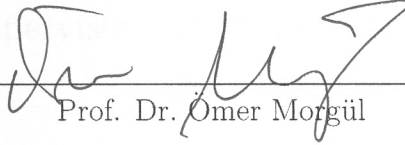
June 2007

I certify that I have read this thesis and that in my opinion it is fully adequate,
in scope and in quality, as a thesis for the degree of Master of Science.



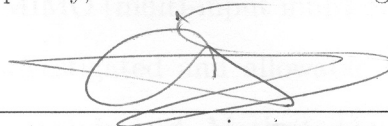
Prof. Dr. Hitay Özbay (Supervisor)

I certify that I have read this thesis and that in my opinion it is fully adequate,
in scope and in quality, as a thesis for the degree of Master of Science.



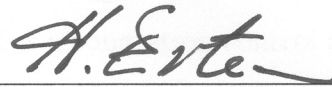
Prof. Dr. Ömer Morgül

I certify that I have read this thesis and that in my opinion it is fully adequate,
in scope and in quality, as a thesis for the degree of Master of Science.



Asst. Prof. Dr. İbrahim Körpeoğlu

Approved for the Institute of Engineering and Sciences:



Prof. Dr. Mehmet Baray
Director of Institute of Engineering and Sciences

ABSTRACT

A NEW PI AND PID CONTROL DESIGN METHOD AND ITS APPLICATION TO ACTIVE QUEUE MANAGEMENT OF TCP FLOWS

Deniz Üstebay

M.S. in Electrical and Electronics Engineering

Supervisor: Prof. Dr. Hitay Özbay

June 2007

PID controllers are continuing to be used in many control applications due to their simple structures. Design of such controllers for unstable systems with time delays is an active research area. Recently, stabilizing PI and PD controllers for a class of unstable MIMO (multi-input multi-output) systems with input/output delays have been investigated and allowable controller gain intervals for such controllers have been maximized. Motivated by these studies, this thesis proposes a new method for tuning the parameters of PI, PD and PID controllers for integrating processes with time delays. The method is based on selecting the centers of the maximized gain intervals as the controller gains for the purpose of obtaining optimal controllers. As an application of this method, controllers for AQM (Active Queue Management) of TCP (Transmission Control Protocol) flows have been designed. AQM is a congestion control method used in computer networks to increase link utilization with less queueing delays. The fluid flow model of TCP's congestion avoidance mode based on delay differential equations supplies the mathematical background for modelling the AQM as a feedback

control system and designing different control schemes accordingly. Firstly, the proposed controller design method has been applied to AQM for the case of time invariant time delay and secondly the method has been supported with switching control technique to obtain optimum system performance in the case of time varying time delay. The performance of the designed controllers for both cases has been illustrated by packet level simulations in ns-2.

Keywords: PID control, time delay, Active Queue Management (AQM), switching control, ns-2

ÖZET

YENİ BİR PI VE PID KONTROLÜ TASARIMI METHODU VE TCP AKIŞLARI İÇİN AQM TASARIMI UYGULAMASI

Deniz Üstebay

Elektrik ve Elektronik Mühendisliği Bölümü Yüksek Lisans

Tez Yöneticisi: Prof. Dr. Hitay Özbay

Haziran 2007

Günümüzde birçok kontrol uygulamasında basit yapıları sayesinde PID kontrol birimlerinin kullanımı devam etmektedir. Zaman gecikmeli kararsız sistemler için de bu tip kontrol birimlerinin tasarımı konusunda birçok çalışma yapılmaktadır. Bir süre önce yayımlanan bir çalışmada, giriş ve/veya çıkışlarında zaman gecikmesi olan bir sınıf çok girişli çok çıkışlı (MIMO) kararsız sistem için PI ve PD kontrol tasarımı metodu geliştirilmiş ve daha sonra bu kontrol birimlerinin izin verilen kazanç aralıklarını maksimize eden yeni bir çalışma yayınlanmıştır. Bu tezde, bahsi geçen çalışmaların sonuçları kullanılarak, tümlevli ve zaman gecikmeli sistemlerde uygulanacak PI, PD ve PID kontrol birimlerinin kazançlarını ayarlamak üzere yeni bir metot sunulmaktadır. Bu yöntemle göre kontrol birimlerinin kazançları izin verilen en yüksek kazanç aralıklarının merkezi olarak alınmakta ve böylece optimal kontrol birimleri elde edilmektedir. Önerilen yöntemin bir uygulaması olarak TCP (İletişim Kontrol Protokolü) akışlarının AQM (Etkin Kuyruk Yönetimi) problemi için kontrol birimleri tasarlandı. Etkin kuyruk yönetimi, bilgisayar ağlarında kullanılan bir sıkışıklığı giderici kontrol yöntemi olup amacı kuyrukta bekleme gecikmelerini azaltmak ve bağlantıların kullanım oranını artırmaktır. Daha önce yapılan çalışmalarda, sıkışıklıktan

kaçınma modunda çalışan TCP'nin türevsel denklemlere dayalı sıvı akış modeli kullanılarak AQM bir geri besleme sistemi olarak modellenmiştir. Bu model kullanılarak uygun kontrol birimleri tasarlandı. Önerilen kontrol metodu öncelikle zamanla değişmeyen zaman gecikmesi durumu için AQM'ye uygulandı. İkinci olarak, metot anahtarlama kontrol yöntemi de kullanılarak zamanla değişen zaman gecikmeli duruma uygulandı. Her iki durum için de geliştirilen kontrol birimlerinin performansları ns-2 platformunda paket düzeyinde benzetimlerle gösterildi.

Anahtar Kelimeler: PID kontrol, zaman gecikmesi, Etkin Kuyruk Yönetimi (AQM), anahtarlama kontrol, ns-2

ACKNOWLEDGMENTS

I gratefully thank my supervisor Prof. Dr. Hitay Özbay for his enthusiasm, encouragement, patience, and never ending guidance. Working under his supervision has been a great pleasure and honor. I have learned a lot from him and his cheerful attitude has always been a good reason to cheer up. I also thank to the members of my thesis committee Prof. Dr. Ömer Morgül and Asst. Prof. Dr. İbrahim Körpeoğlu for their support.

I would like to thank my friend Özlem Güler for being my project mate during graduate courses and for always being by my side. Sengör Altıngövde is one of the most inspiring and funny friends ever and I thank him for being the source of humor during the creation of this thesis. Many thanks go to Rohat Melik, he has helped me with every little detail in these last few months. I also thank my office mates Zeynep Yücel and Mehmet Kök. The friendly atmosphere that we have created in EA227 is to be missed a lot.

Finally, my thanks go to my family. Their efforts beginning from the first day of my life and their unlimited support made everything possible.

Contents

1	Introduction	1
2	PI, PD and PID Control for Integrating Systems with Time Delay	5
2.1	PD Controller	6
2.2	PI Controller	8
2.3	PID Controller	10
2.4	Switching Control	11
3	Application to AQM	12
3.1	Mathematical Model of AQM Supporting TCP Flows	12
3.1.1	Linearization	14
3.2	Application of the proposed control methods	18
3.2.1	An alternative PID controller design	19
3.2.2	Switching Control	20

3.2.3	Digital implementation of the controllers	22
4	Simulation Results	24
4.1	Simulation results for the nominal network parameters	24
4.2	Robustness to uncertainties in the network parameters	27
4.3	Switching Control	29
5	Conclusions	37
A	ns-2 C++ function adapted for PID controller	39
B	ns-2 tcl code (PI controller case)	41

List of Figures

1.1	Delay representation	2
2.1	Feedback system	6
3.1	RED: Probability of packet marking vs. average queue size	16
3.2	Linearized Feedback System	18
3.3	$\Gamma(\omega)/R_o$ versus ω for $f(s)=\frac{e^{-R_o s}}{1+R_o s}$	19
3.4	$\Gamma(\omega)/R_o$ versus ω for $f_o(s)=\frac{e^{-R_o s}(8\theta s+1)}{(KR_o s+1)(R_o s+1)}$	20
3.5	RTT	21
4.1	Network Topology	25
4.2	ns-2 simulations: queue length vs time.	26
4.3	ns-2 simulations: queue length vs time for $c=1563$ packets/sec.	29
4.4	ns-2 simulations: queue length vs time for $N=40$ TCP flows.	30
4.5	ns-2 simulations: queue length vs time for $RTT=0.4$ sec.	31
4.6	Mean and standard deviation of queue length under load variations	32

4.7	Errors under load variations	32
4.8	Errors under load variations	32
4.9	Mean and standard deviation of queue length under RTT variations	33
4.10	Errors under RTT variations	33
4.11	Errors under RTT variations	33
4.12	Mean and standard deviation of queue length under capacity variations	34
4.13	Errors under capacity variations	34
4.14	Errors under capacity variations	34
4.15	ns-2 simulations: (a) RTT_1 (b) single controller, K_{pi0} (c) single controller, K_{pi1} (d) single controller, K_{pi2} (e) two switching controllers, K_{pi1} and K_{pi2} , (f) N=16 switching controllers	35
4.16	ns-2 simulations: (a) RTT_2 (b) single controller, K_{pi0} (c) single controller, K_{pi1} (d) single controller, K_{pi2} (e) two switching controllers, K_{pi1} and K_{pi2} , (f) N=16 switching controllers	36

List of Tables

4.1	The analysis of simulation results	26
4.2	The analysis of simulation results for $c=1563$ packets/sec	28
4.3	The analysis of simulation results for $N=40$ TCP flows	28
4.4	The analysis of simulation results for $RTT=0.4$ sec	28
4.5	The analysis of simulation results for RTT_1	35
4.6	The analysis of simulation results for RTT_2	36

To my parents, Sultan and Fettah

Chapter 1

Introduction

Proportional-Integral-Derivative (PID) controllers are widely used in various engineering applications since they are easy to implement using low storage memory, their computational complexity is low and they are applicable to most control systems, [1], [6]. PID controllers may not provide optimal control in many situations but they have proved their usefulness in providing satisfactory control with the advantages above. In fact, today more than half of the industrial controllers utilize PID or modified PID control schemes, [16].

In the most general form PID controllers are second order systems

$$K_{pid}(s) = K_p + \frac{K_i}{s} + K_d s \quad (1.1)$$

where K_p is the proportional gain, K_i is the integral gain and, K_d is the derivative gain. Furthermore, (1.1) with $K_d=0$ is a PI controller and (1.1) with $K_i=0$ is a PD controller.

The main issue in PID control design is the “optimal” tuning of the gains, K_p, K_i, K_d and many different types of tuning guidelines have been proposed in the literature. Some current techniques for PID controller design include the Ziegler-Nichols tuning methods, internal model controller (IMC) based tuning



Figure 1.1: Delay representation

methods and dominant pole design (Cohen-Coon method). A comprehensive survey on PID controller design methods can be found in [1].

In this study we consider integrating systems with time delays (i.e. the plant contains an integrator, a time delay, and possibly other stable terms). Delays are present in a system if a signal or a physical variable originating from one part of the system becomes available in another part of the system after a lapse of time. The block diagram of a delay element is given in Figure (1.1) and in Laplace domain this delay is represented as e^{-Ts} . Many problems in control engineering involve time delays such as chemical processes, aerodynamics and communication networks. Tuning of PID controllers for this type of unstable time delay systems is an active research area, see [13], [20], [24] and the references therein.

In a recent study, stabilizing PID controllers are obtained for a class of MIMO (multi-input multi-output) unstable plants with delays in the input and output channels, [17]. Using the results of this study resilient PI (Proportional-Integral) and PD (Proportional-Derivative) controllers with largest allowable controller gain intervals are investigated for plants with at most two unstable poles, [7]. Maximization of the controller gain intervals is an important problem since this way the sensitivity of the closed loop stability to perturbations in the controller parameters can be minimized. In this thesis, the results of [17] and [7] are used to obtain the largest allowable gain intervals for a SISO (single-input single-output) plant with a given set of nominal plant parameters and the center of these maximized intervals is selected as the optimal gains of the controllers. Controllers with perturbed gains can be seen as “optimal” controllers for plants with perturbed nominal parameters. So, the controllers proposed here are expected to

work for a wide range of plant parameters. Controllers using this method can be applied to integrating time delay systems appearing in data flow control in computer networks, target tracking problems in robotics applications, and material transport systems encountered in process control.

In this study, proposed PID control method is applied to AQM schemes supporting TCP flows. TCP is currently the dominant transport protocol used in the Internet and with the addition of AQM, routers are able to detect congestion before the queues overflow. Congestion occurs in a network when the demand for the link bandwidth is higher than the available capacity of the resource, [23]. When some of the links are congested in a network, buffers at the routers overflow and any new incoming packets are dropped or lost. This situation leads to long return-trip-times (RTT), i.e. delays, and may even result in an instability in the network (e.g. congestion collapse in the Internet), [5]. On the other hand, having an empty queue at a buffer means that the link capacity is not fully used, i.e. network resources are under-utilized in this case. Therefore, the goal of AQM is to keep the queue size at the buffers at a certain desired level. For this purpose, most AQM schemes mark the Explicit Congestion Notification (ECN) bit of the packets passing through the link according to a certain rule, [21]. In early AQM methods, such as RED (Random Early Detection) [4] and REM (Random Exponential Marking) [2], packet marking probability (control signal) is a static function of the queue (or average queue) size. In [11], [14] mathematical models of AQM schemes supporting TCP flows have been proposed and with the help of these models a control-theory based approach to analyze or design AQM schemes has been possible. First attempts to design a dynamic controller appear in [8, 9], where the authors use the fluid model of the TCP flows developed in [14] to design a PI controller. This AQM scheme is currently implemented in ns-2, [15] and in [9] it is shown that it performs better than RED. Therefore, PI controller of [9] will be taken as the “benchmark” design for comparisons with the alternative AQM controller tuning method proposed here.

Different schemes of AQM deploying control theory methods have been proposed in the literature. In [12], a rate based controller method called adaptive virtual queue (AVQ) has been proposed. A PID controller has been designed for AQM with time domain analysis methods in [22]. Another rate based control mechanism which is called virtual rate control (VRC) has been proposed in [19] and in fact it is just a PID controller. In [18], an H_∞ controller has been designed and later in [26], H_∞ performance of the designed controller with respect to uncertainty bound of RTT has been analyzed. Another robust H_∞ based AQM control scheme (RHC) has been introduced in [3] to maintain queue stabilization in the presence of variations of the network parameters. In [25] a variable structure sliding mode controller based AQM scheme has been designed for better performance in terms of packet loss ratio, link utilization and queue fluctuation. With the exception of [26], where H_∞ based AQM techniques are used, the papers mentioned above do not consider time varying propagation delays, which may occur due to changes in the communication channels. [26] proposes switching among a set of robust controllers designed at small operating ranges for better performance than a single controller for the whole range of time delay. However the proposed H_∞ control method of [26] is relatively complicated to implement in real networks. Therefore in this thesis switching is done among PI controllers since they are much simpler and easy to implement.

Remaining parts of the thesis are organized as follows. Details of the controller parameter tuning is given in Section 2. Application of this tuning method to the AQM problem can be found in Section 3 whereas the simulation results of this application along with the comparisons of the benchmark design with the proposed design can be found in 4. Concluding remarks are made in Section 5.

Chapter 2

PI, PD and PID Control for Integrating Systems with Time Delay

In a recent study [7], existence of stabilizing PID (Proportional-Integral-Derivative) controllers is shown for LTI (linear time-invariant) MIMO (multi-input multi-output) unstable plants with possibly uncertain time delays in the input and output channels. For plants with one or two unstable poles the parameters of these PID controllers are explicitly formulated from a small gain argument. Using the results of this study, stabilizing PI (Proportional-Integral) and PD (Proportional-Derivative) controllers with largest allowable gain intervals are investigated in [17] for plants with only one unstable pole. Using the results of these two studies we propose a new method for tuning the parameters of PI and PID controllers for integrating processes with time delays.

Consider the unity feedback system in the Figure 2.1. The transfer function of the plant is in the form

$$G_{\Lambda}(s) = \frac{K_o}{s} e^{-hs} g(s), \quad h > 0 \quad (2.1)$$

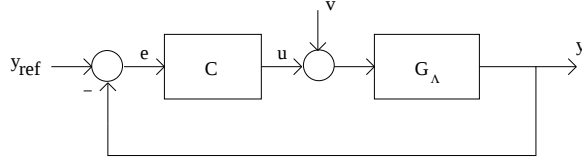


Figure 2.1: Feedback system

where $g(s)$ is an arbitrary stable rational transfer function satisfying $g(0) = 1$ and therefore the only unstable pole of the plant is at $s=0$. For notational convenience we define

$$f(s) := e^{-hs}g(s) \quad (2.2)$$

and let

$$G(s) = \frac{K_o}{s} g(s) \quad (2.3)$$

be the finite dimensional part of the plant. A coprime factorization of $G(s)$ is in the form $G(s) = X(s)Y(s)^{-1}$ with $X(0) = sG(s)|_{s=0} \neq 0$. In this case, $Y(s) = s/(as + 1)$, for any $a > 0$, and since $g(0) = 1$, we have $X(0) = K_o$.

2.1 PD Controller

In [7] it is shown that a PD controller in the form

$$K_{pd}(s) = \alpha X(0)^{-1}(1 + \hat{K}_d s) \quad (2.4)$$

stabilizes the feedback system of Figure 2.1 for any $\alpha \in \mathbb{R}$ satisfying

$$0 < \alpha < \|\Phi_\Lambda\|_\infty^{-1} = (\sup_\omega \|\Phi_\Lambda(j\omega)\|)^{-1} \quad (2.5)$$

where

$$\Phi_\Lambda := \frac{G_\Lambda(s)K_{pd}(s)}{\alpha} - s. \quad (2.6)$$

This overall gain interval is maximized in [17] for the purpose of obtaining a resilient controller. Maximizing $\|\Phi_\Lambda\|_\infty^{-1}$ is equivalent to minimizing

$$\mu := \|\Phi_\Lambda\|_\infty = \|s^{-1}(f(s)(1 + \hat{K}_d s) - 1)\|_\infty \quad (2.7)$$

For $\widehat{K}_d$ being the free parameter this problem reduces to finding a $\widehat{K}_d \in \mathbb{R}$ such that

$$\mu = \left\| \frac{f(s) - 1}{s} + \widehat{K}_d f(s) \right\|_{\infty} \quad (2.8)$$

is minimized. Let the optimal solution be $\widehat{K}_{d, \text{opt}}$. This is a single parameter scalar $\mathcal{H}_{\infty}$ norm minimization problem which can be solved numerically with brute force search. The following algorithm steps will find a solution:

1. Choose candidate values of $\widehat{K}_d = K_1, \dots, K_N$ and frequency values $\omega = \omega_1, \dots, \omega_M$. The optimization will be done over the K values whereas norm will be calculated over the ω values.

2. For $k = 1, \dots, N$ and $l = 1, \dots, M$ compute

$$\Psi(K_k, \omega_l) := \left\| \frac{f(j\omega_l) - 1}{j\omega_l} + K_k f(j\omega_l) \right\|. \quad (2.9)$$

3. Define $\mu(K_k) = \max_{\omega_l} \Psi(K_k, \omega_l)$.

4. Optimal $\widehat{K}_d$ is $\widehat{K}_{d, \text{opt}} = \arg \min_{K_k} \mu(K_k)$.

One can find a solution to (2.8) with this algorithm but that solution will be sensitive to the number of grid points chosen for K and ω . Therefore, a direct computation of the optimal solution will be more useful.

A closed form expression of the solution is given in Proposition 2 of [17] as: Let $f(s) = g(s)e^{-hs}$, with $g \in \mathcal{H}_{\infty}$, $g(0) = 1$, and $h > 0$. Also let $\rho(\omega) := |f(j\omega)|$ and $\phi(\omega) := \angle f(j\omega)$ being the magnitude and phase functions of $f(j\omega)$ and

$$\eta(\omega) := \left| \frac{\rho(\omega) - \cos(\phi(\omega))}{\omega} \right| \quad (2.10)$$

being a function with maximum, η_o , which is attained at a single frequency ω_o , in other words,

$$\eta_o := \max_{\omega} \left| \frac{\rho(\omega) - \cos(\phi(\omega))}{\omega} \right| = \eta(\omega_o).$$

Then, the optimal solution of

$$\widehat{K}_{d, \text{opt}} = \arg \min_{\widehat{K}_d \in \mathbb{R}} \left\| \frac{f(s) - 1}{s} + \widehat{K}_d f(s) \right\|_{\infty} \quad (2.11)$$

is given by

$$\widehat{K}_{d, \text{opt}} = -\frac{\sin(\phi(\omega_o))}{\omega_o \rho(\omega_o)} \quad (2.12)$$

if

$$\Gamma(\omega) := \eta_o^2 - \eta^2(\omega) - |K_o - \widehat{K}_{d, \text{opt}}(\omega)|^2 \rho^2(\omega) \geq 0 \quad \forall \omega \quad (2.13)$$

is satisfied. Note that to find $\widehat{K}_{d, \text{opt}}$ we have to just find ω_o numerically whereas the algorithm given above requiring two dimensional search is much complicated.

Therefore (2.12) is the optimal solution of (2.8) which maximizes the lower bound for $\|\Phi_\Lambda\|_\infty^{-1}$ and the corresponding maximal lower bound for $\|\Phi_\Lambda\|_\infty^{-1}$ is found as $1/\eta_o$.

We propose to take $\widehat{K}_{d, \text{opt}}$ as the derivative gain and the center of the maximum allowable overall gain interval as the overall gain, i.e. $\alpha=1/2\eta_o$ and the resulting PD controller is in the form

$$K_{pd}(s) = \frac{1}{2\eta_o} \left(\frac{1}{K_o} + \widehat{K}_d s \right). \quad (2.14)$$

The above PD controller design technique is valid for all plants in the form (2.1) where $g(s)$ can be any stable rational transfer function satisfying $g(0) = 0$, with one caveat: one has to check that the corresponding η has a single maximum. For a first order strictly proper $g(s)$ (as in the AQM problem to be shown) this is automatically satisfied.

2.2 PI Controller

According to [7] a PI controller in the form

$$K_{pi}(s) = \alpha X(0)^{-1} \left(1 + \frac{\gamma}{s} \right) \quad (2.15)$$

stabilizes the feedback system in Figure 1 for any $\alpha \in \mathbb{R}$ satisfying (2.5) with $\widehat{K}_d = 0$ and any $\gamma \in \mathbb{R}$ satisfying

$$0 < \gamma < \gamma_{max} := \left\| \frac{\frac{\alpha}{s} f(s) (1 + \frac{\alpha}{s} f(s))^{-1} - 1}{s} \right\|_\infty^{-1}. \quad (2.16)$$

For such a PI controller, the maximum integral action gain interval is studied for fixed overall controller gain in [17]. First the function $\theta := \|\frac{f(s)-1}{s}\|_\infty$ is defined and then (2.5) can be rewritten as

$$0 < \alpha < \theta^{-1} \quad (2.17)$$

and the lower bound for γ_{max} is found as

$$\gamma_* = \alpha(1 - \alpha\theta) \leq \gamma_{max} . \quad (2.18)$$

With straightforward calculation it can be shown that the optimal α maximizing this lower bound is $\alpha_* = 1/2\theta$ and the maximal γ_* corresponding to this optimal α is $\gamma_{*,max} = \alpha_*/2$.

To make the controller robustly stable with respect to largest perturbations in the controller parameters we propose to choose controller's proportional gain as α_* and integral gain as $\gamma_{*,max}/2$, which is the center of the maximal interval. The resulting PI controller is in the form

$$K_{pi}(s) = \frac{1}{2\theta} \frac{1}{K_o} \left(1 + \frac{1}{8\theta s}\right). \quad (2.19)$$

It should be noted that, while we tried to maximize the upper bound of the overall controller gain, the optimal overall gain came up as the center of the overall controller gain interval. Therefore both the overall controller gain and integral gain of the optimal PI controller are at the center of their corresponding intervals.

The computation of (2.19) above is rather simplified thanks to conservatism introduced by (2.18). In fact, γ_{max}^{-1} is a function of α :

$$\gamma_{max}^{-1} = \sup_{\omega} \left| \frac{1}{j\omega + \alpha f(j\omega)} \right| =: Q(\alpha).$$

This norm should be calculated for every α in the interval $0 < \alpha < 1/\theta$. Then γ_{max}^{-1} could be chosen as the minimum of $Q(\alpha)$. However, this method requires

$M * N$ calculations for M being the ω grid points for $\mathcal{H}_\infty$ norm computation and N being the α grid points. Using (2.18) we skip the H_∞ norm computation for every fixed α . The conservatism introduced by (2.18) is currently under investigation.

2.3 PID Controller

A PID controller can be obtained from the product of PI and PD controllers,

$$K_{pid}(s) = (K_{p1} + \frac{K_i}{s})(K_{p2} + K_d s) = (K_{p1}K_{p2} + K_i K_d) + \frac{K_i K_{p2}}{s} + K_{p1} K_d s.$$

So, we will first design a PD controller, then design a PI controller for the PD controlled open loop plant. The transfer function of the PD controlled plant is

$$G_\Lambda(s)K_{pd}(s) = \frac{K_o}{s} e^{-hs} g(s) \frac{1}{2\eta_o} (\frac{1}{K_o} + \widehat{K}_d s) =: \frac{1}{2\eta_o} \frac{e^{-hs} g_o(s)}{s}$$

with

$$g_o(s) := g(s) (1 + K_o \widehat{K}_d s).$$

Let's define

$$\theta_o := \left\| \frac{g_o(s)}{s} \right\|_\infty \quad (2.20)$$

such that a PI controller is formed with the method explained in Section 2.2:

$$K_{pi}(s) = \frac{1}{2\theta_o} 2\eta_o (1 + \frac{1}{8\theta_o s}). \quad (2.21)$$

So, the PID controller is

$$K_{pid}(s) = \frac{(1 + K_o \widehat{K}_d s)}{16K_o \theta_o^2} \frac{(1 + 8\theta_o s)}{s}. \quad (2.22)$$

It is also possible to obtain a PID controller by first designing a PI controller and then designing a PD controller for the PI controlled open loop plant. Such a plant will have two poles at zero, one from the original plant (2.1) and one from the PI controller (2.19) however the method proposed in [17] is valid for

plants with only one unstable pole. Therefore, we will not use this method for designing a PID controller for (2.1). On the other hand, in the application problem (AQM) we will find an alternative to overcome this obstacle, see Section (3.2.1).

2.4 Switching Control

For real systems with time varying time delays, the above controller design methods can be further improved. In [26], it was proposed to divide the range of time delay into a number of smaller regions and designing controllers for each of these regions. Then, while the time delay varies, switching between these controllers accordingly is proposed to give better results than using a single controller designed for the entire region of time delay. In that study, the application was done for two switching controllers, here we increase the number of controllers and we apply the idea to our proposed controllers. This method is detailed in Section 3.2.2 for the studied application problem.

Chapter 3

Application to AQM

3.1 Mathematical Model of AQM Supporting TCP Flows

This thesis considers a network configuration consisting of a single bottleneck link supporting N TCP flows and the routers mark the ECN bits of packets to inform the TCP sources of congestion as proposed in [21]. The dynamical model of TCP was developed using stochastic differential equation analysis and fluid flow approximation in [14]. In this study a simplified version of this model which was introduced in [9] is used. The model provides means for analysis and design for TCP in congestion avoidance mode only, thus ignores the slow-start and timeout mechanisms of TCP. In this model TCP dynamics are represented with the following nonlinear differential, time-delayed equations:

$$\dot{W}(t) = \frac{1}{R(t)} - \frac{W(t)}{2} \frac{W(t - R(t))}{R(t - R(t))} p(t - R(t)) \quad (3.1)$$

$$\dot{q}(t) = N(t) \frac{W(t)}{R(t)} - c(t) \quad \text{when } q(t) > 0 \quad (3.2)$$

where $\dot{x}(t)$ denotes time derivative and

W : average TCP window size (packets),
 q : average queue length (packets),
 R : round-trip time (seconds),
 c : link capacity (packets),
 N : number of TCP flows (i.e. load),
 p : probability of packet mark.

Here, RTT (total delay in the feedback path) is expressed by

$$R(t) = T_p(t) + \frac{q(t)}{c(t)} \quad (3.3)$$

where $T_p(t)$ is the propagation delay and obviously, $q(t)/c(t)$ is the queueing delay. Note that in this study both time invariant and time varying round trip time cases are considered. For the later case, the assumption is that the variations of $T_p(t)$ are slow compared to the variations of $q(t)/c(t)$, but the magnitude of the variations of the propagation delay is larger than the variations of queueing delay. Therefore, the time variation in RTT will be taken to be due the variation in the propagation delay.

Equation 3.1 specifies the TCP window dynamic in congestion avoidance mode incorporating the additive increase and multiplicative decrease (AIMD) behavior of TCP. In the right side of this equation, the $1/R$ term represents the increase in the sending rate when no congestion is detected and the $W/2$ term models the decrease in the sending rate when congestion is detected, i.e. packets are marked with probability p .

Likewise, Equation 3.2 models the queue length dynamic as the difference between incoming flow rate and outgoing flow rate (i.e. link capacity). It is possible to use these equations to describe TCP as a feedback control system, where p is the control input generated by a feedback from q , i.e.

$$p = \mathcal{K}(q) \quad (3.4)$$

where $\mathcal{K}$ is the feedback control operator. Clearly, the above system is nonlinear and it depends on an implicit function $R(t - R(t))$. In general, "optimal" controllers for such nonlinear dynamical systems are difficult to obtain. Typically the system is linearized around an equilibrium point using small signal analysis.

3.1.1 Linearization

To perform linearization on equations (3.1) and (3.2), we first assume that the link capacity and the number of TCP sessions are constant, $c(t) = c$ and $N(t) = N$, as in [9]. The following equations should be satisfied for the equilibrium point:

$$\dot{W} = 0 \implies \frac{W_o^2 p_o}{2} = 1 \quad (3.5)$$

$$\dot{q} = 0 \implies \frac{W_o N}{R_o} = c. \quad (3.6)$$

Then, given the set of network parameters $\zeta \doteq (N, C, T_p)$, the operating point is defined by the solution (R_o, W_o, p_o) of the following equations:

$$R_o = \frac{q_o}{c} + T_p \quad (3.7)$$

$$W_o = \frac{R_o C}{N} \quad (3.8)$$

$$p_o = \frac{2}{W_o^2} \quad (3.9)$$

for a desired equilibrium queue size q_o . Additionally, $W_o \in (0, \bar{W})$, $q_o \in (0, \bar{q})$, $p_o \in (0, 1)$ should also be satisfied where $\bar{W}$ is the maximum window size and $\bar{q}$ is the buffer capacity. We also assume that the time delay element $R(t)$ in $(t - R(t))$ is constant and equal to R_o . Then, the equations (3.1) and (3.2) can be rewritten as:

$$\dot{W}(t) = \frac{1}{\frac{q(t)}{c} + T_p} - \frac{W(t)}{2} \frac{W(t - R_o)}{\frac{q(t - R_o)}{c} + T_p} p(t - R_o) \quad (3.10)$$

$$\dot{q}(t) = N(t) \frac{W(t)}{R(t)} - c(t) \quad \text{when } q(t) > 0 \quad (3.11)$$

In order to linearize these two equations we first define

$$f(W, W_d, q, q_d, p_d) \doteq \frac{1}{\frac{q}{c} + T_p} - \frac{W W_d}{2 \frac{(q_d)}{c} + T_p} p_d \quad (3.12)$$

$$g(W, q) \doteq \frac{N}{\frac{q}{c} + T_p} W - c \quad (3.13)$$

where $W_d \doteq W(t - R_o)$, $q_d \doteq q(t - R_o)$, $p_d \doteq p(t - R_o)$ are the delayed variables.

Using the operating point relationships (3.5) and (3.6), the partial derivatives of f and g at the operating point are found as:

$$\begin{aligned} \frac{\partial f}{\partial W} &= -\frac{W_o p_o}{2R_o} = -\frac{1}{2R_o} = -\frac{1}{W_o R_o} = -\frac{N}{R_o^2 C} \\ \frac{\partial f}{\partial W_d} &= \frac{\partial f}{\partial W_d} = -\frac{N}{R_o^2 C} \\ \frac{\partial f}{\partial q} &= -\frac{1}{R_o^2 c} \\ \frac{\partial f}{\partial q_d} &= -\frac{W_o^2 p_o}{2R_o^2} \left(-\frac{1}{c}\right) = \frac{1}{R_o^2 c} \\ \frac{\partial f}{\partial p_d} &= -\frac{W_o^2}{2R_o} = -\frac{R_o c^2}{2N^2} \\ \frac{\partial g}{\partial q} &= -\frac{NW_o}{R_o^2 c} = -\frac{1}{R_o} \\ \frac{\partial g}{\partial W} &= -\frac{N}{R_o}. \end{aligned}$$

We then obtain

$$\begin{aligned} \delta \dot{W}(t) &= \delta W(t) \frac{\partial f}{\partial W} + \delta W(t - R_o) \frac{\partial f}{\partial W_d} + \delta q(t) \frac{\partial f}{\partial q} + \delta q(t - R_o) \frac{\partial f}{\partial q_d} + \delta p(t - R_o) \frac{\partial f}{\partial p_d} \\ &= -\frac{N}{R_o^2 C} (\delta W(t) + \delta W(t - R_o)) - \frac{1}{R_o^2 c} (\delta q(t) - \delta q(t - R_o)) - \frac{R_o c^2}{2N^2} \delta p(t - R_o) \end{aligned} \quad (3.14)$$

$$\delta \dot{q}(t) = \delta W(t) \frac{\partial g}{\partial W} + \delta q(t) \frac{\partial g}{\partial q} = \frac{N}{R_o} \delta W(t) - \frac{1}{R_o} \delta q(t) \quad (3.15)$$

with

$$\delta_W = W - W_o,$$

$$\delta_q = q - q_o,$$

$$\delta_p = p - p_o$$

representing the small signal variables.

We will now consider the Random Early Detection (RED), [4], algorithm as the AQM scheme, and show that q_o is an equilibrium point. RED takes an average measure of the router's queue length and randomly marks ECN bits of packets. The averaging which is achieved by a low-pass filter, is for eliminating the effects of restarts and time-outs on the feedback signal. The randomness on the other hand, is meant to overcome flow synchronization, [9]. The packet marking probability as a function of average queue size is illustrated in Fig. 3.1.

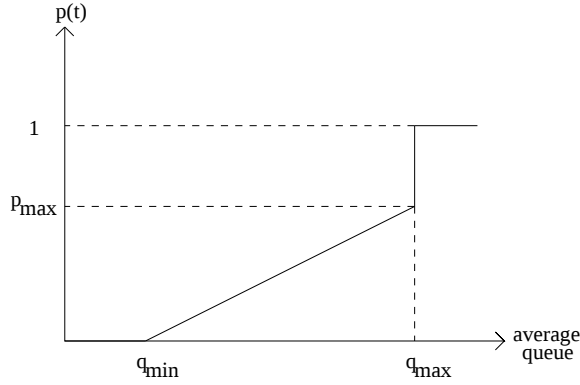


Figure 3.1: RED: Probability of packet marking vs. average queue size

As shown in the figure, packets are marked when the average queue size is between maximum and minimum threshold values, q_{max} and q_{min} , with probability

$$p(t) = \left(\frac{p_{max}}{q_{max} - q_{min}} \right) (q_{av}(t) - q_{min}). \quad (3.16)$$

The goal of RED is keeping $q_{av}(t)$ near its equilibrium point between q_{max} and q_{min} . At equilibrium $q_{av_o} = q_o$ and the slope in Fig. 3.1 is

$$L \doteq \frac{p_{max}}{q_{max} - q_{min}}. \quad (3.17)$$

Then the probability of packet mark satisfies:

$$p_o = \frac{p_{max}}{q_{max} - q_{min}} q_o - \frac{p_{max}}{q_{max} - q_{min}} q_{min} \doteq \alpha q_o - \beta. \quad (3.18)$$

Using equations (3.7), (3.8) and, (3.9)

$$p_o = \frac{2}{W_o^2} = \frac{2N^2}{c^2 R_o^2} = \frac{2N^2}{q_o + cT_p^2} = \alpha q_o - \beta \quad (3.19)$$

is found. Then q_o should satisfy:

$$\alpha q_o(q_o + cT_p)^2 - \beta(q_o + cT_p)^2 - 2N^2 = 0 \quad (3.20)$$

$$q_o^3 + (2cT_p - \frac{\beta}{\alpha})q_o^2 + cT_p(cT_p - 2\frac{\beta}{\alpha})q_o - (\frac{\beta}{\alpha})c^2T_p^2 - \frac{2N}{\alpha} = 0. \quad (3.21)$$

From (3.18) it is obvious that

$$\frac{\beta}{\alpha} = q_{min}. \quad (3.22)$$

Assuming $q_{min} = 0$, equation (3.21) can be rewritten as

$$q_o^3 + 2cT_pq_o^2 + c^2T_p^2q_o - \frac{2N}{L} = 0. \quad (3.23)$$

Let's write the Routh Hurwitz array for this equation:

$$\begin{array}{ccc} s^3 & 1 & c^2T_p^2 \\ s^2 & 2cT_p & -2N^2L \\ s & \mathcal{X} & \\ 1 & -2N^2L & \end{array}$$

From the array it is found that: $\mathcal{X} = \frac{2c^3T_p^3 + 2N^2L}{2cT_p}$. However, independent of the sign of $\mathcal{X}$, there is always one sign change in the first row of the array. Therefore, there is always one root of q_o in the right half plane. This proves that the equilibrium point of equations (3.1) and (3.2) exists and is unique.

Turning back to the linearization problem, the differential equation for the low-pass filter of RED is

$$\dot{q}_{av}(t) = -aq_{av}(t) + aq(t) \quad (3.24)$$

and with simplifying (3.1.1) as:

$$\delta\dot{W} \cong -\frac{2N}{R_o^2C}\delta W - \frac{R_o c^2}{2N^2}\delta p \quad (3.25)$$

we obtain a block diagram representation of the linearized TCP dynamics for RED with separated window and queue dynamics as in Figure (3.2). The transfer

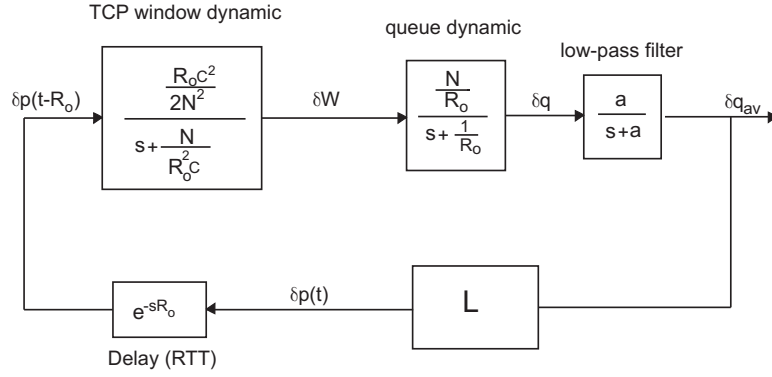


Figure 3.2: Linearized Feedback System

function $G_{pq}(s)$ from the control input δ_p to the output to be regulated δ_q can be obtained as:

$$G_{pq}(s) = e^{-R_o s} \frac{R_o c_o K}{R_o s + K^{-1}} \frac{1}{R_o s + 1}, \quad K = \frac{R_o c_o}{2N_o}. \quad (3.26)$$

In this feedback system, the effect of the RED is the constant gain controller $L = \frac{p_{max}}{q_{max} - q_{min}}$ and the main idea behind designing controllers for AQM is designing controllers and replacing L with these controllers.

3.2 Application of the proposed control methods

With the assumption of $K \gg 1$, the “plant” (3.26) is approximately in the form (2.1) with $K_o := c_o K$, $h = R_o$ and $g(s) = (1 + R_o s)^{-1}$. Hence, we can apply the technique proposed of Section 2 to design PI, PD and PID controllers. However, for the design of PD controller one should not forget to check whether condition (2.13) is satisfied. Figure 3.3 shows that it is satisfied for the plant model of AQM.

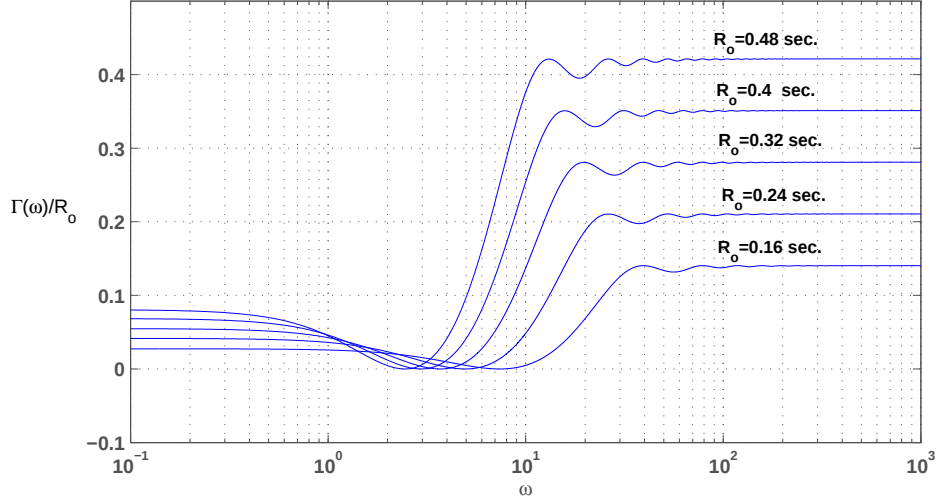


Figure 3.3: $\Gamma(\omega)/R_o$ versus ω for $f(s) = \frac{e^{-R_o s}}{1 + R_o s}$

3.2.1 An alternative PID controller design

As addressed at the end of Section 2.3 there is a second possible PID control design method with first designing a PI controller and then designing a PD controller for the PI controlled plant. This PID controller design method is not applicable to plant (2.1) since the PI controlled plant has two unstable poles and the PD control design method is valid for those with only one unstable pole. However, for the special case of (3.26) we propose an alternative approach. We first design the PI controller as proposed and after that we release the assumption of $K \gg 1$. Then the transfer function of the PI controlled plant becomes

$$G_{pq}(s)K_{pi}(s) = \frac{KR_o}{16\theta^2} \frac{(8\theta s + 1)e^{-R_o s}}{s(KR_o s + 1)(R_o s + 1)} =: \frac{KR_o}{16\theta^2} \frac{g_o(s)}{s} \quad (3.27)$$

with only one unstable pole which comes from the PI controller. Hence the PD design method of Section 2.1 is valid for (3.27). Note that, Figure 3.4 shows that condition (2.13) is satisfied for the PI controlled plant.

For designing the PD controller, first define $f_\nu = e^{-R_o s} g_o(s)$ then, set $\rho_\nu(\omega) := |f_\nu(j\omega)|$ and $\phi_\nu(\omega) := \angle f_\nu(j\omega)$ as the new magnitude and phase functions. Then set $\eta_\nu(\omega) := \left| \frac{\rho_\nu(\omega) - \cos(\phi_\nu(\omega))}{\omega} \right|$. Assume that $\eta_\nu(\omega)$ has a maximum, $\eta_{\nu,o}$, which is

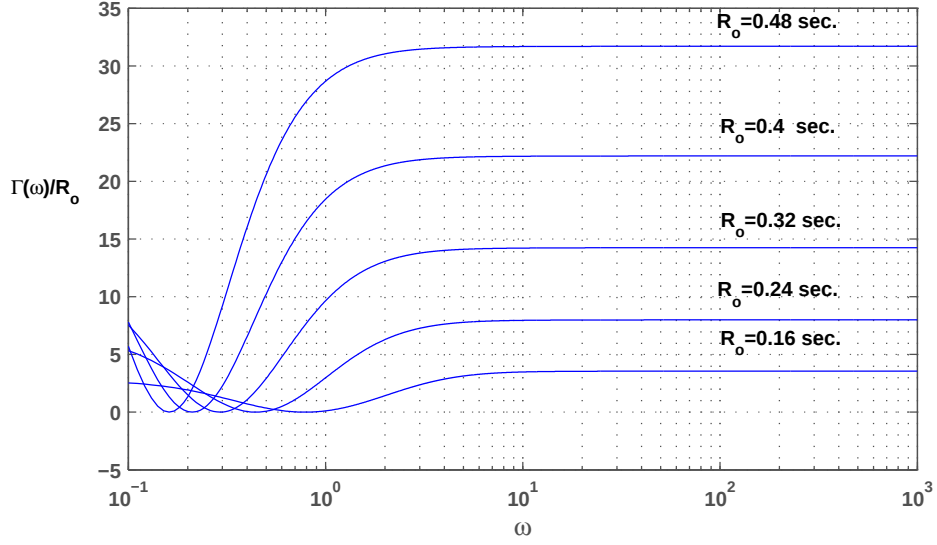


Figure 3.4: $\Gamma(\omega)/R_o$ versus ω for $f_o(s) = \frac{e^{-R_o s}(8\theta s + 1)}{(KR_o s + 1)(R_o s + 1)}$

attained at a single frequency $\omega_{\nu,o}$. With this assumption define

$$\hat{K}_{d\nu} := -\frac{\sin(\phi_\nu(\omega_{\nu,o}))}{\omega_{\nu,o} \rho_\nu(\omega_{\nu,o})}.$$

Then the PD controller is

$$K_{\nu,pd}(s) = \frac{1}{2\eta_{\nu,o}} \left(\frac{16\theta^2}{KR_o} + \hat{K}_{d\nu}s \right).$$

Thus the overall PID controller for the plant (3.26) is the product of $K_{pi}(s)$ and $K_{\nu,pd}(s)$, that is,

$$K_{pid}(s) = \frac{(16\theta^2 + \hat{K}_{d\nu}s)}{32\eta_{\nu,o}c_oR_oK^2} \frac{(1 + 8\theta s)}{s}. \quad (3.28)$$

3.2.2 Switching Control

In computer networks RTT is probably time varying or uncertain and for better performance, in [26], switching among a set of robust controllers was proposed. In this section we use this method to design PI controllers for smaller operating ranges of RTT and switch among them accordingly for the case of known RTT variation. Note that, we only explain the method for PI controller case but it can be easily applied to PD, PID or any other controller. However, we will give

results in Section 4.3 only for switching PI controllers since they give satisfactory results with less parameter computation.

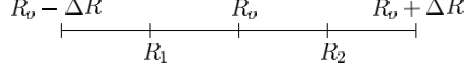


Figure 3.5: RTT

For the case shown in Fig. 3.5, the nominal value of RTT is R_o and we assume that RTT takes values between $R_o - \Delta R$ and $R_o + \Delta R$. If we are to design a single PI controller for this plant we can assume

- (i) the plant is nominal, let $R = R_o$, and implement K_{pi0} ,
- (ii) for $R_o - \Delta R < RTT < R_o$, let $R = R_1 = R_o - \frac{\Delta R}{2}$ and implement K_{pi1}
- (iii) for $R_o < RTT < R_o + \Delta R$, let $R = R_2 = R_o + \frac{\Delta R}{2}$ and implement K_{pi2} .

As the designed PI controllers are robust to the changes in RTT (to be shown in Section 4.2), these three controllers are expected to have good performance in the neighborhood of RTT values they are designed for. In this thesis, we illustrate that it is possible to improve the performance in the case of time varying RTT by applying switching control. Two different configurations are investigated:

- a) Using two of the PI controllers above, we perform mid-point switching. When RTT is in $[R_o - \Delta R, R_o]$ interval K_{pi1} is active and when RTT is in $[R_o, R_o + \Delta R]$ interval K_{pi2} is active.
- b) Instead of dividing $[R_o - \Delta R, R_o + \Delta R]$ interval into two, we divide it into $N \gg 1$ intervals. Therefore, we design N different PI controllers for each of these intervals and as RTT varies among these intervals, the controller parameters switch accordingly.

The switching mentioned above is actually mid-point switching since RTT is assumed to be varying slowly, free of sudden large deviations. It should be noted that, for a more frequently changing time delay case (i.e., chattering), hysteresis switching may be beneficial.

3.2.3 Digital implementation of the controllers

When implementing these controllers in discrete time we need to convert the differential equations into difference equations, there are several ways to do this, see e.g. [10]. In [8] a method for the digital implementation of the PI controller for ns-2 implementation is given. We slightly modify this method and also apply it to the PID controller case. For digital implementation, z -transform of the controller transfer function is calculated without any approximation and a proper sampling frequency, f_s , is chosen. It is advised to choose f_s as 10-20 times the loop bandwidth. In our case, the loop bandwidth is between 0.1-0.3 Hz , so we choose $f_s = 4 Hz$.

The PI controller transfer function is in the form (1.1) with $K_d = 0$, and in z -domain it becomes:

$$K_{pi}(z) = \frac{a - bz^{-1}}{1 - z^{-1}} \quad (3.29)$$

where

$$a = K_p \quad (3.30)$$

$$b = K_p - K_i T_s \quad (3.31)$$

and, $T_s = 1/f_s$. Note that, since $K_{pi}(z)$ is the transfer function from $\delta_q(t)$ to $\delta_p(t)$, (3.1), and assuming p_o is equal to 0, we can write:

$$\frac{p(z)}{\delta_q(z)} = \frac{a - bz^{-1}}{1 - z^{-1}}. \quad (3.32)$$

This transfer function can be converted to the following difference equation for $t = kT_s$:

$$p(kT_s) = a\delta_q(kT_s) - b\delta_q((k-1)T_s) + p((k-1)T_s) \quad (3.33)$$

The digital implementation of this difference equation is done with the following pseudo code:

$$p = a(q - q_o) - b(q_{old} - q_o) + p_{old} \quad (3.34)$$

$$p_{old} = p \quad (3.35)$$

$$q_{old} = q \quad (3.36)$$

Similarly the PID controller (1.1) in z -domain can be given as:

$$K_{pid}(z) = \frac{a_0 + a_1 z^{-1} + a_2 z^{-2}}{b_0 + b_1 z^{-1} + b_2 z^{-2}} \quad (3.37)$$

where

$$a_0 = K_p + K_d \quad (3.38)$$

$$a_1 = -K_p - 2K_d + K_i T_s \quad (3.39)$$

$$a_2 = K_d \quad (3.40)$$

$$b_0 = 1 \quad (3.41)$$

$$b_1 = -1 \quad (3.42)$$

$$b_2 = 0 \quad (3.43)$$

and, $T_s = 1/f_s$. As in the above case, we can write

$$\frac{p(z)}{\delta_q(z)} = \frac{a_0 + a_1 z^{-1} + a_2 z^{-2}}{b_0 + b_1 z^{-1} + b_2 z^{-2}}. \quad (3.44)$$

and for $t = kT_s$ the difference equation is

$$p(kT_s) + b_1 p((k-1)T_s) + b_2 p((k-2)T_s) = a_0 \delta_q(kT_s) - a_1 \delta_q((k-1)T_s) + a_2 \delta_q((k-2)T_s) \quad (3.45)$$

The digital implementation of this difference equation is done with the following pseudo code:

$$p = a_0(q - q_o) + a_1(q_{old} - q_o) + a_2(q_{elder} - q_o) - b_1 p_{old} - b_2 p_{elder} \quad (3.46)$$

$$p_{elder} = p_{old} \quad (3.47)$$

$$q_{elder} = q_{old} \quad (3.48)$$

$$p_{old} = p \quad (3.49)$$

$$q_{old} = q \quad (3.50)$$

The C++ function derived from this pseudo code can be found in Appendix A. It is an adaptation of a similar function which is implemented in the PI controller in ns-2.

Chapter 4

Simulation Results

4.1 Simulation results for the nominal network parameters

The performances of the designed PI, PD and PID controllers are tested via ns-2 simulations. The simulation topology is a dumbbell network with a single bottleneck link shared by N TCP connections which generate FTP flows as in [25]. This bottleneck link is of capacity $C_0 = 10$ Mbps with $T_0 = 20$ ms delay. All the remaining links in the network, namely the links between the TCP sources and the first router, and the links between the TCP sinks and the second router are $C_1 = C_2 = 10$ Mbps links with $T_1 = T_2 = 40$ ms propagation delay, see Figure 4.1. The buffer sizes of both routers are set to 300 packets and each packet is of size 1000 Bytes. The ns-2 code constructed for this configuration can be found in Appendix B.

The nominal system parameters are: $N_o = 30$ TCP flows, $c = C_0 = 1250$ packets/s, $q_o = 150$ packets, $R_o = 0.32$ s. The simulation is carried out for 200 seconds. The queue length plots of the controllers are given in Figure 4.2.

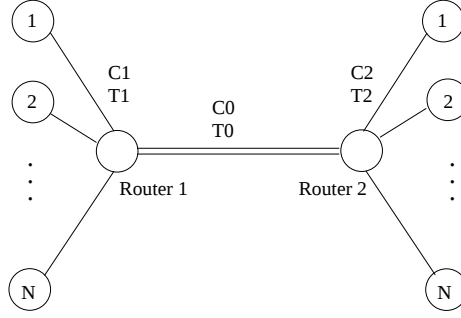


Figure 4.1: Network Topology

Note that, the results for PD controllers are not included in this thesis, since they do not provide improvement in the queue size compared to PI and PID controllers. Furthermore, the designed controllers are not compared with RED since the benchmark PI controller of [9] is already shown to perform better than RED.

In order to evaluate these simulation results we use several error metrics. The first error metric is the RMS value of the percentage error of the queue length with respect to the desired queue length, $q_d = 150$ packets, more precisely,

$$\text{RMS error} = \left(\frac{1}{M} \sum_{k=1}^M \left(\frac{q(k) - q_d}{q_d} \right)^2 \right)^{\frac{1}{2}}, \quad (4.1)$$

here M is the total number of the samples generated by ns-2. To define the second error metric, we specify a tolerance for tracking of the desired queue: let $\mathbf{Q}_d := [q_d - q_{\text{tol}}, q_d + q_{\text{tol}}]$ be a neighborhood of the desired queue. In our case we have $q_d = 150$ packets, and we choose $q_{\text{tol}} = 30$ packets. Define T_0 to be the total length of the time intervals for which $q(t) \notin \mathbf{Q}_d$, and let T_{total} be the length of the total simulation time (in our setting $T_{\text{total}} = 200$ seconds). Then the second error metric we are interested in is $\mathcal{E} = T_0/T_{\text{total}}$. This error provides a better look on the variations of the queue length, and its small values bound the delay jitter.

Since, the queue length settles at approximately 60 seconds, we divide time to two intervals. Firstly, for $t \in [0, 60]$ RMS error is calculated which gives a means of understanding the transient behavior. Secondly, we evaluate $\mathcal{E}$ over the interval

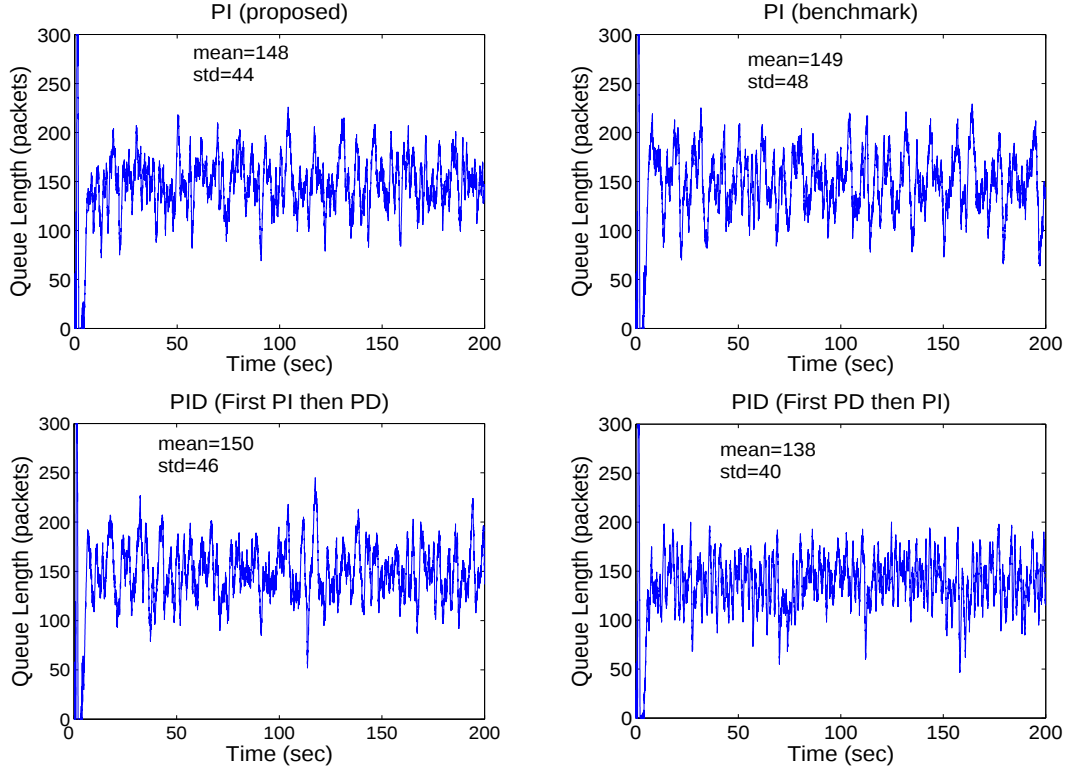


Figure 4.2: ns-2 simulations: queue length vs time.

$t \in (60, 200]$. This analysis reveals the steady state tracking characteristics of the system.

Table 4.1 shows that the proposed PI controller performs better than the benchmark PI controller for all error metrics defined above. Depending on the cost function taken, it may be beneficial to use PID controllers by either first PI then PD design or first PD then PI design. Note that all cost functions are normalized with respect to the time interval. The new proposed PI design gives

Table 4.1: The analysis of simulation results

Controller	RMS error	$\mathcal{E}$	$t \in [0, 60]$	$t \in (60, 200]$
			RMS error	$\mathcal{E}$
PI (proposed)	0.29	0.28	0.47	0.22
PI (benchmark)	0.32	0.37	0.49	0.30
PID (first PI then PD)	0.30	0.30	0.49	0.21
PID (first PD then PI)	0.27	0.29	0.42	0.24

an improvement of 2% to 9% depending on the cost taken; when integrated over time, the benefits of the new design can be significant.

4.2 Robustness to uncertainties in the network parameters

We investigate the robustness of the four controller schemes under variations of number of TCP connections (load) N , the round trip time RTT and, the link capacity c . The controller designs are for the nominal values of the system parameters, as in the previous section, and are fixed throughout the robustness tests. In Figures (4.6, 4.7, 4.8) we vary the number of TCP connections from 10 to 60. In Figures (4.9, 4.10, 4.11) the round trip time is varied between 160 msec and 480 msec. In Figures (4.12, 4.13, 4.14) the capacity is changed from 625 packets/second to 1875 packets/second (from 5 Mbps to 15 Mbps). These figures illustrate the mean and standard deviations of the queue lengths and the error metric comparisons. We omitted the results for the second PID controller (first PD then PI), because it yields similar results to that of the first PID controller (first PI then PD). The time graphs for some of the robustness cases are also given in Figures (4.3, 4.4, 4.5) and analyzed in Tables (4.2, 4.3, 4.4).

As seen in all the figures related to performance robustness analysis, the proposed PI controller performs better than the PI benchmark design except for small values of N . The PI controller scheme that we propose here gives better results than PI benchmark for standard deviation and for all of the error metrics. Simulation results showed that in certain situations the proposed PI controller is better than the PID controller as far as the robustness is concerned.

Table 4.2: The analysis of simulation results for c=1563 packets/sec

Controller	RMS error	$\mathcal{E}$	$t \in [0, 60]$	$t \in (60, 200]$
			RMS error	$\mathcal{E}$
PI (proposed)	0.26	0.29	0.68	0.39
PI (benchmark)	<i>0.27</i>	<i>0.32</i>	<i>0.74</i>	<i>0.40</i>
PID (first PI then PD)	0.27	0.32	0.72	0.40
PID (first PD then PI)	0.31	0.48	0.79	0.43

Table 4.3: The analysis of simulation results for N=40 TCP flows

Controller	RMS error	$\mathcal{E}$	$t \in [0, 60]$	$t \in (60, 200]$
			RMS error	$\mathcal{E}$
PI (proposed)	0.27	0.22	0.66	0.43
PI (benchmark)	<i>0.29</i>	<i>0.30</i>	<i>0.72</i>	<i>0.46</i>
PID (first PI then PD)	0.29	0.34	0.73	0.45
PID (first PD then PI)	0.25	0.21	0.67	0.40

Table 4.4: The analysis of simulation results for RTT=0.4 sec

Controller	RMS error	$\mathcal{E}$	$t \in [0, 60]$	$t \in (60, 200]$
			RMS error	$\mathcal{E}$
PI (proposed)	0.34	0.36	0.76	0.56
PI (benchmark)	<i>0.37</i>	<i>0.42</i>	<i>0.76</i>	<i>0.58</i>
PID (first PI then PD)	0.36	0.40	0.77	0.58
PID (first PD then PI)	0.34	0.45	0.80	0.49

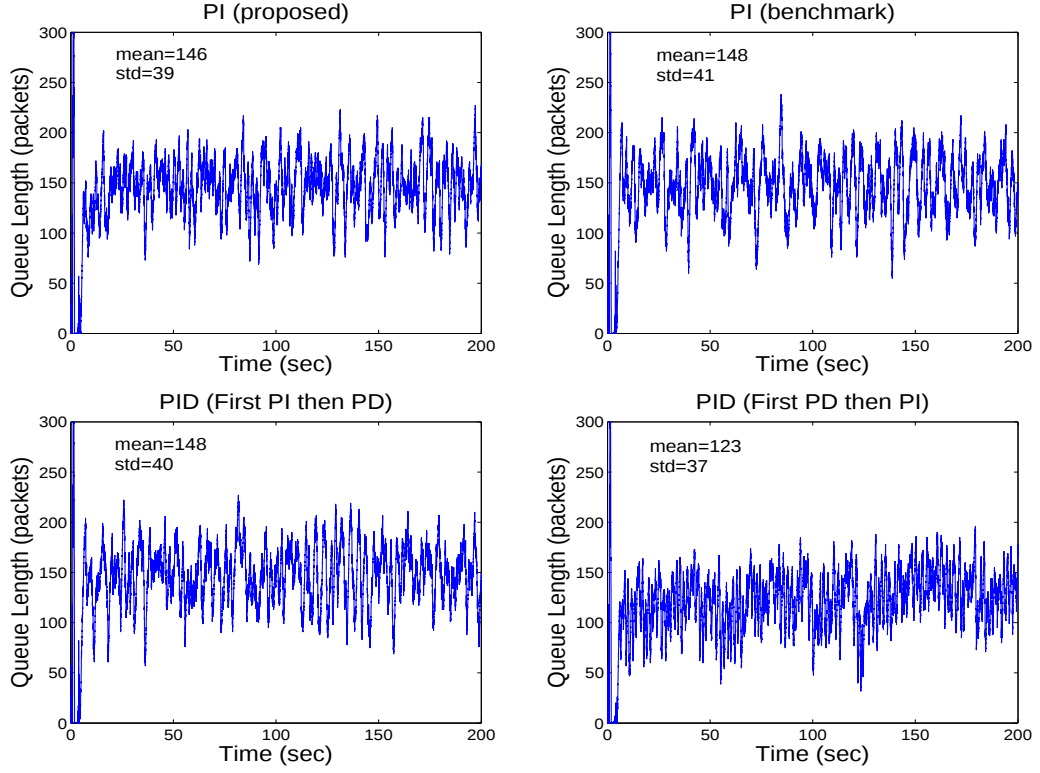


Figure 4.3: ns-2 simulations: queue length vs time for $c=1563$ packets/sec.

4.3 Switching Control

We investigate the performance of five different configurations:

- (a) single controller, K_{pi0} ,
- (b) single controller, K_{pi1} ,
- (c) single controller, K_{pi2} ,
- (d) two switching controllers, K_{pi1} and K_{pi2} ,
- (e) $N=16$ switching controllers.

This experiment is done twice, first for an RTT function, RTT_1 , as in Fig. 4.15(a) and then for a more quickly changing RTT function, RTT_2 , as in Fig. 4.16(a). The plots of queue lengths are given in Fig. 4.15 and Fig. 4.16. The results are given in Tables 4.5 and 4.6. According to the figures and tables,

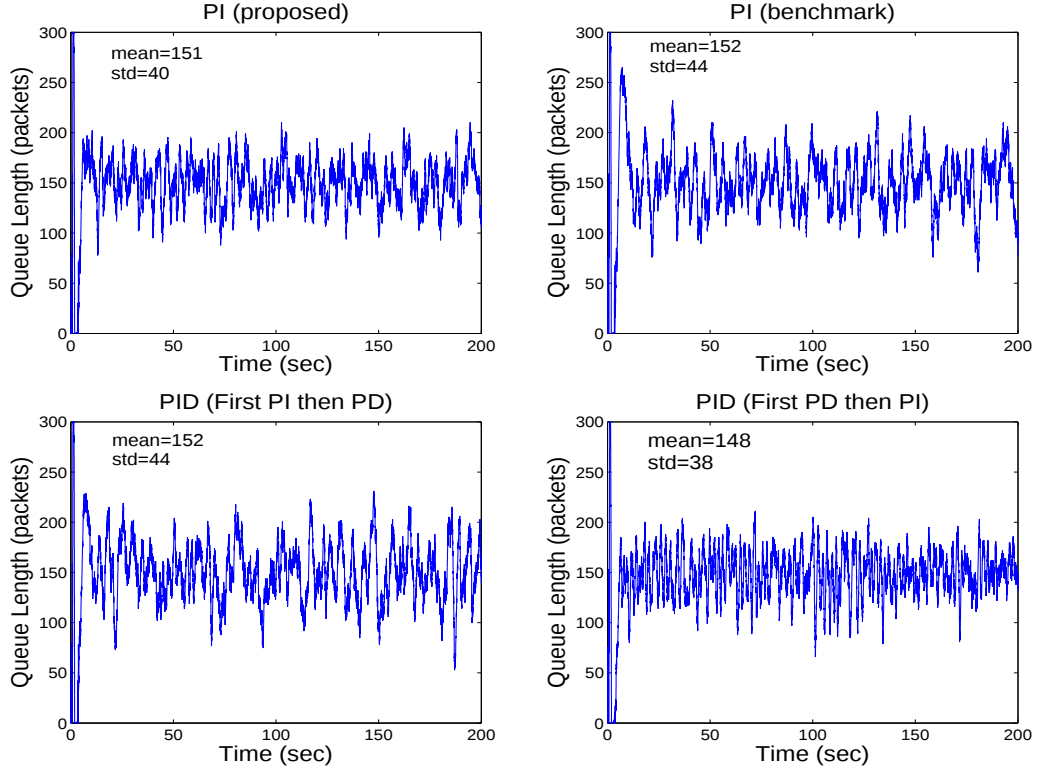


Figure 4.4: ns-2 simulations: queue length vs time for $N=40$ TCP flows.

switching between two controllers improves the tracking and mean, standard deviation, RMS and $\mathcal{E}$ values for both RTT functions. Using 16 controllers does not provide a significant improvement according to the results in Tables 4.5 and 4.6. However for RTT_2 case, around $t \in [30, 40]$ there is a sudden drop in queue length which cannot be compensated by two switching controllers, see Fig. 4.16. This sudden drop in $q(t)$ is not seen when we use K_{pi1} (designed for smallest nominal RTT) or 16 switched controllers. Since the numerical results show that K_{pi1} is not able to track the queue length properly, the sudden drop problem can be solved by 16 controllers with good performance.

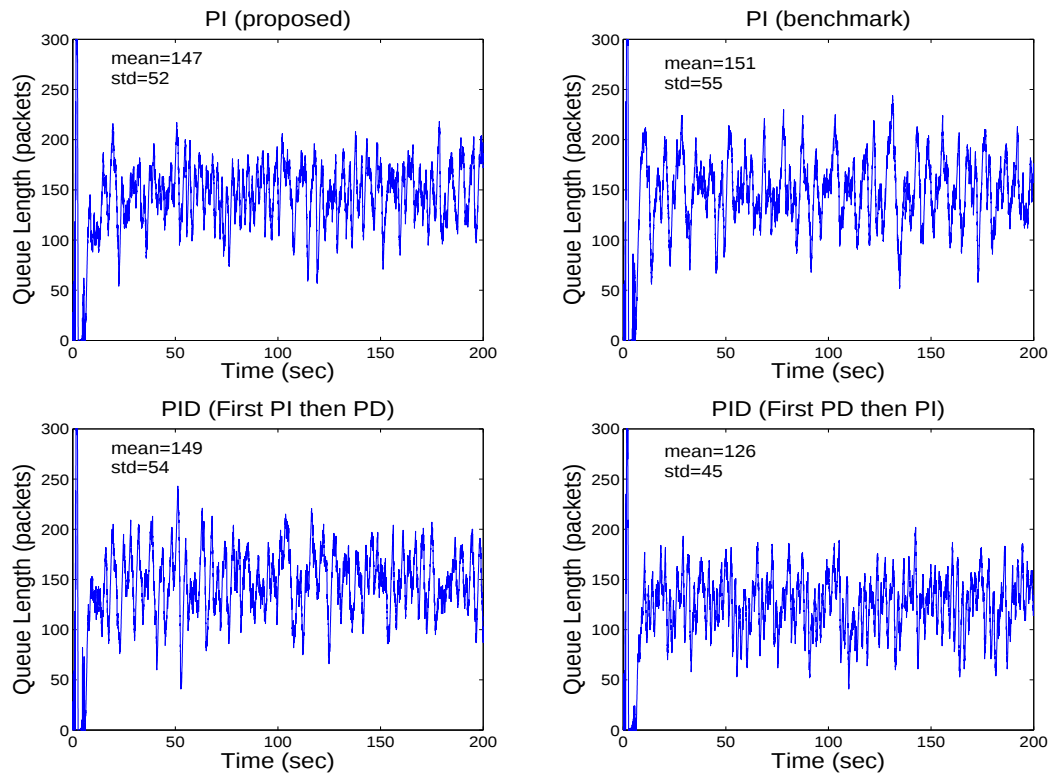


Figure 4.5: ns-2 simulations: queue length vs time for $RTT=0.4$ sec.

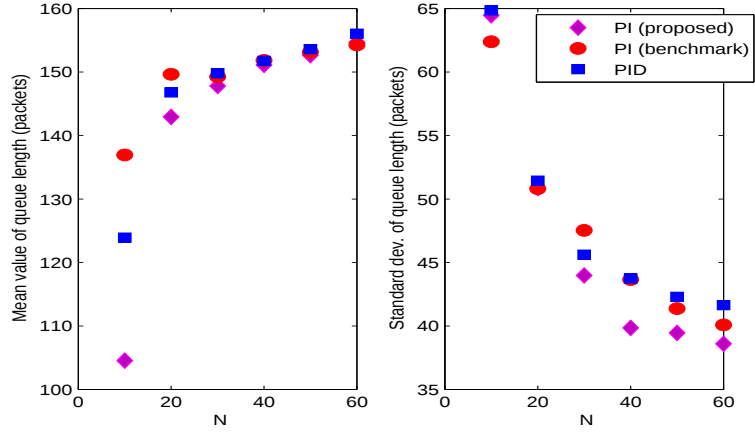


Figure 4.6: Mean and standard deviation of queue length under load variations

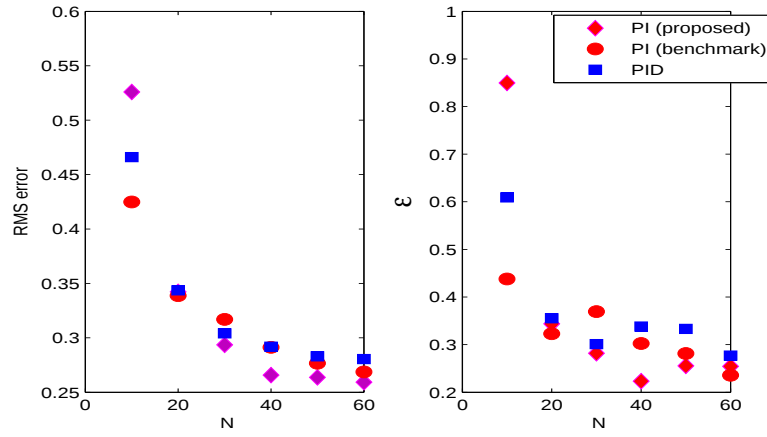


Figure 4.7: Errors under load variations

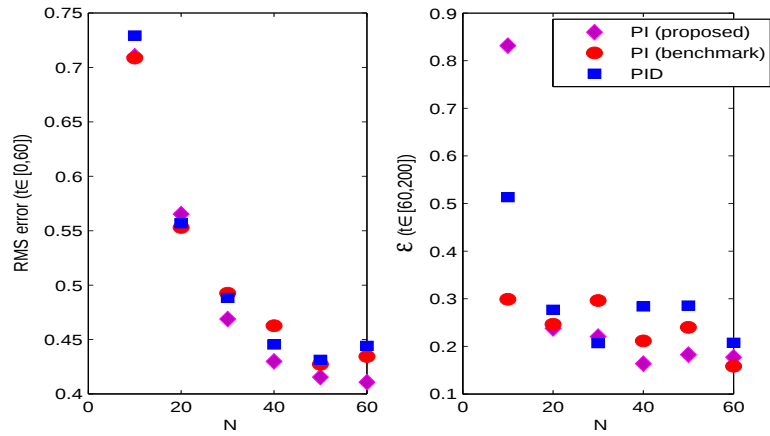


Figure 4.8: Errors under load variations

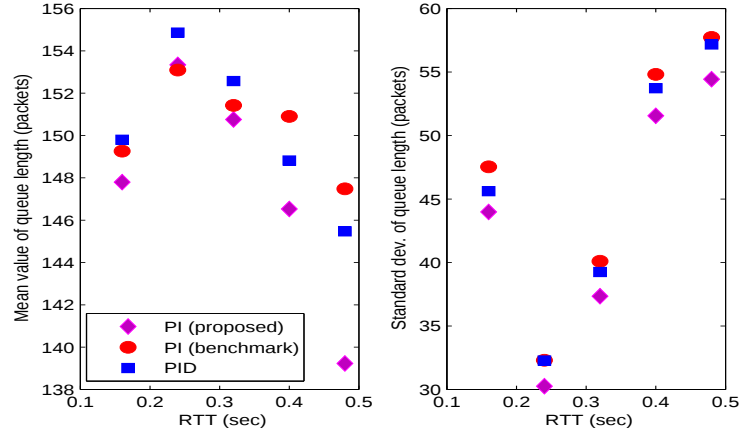


Figure 4.9: Mean and standard deviation of queue length under RTT variations

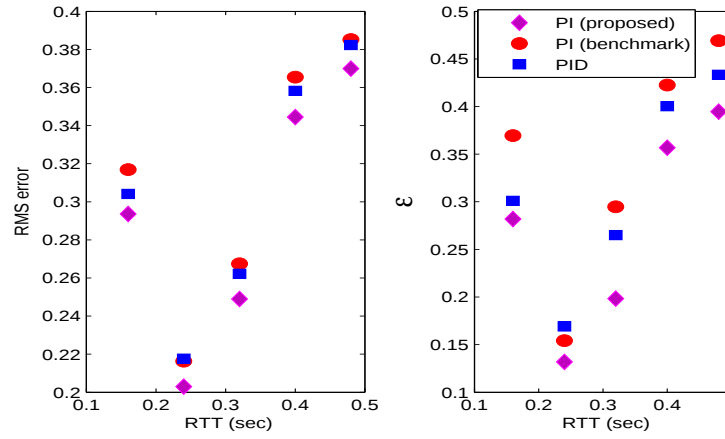


Figure 4.10: Errors under RTT variations

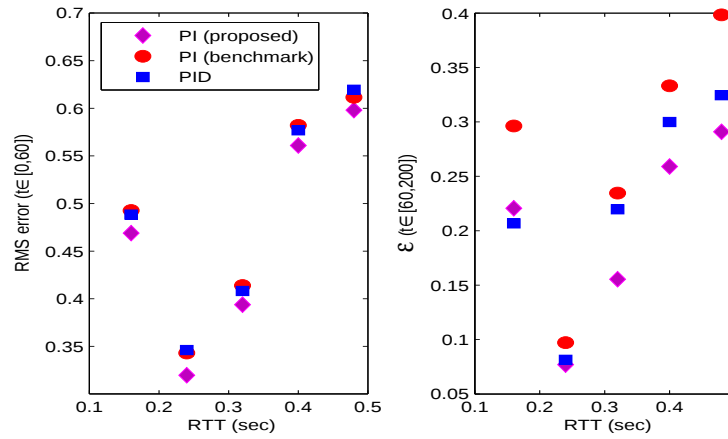


Figure 4.11: Errors under RTT variations

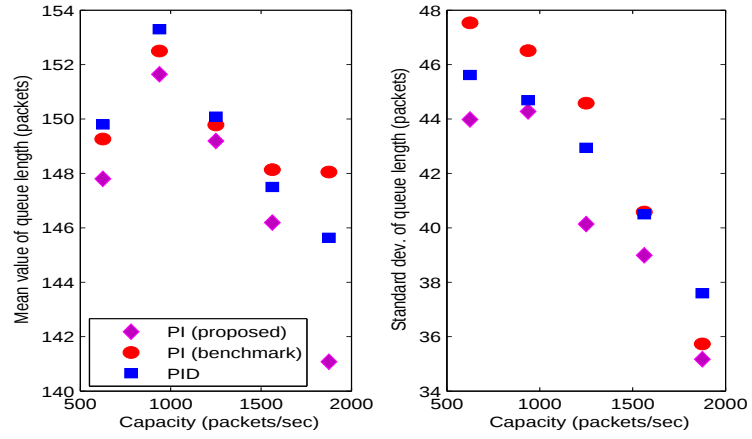


Figure 4.12: Mean and standard deviation of queue length under capacity variations

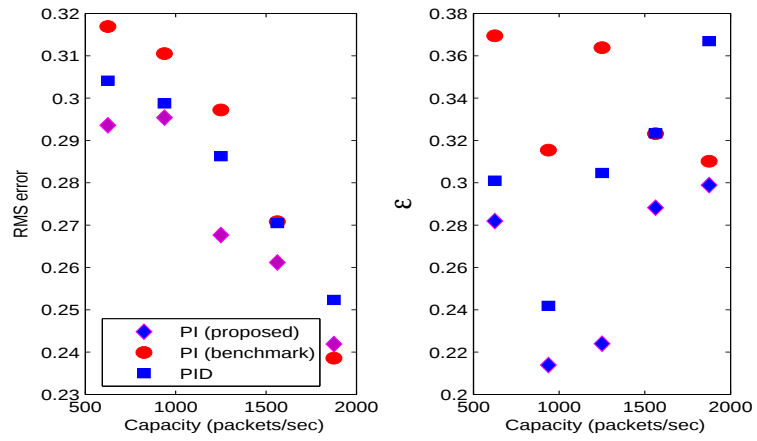


Figure 4.13: Errors under capacity variations

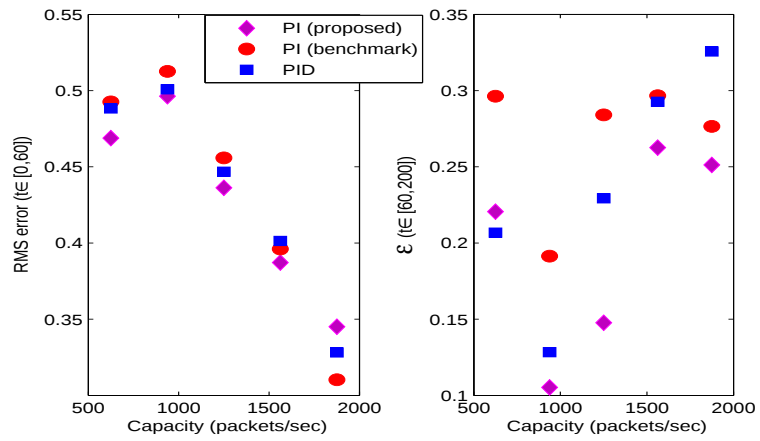


Figure 4.14: Errors under capacity variations

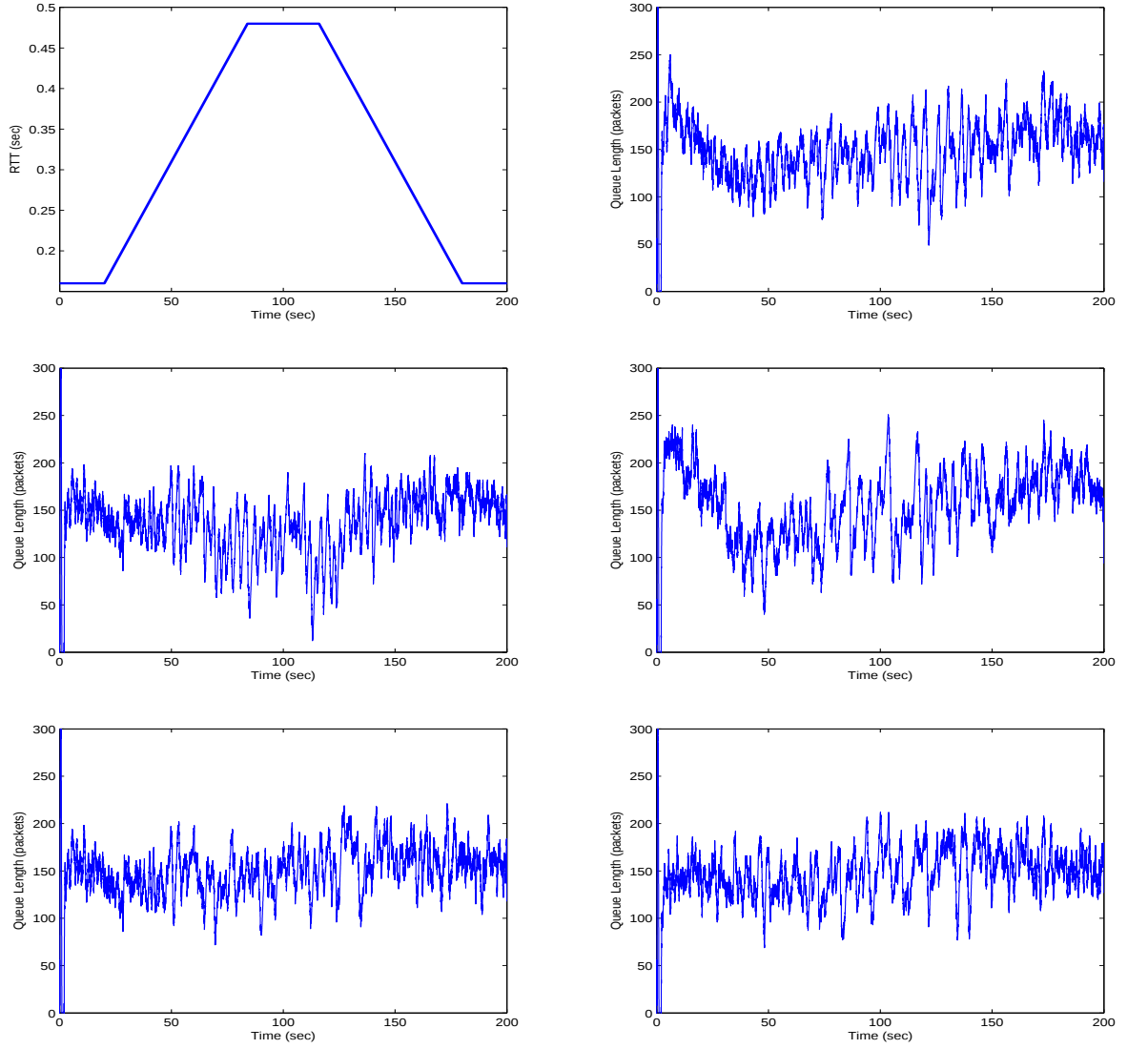


Figure 4.15: ns-2 simulations: (a) RTT_1 (b) single controller, K_{pi0} (c) single controller, K_{pi1} (d) single controller, K_{pi2} (e) two switching controllers, K_{pi1} and K_{pi2} , (f) N=16 switching controllers

Table 4.5: The analysis of simulation results for RTT_1

Controller	Mean	Std	RMS error	$\mathcal{E}$
K_{pi0}	153.34	39.64	0.27	0.33
K_{pi1}	142.60	39.14	0.27	0.27
K_{pi2}	159.91	45.96	0.31	0.49
K_{pi1} - K_{pi2} Switching	152.52	35.07	0.23	0.22
16 Switching Controllers	148.77	34.63	0.23	0.23

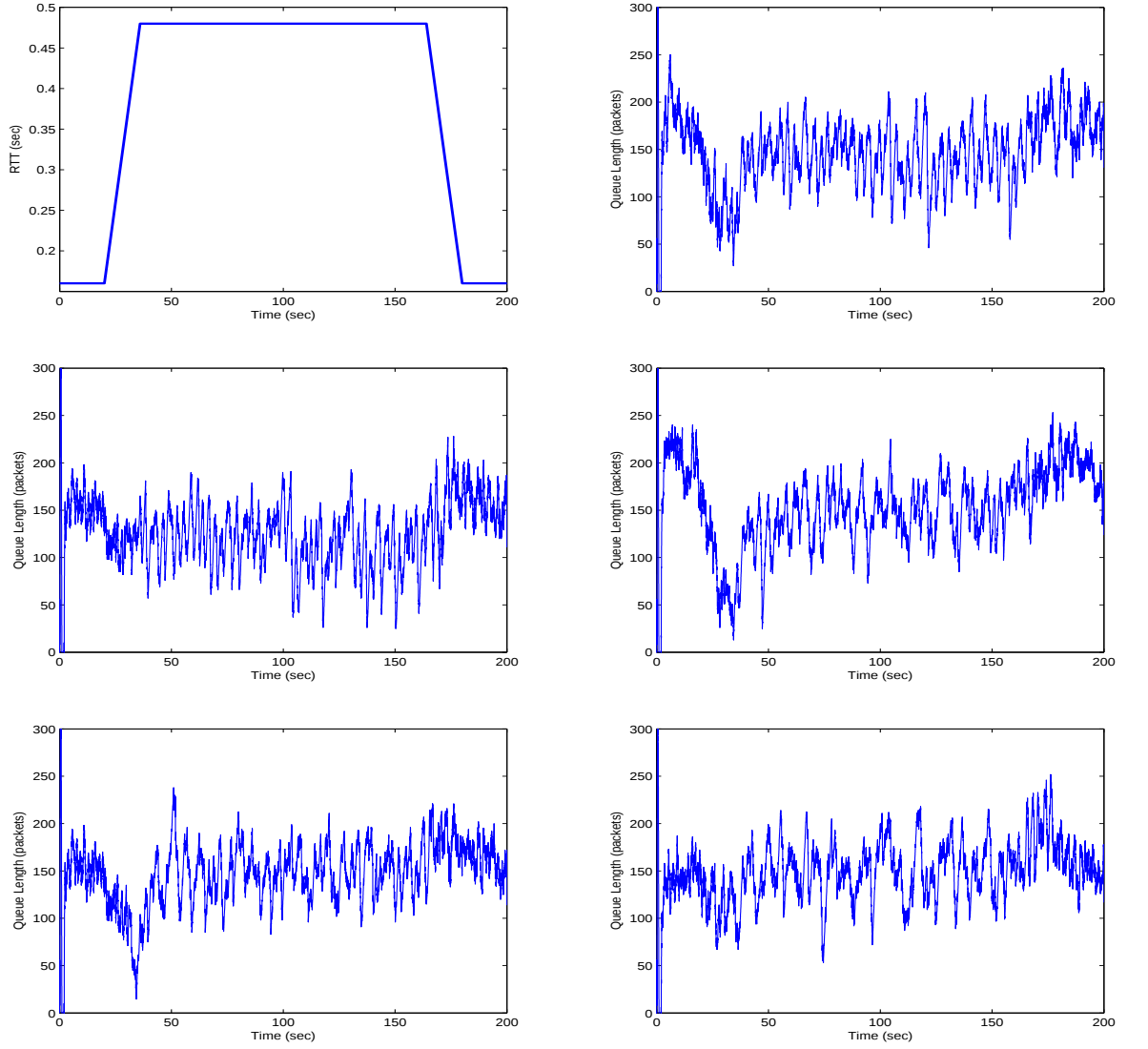


Figure 4.16: ns-2 simulations: (a) RTT_2 (b) single controller, K_{pi0} (c) single controller, K_{pi1} (d) single controller, K_{pi2} (e) two switching controllers, K_{pi1} and K_{pi2} , (f) N=16 switching controllers

Table 4.6: The analysis of simulation results for RTT_2

Controller	Mean	Std	RMS error	$\mathcal{E}$
K_{pi0}	152.28	46.03	0.31	0.42
K_{pi1}	134.68	43.60	0.31	0.42
K_{pi2}	159.47	51.80	0.35	0.52
K_{pi1} - K_{pi2} Switching	150.67	41.69	0.28	0.32
16 Switching Controllers	150.57	40.31	0.27	0.32

Chapter 5

Conclusions

In this thesis we proposed new PI, PD and PID controller tuning methods for integrating systems with time delay using the analysis of allowable PID gains done in [7, 17]. The PI and PD controller expressions (2.19), (2.14) appearing in Chapter 2 are valid for all integrating systems whose transfer function is in the form (2.1) where $g(s)$ is an arbitrary stable transfer function containing a time delay and satisfying $g(0) = 0$.

We have illustrated this method on the AQM problem for a single bottleneck network of TCP flows. Performance of the proposed designs are compared with the benchmark design of [8, 9] which is currently implemented in ns-2. Simulations show that steady state queue variations are lower in the new PI design, and the transient response performance is better. On the other hand, designed PID controllers also perform better than benchmark PI scheme but their advantages over proposed PI controller is only for some error metrics. Robustness of the designs with respect to variation in R , N and c are also analyzed and we have seen that in almost all cases the new PI controller performs better than the benchmark PI controller.

We also proposed switching PI controllers for time varying time delayed single bottleneck network. Simulations show that switching between two controllers gives better results compared to a single controller case.

Theoretical proof of performance improvement by using switched controllers in such a complicated nonlinear system (packet level simulation setting) is not easy to obtain. In fact, even for simplified flow model it can be shown that arbitrary switching between two controllers may even destabilize the feedback system. Therefore our simulation results illustrate the value of mid-point switched PI controllers for this AQM problem.

There are several other dynamic AQM schemes using control theory techniques such as H_∞ and sliding mode control. However implementation of these controllers are rather complicated and in this thesis PID controllers are discussed in order to provide simple controllers with satisfactory performances. As a future work, the performance of PID controllers can be compared with the mentioned complicated controllers in the scope of the AQM problem. Furthermore, it is possible to test the proposed controller schemes for multi bottleneck networks and for networks with unresponsive flows.

APPENDIX A

ns-2 C++ function adapted for PID controller

```
double PIDQueue::calculate_p() {  
    //double now = Scheduler::instance().clock();  
    double p;  
    int qlen = qib_ ? q_->byteLength() : q_->length();  
  
    if (qib_) {  
  
        p=edp_.a0*(qlen*1.0/edp_.mean_pktsize-edp_.qref)+  
          edp_.a1*(edv_.qold*1.0/edp_.mean_pktsize-edp_.qref)+  
          edp_.a2*(edv_.qelder*1.0/edp_.mean_pktsize-edp_.qref)-  
          edp_.b1*edv_.v_prob-  
          edp_.b2*edv_.pelder;  
    }  
    else {  
        p=edp_.a0*(qlen-edp_.qref)+  
          edp_.a1*(edv_.qold-edp_.qref)+  
          edp_.a2*(edv_.qelder-edp_.qref)-
```

```

        edp_.b1*edv_.v_prob-
        edp_.b2*edv_.pelder;
    }

    if (p < 0) p = 0;
    if (p > 1) p = 1;

    edv_.pelder=edv_.v_prob;
    edv_.v_prob = p;

    edv_.qelder=edv_.qold;
    edv_.qold = qlen;

    CalcTimer.resched(1.0/edp_.w);
    return p;
}

```

APPENDIX B

ns-2 tcl code (PI controller case)

```
#Create a simulator object
set ns [new Simulator]

#set number of sources
set M 30

#making routers
set n(1) [$ns node]
set n(2) [$ns node]

#making sources and destinations
for {set i 1} {$i <= $M} {incr i}{
    set s($i) [$ns node]
    set r($i) [$ns node]
}

#creating the network link

$ns duplex-link $n(1) $n(2) 15Mb 20ms PI
```

```

#creating the edge links
for {set i 1} {$i <= $M} {incr i} {
    $ns duplex-link $s($i) $n(1) 10Mb 40ms DropTail
    $ns duplex-link $n(2) $r($i) 10Mb 40ms DropTail
}

#Setup TCP connections
for {set i 1} {$i <= $M} {incr i} {
    set tcp($i) [new Agent/TCP/Reno]
    set sink($i) [new Agent/TCPSink]
    $ns attach-agent $s($i) $tcp($i)
    $ns attach-agent $r($i) $sink($i)
    $ns connect $tcp($i) $sink($i)

    $tcp($i) set ecn_ true
    $tcp($i) set fid_ $i
    $tcp($i) set window_ 300
}

#setup FTP applications
for {set i 1} {$i <= $M} {incr i} {
    set ftp($i) [new Application/FTP]
    $ftp($i) attach-agent $tcp($i)
    $ftp($i) set type_ FTP
}

#start FTP applications
for {set i 1} {$i <= $M} {incr i} {
    $ns at 0 "$ftp($i) start"
    $ns at 200 "$ftp($i) stop"
}

# Tracing a queue
set redq [[$ns link $n(1) $n(2)] queue]
$redq set setbit_ true

```

```

$redq set limit_ 300
$redq set mean_pktsize_ 1000
$redq set qref_ 150
$redq set w_ 4
$redq set a_ 0.000045351
$redq set b_ 0.000042821

set tchan_ [open PI_new_09_09.txt w]
$redq trace curq_ $redq
attach $tchan_

# Define 'finish' procedure (include post-simulation processes)
proc finish {} {
    global tchan_
    set awkCode {
        {
            if ($1 == "Q" && NF>2) {
                print $2, $3 >> "temp.q";
                set end $2
            }
            else if ($1 == "a" && NF>2)
                print $2, $3 >> "temp.a";
        }
    }
    set f [open temp.queue w]
    puts $f "TitleText: PI_new_09_09"
    puts $f "Device: Postscript"

    if { [info exists tchan_] } {
        close $tchan_
    }
}

```

```

exec rm -f temp.q temp.a

exec touch temp.a temp.q


exec awk $awkCode PI_new_09_09.txt


puts $f \"queue
exec cat temp.q >@ $f
    #puts $f \"ave_queue
#exec cat temp.a >@ $f
close $f
exec xgraph -bb -tk -x time -y queue temp.queue &
exit 0
}

#Call the finish procedure after 200 seconds of simulation time

$ns at 201 "finish"

#Run the simulation
$ns run

```

Bibliography

- [1] K. J. Aström, T. Hagglund, *PID Controllers: Theory, Design, and Tuning*, 2nd Ed., Instrument Society of America, Research Triangle Park, NC, 1995.
- [2] S. Athuraliya, S. H. Low, V. H. Li, Q. Yin, “REM: Active Queue Management,” *IEEE Network*, vol. 15, no. 3, May-June 2001, pp. 48-53.
- [3] M. di Bernardo, F. Garofalo, and S. Manfredi, “Performance of Robust AQM controllers in multibottleneck scenarios,” *Proc. of the 44th IEEE Conference on Decision and Control, and the European Control Conference 2005* Seville, Spain, December 2005, pp. 6756–6761.
- [4] S. Floyd, V. Jacobson, “Random early detection gateways for congestion avoidance,” *IEEE/ACM Transactions on Networking*, vol. 1, no. 4, Aug. 1993, pp. 397-413.
- [5] S. Floyd, K. Fall, “Promoting the use of end-to-end congestion control in the Internet,” *IEEE/ACM Transactions on Networking*, vol. 7 (1999), pp. 458–472.
- [6] G. C. Goodwin, S. F. Graebe, and, M. E. Salgado *Control System Design*, Prentice Hall, 2001.
- [7] A. N. Gündes, H. Özbay, A. B. Özgüler, “PID Controller Synthesis for a Class of Unstable MIMO plants with I/O Delays,” *Automatica*, vol. 43 (2007), pp. 135–142.

- [8] C. Hollot, V. Misra, D. Towsley, W. Gong, "On designing improved controllers for AQM routers supporting TCP flows," in *IEEE INFOCOM*, Alaska, April 2001, pp. 1726-1734.
- [9] C. Hollot, V. Misra, D. Towsley, W. Gong, "Analysis and design of controllers for AQM routers supporting TCP flows," *IEEE Transactions on Automatic Control*, vol. 47, no. 6, June 2002, pp. 945-959.
- [10] R. Isermann, *Digital Control Systems Volume 1: Fundamentals, Deterministic Control*, 2nd revised Ed., Springer-Verlag, 1989.
- [11] F. Kelly, "Mathematical modeling of the Internet," *Mathematics Unlimited-2001 and Beyond*, B. Engquist and W. Schmidt Eds., Berlin, Germany, Springer-Verlag, 2001.
- [12] S. Kunniyur, R. Srikant, "Analysis and design of an adaptive virtual queue (AVQ) algorithm for active queue management," in *Proc. ACM SIGCOMM*, San Diego, CA, Aug. 2001, pp. 123-134.
- [13] Y. Lee, J. Lee, S. Park, "PID controller tuning for integrating and unstable processes with time delay," *Chemical Engineering Science*, vol. 55, 2000, pp. 3481-3493.
- [14] V. Misra, W. Gong, D. Towsley, "Fluid-based analysis of a network of AQM routers supporting TCP flows with an application to RED," in *Proc. ACM SIGCOMM*, Stockholm, Sweden, Sep. 2000, pp. 151-160.
- [15] ns-2 Network Simulator, version: 2.27. [Online]. Available: <http://www.isi.edu/nsnam/ns/>.
- [16] K. Ogata, *Modern Control Engineering*, 3rd Ed., Prentice Hall, 1997.
- [17] H. Özbay, A. N. Gündes, "Resilient PI and PD Controllers for a Class of Unstable MIMO plants with I/O Delays," in *CDROM Proceedings of 6th IFAC Workshop on Time Delay Systems*, L'Aquila, Italy, July 2006.

- [18] P-F. Quet, H. Özbay, “On the Design of AQM Supporting TCP Flows Using Robust Control Theory ,” *IEEE Transactions on Automatic Control*, vol. 49 (2004), no. 6, pp. 1031-1036.
- [19] E-C. Park, H. Lim, K-J. Park, C-H. Choi, “Analysis and design of the virtual rate control algorithm for stabilizing queues in TCP networks,” *Computer Networks*, vol. 44 (2004), pp. 17-41.
- [20] E. Poulin, A. Pomerlau, “PI Settings for Integrating Processes Based on Ultimate Cycle Information” *IEEE Transactions on Control Systems Technology*, vol. 7, 1999, pp. 509-511.
- [21] K. Ramakrishnan, S.Floyd, “A Proposal to add Explicit Congestion Notification (ECN) to IP,” RFC 2481, January 1999.
- [22] S. Ryu, C. Rump, C. Qiao, “A predictive and robust active queue management for Internet congestion control,” in *Proc. of the 8th IEEE Symposium on Computers and Communications*, Kemer, Antalya, Turkey, June-July 2003, pp. 991–998.
- [23] S. Ryu, C. Rump, C. Qiao, “Advances in Internet Congestion Control,” in *IEEE Communications Surveys and Tutorials*, vol. 5, no. 1, Third Quarter 2003, pp. 28-38.
- [24] G.J. Silva, A. Datta, S. P. Bhattacharyya, *PID Controllers for Time-Delay Systems*, Birkhäuser, Boston, 2005.
- [25] P. Yan, Y. Gao, H. Özbay, “A Variable Structure Control Approach to Active Queue Management for TCP with ECN,” *IEEE Transactions on Control Systems Technology*, vol. 13, no. 2, March 2005, pp. 203-215.
- [26] P. Yan and H. Özbay, “Robust Controller Design for AQM and H_∞ Performance Analysis,” in *Advances in Communication Control Networks*, S. Tarbouriech, C. Abdallah, J. Chiasson Eds., Springer-Verlag LNCIS, Vol. 308, 2004, pp. 49-64.