A New PI and PID Control Design Method for Integrating Systems with Time Delays: Applications to AQM of TCP Flows

DENİZ ÜSTEBAY
Dept. of Electrical & E. Eng.
Bilkent University
06800 Ankara
TURKEY
deniz@ee.bilkent.edu.tr

HİTAY ÖZBAY
Dept. of Electrical & E. Eng.
Bilkent University
06800 Ankara
TURKEY
hitay@bilkent.edu.tr

NAZLI GÜNDEŞ
Dept. of Electrical & C. Eng.
Univ. of California, Davis
95616 CA
USA
angundes@ucdavis.edu

Abstract: - In computer networks Active Queue Management (AQM) is used for preventing buffer overfull in the routers and under-utilization of the available link capacity. Many different types of static and dynamic AQM schemes have been proposed in the past. Dynamic approaches have been analyzed from control theory point and various designs have been introduced. In this paper, by using the results of a recent study we propose a new method for tuning the parameters of PI and PID controllers for integrating processes with time delays, and apply this technique to AQM designs for TCP flows. Performance of this new scheme is compared to the existing "benchmark'" PI design implemented in ns-2.

Key-Words: - Active Queue Management, PID Controllers, Time Delay, Integrating Processes

1 Introduction

One of the most important problems in computer networks is congestion. When some of the links are congested, buffers at the routers are full, which means incoming packets are lost and re-transmission occurs. This 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 Active Queue Management (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. In early AQM methods, such as RED [4] and REM [2], packet marking probability (control) is a static function of the queue (or average queue). First attempts to design a dynamic controller appear in [7, 8], where the authors use the fluid model of the TCP flows developed in [11] to design a PI (Proportional plus Integral) controller. This AQM scheme is currently implemented in ns-2, [12]. It will be taken as the ``benchmark" design for comparisons with the alternative AQM controller tuning methods.

In this paper we propose a new PI and PID control method for integrating processes with time delays (i.e. the plant contains an integrator, a time delay, and possibly other stable terms), and apply it to AQM problem. The method is based on the study of allowable PI (Proportional+Integral) and PD (Proportional+Derivative) controller parameters appearing in [6, 13]. For a given set of nominal network parameters (RTT, number of TCP flows, link capacity) we try to find the largest allowable intervals for the PI and PD controller parameters, and select the optimal gains as the center of these intervals (in this case feedback system is robustly stable with respect to largest possible perturbations in the controller parameters). Controllers with perturbed gains can be seen as "optimal" controllers for networks with perturbed nominal parameters. So, the controllers proposed here are expected to work for a wide range of network parameters; this is illustrated via ns-simulations. Details of the controller parameter tuning and its application to the AQM problem are given in Section 2. Comparisons with the benchmark PI design are done with ns-simulations; the results can be found in Section 3. Concluding remarks are made in Section 4.

2 PI/PID Control for AQM

PID (Proportional+Integral+Derivative) controllers are widely used in control applications, [1]. One of the most attractive features of these types of controllers is their simplicity: in the most general form they are second order systems

$$K_{pid}(s) = K_p + \frac{K_i}{s} + K_d s \qquad (1)$$

and can be implemented digitally by discretization. The main issue in PID control design is the "optimal" tuning of the gains, K_p, K_i, K_d. We will use the approach proposed in [6,13] to address this issue specifically for the AQM problem.

2.1 Fluid Model of TCP Flows

For AQM design a PI tuning method is proposed in [6,7] using a dynamical model of the TCP. This model is introduced in [11], and has been tested and used by many researchers, see e.g. [3, 10, 14 15, 18], and references therein. By making fluid approximations the model is expressed in continuous time domain as follows, [7]

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

$$\dot{q}(t) = N(t)\frac{W(t)}{R(t)} - c(t) \quad \text{when } q(t) > 0 \quad (3)$$

where W is the TCP window size, q is the queue length, N is the number of TCP flows passing through the link, c is the link capacity (outgoing flow rate), $R(t) = T_o + \frac{q(t)}{c(t)}$ is the RTT (total delay in the feedback path), T_o is the propagation delay, and p is the probability of a packet being marked. In this system p can be seen as the control input generated by a feedback from q, i.e. $p = K(q)$ where K is the feedback control operator. First AQM schemes developed, e.g. RED, use static maps from q to p. Dynamic controllers are developed recently using control theoretic design methods. 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, linearization is done around an equilibrium. Let $W(t)=W_o+\delta_W(t), q(t)=q_o+\delta_q(t), c(t)=c_o+\delta_c(t)$, $N(t)=N_o+\delta_N(t), p(t)=p_o+\delta_p(t)$ and $R(t)=R_o+\delta_R(t)$, with $R_o = T_o + \frac{q_o}{c_o}$, where $q_o, W_o, c_o, N_o, p_o, R_o$ are the nominal values determined from equilibrium conditions. Then, a transfer function, $G_{pq}(s)$, from the control input δ_p to the output to be regulated δ_q can be obtained using small signal analysis, see e.g. [3, 7, 8, 15],

$$G_{pq}(s) = e^{-R_o s}\frac{R_o c_o K}{R_o s + K^{-1}}\frac{1}{R_o s+1}, \quad K = \frac{R_o c_o}{2N_o}. \quad (4)$$

Now our goal is to design a PID controller (1) for the "plant" (4). Such dynamical controllers can be implemented easily as shown in [7, 15].

2.2 Tuning of PI and PD Control Parameters

Allowable ranges of PI and PD controller gains has been investigated in [6, 13]. Here we use these results to derive a controller for a plant $G_{pq}(s)$. First, for the PI controller design we assume that $K \gg 1$. Then,

$$G_{pq}(s) \approx \frac{c_o K}{s} f(s) \quad \text{where } f(s) := \frac{e^{-R_o s}}{(R_o s+1)} \quad (5)$$

We define $g(s) := \frac{1}{R_o s+1}$ and let $G(s) = \frac{c_o K}{s} g(s)$ 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 our case, $Y(s) = s/(as+1)$, for any $a > 0$, and since $g(0) = 1$, we have $X(0) = c_o K$. Then a PI controller is in the form

$$K_{pi}(s) = \alpha X(0)^{-1}(1+\frac{\gamma}{s}) \quad (6)$$

for $\alpha \in R$ and $\gamma \in R$. The bound on the integral action gain γ can be defined by $0 < \gamma < \gamma_{max}$, where

$$\gamma_{max}^{-1} = \left\| \frac{\frac{\alpha}{s} f(1+\frac{\alpha}{s}f)^{-1} - 1}{s} \right\|_\infty$$

with α satisfying

$$0 < \alpha < \left\| \frac{f(s)-1}{s} \right\|_\infty^{-1}.$$

Let us define

$$\theta := \left\| \frac{f(s)-1}{s} \right\|_\infty.$$

Then the lower bound for γ_{max}^{-1} can be found as

$$\gamma_* = \alpha(1-\alpha\theta) \leq \gamma_{max}. \quad (7)$$

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

Here 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

$$K_{pi}(s) = \frac{1}{2\theta}\frac{1}{c_o K}(1+\frac{1}{8\theta s}). \qquad (8)$$

The computation of (8) above is rather simplified thanks to conservatism introduced by (7). In fact, γ_{max}^{-1} is a function of α :

$$\gamma_{max}^{-1} = \sup_{\omega}\left|\frac{1}{j\omega + \alpha f(j\omega)}\right| =: Q(\alpha). \qquad (9)$$

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 H_∞ norm computation and N being the α grid points. Using (7) we skip the H_∞ norm computation for every fixed α. The conservatism introduced by (7) is currently under investigation.

For a PD controller written in the form $\widehat{K}(1+\widehat{K}_d s)$, [13] determined the optimal $\widehat{K}_d$ maximizing the allowable interval for the overall controller gain $\widehat{K}$ within the framework of the approach introduced in [6]. In this paper, we propose to take this optimal derivative action gain, and the center of the maximal allowable interval for $\widehat{K}$, for the plant (5). For this purpose, define $\rho(\omega) := |f(j\omega)|$ and $\phi(\omega) := \angle f(j\omega)$ as the magnitude and phase functions. Then set

$$\eta(\omega) := \frac{\rho(\omega) - \cos(\phi(\omega))}{\omega}.$$

Assume that $\eta(\omega)$ has a maximum, η_o, which is attained at a single frequency ω_o. Then, define

$$\widehat{K}_d := -\frac{\sin(\phi(\omega_o))}{\omega_o \rho(\omega_o)}.$$

With the above definitions, the PD controller is

$$K_{pd}(s) = \frac{1}{2c_o K\eta_o}(1+\widehat{K}_d s). \qquad (10)$$

The above PD controller design technique is valid for all plants in the form (5) 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) this is automatically satisfied.

A PID controller can be obtained from the product of PI and PD controllers. So, we will design a PD controller, then design a PI controller for the PD controlled open loop plant $G_{pq}(s)K_{pd}(s)$. For this purpose we define

$$G_{pq}(s)K_{pd}(s) \approx \frac{c_o K}{s} g(s) \frac{e^{-R_o s}}{2c_o K\eta_o}(1+\widehat{K}_d s) \qquad (11)$$

$$=: \frac{1}{2\eta_o}\frac{e^{-hs} g_o(s)}{s}, \quad g_o(s) := g(s)(1+\widehat{K}_d s)$$

$$\theta_o := \left\|\frac{g_o(s)-1}{s}\right\|_\infty \text{ and } K_{pi}(s) = \frac{1}{2\theta_o} 2\eta_o (1+\frac{1}{8\theta_o s}).$$

So, the PID controller is

$$K_{pid}(s) = \frac{(1+\widehat{K}_d s)}{16K_o\theta_o^2}\frac{(1+8\theta_o s)}{s}. \qquad (12)$$

Note that for (4) we can also design a PI controller and then the PI controlled system can be considered as an open loop plant with an integrator so that we can apply PD control. This will be an alternative method of obtaining a PID controller. For the plant (4) the transfer function of the PI controlled system can be written as

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

$$=: \frac{KR_o}{16\theta^2}\frac{g_o(s)}{s}.$$

For designing a PD controller, first define $\rho_v(\omega) := |g_v(j\omega)|$ and $\phi_v(\omega) := \angle g_v(j\omega)$ as the new magnitude and phase functions. Then set

$$\eta_v(\omega) := \left|\frac{\rho_v(\omega) - \cos(\phi_v(\omega))}{\omega}\right|.$$

Assume that $\eta_v(\omega)$ has a maximum, $\eta_{v,o}$, which is attained at a single frequency $\omega_{v,o}$. With this assumption define

$$\widehat{K}_{dv} := -\frac{\sin(\phi_v(\omega_{v,o}))}{\omega_{v,o}\rho_v(\omega_{v,o})}.$$

Then the PD controller is

$$K_{v,pd}(s) = \frac{1}{2\eta_{v,o}}\frac{16\theta^2}{KR_o}(1+\widehat{K}_{dv}s).$$

Thus the overall PID controller for the plant (4) is the product of $K_{pi}(s)$ and $K_{pd}(s)$, that is,

$$K_{pid}(s) = \frac{(1+\widehat{K}_{dv}s)}{2\eta_{v,o}c_o R_o K^2} \frac{(1+8\theta s)}{s}. \qquad (6)$$

2.3. Digital implementation of the PI and PID 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. [9]. In [7] 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. For digital implementation the z-transform of the controller transfer function is calculated without approximations 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) with $K_d = 0$, and in z-domain it becomes:

$$K_{pi}(z) = \frac{a - bz^{-1}}{1 - z^{-1}} \qquad (13)$$

where $a = K_p$, $b = K_p - K_i T_s$ and, $T_s = 1/f_s$.

Similarly the PID controller (1) in z-domain can be given as:

$$K_{pid}(z) = \frac{a_0 + a_1 z^{-1} + a_2 z^{-2}}{b_0 + b_1 z^{-1} + b_2 z^{-2}} \qquad (14)$$

where $a_0 = K_p + K_d$, $a1 = -K_p - 2K_d + K_i T_s$, $a_2 = K_d$, $b_0 = 1$, $b_1 = -1$, $b_2 = 0$ and, $T_s = 1/f_s$.

3. Simulation Results

3.1. Simulation results for the nominal network parameters

The performances of the designed PI 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 [17]. 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 1. The buffer sizes of both routers are set to 300 packets and each packet is of size 1000 Bytes.

The nominal values of the system parameters are: $N_o = 30$ TCP sessions, $c_o = 1250$ 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 2.

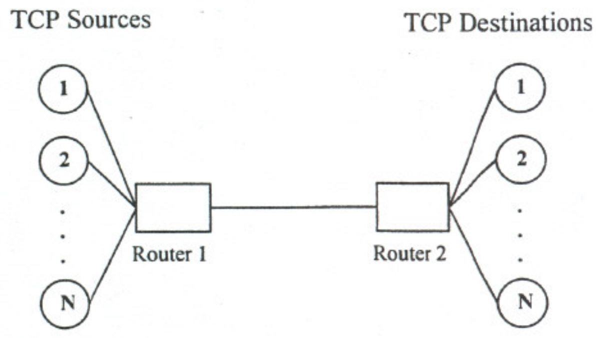

Fig 1. Network Topology

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,

$$\text{RMS error} = \left(\frac{1}{M} \sum_{k=1}^{M} \left(\frac{q(k) - q_d}{q_d} \right)^2 \right)^{\frac{1}{2}}, \qquad (15)$$

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 $\mathrm{Q_d} := [q_d - q_{\text{tol}}, q_d + q_{\text{tol}}]$ be a neighborhood of the desired queue. In our case we have q_d=150 packets, and we choose q_{tol}=30 packets. Define T_0 to be the total length of the time intervals for which $q(t) \notin \mathrm{Q_d}$, and let T_{total} be the length of the total simulation time (in our setting $T_{total} = 200$ seconds). Then the second error metric we are interested in is $\varepsilon = T_0/T_{total}$. This error provides a better look on the variations of the queue length, and its small values bound the delay jitter.

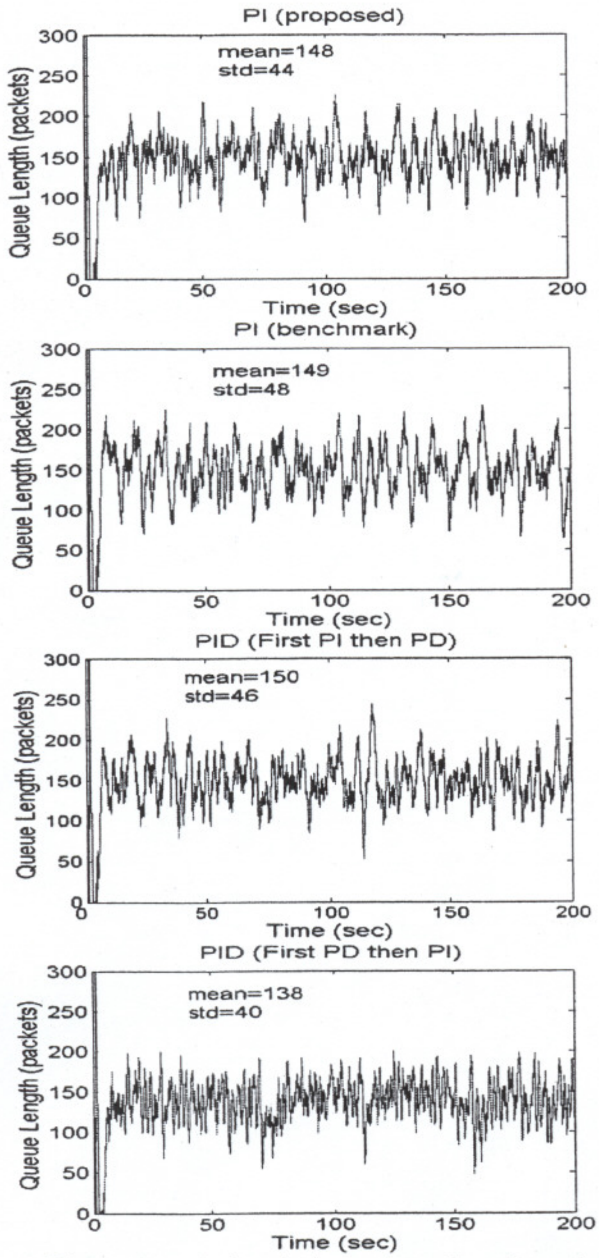

Figure 2: ns-2 simulations: queue length vs time.

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

Table 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 proposed by [13]. Note that all cost functions are normalized with respect to the time interval. The new proposed PI design gives an improvement of 2% to 9% depending on the cost taken; when integrated over time, the benefits of the new design can be significant.

3.2. 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.

Table 1: The analysis of simulation results

Controller	RMS Error	ε
PI (proposed)	0.29	**0.28**
PI (benchmark)	0.32	0.37
PID (first PI then PD)	0.30	0.30
PID (first PD then PI)	**0.27**	0.29

Controller	RMS Error	ε
PI (proposed)	0.47	0.22
PI (benchmark)	0.49	0.30
PID (first PI then PD)	0.49	**0.21**
PID (first PD then PI)	**0.42**	0.24

In Figures 3, 4, 5 we vary the number of TCP connections from 10 to 60. In Figures 6, 7, 8 RTT is varied between 160 msec and 480 msec. In Figures 9, 10, 11 the capacity is changed from 625 packets/second to 1875 packets/second (from 5 Mbps to 15 Mbps). All these figures are given in the Appendix. They 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).

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.

4 Conclusions

In this paper 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 [6,13]. The PI and PD controller expressions (8), (10) appearing in Section 2 are valid for all integrating systems whose transfer function is in the form (5) where $f(s)$ is an arbitrary stable transfer function containing a time delay and satisfying $f(0)=0$.

We have illustrated this method on the Active Queue Management problem for a single bottleneck network of TCP flows. Performance of the proposed designs are compared with the benchmark design of [7,8] implemented in ns-2. Simulations show that steady state queue variations are lower in the new design, and the transient response performance is better. 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.

References:

[1] K. J. Astrom, 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 (2001), no. 3, 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 CDC, and the ECC 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 (1993), no. 4, 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] A. N. Gündeş, H. Özbay, A. B. Özgüler, “PID Controller Synthesis for a Class of Unstable MIMO plants with I/O Delays,” Proc. of 6th IFAC Workshop on Time Delay Systems, L’Aquila, Italy, July 2006.

[7] 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.

[8] 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 (2002), no. 6, pp. 945-959.

[9] R. Isermann, *Digital Control Systems Volume 1: Fundamentals, Deterministic Control* 2nd revised Ed., Springer-Verlag, 1989.

[10] S. Low, F. Paganini, J. Wang, S. Adhalka, J. Doyle, “Dynamics of TCP/RED and a scalable control,” in *Proc. IEEE INFOCOM*, New York, June 2002, pp 239-248.

[11] V. Misra, W. Gong, D. Towsley, “A fluid-based analysis of a network of AQM routers supporting TCP flows with an application to RED,” in *Proc. ACM SIGCOMM*, Stockholm, Sweeden, Sep. 2000, pp. 151-160.

[12] ns-2 Network Simulator, version: 2.27. [Online]. Available: http://www.isi.edu/nsnam/ns/.

[13] H. Özbay, A. N. Gündeş, "Resilient PI and PD Controllers for a Class of Unstable MIMO plants with I/O Delays," Proc. of 6th IFAC Workshop on Time Delay Systems, L'Aquila, Italy, July 2006.

[14] 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.

[15] 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.

[16] 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.

[17] 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 (2005), no. 2, pp. 203-215.

[18] 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.

Acknowledgements: This work is supported by the European Commission (contract no. MIRG-CT-2004-006666) and by TÜBİTAK (grant no. EEEAG-105E156)

5 Appendix

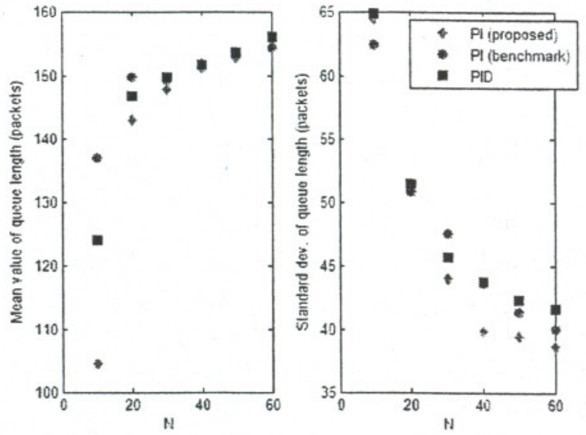

Figure 3: Mean and standard deviation of queue length under load variations

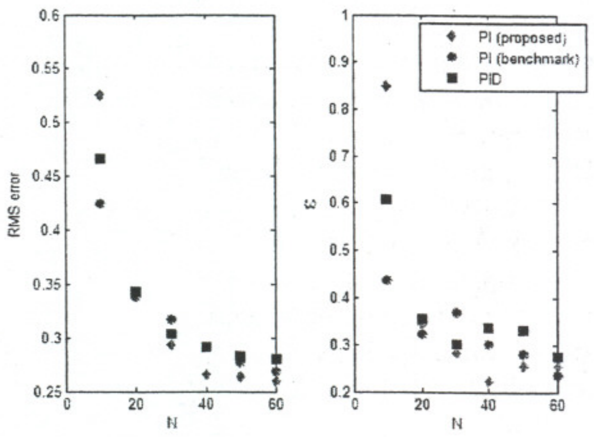

Figure 4: Errors under load variations

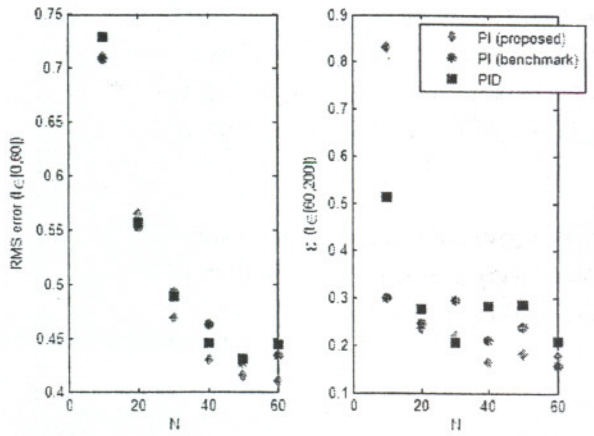

Figure 5: Errors under load variations

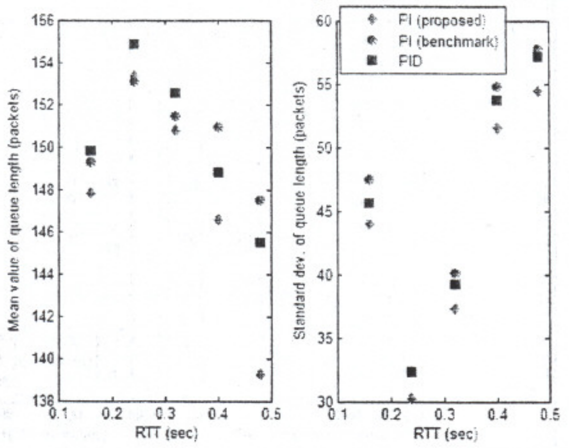

Figure 6: Mean and standard deviation of queue length under RTT variations

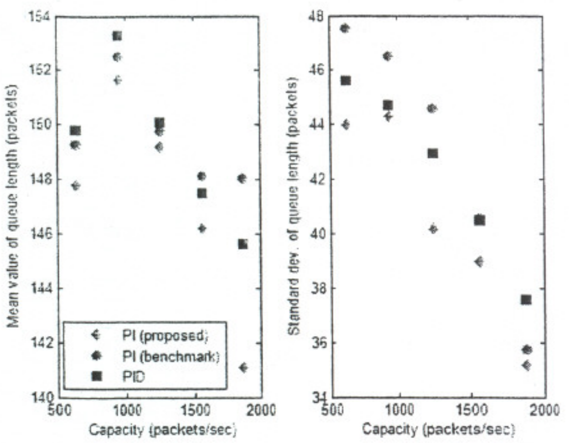

Figure 9: Mean and standard deviation of queue length under capacity variations

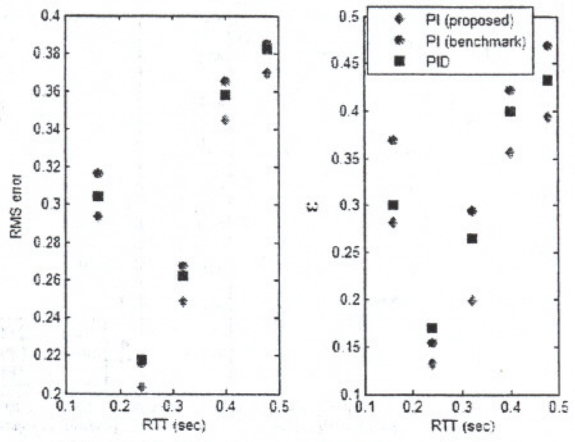

Figure 7: Errors under RTT variations

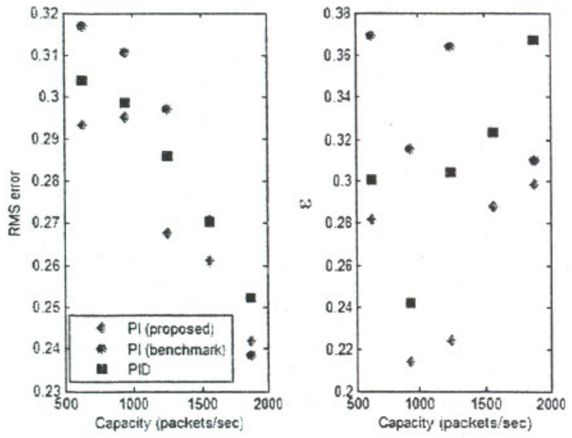

Figure 10: Errors under capacity variations

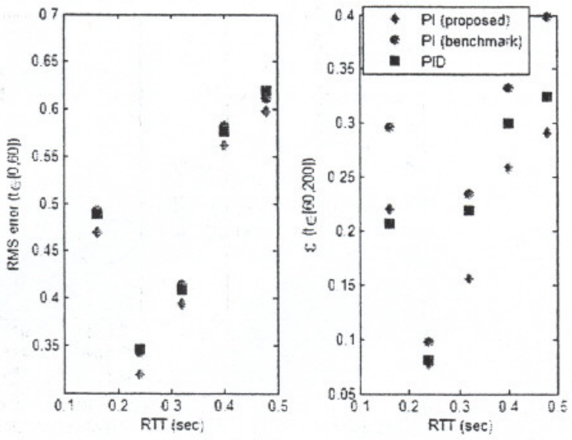

Figure 8: Errors under RTT variations

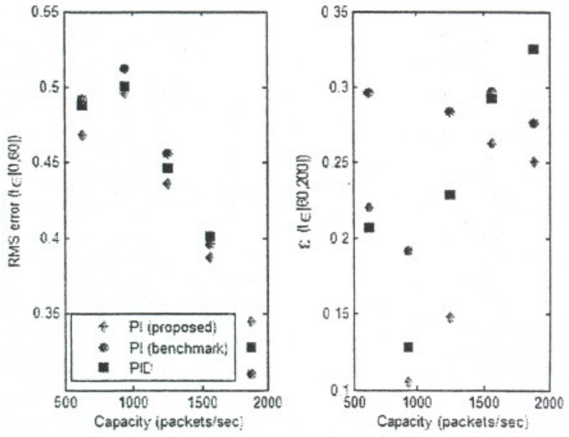

Figure 11: Errors under capacity variations