A control theoretical approach to congestion control of TCP/AQM networks

Teresa Alvarez, Anuar Salim and J.M. Maestre

***Abstract*— This paper shows how control techniques, such as PID or generalized predictive control, can improve the performance of TCP/IP networks when dealing with congestion. Drop tail, or more sophisticated AQM techniques such as RED, perform worse than process control techniques. A comparison between the different methods in different working scenarios is analyzed. Two different GPC controllers are evaluated. One considers the model resulting from on-line identification and the other is the linearization of the non-linear model.**

I. Introduction

As the growth of Internet increases and users demand new applications and better performance, the rise in data volume generates such problems as long delays in delivery, lost and dropped packets, oscillations and synchronization problems ([1], [2], [3]). These troubles are due to congestion, which happens when there are too many sources sending too much data too fast for the network to handle, and it is a very serious drawback. Thus, it is necessary to reduce this problem as much as possible. At present, there are methodologies to deal with this issue ([4], [5]): *congestion control* which is used after the network is overloaded and *congestion avoidance* which takes action before the problem appears. This paper deals with congestion control because it is where feedback control techniques can be openly and easily applied.

Internet congestion control is carried out in the transport layer at the sources (end systems) and has two parts: the end-to-end protocol TCP (Transmission Control Protocol), and the active queue management (AQM) scheme, implemented in routers. As explained in [5], [6] and [7], the most common AQM objectives are: *efficient queue utilization* (to minimize the occurrences of queue overflow and underflow, thus reducing packet loss and maximizing link utilization), *queueing delay* (to minimize the time required for a data packet to be serviced by the routing queue) and *robustness* (to maintain closed-loop performance in spite of changing conditions).

These AQM schemes enhance the performance of TCP. This has been a subject of research for several years and different algorithms have been proposed: RED [9], PI [6], or REM [10]. These AQM schemes have both advantages and disadvantages as they do not work perfectly under every traffic situation. For instance, RED can detect and respond to long-term traffic patterns, but it cannot detect congestion caused by short-term traffic load changes. From the moment that researchers published mathematical models of AQM, control theory based approaches have been used to analyze and design AQM ([5] and the references therein,[6]).

Model Based Predictive Control (MBPC) [13] has been successfully applied to several real systems. It is one of the few control techniques that can handle constraints, systems with simultaneous time-delay and time varying parameters. The only drawback is that it can be computationally expensive. The most relevant MBPC approaches to the TCP congestion control problem found in the literature are now presented. Their solutions are innovative and give good results. [14] worked with a model-free LQC subspace predictive control. Results are very good, but no constraints are considered. [15] applied a GPC [16] to solve the problem and compared the results with a P and PI controller. Again, no constraints are taken into account. The same can be said for [12] and [17]. The predictive controllers are very smart and non-computationally taxing. [18] considered a neural predictive controller and presented a comparison with a PI. In a previous work by the authors [19], a constrained GPC was compared with a PI and a RED/AQM controller. Results were promising but further tests in a more realistic environment were clearly necessary.

In this paper, we apply drop tail and RED (classical congestion control techniques) and compare them with P, PI, PID and GPC controllers. From the results, it can be concluded that the PID and the constrained GPC perform better than the classic strategies such as DROP TAIL or RED. Although the PID gives very good results, the predictive controller outperforms it. When constraints are taken to the maximum, packet drop probability can be limited and smoother responses are obtained.

The paper is organized as follows. Section II introduces the TCP/AQM dynamic model. Section III describes the AQM control problem, DROP TAIL, RED, PID and predictive control. Section IV shows simulation results and the comparison between the different control techniques

This work was supported in part by the Junta de Castilla y León under Grant VA037A08.

T. Alvarez is with the Department of Engineering Science and Automatic Control, School of Computer Engineering, University of Valladolid, Valladolid, Spain (tel: +34983423276, email: tere@autom.uva.es, corresponding author).

A. Salim is with the Department of Engineering Science and Automatic Control, Faculty of Science, University of Valladolid, Valladolid, Spain (email: anuar.salim@alumnos.uva.es).

J.M. Maestre is with the Dept. de Ingeniería de Sistemas yAutomática, Escuela Superior de Ingenieros, University of Seville, Spain (email: pepemaestre@cartuja.us.es)

applied to the system. Finally, some conclusions and future work are given.

II. Dynamic Model of an AQM Router

This section presents the modelling of an AQM router. First, a non-linear model is given and then the linear version is derived.

A. Non-linear model

The model was developed using fluid-flow and stochastic differential equation analysis (presented in [6]). For simplicity, this paper considers a reduced version that ignores the TCP timeout mechanism. The model relates the average value of the network variables and is described by the following coupled, nonlinear differential equations:

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

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

Where

$W \approx$ average TCP window size (packets),

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

$R \approx$ round-trip time $= \dfrac{q}{C} + T_p$ (secs),

$C \approx$ link capacity (packets/sec),

$T_p \approx$ propagation delay (secs),

$N \approx$ load factor (number of TCP sessions),

$p \approx$ packet drop probability.

As explained by [6], 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 packet arrival rate and link capacity. The queue length and window size are positive, bounded quantities, i.e., $q \in [0, \bar{q}]$ and $W \in [0, \overline{W}]$, where $\bar{q}$ 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. Moreover, it has been assumed that the AQM scheme implemented at the router marks packets using Explicit Congestion Notification (ECN, [20]) to inform the TCP sources of impending congestion.

B. Linear Model

Although an AQM router is a non-linear system, in order to analyze certain types of properties and design controllers, we need a linear model which is presented in this sub-section.

To linearize (1), we assume that the number of active TCP sessions and the link capacity are constant, i.e., $N(t)=N$ and $C(t)=C$. The dependence of the time delay argument $t-R$ on queue length q, is ignored and it is assumed to be fixed to $t-R_0$. Then, local linearization of (1) about the operating point results in the following equation:

$$\partial\dot{W}(t) = -\frac{N}{R_0^2 C}\left(\partial W(t) - \partial W(t-R_0)\right) - \frac{1}{R_0^2 C}\left(\partial q(t) - \partial q(t-R_0)\right) - \frac{R_0^2 C}{2N^2}\partial p(t-R_0)$$

$$\partial\dot{q}(t) = \frac{N}{R_0}\partial W(t) - \frac{1}{R_0}\partial q(t) \quad (2)$$

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:

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

Equation (2) can be further simplified by separating the low frequency ('nominal') behavior ($P(s)$ in (4)) of the window dynamic from the high frequency behavior ($\Delta(s)$ in (4)) which is accounted as parasitic.

$$P(s) = \frac{C^2/(2N)}{\left(s + (2N)/(R_0^2 C)\right)\left(s + 1/R_0\right)},\ \Delta(s) = \frac{2N^2}{R_0 C^3}\left(1 - e^{-R_0 s}\right) \quad (4)$$

III. The AQM Control Problem

This section introduces the control formulation of an AQM router and how RED and PI control can be applied.

A. AQM as feedback control

Taking (4) as starting point, [6] gives a feedback control system of AQM (Figure 2). 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 $\Delta(s)$ denotes the high-frequency window dynamics and $P(s)$ (plant dynamics) relates how p dynamically affects q.

A. AQM using RED

Random Early Detection (known as RED) was presented by [9]. 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 cooperative, 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. [6] proposed the following transfer function model for the RED controller:

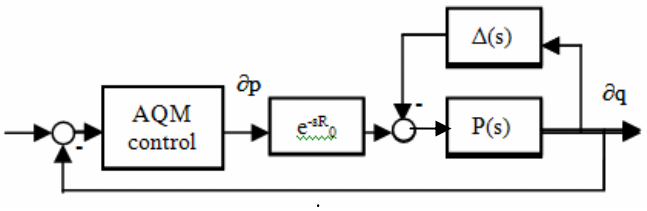

Fig. 1. Block diagram of AQM as feedback control system

$$C(s) = \frac{K \cdot L_{red}}{s + K} = \frac{K_{red}}{s/k_{red} + 1} \quad (5)$$

Following the guidelines in [11]:

$$K_{red} = \frac{R_0^3 C^2}{(2N)^2} L_{red}\ ,\ L_{red} = \frac{p_{\max}}{\max_{th} - \min_{th}} \text{ and}$$

$$k_{red} = -C \ln(1 - \alpha_{red}) \quad (6)$$

Where α_{red} is RED's queue-averaging weight. The corresponding block diagram is shown in Fig. 2,

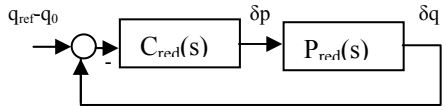

Fig. 2. RED linearized block diagram

where: $P_{red}(s) = \frac{-C^2/(2N)}{\left(s + (2N)/(R_0^2 C)\right)\left(s + 1/R_0\right)} e^{-R_0 s}$ (7).

As explained in [5], RED introduces a range of reference input values, rather than a reference input. So, RED shows oscillatory queue length dynamics and gives poor performance under a wide range of traffic environments. Nevertheless, the RED transfer function is quite useful for studying certain properties.

B. AQM using PID Control

The PID [21] is the most common form of feedback. It can be described by (8):

$$u(t) = K_p\left(e(t) + \frac{1}{T_i}\int_0^t e(\tau)d\tau + T_d \frac{de(t)}{dt}\right) \quad (8).$$

The PID implemented in the computer follows the guidelines [21]. The block diagram is the same as in the RED case.

C. AQM as predictive control

Model Based Predictive Control (MBPC) ([13]) is a control strategy based on the explicit use of a model to predict the process output over a long-range time period. A receding control horizon technique is used: only the first control signal is applied (so all the changes that take place between two control signal calculations are considered). A cost function is minimized at each sampling time.

Generalized Predictive Control (GPC, [16]) is a classic predictive controller. There are MIMO formulations, but as we are working with a SISO congestion control formulation, this is the description that will be given below (9).

$$A(q^{-1})y(t) = B(q^{-1})u(t) + \frac{T}{\Delta}\xi(t) \quad (9)$$

where: A, B and T are polynomials in the q^{-1} operator, $\Delta = 1 - q^{-1}$ and ξ is white noise. The predicted future values of the controlled variables are given by (10).

$$\hat{y}(t+j) = \sum_{k=1}^{j} g_k \Delta u(t-k+j) + p(t+j) \quad (10)$$

where g_j is the step response between y (router queue) and u (probability of discarding a packet) and p is the free response. Then, the sequence of changes of the control variable in a control horizon Nu: Δu(t+j), j=0,..., Nu-1 are calculated. The predicted output is as close as possible to the internal reference r(t+j). This is translated into an optimization problem, where a quadratic cost function of the tracking error and the manipulated variable is minimized, taking into account constraints on Δu, u, y, and any other constrained variable that depends on Δu. This optimization problem can be stated as (11) and (12):

$$J = \sum_{j=N1}^{N2} \gamma\left(\hat{y}(t+j) - r(t+j)\right)^2 + \sum_{j=0}^{Nu-1} \beta \Delta u(t+j) \quad (11)$$

$$\left.\begin{aligned} D_m \le \Delta u(t+j) \le D_M, j = 0,..., N_u - 1 \\ U_m \le u(t+j) \le U_M, j = 0,..., N_u - 1 \\ L_m \le \tilde{y}(t+j) \le L_M, j = N_1,...N_2 \end{aligned}\right\} \quad (12)$$

Where the coefficients γ and β give the relative weight of every prediction error or change in the control variable. When no constraints (12) are considered, there exists an explicit solution of (11).

In the AQM scheme in Figure 2, there is one input (p), one output (q) and no measured disturbances. Thus, the transfer function that will be used as a model to predict the future outputs is given by the low frequency component of the model in (4):

$$P_{GPC}(s) = \frac{-C^2/(2N)}{\left(s + (2N)/(R_0^2 C)\right)\left(s + 1/R_0\right)} \quad (13)$$

The system delay will be taken into account in the control and prediction horizons. Choosing the sampling time T_s as $0.2\sqrt{\tau_1^2 + \tau_2^2}$ ([12]) where $\tau_1 = R_0^2 C/(2N)$ and $\tau_0 = R_0$. Applying the zero-order-hold transformation, the discrete transfer function of (11) is represented by:

$$P_{GPC}(z^{-1}) = z^{-d} \frac{b_1 z^{-1} + b_2 z^{-2}}{1 + a_1 z^{-1} + a_2 z^{-2}} \quad (14)$$

where the coefficients a_i and b_i are directly calculated from the discretization, and d represents the system delay in time samples. In our case, simplifying, if R_0 is the continuous time-delay, then d=round_upper(R_0/T_s).

IV. Simulation

In this section, simulation results under different working conditions are presented. The simulations have been carried out using ns-2 [22]. Ns-2 is a discrete event simulator targeted at networking research. The PID and the GPC controllers have been added to the congestion control

procedures available in ns. The P and PI controllers are also implicitly included.

The network topology is depicted in Figure 4. It is a typical single bottleneck topology and reflects the working scenario defined in [12]: N=40 TCP sessions, C = 250 packets/sec., Tp=0.3 sec., so R_0=0.7 and W_0= 4.375 packets.

The controllers have been tuned using the nominal values given by: N=50 TCP sessions, C = 300 packets/s, Tp=0.2 s., R_0=0.533 and W_0= 3.2 packets.

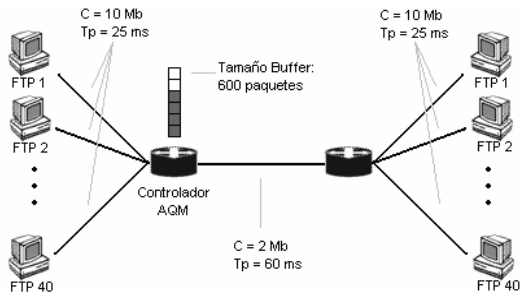

Fig. 3. Dumbbell topology

For comparison, a drop tail strategy, the RED AQM [9], a P, PI, PID and a constrained GPC that minimizes (10) subject to (11), were implemented and simulated under different situations.

The parameters of the different controllers are:

- P: K_p = - 0.0020
- PI: K_p = -0.0013 and T_i = 0.8816.
- PID: K_p = -0,0022, T_i = 1.8207 and T_d = 0.32.
- RED: min_{th} = 70, max_{th} = 120, p_{max} = 0.1 and w = 0.002.

The GPC controller has the following settings:

- Sampling time: T_s=0.2013 s.
- Model 1: $A(z^{-1}) = T(z^{-1}) = 1 - 1.476z^{-1} + 0.5416z^{-2}$, $B(z^{-1}) = z^{-3}\left(-14.9 - 12.15z^{-1}\right)$
- Model 2: $A(z^{-1}) = T(z^{-1}) = 1 - 1.03z^{-1} + 0.15z^{-2}$, $B(z^{-1}) = z^{-3}\left(-165 - 82z^{-1}\right)$
- Prediction horizon: N1=1, N2=5
- Control horizon: Nu=3.
- γ=100, β=0.
- Default values for constraints are:
  - $0 \le p \le 1$, $-0.05 \le \Delta p \le 0.05$
  - $0 \le q \le 300$

Model 1 is the discretization of (13) and model 2 has been calculated applying identification techniques to the system of Figure 3 in ns-2. The first GPC gives good results (as will be seen later on this section), but clearly the GPC with the second model does better than the other one.

The most obvious advantage of the first method is that it is possible to obtain a model for a different network configuration with few calculations, but when using the second model the process of finding a new model is more tedious (not more difficult, but it would take longer as it is not a straightforward calculation).

Sub-section A shows the results when we consider drop tail as the congestion control algorithm. This technique does not allow a reference to be set; we can only change the maximum queue size. So the results we can obtain are independent of the chosen reference. Sub-section B and following ones present different simulations and results.

A. *Drop tail*

Drop tail [24] is a simple queue management algorithm used by Internet routers to decide when to drop packets. All packets are treated identically. When the queue is filled to its maximum capacity, the newly arriving packets are dropped until the queue has enough room to accept incoming traffic.

We set the maximum queue size to 600 packets. Upper Fig. 4 shows the queue size during the simulation and lower Fig. 4 shows the rate of discarded packets.

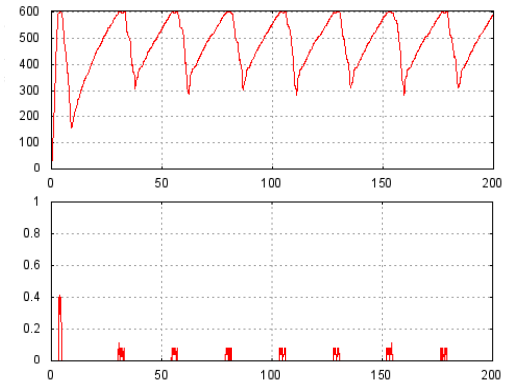

Fig. 4. Drop tail instant queue and discarding rate

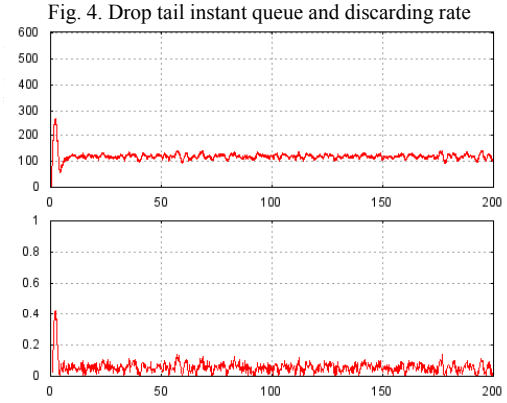

Fig. 5. PID controller

While the queue is not full, the TCP agents go on sending packets. Only when the maximum size is reached does the router begin to discard packets and the agents are notified. The result is the saw teeth visible on the graph. This is not effective as the RTT increases, which is bad for transmissions.

B. *Constant reference*

In order to obtain some statistics, we consider a constant reference: 120 packets. Figs. 5-8 show the instant queue values (upper graph) and the packet drop probability (lower graph).

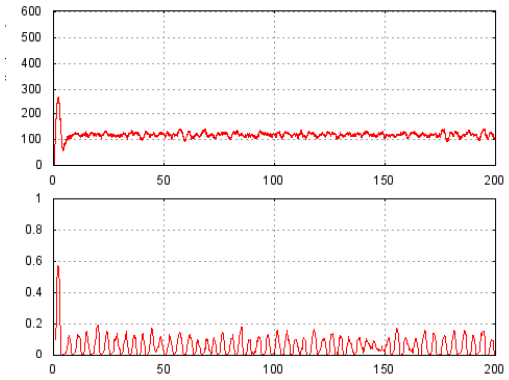

Fig. 6. GPC controller (first model)

TABLE 1: STATISTICS

	Mean	Std. dev.
Drop tail	476.99	99.0385
PID	120.38	15.5302
GPC1	121.63	20.5657
GPC2	120.18	14.3173
RED	109.03	18.6809

C. *Variable reference*

Table 1 summarizes the mean and standard deviation for each controller. Clearly, drop tail and RED achieved the worst results, as expected. In the case of the first one, these results are due to its intrinsic behavior. RED's parameters are fixed at the beginning and do not change during the simulation.

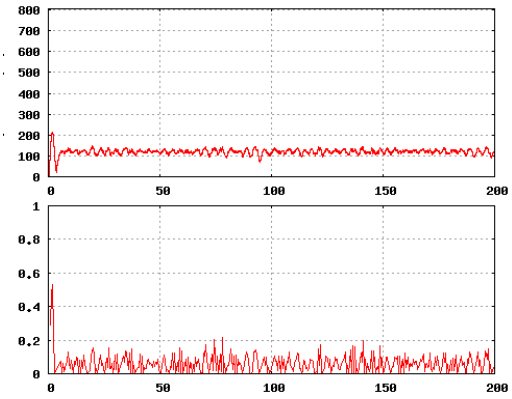

Fig. 7. GPC controller (second model)

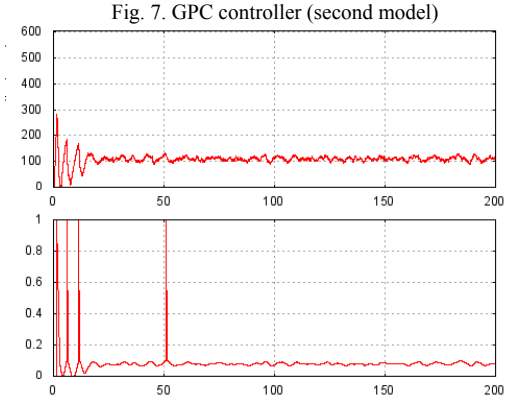

Fig. 8. Red controller

As explained in the previous section, RED does not follow a reference, but tries to maintain the queue between two values without surpassing the maximum probability. Both predictive controllers show a smaller overshoot than the other techniques. Now, the queue reference value is changed: t=0, q=120 pkts, t=100, q=50 pkts, t=200, q=250 pkts.

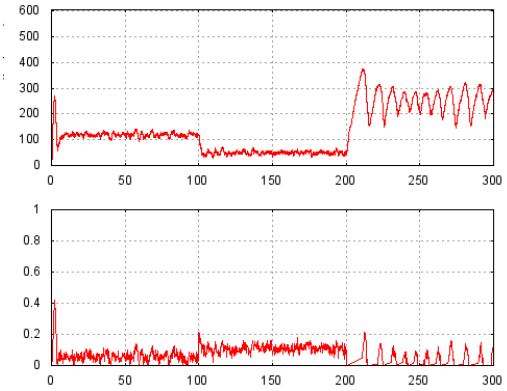

Fig. 9. PID, variable reference

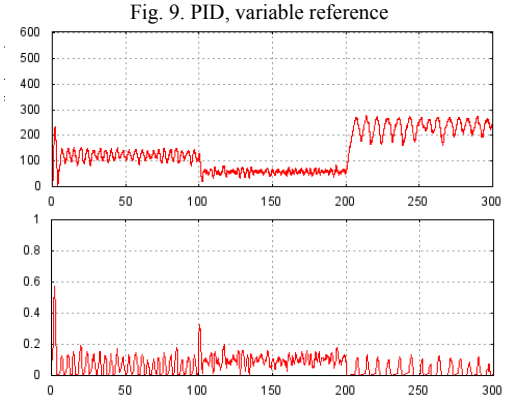

Fig. 10. GPC (first model)

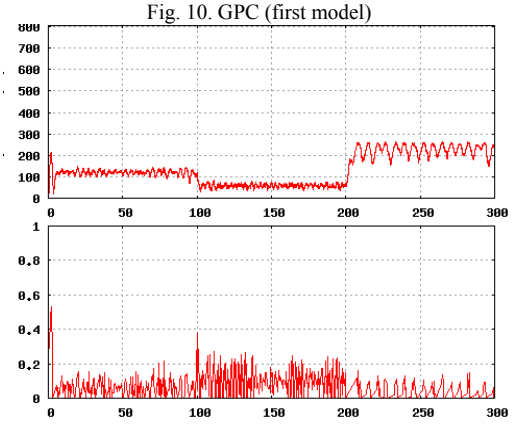

Fig. 11. GPC (second model)

We only show results for the PID (Fig. 9) and the GPC (Figs. 10-11) controllers. As the RED technique does not include information on reference changes, the results would be the same as in the previous sub-section. All three give a good performance, but both predictive controllers present a better behavior when the maximum or minimum values allowed for the variables are reached. This is the case when the reference is set at 250 packets. Sometimes, the packet drop probability should be smaller than 0, but as this is not possible, the PID saturates its value to 0, but the predictive controller can take this into account in the constraints.

D. *Limiting the packet drop probability maximum value*

One of the characteristics of RED is that the maximum packet drop probability is a tuning parameter. Figure 15

shows the GPC evolution in a reference changing environment with packet drop probability limited to 0.1 (as in the RED/AQM). From Fig. 12, we can conclude that the predictive controller gives good performance when changing the reference and, at the same time, the probability has been limited. We can thus ensure that this variable is maintained within range and a reference is followed as closely as the allowed values permit. This is one of the advantages of predictive controllers when compared with PID.

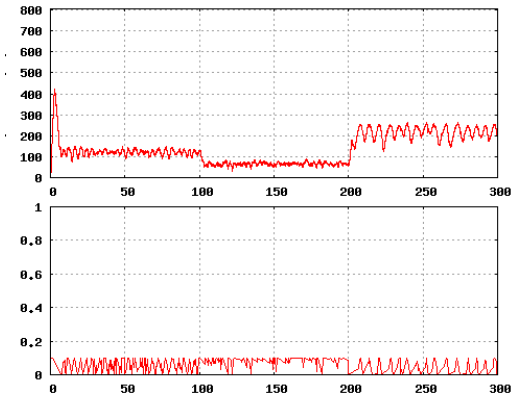

Fig. 12. GPC, 0≤p≤0.1

V. CONCLUSIONS

This paper has presented a comparison between different techniques that can help with the congestion control problem. The application of PID and GPC controllers as active queue management methodologies for TCPI/IP networks gives promising results. Predictive controllers allow changing references to be followed and the variable ranges with the constraints to be limited. In this work, we have limited the packet drop probability range of values with the constraints, but we do not penalize changes in the cost function. It is not as if we were working with a valve as input variable.

The PID controller works fine in most situations, but when we need queues in the extremes of the desired range, the GPC gives a better performance. Furthermore, the use of constraints on signals has been rather important.

Further work will include MIMO and time varying network models, so changing conditions can be included. This would help to control congestion at several nodes. Moreover, as the number of packets in a queue is an integer value, hybrid predictive control could also be explored. It would also be interesting to compare results with other RED variants such as ARED [25], DRED [26] or SRED [27], where some of the parameters could be adapted dynamically.

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] Xiong, N., Y. He, Y. Yang, B. Xiao, X. Jia (2005). On designing a novel PI controller for AQM routers supporting TCP flows. APWEB 2005, 991-1002.

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

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

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

[6] 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.

[7] Hayes. M.J., S.M. Mahdi Alavi, P. Van de Ven (2007). An Improved Active Queue Management scheme using a two-degree-of-freedom feedback controller. European Control Conference (ECC 2007), July 2-5, 2007, Kos, Greece.

[8] Sun, J., S. Chan, K.-T. Ko, G. Chen, M. Zukerman (2007). Instability effects of two-way traffic in a TCP/AQM system. *Computer Communications*, 30, 2172-2179.

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

[10] Athuraliya, S., S.H. Low, V.H. Li, Q. Yin (2001). REM: active queue management. *IEEE Networking*, **15**, 48-53.

[11] Durresi, A., P. Kandikuppa, M. Sridharan, S. Chellappan, L. Barolli, R. Jain (2006). LED: Load Early Detection: A Congestion Control Algorithm based on Router Traffic Load. *Journal of Information Processing Society of Japan*, **47**, 94 – 107.

[12] Marami, B., N. Bigdeli and M. Haeri (2007). Active Queue Management of TCP/IP Networks Using Rule-Based Predictive Control. IEEE International Symposium on Industrial Electronics (ISIE 2007), June 4-7, 2007, Vigo, Spain.

[13] Camacho, E.F., C. Bordons, C. (2007). *Model Predictive Control.* Springer Verlag, 1995, 2nd ed. 2004., Corr. 2nd printing, 2007.

[14] Chiera, B.A. and L.B. White (2007). Application of model-free LQC subspace predictive control to TCP congestion control. *International Journal of Adaptive Control and Signal Processing*, published on-line 22/08/2007.

[15] Zhu, R., H. Teng and W. Hu (2004). A predictive controller for AQM router supporting TCP with ECN. In AWCC 2004, C.-H. Chi and K.-Y. Lam (Eds), 131-136.

[16] Clarke, D.W., Mohtadi, C., and Tuffs, P.S.: 'Generalised predictive control-Part I: The basic algorithm and Part II: Extensions and Interpretations', *Automatica*, 1987, 23, (2), pp. 137-160.

[17] Marami, B.and M. Haeri (2007). Neural network approximation of model predictive controller for congestion control of TCP/AQM networks. International Conference on Control, Automation and Systems, Seoul, Korea, 2591-2596.

[18] Rahnamai, K., K. Gorman and A. Gray (2006). Model predictive neural control of TCP flow in AQM network. IEEE NAFIPS 2006, 493-498.

[19] T. Alvarez and S. Cristea, *AQM Control of TCP/IP Networks using Generalized Predictive Control*, in Proc. UKACC International Conference on Control 2008, Manchester, UK, Sep 2-4, 2008.

[20] Ramakrishnan, K., S. Floyd (1999). Explicit Congestion Notification. *ACM SIGCOMM Computer Communications Review*, **24**, 8-23.

[21] K.J. Aström and T. Hägglund, *Advanced PID control*. ISA. NC, 2006.

[22] Network Simulator, *ns-2*, http:/www.isi.edu/nsnam/ns/

[23] W.K. Chen, *Linear Networks and Systems* (Book style). Belmont, CA: Wadsworth, 1993, pp. 123–135.

[24] Comer, Douglas E. (2006). *Internetworking with TCP/IP*, 5, Prentice Hall: Upper Saddle River, NJ..

[25] *W. Feng et al., "A self configuring RED gateway", Proc. INFOCOM '99,* 1999.

[26] J. Aweya, M. Oulette and D.Y. Montuno, "A control theoretic approach to Active Queue Management", Computer Networks, vol. 36, no. 2-3, 2001, pp. 203-35.

[27] T.J. Ott, V. Lakshman, and L. Wong, "SRED: Stabilized RED," *Proc. IEEE INFOCOM '99*, 1999, ftp://ftp.telcordia.com/ pub/tjo/SRED.ps