UKACC International Conference on Control 2012
Cardiff, UK, 3-5 September 2012

How to Reduce Congestion on TCP/AQM Networks with Simple Adaptive PID Controllers

Teresa Alvarez, Anuar Salim
Department of Engineering Science and Automatic Control,
Escuela de Ingenierías Industriales (Sede Doctor Mergelina), University of Valladolid
Universidad de Valladolid, Valladolid, Spain
e-mail: tere@autom.uva.es

Abstract—**Congestion is a problem in real networks. Users do not want to lose information and data should be delivered as fast and reliably as possible. This is really difficult to achieve. Moreover, networks work in changing environments: number of users, type of traffic, delays in transmission, etc. So this paper presents how to design PID controllers that take network changes into account. Non-linear simulations using ns-2 will show the goodness of the approach when compared with classical PID, drop tail and RED.**

Keywords-component; congestion control; PID; adaptive control; robustness; TCP/AQM

I. INTRODUCTION

Internet is very important in our society. Information, images, videos and data are transmitted from one place to another in a matter of seconds. Nevertheless, there are problems in transmission that affect the quality of the process. Congestion is one of the most annoying obstacles ([1], [2]). It is important to detect this and to act as fast as possible to solve it. Thus, techniques to reduce congestion are of great interest. There are two basic approaches ([3], [4]): c*ongestion control,* which is used after the network is overloaded, and c*ongestion avoidance,* which takes action before the problem appears.

Feedback control can help to solve congestion control. The end-to-end transmission control protocol (TCP), and the active queue management (AQM) scheme define the two parts implemented at the routers' transport layer where congestion control is carried out. The AQM objectives ([5]) are: to minimize the occurrences of queue overflow and underflow, to minimize the time required for a data packet to be serviced by the routing queue and to maintain closed-loop performance in spite of changing conditions (robustness).

AQM schemes enhance the performance of TCP, although they do not work perfectly in every traffic situation. Numerous algorithms have been proposed (a good survey can be found in [4]). RED [6] is the basic algorithm used for comparison. It can detect and respond to long-term traffic patterns, but it cannot detect congestion caused by short-term traffic load changes. The most widely used AQM mathematical models were published in [4]. Since then, several control approaches have been applied, such as fuzzy, predictive control or robust control. The automatic control techniques that have received most attention are the PI controller, followed by the PID ([5]).

AQM techniques should be robust and give good results when the network is not operating in a nominal situation, when the delays are changing, or the number of users or the intensity of the packets' flow vary. So the AQM congestion control algorithm should be robust, stable and do better in a changing environment. If PID is chosen as the AQM algorithm, there are many useful references in the literature ([7]-[9]). These works do not consider what happens with the tuning when the traffic conditions change. They explicitly consider delays in the design of the PIDs and this is the situation that arises when working with networks: a system with delay.

Motivated by these issues, this paper presents how to derive PIDs, with guaranteed stability and robustness, which perform well in a network with changing traffic conditions. The work presented in this paper applies the generic method by [7] for finding stable PID controllers for time delay systems to the dynamic TCP/AQM router model. A linear gain scheduling is included in the design (reducing the gain variation) to enhance the robustness and ensure an adequate behaviour in extreme working scenarios. A similar approach was described in [10], but further study has shown that it can be simplified, as shown in this paper. The metrics applied to study the controller performance are: the router queue size, the link utilization and the probability of packet losses.

Non-linear simulations with ns-2 will show the performance of the technique by applying it to a problem of two routers connected in a Dumbbell topology, which represents a single bottleneck scenario. A comparison between the new PID, a classic PID and the RED approach is performed.

This paper is organized as follows. Section II briefly describes the TCP NewReno modeling. Section III describes a fluid model for the system. Next, the gain scheduling PID, classic PID and RED are explained and their methodology is described. Simulation results are shown in section V. Finally, some conclusions are presented.

II. TCP/IP Network Modelling

A. TCP/IP NewReno dynamic model

Now, the TCP/IP NewReno network dynamics is presented. The dynamics of an AQM router are complex due to the number of variables that come into play: packet sources, protocols, etc. Nevertheless, it is possible to obtain a nonlinear model that represents the dynamics of the system ([4]), considering that the protocol used is TCP. The model relates to the average value of the network variables and is described by the following coupled, nonlinear differential equations:

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

where

W: average TCP window size (packets),

$\dot{q}$: average queue length (packets),

R: round-trip time = $q/C+T_p$ (secs),

C: link capacity (packets/sec),

T_p: propagation delay (secs),

N_{TCP}: load factor (number of TCP sessions) and

p: probability of packet mark.

As explained in [4], the first differential equation in (1) describes the TCP window control dynamic, and the second equation models the bottleneck queue length, as an accumulated difference between the packet arrival rate and the link capacity. The queue length and window size are positive, bounded quantities, i.e., $q\in[0,\overline{q}]$ and $W\in[0,\overline{W}]$, where $\overline{q}$ and $\overline{W}$ denote buffer capacity and maximum window size, respectively. In this formulation, the congestion window size $W(t)$ is increased by one every round-trip time if no congestion is detected, and is halved when congestion is detected.

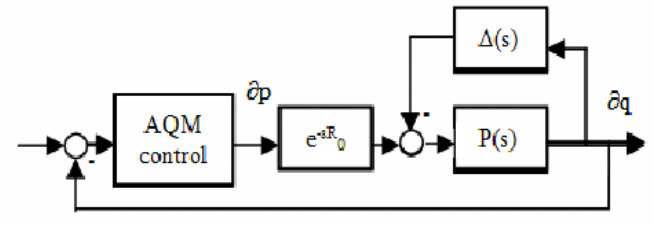

Fig. 1. Block diagram of AQM as a feedback control system

B. Linearized model

Although an AQM router is a non-linear system, in order to analyze certain types of properties and design controllers, a linearized model is used. To linearize (1), it is assumed that the number of active TCP sessions and the link capacity are constant, i.e., $N_{TCP}(t)= N$ and $C(t)=C$.

The dependence of the time delay argument $t-R$ on queue length q is ignored ([4]) and it is assumed to be fixed at $t-R_0$. This is a big assumption and there will be situations where it may not be acceptable. In [11], the authors deduced that this is a good approximation when the round-trip time is dominated by the propagation delay. This occurs when the capacity C is large. R_0 should be chosen such that the above hypothesis can be ensured, so a value that works in the worst case scenario should be advisable. When the model is linearized, the same supposition is made. It cannot be denied that calculations are significantly simplified, but further research in the matter would be advisable. Local linearization of (1) around the operating point results in the following differential equations:

$$\left.\begin{aligned}\partial\dot{W}(t)&=-\frac{N}{R_0^2C}(\partial W(t)-\partial W(t-R_0))\\&\quad-\frac{1}{R_0^2C}(q(t)-\partial q(t-R_0))-\frac{R_0^2C}{2N^2}\partial p(t-R_0)\\ \partial\dot{q}(t)&=\frac{N}{R_0}\partial W(t)-\frac{1}{R_0}\partial q(t)\end{aligned}\right\},\quad (2)$$

where $\partial\dot{W}(t)=W-W_0$, $\partial q=q-q_0$ and $\partial p=p-p_0$, represent the perturbed variables. The operating point for a desired equilibrium queue length q_0 is given by:

$$R_0=\frac{q_0}{C}+T_p,\ W_0=\frac{R_0C}{N_{TCP}}\text{ and }p_0=\frac{2}{W_0^2}. \quad (3)$$

The linearized model (2) can be rewritten by separating the low frequency ('nominal') behaviour $P(s)$ of the window dynamic from the high frequency behaviour $\Delta(s)$ which is considered as parasitic.

$$P(s)=\frac{C^2/(2N_{TCP})}{\left(s+(2N_{TCP})/(R_0^2C)\right)\left(s+1/R_0\right)},$$

$$\Delta(s)=\frac{2N_{TCP}^{\ 2}}{R_0C^3}\left(1-e^{-R_0s}\right) \quad (4)$$

Taking (4) as a starting point, Hollot et al. (2002) give a feedback control description of AQM (Fig. 1). The action implemented by an AQM control law is to mark packets with a discard probability $p(t)$, as a function of the measured queue length $q(t)$. The larger the queue, the greater the discard probability becomes.

III. Design of AQM Controllers

This section introduces the control formulation of an AQM router and how RED, classic PID and the gain scheduling PID can be applied. The latter will be described in depth, as it is the approach that outperforms the other two.

A. AQM as feedback control

Taking (4) as the starting point, [5] gives a feedback control system of AQM (Figure 1). The action of an AQM control law is to mark packets with probability *p*, as a function of the measured queue length *q*. Following (4), the transfer function Δ(s) denotes the high-frequency window dynamics and *P(s)* (plant dynamics) relates how *p* dynamically affects *q*.

B. AQM using RED

Random Early Detection (known as RED) was presented in [6]. A RED gateway calculates the average queue size, using a low-pass filter with an exponential weighted moving average. The average queue size is compared to two thresholds (minimum and maximum). When the average queue size is less than the minimum threshold, no packets are marked. When the average queue size is greater than the maximum threshold, every arriving packet is marked. If marked packets are in fact dropped, or if all source nodes are co-operative, this ensures that the average queue size does not significantly exceed the maximum threshold. When the average queue size is between the minimum and the maximum threshold, each arriving packet is marked with probability *p*, where *p* is a function of the measured queue length *q*.
RED does not work with a proper reference, set-point. As explained in [7], RED introduces a range of reference input values, rather than a proper set-point. We will use it as a classic AQM algorithm for the sake of comparison.

C. AQM using PID control

PID controllers [5] are the most common form of feedback. They can be described by (5). This is the classic formulation.

$$\begin{aligned} u(t) &= K_p\left(e(t)+\frac{1}{T_i}\int_0^t e(\tau)d\tau + T_d\frac{de(t)}{dt}\right) \\ &= K_p e(t) + K_i\int_0^t e(\tau)d\tau + K_d\frac{de(t)}{dt} \end{aligned} \tag{5}$$

D. AQM using gain scheduling PID controller

This subsection presents the approach that has been followed to deal with the AQM congestion control problem in a network working with TCP Newreno (it should be noted that the approach can be applied to other TCP variants, [12]) under varying traffic. There are parameters, such as the link capacity *C*, which do not usually change, but the round trip delay, R_0, and the number of users, N_{TCP}, frequently vary.

The proposed technique is simple, but works well in networks with a wide range of users and varying delays. As network traffic changes constantly, this affects the performance of the system. We tune the PID for a certain set of conditions, but they change. The method proposed here will improve the router's performance no matter how many users or delay there may be in the network.

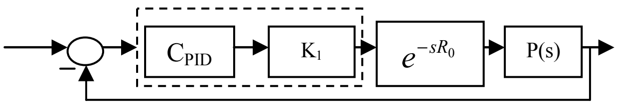

Figure 2: Block diagram of the AQM feedback control system

The PID is tuned following a model-based approach, taking as its starting point the linearized dynamic model of TCP described in the previous section. Figure 2 depicts the block diagram corresponding to the feedback control approach followed in the paper: P(s) is the transfer function obtained in the previous section. The delay e^{-sR_0} is explicitly considered in the design. There are two elements in the controller C(s):

- C_{PID}(s) is a PID controller and
- K_1 is a variable gain; a simple gain scheduling that allows the controller to work satisfactorily in a broad range of situations.

The steps in the design are:

- Choose the worst network scenario in terms of N_{TCP} and R_0: i.e., the biggest delay and the smallest number of users.
- Choose the best network scenario in terms of N_{TCP} and R_0: i.e., the smallest delay and the biggest number of users.
- Design a controller C_{PID}(s) that is stable in these two situations and the scenarios in between.
- Add a simple gain scheduling K_1 to improve the system's response, based on the expected variations of the system's gain.

We have chosen the PID controller (6) as the basic AQM congestion control algorithm, as it is the most common form of feedback control technique. The PID should be tuned to be stable in the above mentioned situations. It is also required to have good responses in terms of settling time and overshoot. Taking this into account, the tuning of the controller is done following the method presented in [7]. It is relevant to note that the system can be of any order and the roots can be real or complex. Using the provided Matlab toolbox, a three dimensional region is obtained. If K_p, K_i and K_d are chosen inside this region, the closed-loop response is stable.

The simple gain scheduling approach that we propose takes into consideration the big gain variation that the AQM dynamic model has:

$$\frac{-C^2}{2N} \tag{7}$$

Moreover, if a fair comparison among all the network configurations is to be done, the open-loop gain of the systems should be the same, so the independent term of the denominator of the transfer function should be included:

$$\frac{2N}{C\cdot R_0^3} \tag{8}$$

This term greatly affects the size of the stable region and the magnitude of the PID's tuning parameters (as will be shown in next section). So we take out this term from P(s) and calculate the stable region for:

$$P_{normalized}(s) = P_n(s) = \frac{1}{s^2 + a \cdot s + 1}, \qquad a = \frac{2N}{R_0^2 C} + \frac{1}{R_0} \tag{9}$$

And:

$$P(s) = \frac{-\frac{C^2}{2N}}{\frac{2N}{C \cdot R_0^3}} P_n(s) = -\frac{C^3 R_0^3}{4N^2} P_n(s) = P_{cte} \cdot P_n(s) \tag{10}$$

So, the gain scheduling term is given by (11).

$$K_1 = -\frac{4N^2}{C^3 R_0^3} \tag{11}$$

It is important to choose the worst and best scenarios properly. A good knowledge of the network is required. Section IV defines the network parameters. The stable regions are depicted and the PID tuned. Then linear and non-linear simulations will show the goodness of the method.

IV. Simulations

This Section presents how the new PID is tuned. A comparison between the proposed approach and a classical PID is shown using a linear simulation. Finally, ns-2 is used to test the new controller and to compare it with a classical PID and the AQM/RED congestion control technique.

A. *Tuning the Controller*

The basic network topology, used as an example to test the controller, is depicted in Figure 3. It is a typical, single bottleneck topology. Although the network capacity C could change, its adjustments are more related with network updates than with everyday traffic conditions. The different scenarios come from changing the number of users ($N=N_{TCP}$) and the round trip time (R_0) (see Table I). For each of the values in Table 1, the link capacity takes two values: 1875 and 940 packets/sec.

TABLE I.

	N	R_0	Results (Fig. 4)
Case 1	100	0.5	Red
Case 2	150	0.4	Green
Case 3	40	0.9	Blue
Case 4	60	0.24	Yellow

The situations deal with a broad range of conditions: few users and big delay, many users and small delay and in-between situations. In [13] and [14], the effect of N and R_0 on the size of the controller's stability region was studied. They concluded (and our experiments also agree) that the smallest stability region is obtained for the smallest number of users (N) and the minimum round trip delay (R_0)

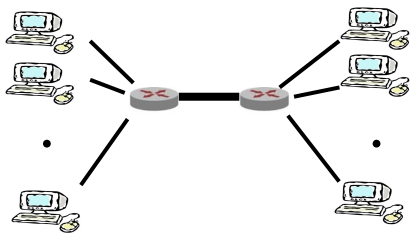

Figure 3: Dumbbell topology

Figure 4 shows the open loop poles of the system, without considering the delay. The lower graph depicts the open loop poles and zeros if a Padé approximation of first order is used for the delay of each of the scenarios.

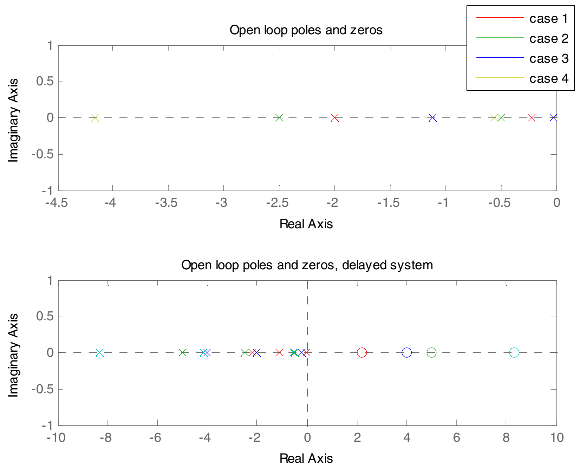

Figure 4: Open loop poles and zeros

The open loop step response for each of the studied scenarios of the original linear system (*P(s)*) and the normalized one (*$P_n(s)$*) are shown in Figure 5 (upper and lower, respectively). Both systems have the same settling time, but the behaviour will be very different once the loop is closed. Case 3 gives the slowest open loop response (settling time: 150 sec.) and case 4 gives the fastest (settling time: 7.53 sec.).

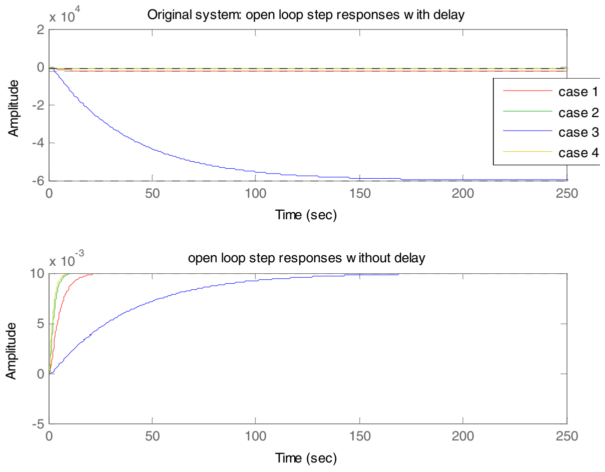

Figure 5: Step responses

TABLE II. PID PARAMETERS

	Normalized	Standard
Kp	1.2	-0.000006
Ki	0.2	-0.0000012
Kd	0.05	-0.00000432

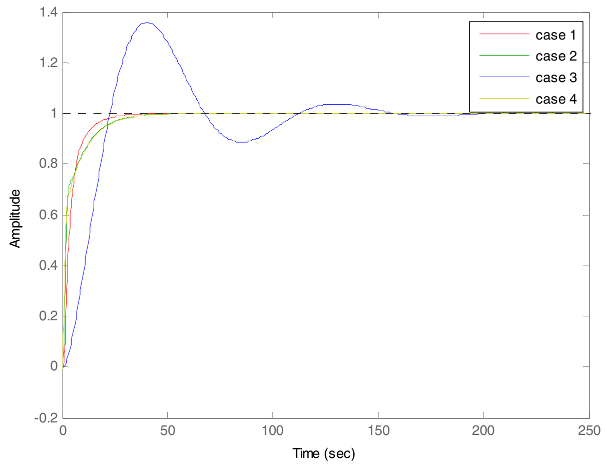

Figure 8: Closed-loop step response with proposed method

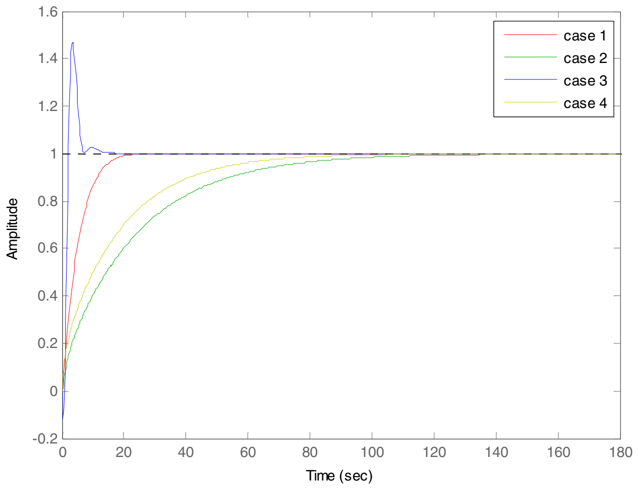

Figure 9: Closed-loop step response with classical approach

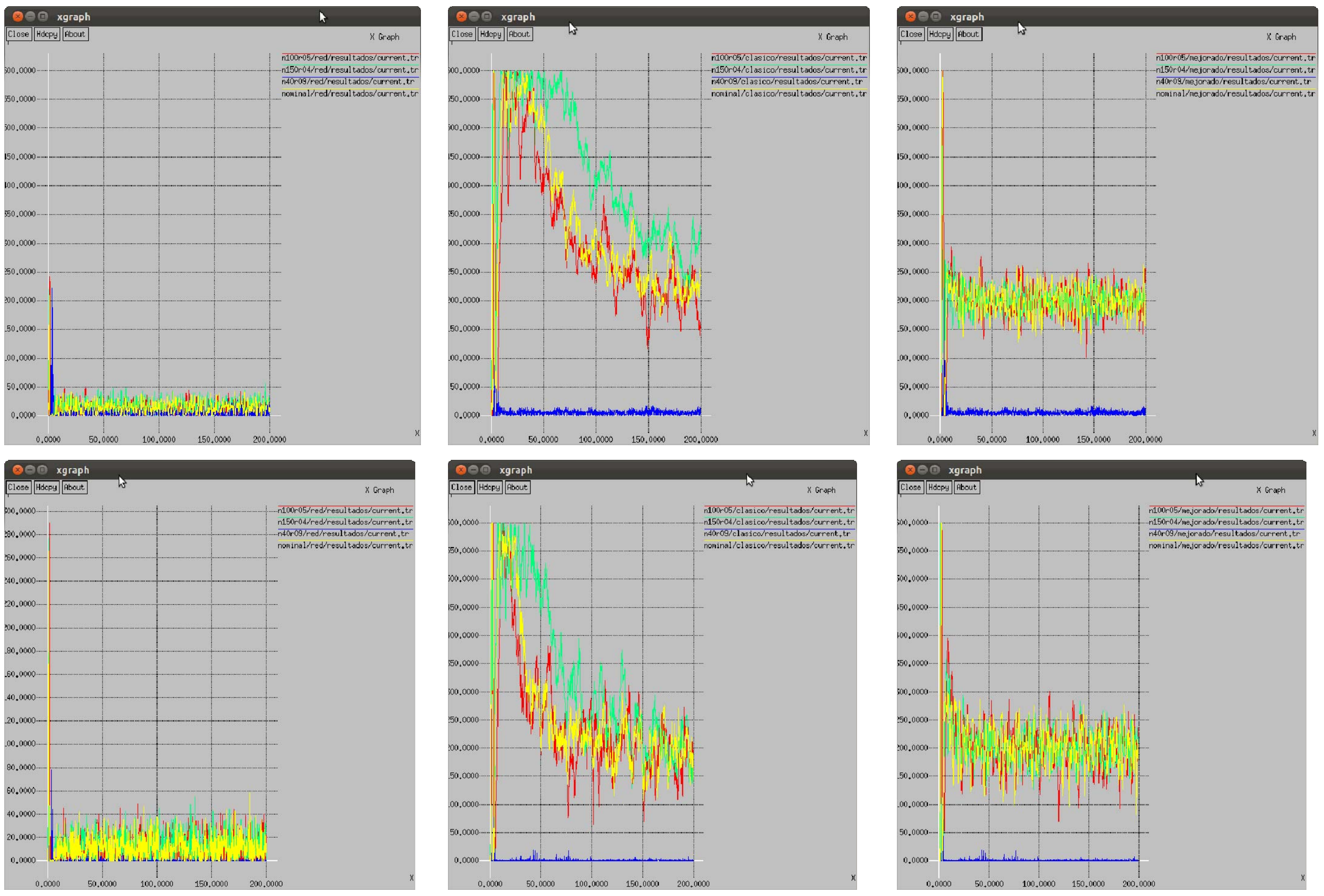
xgraph
Close
Hdcpy
About
X Graph
n100r05/red/resultados/current.tr
n150r04/red/resultados/current.tr
n40r09/red/resultados/current.tr
nominal/red/resultados/current.tr
n100r05/clasico/resultados/current.tr
n150r04/clasico/resultados/current.tr
n40r09/clasico/resultados/current.tr
nominal/clasico/resultados/current.tr
n100r05/mejorado/resultados/current.tr
n150r04/mejorado/resultados/current.tr
n40r09/mejorado/resultados/current.tr
nominal/mejorado/resultados/current.tr

Figure 10: Non-linear simulations in ns-2

Using the approach in [7], a PID is tuned for both linear systems (Table II). The closed loop for these settings is stable for both approaches: the classical and the one presented in the paper. This system has negative gain. With the new PID, the sign is included in the gain scheduling, but the classic PID should be taken into account.

B. Linear Simulations

Figures 8 and 9 show the closed loop step response of the normalized and classical systems. The proposed PID greatly improves the response when the network deals with cases 1, 2 and 4. Case 3 results are slower than when the classical approach is used.

C. Non-linear simulations

This sub-section shows a comparison between the PID proposed in the paper, a classical PID and RED. Table I summarizes the different network parameters that have been used. The PIDs have been tuned as in Table II. RED minimum and maximum thresholds are set to 180 and 220, respectively.

Figure 10 depicts the results of the non-linear simulations that have been carried out using the very well known ns-2. The results with the link capacity C set to 1875 packet/sec are shown in the first row of Figure 10. The second row depicts the results when C=940 packets/sec. In both cases the reference is set to 200 packets, i.e., the queue at the router should take this value. The simulation lasts 200 seconds. The first column illustrates the RED algorithm; the classic PID is in the second column and the results of the gain scheduling discussed in the paper correspond to the third column.

The experiment has been carried out for the four scenarios described at the beginning of this section. The RED algorithm cannot deal properly with the requirements. This is logical because this technique is not intended for following a reference. Packets are marked following the steps described in Section III.

The classic PID gives good results, but it is slow. If C=940 packets/sec., the queue does not reach the reference value at any of the scenarios until t=150 sec. This would be almost unacceptable in a real network. When, in case 3 (blue line), the link is not congested, no packets are queued at the router.

Finally, the gain scheduling PID gives the best results. The response is very fast and there is almost no difference (in terms of settling time or final value) for any of the scenarios. As in the classic PID approach, when N=40 and the delay is 0.9 sec., the link is not congested. Results are very promising and further tests would give us a better insight.

V. CONCLUSION

This paper has presented a comparison between a PID with linear gain scheduling and normalized values, a classic PID and an AQM/RED algorithm using the non-linear simulation environment ns-2. The controller is tuned for the worst scenario and works properly in a wide range of situations.

The main goal was to show that when traffic network changes and the delay, the number of users or the link's capacity are not the nominal values, the controller gives good results. The proposed PID outperforms the other approaches, as linear and non-linear simulations confirm.

The technique presented in the paper gives more uniform results than the classical approach. It does not matter if the scenario is close to the nominal situation or if it is a worst case setting, the controller reacts adequately and the settling times are all of the same order, whereas the classic approach cannot deal well with extreme situations. The RED algorithm gives the worst results of the three.

Future work will include testing the technique in a network with more congested routers.

REFERENCES

[1] Azuma, T., T. Fujita, M. Fujita (2006). Congestion control for TCP/AQM networks using State Predictive Control. Electrical Engineering in Japan, 156, 1491-1496.

[2] Deng, X., S. Yi, G. Kesidis. C.R. Das (2003). A control theoretic approach for designing adaptive AQM schemes. GLOBECOM'03, 5, 2947 – 2951.

[3] Jacobson, V. (1988). Congestion avoidance and control. ACM SIGCOMM'88.

[4] Ryu, S., C. Rump, C. Qiao (2004). Advances in Active Queue Management (AQM) based TCP congestion control. Telecommunication Systems, 25, 317-351.

[5] Hollot, C.V., V. Misra, D. Towsley, W. Gong (2002). Analysis and Design of Controllers for AQM Routers Supporting TCP flows. IEEE Transactions on Automatic Control, 47, 945-959.

[6] Aström K.J. and T. Hägglund (2006), Advanced PID control. ISA. NC.

[7] S. Floyd &V. Jacobson, Random early detection gateways for congestion avoidance. *IEEE/ACM Transactions on Networking*, *1*(4), 1993, 397-413.

[8] Hohenbichler, N. (2009). All stabilizing PID controllers for time delay systems. Automatica, 45, 2678-2684.

[9] Silva, G., A. Datta and S. Bhattacharyya. PID controllers for time-delay systems. Birkhäuser, 2005.

[10] Long, G.E., B. Fang, J.S. Sun and Z.Q. Wang (2010). Novel Graphical Approach to Analyze the Stability of TCP/AQM Networks. Acta Automatica Sinica, 36, 314-321.

[11] Alvarez, T. (2010). "Practical Design of PID Controllers for TCP/AQM". UKACC Control 2010, Coventry, United Kingdom.

[12] Hollot, C.V. and Y. Chait (2001). Nonlinear stability analysis for a class of TCP/AQM networks". In Proceedings of the 40th IEEE Conference on Decision and Control, Orlando, USA.

[13] Alvarez, T. (2012). Design of PID Controllers for TCP/AQM Wireless Networks. WCE 2012, London.

[14] Al-Hammouri, A.T. , V. Liberatore, M.S. Branicky, and S.M. Phillips (2006b). Parameterizing PI congestion controllers. Proc. First International Workshop on Feedback Control Implementation and Design in Computing Systems and Networks, Vancouver, CANADA.

[15] Al-Hammouri, A.T., V. Liberatore, M.S. Branicky, and S.M. Phillips (2006a). Complete stability region characterization for PI-AQM. SIGBED Review, 3(2).