This is a postprint version of the following published document: N. Merayo, D. Juárez, J.C. Aguado, I. de Miguel, R. J. Durán, P. Fernández, R. M. Lorenzo, and E. J. Abril . PID Controller Based on a Self-Adaptive Neural Network to Ensure QoS Bandwidth Requirements in Passive Optical Networks. In: IEEE/OSA Journal of optical communications and networking, 2017, vol.9, no. 5, pp. 433-445. DOI: https://doi.org/10.1364/JOCN.9.000433

A PID controller based on a Self-adaptive Neural Network to ensure QoS bandwidth requirements in Passive Optical Networks

N. Merayo, D. Juárez, Juan C. Aguado, I. de Miguel, R.J. Durán, P. Fernández, R.M. Lorenzo, E.J. Abril

***Abstract*—In this paper it is proposed a Proportional-Integral-Derivative (PID) controller integrated with a Neural Network (NN) to ensure QoS bandwidth requirements in Passive Optical Networks (PONs). To the best of our knowledge, this is the first time an approach that implements a NN to tune a PID to deal with QoS in PONs is used. In contrast to other tuning techniques such as Ziegler-Nichols (ZN) or genetic algorithms (GA), our proposal allows a real-time adjustment of the tuning parameters according to the network conditions. Thus, the new algorithm provides an online control of the tuning process unlike the ZN and the GA techniques, whose tuning parameters are calculated offline. The algorithm, called NN-SPID (Neural Network Service level PID), guarantees minimum bandwidth levels to users depending on their SLA (Service Level Agreement) and it is compared with a tuning technique based on genetic algorithms (GA-SPID). The simulation study demonstrates that NN-SPID continuously adapts the tuning parameters achieving lower fluctuations than GA-SPID in the allocation process. As a consequence, it provides a more stable response than GA-SPID, since it needs to launch the GA to obtain new tuning values. Besides, NN-SPID guaranties the minimum bandwidth levels faster than GA-SPID. Finally, NN-SPID is more robust than GA-SPID under real time changes of the guaranteed bandwidth levels as GA-SPID shows high fluctuations in the allocated bandwidth, especially just after the change is made.**

Index Terms—Proportional-Integral-Derivative (PID); Neural Network (NN), Passive Optical Network (PON); Dynamic Bandwidth Allocation (DBA); Quality of Service (QoS); Service Level Agreement (SLA)

Manuscript received December 15, 2016.

N. Merayo, D. Juárez, J.C. Aguado, I. de Miguel, R.J. Durán, P. Fernández, R.M. Lorenzo and E. Abril are with the Optical Communications Group of the Department of Signal Theory, Communications and Telematic Engineering. E.T.S.I. Telecomunicación, Universidad de Valladolid (Spain), Campus Miguel Delibes, Paseo de Belén 15, 47011 Valladolid, Spain, (e-mail: noemer@tel.uva.es).

I. Introduction

Passive Optical Networks (PONs) and LR-PONS (Large Range PONs) are the most deployed access network technologies all over the world in the last few years. In fact, the report presented in the FTTH Council [1] states that in Europe about 22 million homes will be connected by the end of 2018, amounting to 10.6% of all homes. Moreover, 100 million people in the Asia-Pacific region are now subscribed to Fiber to the Home (FTTH) services by means of PON infrastructures. In a typical configuration a Passive Optical Network follows a tree topology between the Optical Line Terminal (OLT), located in the Central Office, and an Optical Network Unit (ONU), located at the user's house (Fig. 1). These access networks use optical fiber as the transmission medium and they are provided with two channels, that is, two independent wavelengths. In the upstream channel (from ONUs to the OLT) PONs have a multipoint-to-point connectivity so every ONU share the same wavelength. As a consequence, a MAC (Medium Access Control) protocol is required to manage data collisions of different users (ONUs). Dynamic Bandwidth Allocation (DBA) algorithms, based on the TDMA (Time Division Multiple Access) protocol, is the most extended option, as they dynamically distribute the available bandwidth depending on the real time bandwidth demand or the Quality of Service (QoS) requirements contracted by end users [2-9]. On the other hand, the connectivity in the downstream channel (from the OLT to the ONUs) is point-to-point so the optical splitter (Fig. 1) broadcasts the input traffic to every ONU.

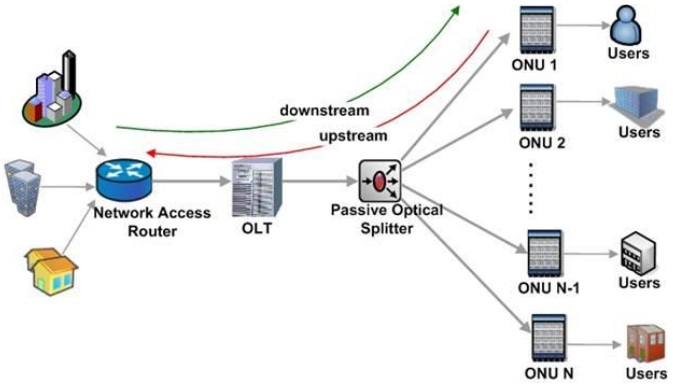


Figure 1.Typical topology of a PON and LR-PON access infrastructure.

Regarding Quality of Service (QoS), access networks have to differ among priority profiles by means of Service Level Agreements (SLA) that are contracted by users with service providers. Each profile consists of different services and applications and the PON should ensure some requirements for each service typically associated with minimum guaranteed bandwidth levels, maximum delays or maximum permitted jitters. These QoS restrictions have to be efficiently fulfilled by DBA algorithms and it is highly desirable that the performance of these algorithms becomes independent from real time network or traffic conditions. In this way, the majority of DBA algorithms are focused on complying with QoS requirements [10-11] stipulated by service providers. The most common type of SLA contract regards to ensure minimum bandwidth levels depending on the profile priority and its related applications or services, so service providers should be provided with robust DBA algorithms that comply with these guaranteed bandwidths. Authors in [12] proposed an algorithm that allows bandwidth and delay bounds for guaranteed service traffic, but they do not fulfill the bandwidth for the entire profile. In [13], authors proposed a DBA algorithm to provide bandwidth guarantees for only high priority subscribers based on the contracted SLA while providing best-effort service to the remaining subscribers. Moreover, the algorithm developed in [14] guarantees fairness among users by allocating excess bandwidth proportional to their subscription rates. On the other hand, many research algorithms apply fixed weighted factors to each SLA to comply with its associated minimum bandwidth requirements [15-16]. However, these algorithms cannot properly react when real time changes are done by service providers in the SLA conditions. Therefore, it is desirable that the DBA algorithms can be independent from the initial weights or bandwidth conditions and they can be able to automatically adapt to the new network conditions. To overcome this situation, Proportional-Integral-Derivative (PID) systems provide a robust way to automatically control QoS requirements to end users in PONs. PID controllers are able to offer a reliable, stable and effective response when controlling different systems and they are used in many environments due to their good performance [17-20]. However, their implementation in optical networks is quite reduced. For example, one PID controller was integrated in dynamic optical burst-switched networks [21] to model a wavelength-locking loop in order to stabilize the output of tunable lasers. In the articles [22-23] authors designed controllers to manage certain tasks in optical grid networks and for the establishment of lightpaths in WDM networks. Besides, authors in [24] proposed a GA-based PID active queue management control design for a class of TCP communication networks. Quite recently, we proposed the design of PID and P controllers to efficiently deal with bandwidth and delay requirements in PONs [25-26]. In particular, the algorithm in [25] proposes a PID that controls that different profiles are ensured minimum guaranteed bandwidth levels according to their contracted SLA. Besides, in [26] we implemented a P controller that supports QoS requirements regarding maximum delays of high sensitive traffic. Both strategies demonstrated best performance and robustness than other previous existing DBA algorithms.

However, PID systems need to be tuned according to the system under control. A great number of tuning techniques, such as the well-known Ziegler-Nichols frequency response method, are manual and based on experiments, so they are laborious, quite slow and imprecise. As a consequence, many novel tuning approaches implement expert systems techniques to automatically tune PID controllers with high accuracy and robustness [27] because they provide intelligent advice or make intelligent decisions using rule-based programs or expertise knowledge in one specific field. Some of these techniques are fuzzy logic, genetic algorithms, artificial intelligence or neural networks. On the one hand, fuzzy logic sometimes bases the reasoning and decisions on incomplete information similar to that of human beings [28] and if the system is complex, the selection of the fuzzy rules and functions may be a laborious issue [29-30]. In contrast, genetic algorithms have demonstrated better performance, stability and robustness than other techniques providing a range of good solutions for the tuning parameters, as genetic algorithms select and reproduce the best individuals of each population [20] [31-35]. Consequently, in a recent research study [36] we designed and implemented a tuning technique based on a genetic algorithm integrated in a P controller to deal with QoS delay constraints of high priority services in PONs. Although the simulation study showed a very good performance to control network parameters, genetic algorithms are an offline technique, which means that if traffic or network conditions change along the time, it should be necessary to launch the genetic algorithm to look for the best solutions (tuning parameters) under the new conditions. Consequently, it should be quite appealing the integration of online tuning techniques to solve this problem. Therefore, in this proposal we have selected a neural network to design an online and an adaptive tuning process along the time [37-39]. However, traditional Neural Networks (NN) need to be provided with good examples to be trained and consequently the learning process could become quite long, because their performance highly depends on the amount of data they are trained with [28]. In contrast, our proposal focuses on developing a self-learning and adaptive neural network that permits an online and real-time adjustment of the tuning parameters of the PID according to the recent working status of the network. In fact, there are many algorithms in the literature that implement self-adaptive neural networks to design intelligent PID controllers [40-43]. In particular, we propose to develop an expert and intelligent PID controller based on a NN in order to automatically provide minimum bandwidth levels to different profiles. To the best of our

knowledge this is the first bandwidth allocation algorithm that implements a NN technique to optimize the tuning process in PID controllers to efficiently provide QoS bandwidth guarantees in PONs.

The rest of this paper is organized as follows. Section II describes the PID controller implemented to provide bandwidth guarantees to different kind of user profiles. Moreover, it is explained the designed Neural Network to tune the controller. Section III shows the simulation results and the discussion of results. Section IV addresses the conclusions of the paper and some future research proposals.

II. Design of the PID controller with an auto-adaptive Neural Network to guaranteed minimum bandwidth levels to different priority profiles

A. Principles of Neural Networks

Neural Networks are simplified mathematical models that try to emulate the human nervous system so they try to extract brain capacities to solve complex problems. A neural network is a big mesh that propagates signals from one neuron to others and it constantly modifies the strength of the response in order to activate or inhibit the connection between neurons. Artificial neural networks work in the same way [44]. Then, each artificial neuron has an internal state, called activation level, so each neuron receives signals that permit to change its state using the activation function and the received signals. The input of one neuron is calculated as the sum of every input weighted by a factor, as it can be observed in Fig. 2.

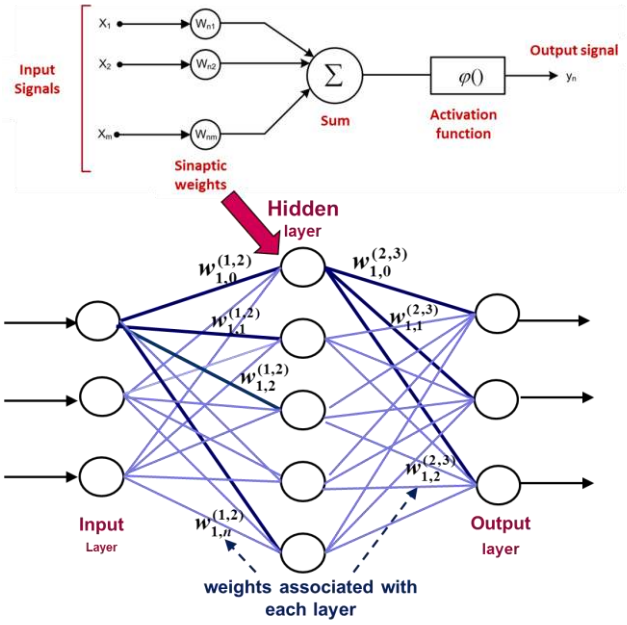


Figure 2. Structure of one artificial neuron and a neural network

Therefore, this synaptic weight (positive, negative, zero) defines the strength of the connection between two neurons. For example, a positive weight works as a positive stimulation and a negative weight acts as an inhibitor. As a consequence, by means of the adjustment of the synaptic weights the neural network adapts its response to different environments. Finally, the input of the neuron is processed by the activation function, that provides the output of the neuron, and this output is propagated to the next connected neurons [44]. On the other hand, the organization of the neurons inside the network is the topology and it depends on the number of layers, the number of neurons at each layer and the degree of connection between neurons. The most common neural networks are multilayer, so the input layer receives information from outside and the last layer is the final output. Furthermore, there are several intermediate layers called hidden layers (Fig. 2).

B. Design of the PID controller to ensure QoS bandwidth requirements in the OLT

In previous works we have developed several algorithms based on PIDs to provide QoS requirements in PON networks. In fact, the DBA algorithm presented in [25], called SPID (Service level agreement PID), efficiently assigns minimum bandwidth levels using a PID controller. The main purpose of PID controllers and its variants (P and PI controllers) is to keep the value of a variable close to a reference value [18][45]. Thus, the controller calculates the error, defined as the difference between the current value of the variable under control and its reference value. According to this committed error ($e[n]$), the PID updates one control signal ($u[n]$), following equation (1). This control signal is composed by the proportional term (regards the current error), the integral term (accumulation of past errors), and the derivative term (prediction of future errors). Moreover, the variable K_p is the proportional gain, T_i is the integral gain and T_d is the derivative gain.

$$u[n] = K_p \cdot e[n] + K_p \cdot \frac{T_{sample}}{T_i} \sum_{m=0}^{n} e[n] + K_p \cdot \frac{T_d}{T_{sample}} (e[n] - e[n-1]) \qquad (1)$$

To assign bandwidth to each ONU_i at every cycle ($B_{alloc}^{onu_i}$), the algorithm implements a polling policy with a limited scheme, the most common and efficient way to allocate bandwidth in PON networks [46]. This limited and simple scheme is defined as $B_{alloc}^{onu_i} = Minimum\left\{B_{required}^{onu_i}, B_{\max}^{onu_i}\right\}$, where $B_{required}^{onu_i}$ is the bandwidth demanded by ONU_i in one cycle and $B_{\max}^{onu_i}$ is the maximum bandwidth allowed to each ONU at one cycle (to avoid the channel monopolization of some ONUs). In order to integrate the controller in the bandwidth allocation process ($B_{alloc}^{onu_i}$), the term $B_{\max}^{onu_i}$ is constantly updated by means of the control signal ($u[n]$) so that $B_{\max}^{onu_i}$ can dynamically evolve to efficiently ensure the minimum bandwidth levels to every profile. The block diagram of the algorithm is shown in Fig. 3. As the PID algorithm controls the guaranteed bandwidth associated with every ONU, the reference value is the minimum guaranteed bandwidth that have to be ensured by the service provider depending on the contracted SLA ($B_{guaranteed}^{sla \in onu_i}$). The term under control is the mean allocated bandwidth associated with each ONU_i ($\overline{B_{alloc}^{onu_i}[n]}$), so to calculate the instantaneous error ($e[n]$) committed at each ONU, it is necessary to subtract the mean allocated bandwidth from its guarantee bandwidth

level ($e[n] = B_{guarantee}^{sla \in onu_i} - \overline{B_{alloc}^{onu_i}}$). Therefore, to update the maximum permitted bandwidth ($B_{\max}^{onu_i}$), the control signal $u[n]$ is added to the previous maximum permitted bandwidth ($B'^{onu_i}_{\max} = B_{\max}^{onu_i} + u[n]$), as it can be observed in Fig. 3. In this way, if for example the mean allocated bandwidth of ONU_i is lower than their guaranteed bandwidth level, the error becomes positive and the control signal also too, so the algorithm increments its maximum permitted bandwidth to allow ONU_i to comply with the QoS bandwidth restrictions. Finally, a delimiter is included in the system to adapt the maximums proportionally to the ones calculated by the controller, to fit in the maximum cycle time of the EPON standard (2 ms) (Fig. 3).

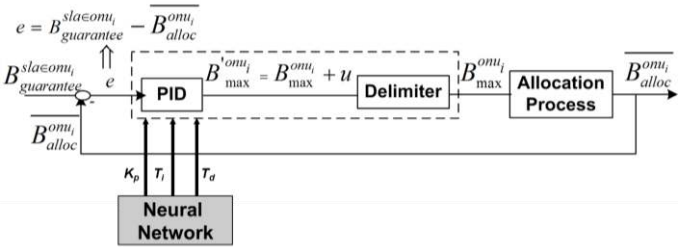


Figure 3. Block diagram of the designed PID to control the QoS bandwidth requirements in PONs

C. Design of the Neural Network to tune the PID controller

To calculate the tuning parameters of the PID controller, the algorithm SPID uses the frequency response method proposed by Ziegler-Nichols [18][45]. However, ZN is a manual technique that may become a time-consuming and a laborious method if the selected values are far from the suitable ones, as it was demonstrated in [36]. In contrast, in [36] we enhanced the performance of SPID by incorporating a genetic algorithm to automate the tuning process, the so-called GA-SPID (Genetic Algorithm SPID). Althought the GA efficiently automates the tuning process and these tuning parameters offer good performance in the bandwidth allocation process, this technique does not permit a real time adaptation of the tuning parameters in case the network or traffic conditions suddely change. Therefore, genetic algorithms are consider an offline tuning technique. However, it is quite desirable to design one method that updates the tuning parameters according to whatever change inside the network, that is, an online tuning technique. In this way, we propose the integration of an auto-adaptive neural network inside the PID controller that auto-learns from the last committed errors depending on the real time network conditions, called NN-SPID (Neural Network SPID) algorithm.

To design the neural network that tunes the controller we apply one alternative formula for the control signal of the PID in the discrete time. This expression is represented in equation (2), where K_p is the proportional constant, K_i is the integral constant and K_d is the derivative constant. In this way, the tuning parameters using neural networks will be K_p, K_i and K_d. However, the relationship between these parameters and the T_i y T_d terms of equation (1) is given by equation (3), where T is the sample time of the PID (the time between two consecutive executions of the PID).

$$\Delta u(k) = \Delta u[k-1] + K_p\left(e[k] - e[k-1]\right) + K_i e[k] + K_d\left(e[k] - 2e[k-1] + e[k-2]\right) \quad (2)$$

$$K_i = K_p \frac{T}{T_i} \qquad K_d = K_p \frac{T_d}{T} \quad (3)$$

Therefore, the general scheme of the controlled process using a NN is shown in Fig. 4. As it can be observed in the picture the tuning parameters K_p, K_i and K_d, are provided by the output of the neural network. Furthermore, our proposed method needs to know the real time error of the PID controller (equation (2)) to efficiently update the tuning parameters. Consequently, this error is provided by the PID controller, so both of them are constantly interchanging information inside the OLT, as it can be noticed in Fig. 3. In fact, this error corresponds to the absolute value of the difference between the guaranteed bandwidth of each ONU ($B_{guaranteed}^{sla \in onu_i}$) and its mean allocated bandwidth ($\overline{B_{alloc}^{onu_i}[n]}$), that is, $e[k] = \left| B_{guarantee}^{sla \in onu_i} - \overline{B_{alloc}^{onu_i}[n]} \right|$. Finally, it is worth saying that the PID controller and the NN are integrated in the OLT, since the bandwidth allocation process of the entire EPON is carry out inside this active component.

On the other hand, to design the architecture of the neural network it is necessary to select some parameters, such as the number of layers, neurons, activation functions and training methods. Regarding the number of layers, the literature reveals that many algorithms implement one hidden layer (apart from the input and the output layer), because it provides a simple structure and good performance [37-39]. The number of neurons at the output layer accords with the tuning parameters of the PID, so it is set to three (K_p, K_i and K_d,) and the number of neurons in the input layer is three because the tuning parameters are calculated based on the three last errors, *e[k], e[k-1]* and *e[k-2]* (following equation (2)). Finally, the number of neurons in the hidden layer depends on the accuracy we want to achieve, but literature points that a number between one and four times the number of inputs is enough, so we select five neurons (simulations have been done to demonstrate this selection) [37-39].

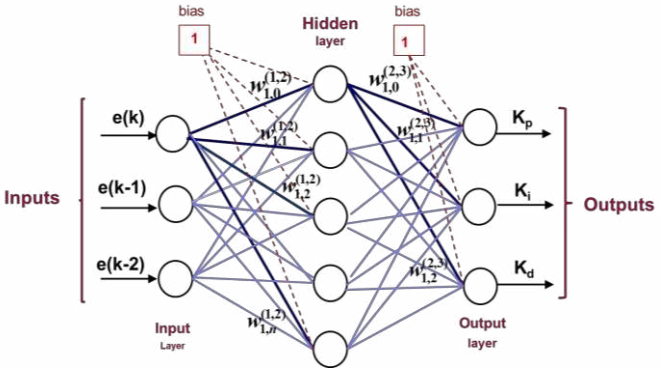


Figure 4. Flow diagram of the designed Neural Network to control QoS bandwidth requirements in NN-SPID.

Moreover, we have added a bias signal at the intermediate and at the output layers to look for a more stable response of the system. Then, the feedforward artificial neural network is a multilayer perceptron that maps sets of input data onto a set of appropriate outputs, that is, good tuning parameters according to the errors (Fig. 4). Furthermore, the selection of the activation function is based on the range of values of the desirable results. In

this way, the sigmoid function is chosen for the input layer since it is one of the most classical and extended. For the output layer, although the sigmoid function is the most used, we select the lineal function since we need a large number of values for the tuning parameters (K_p, K_i and K_d) and the values of the sigmoid function are limited by a lower and a maximum threshold.

Traditional neural networks use learning techniques in which the network is provided with a set of good examples to efficiently learn and work and they keep weights between neurons with fix values. In contrast, our proposal constantly auto-learns from the last errors (applying an incremental learning technique) so it constantly changes the weights between neurons. The mathematical study used to increment the matrix of weights is based on several published studies [40-43]. In this way, the backpropagation (BP) NN is one of the most implemented neural networks at many systems. It uses the backpropagation learning algorithm applied in two stages: data feedforward and error backpropagation. In the data feedforward phase, the error of the PID controller (*e[k], e[k-1]* and *e[k-2]*) is inserted in the input layer and propagated to the next layers (hidden and output layers), and the algorithm obtains the new tuning parameters for the next cycles. On the other hand, in the error backpropagation stage, with the last output values (tuning parameters) it is calculated the committed error (typically using the quadratic performance index) and they are propagated backward (from output to input). Therefore, the weights between neurons are adjusted using some method, typically the gradient descent algorithm, based on these errors. These two phases are continuously repeated along the time to ensure a good performance of the bandwidth allocation process. In particular, in our proposal at each number of iterations of the PID, called *n*, the tuning parameters are updated (feedforward algorithm). Besides, at each number of iterations of the PID (called *m*), the matrix of weights is also updated (backpropagation algorithm). The value of *m* has to be higher than *n* because the tuning parameters have to be updated taken into account the new matrix of weights. If not, it may exist periods with changes in the matrix of weights that are not reflected in the calculation of the new tuning parameters.

In table I…

TABLE I
TABLE OF NOTATIONS REGARDING THE NEURAL NETWORK

Notation	Definition
2	0.046169
4	0.04882
5	0.0497220
6	0.067423
8	0.05567

In particular, the feedforward-propagating algorithm calculates the input and output values of each layer according to the current weights and the last inputs of the NN (the last three errors and the bias signal), achieving the new tuning parameters at the output of the NN. Then, the input layer follows equation (4), in which the inputs *x(i)* corresponds to *e[k], e[k-1], e[k-2]* and the bias input and $O_j^{(1)}(k)$ is the output of the layer. The superscripts in the different equations (1, 2, 3) are associated with the layer, that is, input, hidden and output layer. Moreover, the superscript of the weights refers to weights between two layers, for example $w_{ij}^{(1,2)}$ correspond to the weights between the input and the hidden layer.

$$O_j^{(1)}(k) = x(j) \qquad (j = 0,1...) \tag{4}$$

For the hidden layer (subscript *2*), the inputs ($I_i^{(2)}(k)$) and outputs ($O_i^{(2)}(k)$) follow equation (5) and (6). In this equations, $w_{ij}^{(1,2)}$ are the weights between the input layer and the hidden layer. As it was said before, the sigmoid function (with positive and negative symmetry) is chosen as the activation function for this layer $\left(f(x) = \tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}} \right)$.

$$I_i^{(2)}(k) = \sum_j w_{ij}^{(1,2)} O_j^{(1)} \tag{5}$$

$$O_i^{(2)}(k) = f(I_i^{(2)}(k)) \qquad (i = 0,1...) \tag{6}$$

Finally, the inputs and outputs of the output layer follow equation (7) and (8). The output of this layer corresponds with the three tuning parameters (K_p, K_i and K_d) and the activation function for this layer is the linear function $\left(g(x) = x \right)$. However, the lineal function is truncated for avoiding negative values since tuning parameters have to be positive. The terms $w_{ij}^{(2,3)}$ are the weights between the hidden layer and the output layer.

$$I_l^{(3)}(k) = \sum_i w_{li}^{(2,3)} O_i^{(2)}(k) \tag{7}$$

$$O_i^{(3)}(k) = g(I_l^{(3)}(k)) \qquad (l = 0,1,..)$$

$$So: \quad O_1^{(3)}(k) = k_p, \; O_2^{(3)}(k) = k_i, \; O_3^{(3)}(k) = k_d \tag{8}$$

On the other hand, the backpropagation algorithm modifies the weighted factors in the NN, starting from the output layer to the input layer. To update the weights in the neural network, it is used the quadratic performance index of error (equation (9)), based on the parameter under control (guaranteed bandwidth levels). Furthermore, weights are modified using the gradient descent method. Then, the increments are adjusted according to the negative gradient direction to the weighting coefficients, equation (10), so an inertia term ( ) and a learning rate coefficient ( ) directly impact on the evolution and the convergence speed of the NN. Although both parameters are initially set to *0.1* (following the literature) in the results section we will analyze their impact on the NN. Furthermore, $\Delta W_{li}^{(3)}(k)$ denote the last increments achieved in the last weights modification between the hidden and the output layers.

$$J_1 = \left(\frac{1}{2}\right) e^2[k] = \left(\frac{1}{2}\right) (B_{guarantee}^{sla \in onu_i} - B_{alloc}^{onu_i})^2 \tag{9}$$

$$\Delta w_{li}^{(2,3)}(k) = -\eta \frac{\partial J_1}{\partial w_{li}^{(2,3)}} + \alpha \Delta w_{li}^{(2,3)}(k-1) \tag{10}$$

To calculate $\frac{\partial J_1}{\partial w_{li}^{(2,3)}}$, we follow equation (11). In this equation the term $\frac{\partial y(k)}{\partial \Delta u(k)}$ is unknown so we replace it by the $\operatorname{sgn}\left(\frac{\partial y(k)}{\partial \Delta u(k)}\right)$ approximation (the lack of accuracy can be rectify by η), as it is considered in several studies [40-43].

$$\frac{\partial J_1}{\partial w_{li}^{(2,3)}} = \frac{\partial J_1}{\partial y(k)} \frac{\partial y(k)}{\partial \Delta u(k)} \frac{\partial \Delta u(k)}{\partial O_l^{(3)}(k)} \frac{\partial O_l^{(3)}(k)}{\partial I_l^{(3)}(k)} \frac{\partial I_l^{(3)}(k)}{\partial w_{li}^{(2,3)}} \tag{11}$$

Taken into account equation (2) and (8) we obtain the expressions for $\frac{\partial \Delta u(k)}{\partial O_l^{(3)}(k)}$ (equation (12)).

$$\begin{aligned} \frac{\partial \Delta u(k)}{\partial O_1^{(3)}(k)} &= \frac{\partial \Delta u(k)}{\partial K_p} = e(k) - e(k-1) \\ \frac{\partial \Delta u(k)}{\partial O_2^{(3)}(k)} &= \frac{\partial \Delta u(k)}{\partial K_i} = e(k) \\ \frac{\partial \Delta u(k)}{\partial O_3^{(3)}(k)} &= \frac{\partial \Delta u(k)}{\partial K_d} = e(k) - 2e(k-1) + e(k-2) \end{aligned} \tag{12}$$

Finally, with these expressions we can simplify the formula for the weighted factors $\Delta w_{li}^{(2,3)}(k)$ of the output layer, as it is shown in (equation (13)).

$$\begin{aligned} &\Delta w_{li}^{(2,3)}(k) = \eta \delta_l^{(3)} O_i^{(2)}(k) + \alpha \Delta w_{li}^{(2,3)}(k-1) \\ &where \quad \delta_l^{(3)} = e(k) \operatorname{sgn}\left(\frac{\partial y(k)}{\partial \Delta u(k)}\right) \frac{\partial \Delta u(k)}{\partial O_l^{(3)}(k)} g'\left[I_i^{(3)}(k)\right] \end{aligned} \tag{13}$$

In the same way as the output layer, we modify the weighted factors at the hidden layer, following the expressions of equation (14).

$$\begin{aligned} &\Delta w_{ij}^{(1,2)}(k) = \eta \delta_i^{(2)} O_j^{(1)}(k) + \alpha \Delta w_{ij}^{(1,2)}(k-1) \\ &\delta_i^{(2)} = f'\left[I_i^{(2)}(k)\right] \sum_{l=1}^{3} \delta_l^{(3)} w_{li}^{(2,3)}(k) \end{aligned} \tag{14}$$

III. Simulation Results

A. Simulation Scenario

Simulations have been carried out in an EPON (based on the Ethernet protocol) network with 16 ONUs and one user connected to each ONU using OMNeT++ 4.2 [47]. The network scenario has similar characteristics proposed by other DBA algorithms in PONs [2-9]. Regarding the packet and traffic distribution, a self-similar traffic based on alternating Pareto-distributed on/off periods with a Hurst parameter of H=0.8 is used to emulate the practical Internet traffic, using the self-similar traffic generator developed in [48]. Then, it is assumed Ethernet packets between 64 and 1518 bytes and symmetry in the traffic load of every ONU, as the majority of existing DBA algorithms [16][21]. In this research study we compare the performance of GA-SPID and NN-SPID, that is, an offline and an online tuning technique, respectively. Moreover, both algorithms (GA-SPID and NN-SPID) dynamically modify the maximum permitted bandwidth of every profile (SLAs) depending on the committed error when guaranteeing the minimum stipulated bandwidth levels to each of them. Therefore, it is assumed three SLAs (SLA_0, SLA_1, SLA_2) in the EPON network and different QoS minimum bandwidth levels to each profile in order to check the algorithms' performance. However, the main scenario considers100 Mbps for SLA_0 profile, 75 Mbps for SLA_1 profile and 50 Mbps for SLA_2 profile. Finally, results are contained within 95% of confidence level.

B. Selection of parameters for the Neural Network

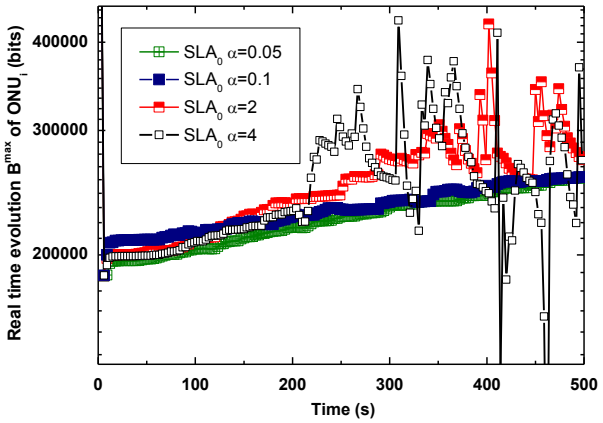
The first simulation study regards the selection of some parameters to model the neural network. This selection has been done running simulations and choosing the values that allows a good performance. The first parameter to analyze is the number of neurons in the hidden layer, so Table II collects the standard deviation of the mean allocated bandwidth of each ONU over its guaranteed bandwidth in Mbps (ONUs of the same SLA show same performance). The standard deviation provides a good measure of the algorithm's performance, as low deviations imply high accuracy. Then, the collected data in Table II show that results considering different number of neurons are quite similar and very low, less than 0.05 Mbps. As literature points that a number between one and four times the number of inputs (three) is enough, we chose an intermediate value of 5 neurons that also allows a very low deviation.

TABLE II
STANDARD DEVIATION OF $\overline{B_{alloc}^{onu_i}}$ OVER $B_{guarantee}^{sla \in onu_i}$ IN MBPS
CONSIDERING DIFFERENT NUMBER OF NEURONS IN THE HIDDEN LAYER

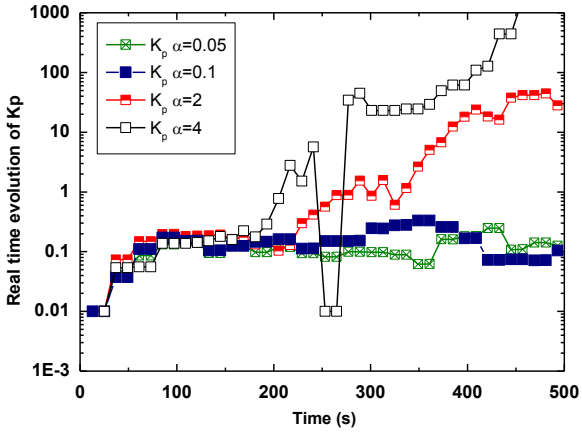
Number of neurons	Standard deviation SLA_0 (Mbps)	Standard deviation SLA_1 (Mbps)	Standard deviation SLA_2 (Mbps)
2	0.046169	0.0378122	0.02310
4	0.04882	0.030392	0.023456
5	0.0497220	0.0297659	0.0234553
6	0.067423	0.02963	0.02432
8	0.05567	0.028723	0.022764

On the other hand, another issue to analyze is the evolution speed of the matrix of weights, as it can be noticed in equation (13). As the goal is to minimize the objective function of the problem, there is a tradeoff between some parameters (α, η), because low values allow a slow convergence to the objective but very high steps can allow that the problem does not properly converge to the correct values. The inertia coefficient α in equation (13) highly affects the convergence speed in the equation, since it multiplies the previous increment of the weighted factors. Then, Fig. 5 (a) and (b) show the impact of this parameter in the evolution of the maximum permitted bandwidth periodically updated by the PID controller and in the real time evolution of the tuning parameters (we select K_p to

reduce the number of graphs). As it can be observed in the graphs, high values of α $(\alpha = 2,4)$, allows an bad convergence response of the PID controller that manage the bandwidth allocation process to efficiently ensure the minimum bandwidth levels (Fig. 5 (a)). This same performance (high fluctuations and incorrect convergence) is translated into the evolution of the K_p values as it can be observed in (Fig. 5 (b)). On the other hand, low values of the inertia coefficient $(\alpha = 0.05, 0.1)$ provide a more stable PID response and lower oscillations of the tuning parameter K_p. Therefore, as many research works use a typical value of 0.1 we select the same value [40-43].



(a)



(b)

Figure 5. Parameters of the neural network in NN-SPID with different α values (a) Real time evolution of the maximum permitted bandwidth (b) Real time evolution of *Kp*.

On the other hand, Fig. 6 shows the impact of the learning coefficient η in the real time evolution of the tuning parameters (assuming $\alpha = 0.1$).

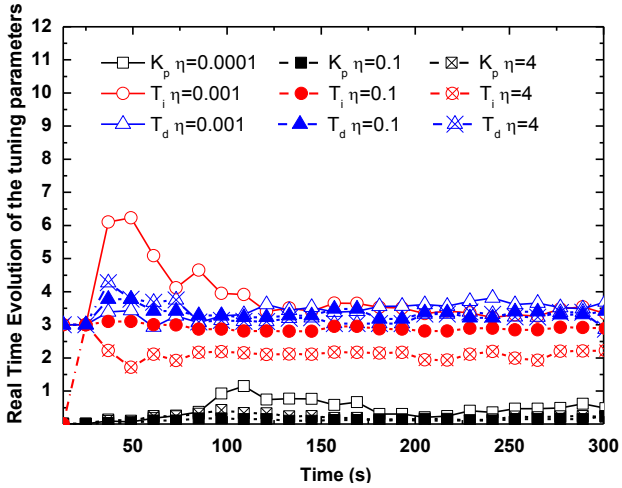


Figure 6. Real time evolution of the K_p, T_i and T_d tuning parameters considering different values of η.

As it can be observed in the graph, the performance of η does not highly impact on the evolution of the tuning parameters as the α parameter does. However, it can be noticed more oscillations for very low values $(\eta = 0.0001)$ and for very high values $(\eta = 4)$, so we select an intermediate value of 0.1 that it is also adopted in many research works [40-43].

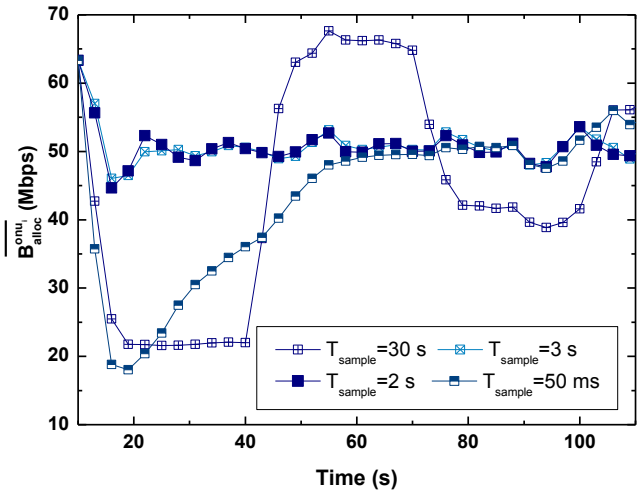
C. Performance of NN-SPID algorithm in comparison with GA-SPID

One of the most important characteristics in both algorithms is the convergence speed to calculated good tuning parameters. Regarding the genetic algorithm, the number of iterations of the PID to update the fitness value, the population size and the number of generations of the stop criterion have to be fixed, in order to simultaneously allow good performance and short tuning time. As it was said before, the genetic algorithm is offline, so once it calculates the best tuning parameters the PID starts to work with these values. The tuning time employed by the genetic algorithm is given by equation (15), where N is the population size of the genetic algorithm, m is the number of iterations of the PID in which each individual is tested to obtain its fitness, T_{sample} is the sample time used by the PID controller to calculate the committed error and N_{Gen} is the number of generations of the stop criterion.

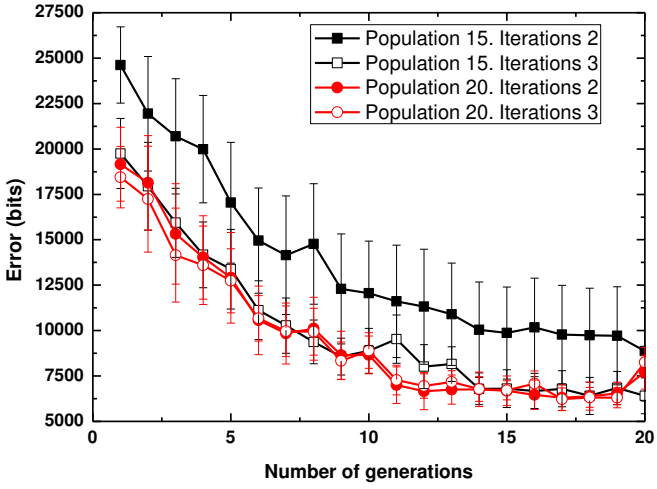
$$T_{tuning} = N \cdot m \cdot T_{sample} \cdot N_{Gen} \quad (15)$$

One of the parameters to analyze the performance of the tuning time in GA-SPID is T_{sample}. Fig. 7 (a) shows the real time evolution of the mean allocated bandwidth when assuming a set of T_{sample} values for the SLA_2 profile (the performance is the same for SLA_0 and SLA_1). As it can be noticed in the figure, if T_{sample} is high (30 s) the evolution to the guarantee bandwidth requirements for this profile, 50 Mbps, becomes more slow and instable. In contrast, a low value for T_{sample} (50 ms) makes the adaptation difficult to control as the PID lacks of enough samples to properly update the error to evolve to the guarantee bandwidth levels. Therefore, intermediate values (2 or 3 s) show a robust and a fast achievement of the stipulated bandwidth

requirements, so as we desire a low tuning time, we chose T_{sample}=*2 s*. Once we select the minimum and efficient value of T_{sample}, we have to determine the remaining parameters that directly impact on the tuning time. Then, Fig. 7 (b) represents the mean committed error (in bits) of the best individual in each generation when considering different population sizes and number of iterations. As it was deeply described in [36] GA-SPID follows the main steps of genetic algorithms. Then, each chromosome corresponds with the three tuning parameters (K_p, T_i, T_d) coded in a binary chain (16 bits per parameter). Furthermore, we have to evaluate each individual (chromosome) using he objective function in order to calculated the fitness of each of them. This objective function is based on the committed error of the PID using this individual (K_p, T_i, T_d), during m iterations of the PID, defined as $F = \frac{1}{m} \cdot \frac{1}{N_{onus}} \sum_{m} \sum_{i=0}^{N_{onus}} \left| e_i[m] \right|$. As a consequence, if we observe Fig. 7 (b) the committed error is reduced as long as the number of generations increases for every combination of population size and number of iterations. However, the worst performance is achieved for a population of 15 individuals and a number of iterations equal to 2. For the other combinations, the results are quite similar. Thus, as we require the lowest tuning time, we select a population of 20 individuals with 2 iterations. Furthermore, as the committed error becomes quite stable for a number of generations higher than 10, we can select this value for our genetic algorithm. As a consequence, taken into account these previous values and substituting them in equation (15), GA-SPID provides a total tuning time (T_{tuning}) equal to 800 seconds.



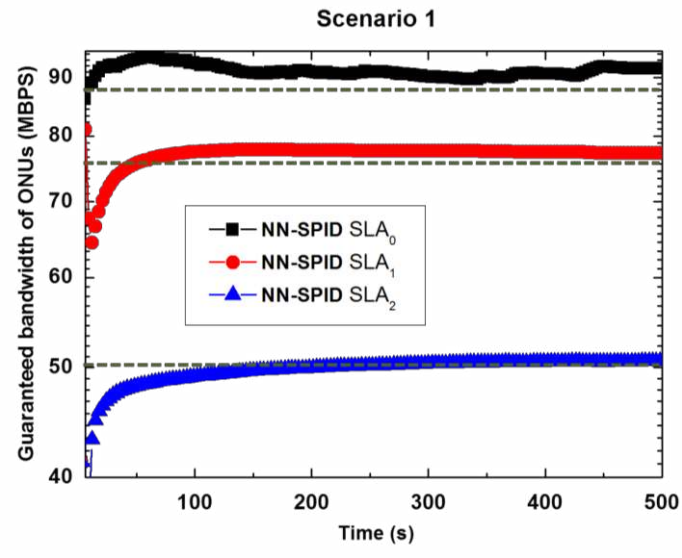
(a)



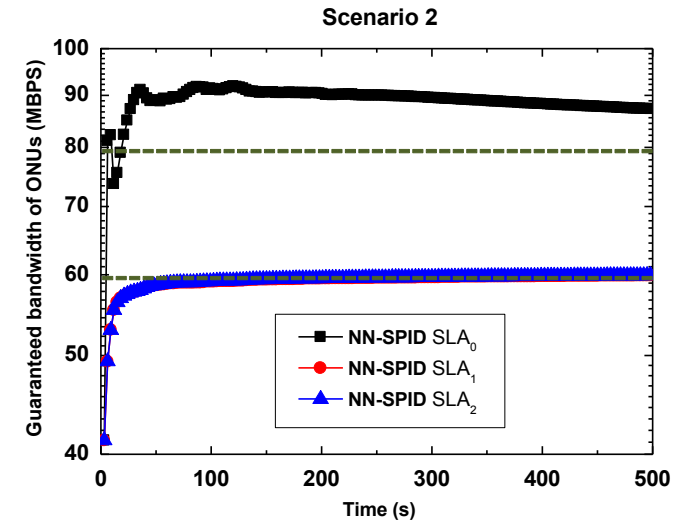
(b)

Figure 7. Parameters of the genetic algorithm in GA-SPID (a) Evolution of the mean allocated bandwidth when assuming different values of T_{sample} (b) Evolution of the mean committed error of the best individual of each generation when considering different population sizes and number of iterations of the PID.

In contrast, our new proposal based on Neural Networks, NN-SPID, permits an online modification of the tuning parameters, because it constantly adapts these values according to the network state to ensure the minimum guarantee bandwidth levels to every profile. In order to show the tuning time required by NN-SPID to achieve the guarantee bandwidth levels, Fig. 8 (a) and (b) represent the real time evolution of the mean allocated bandwidth for every profile. We check the performance for two different guaranteed bandwidth scenarios (Scenario 1: SLA_0=100 Mbps, SLA_1=75 Mbps, SLA_2=50 Mbps and Scenario 2: SLA_0=80 Mbps, SLA_1=60 Mbps, SLA_2=60 Mbps). For both scenarios, it can be noticed than NN-SPID is faster than SPID since it only needs around 100 seconds to ensure the minimum bandwidth levels for every profile in Scenario 1 (Fig. 8 (a)) and less than 50 seconds for Scenario 2 (Fig. 8 (b)). In Scenario 1 SLA_0 is given around 90 Mbps since its load is 0.9 (90 Mbps) so NN-SPID gives all demanded bandwidth to this profile. On the contrary, GA-SPID requires 800 seconds to provide good individuals to tune the PID. Consequently, it is demonstrated that NN-SPID provides the QoS bandwidth requirements faster than GA-SPID independently from the considered bandwidth scenario.



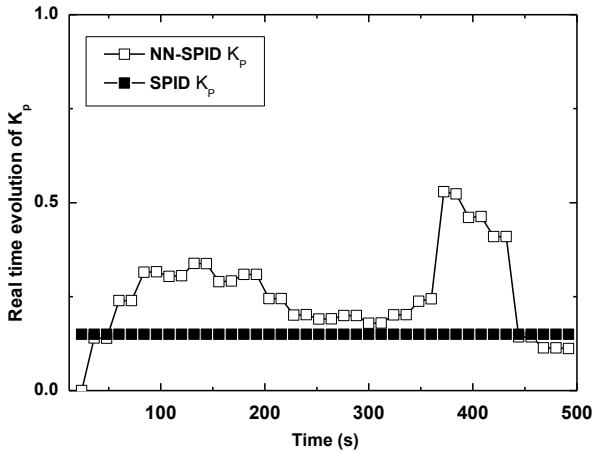
(a)



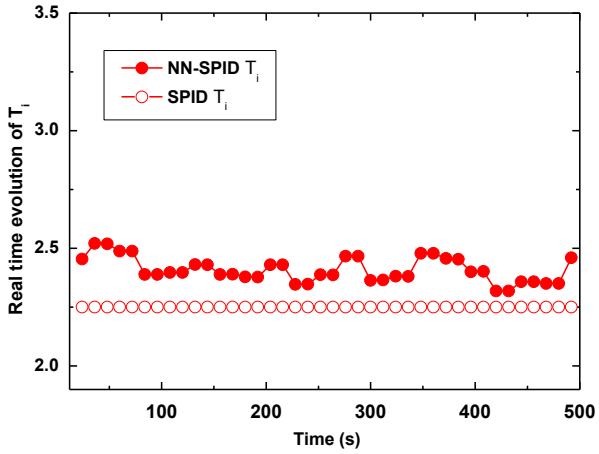
(b)

Figure 8. Real time evolution of the mean allocated bandwidth of NN-SPID for different Scenarios (a) Scenario 1 (b) Scenario 2.

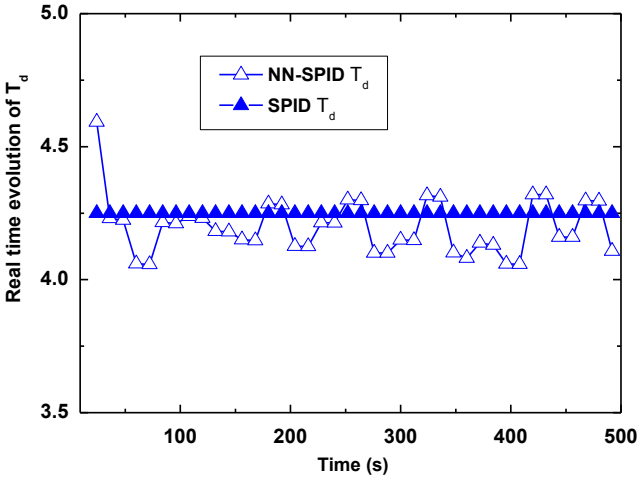
Furthermore, Fig. 9 (a) (b) (c) shows the real time evolution of the tuning parameters (K_p, T_i, T_d) for NN-SPID and GA-SPID in order to observe the differences between an adaptive and a not adaptive technique. It can be noticed that NN-SPID periodically adapts the tuning parameters, so it modifies the control signal of the PID to face real time changes in the network conditions. In contrast, the genetic algorithm does not modify the tuning parameters and it cannot immediately react when changes in the network conditions may appear.



(a)



(b)



(c)

Figure 9. Real time evolution of the tuning parameters for NN-SPID and GA-SPID (a) K_p (b) T_i (c) T_d.

On the other hand, to compare the robustness of both algorithms Fig. 10 represents the real time evolution of the maximum permitted bandwidth (periodically updated by the PID) when comparing NN-SPID and GA-SPID assuming different good individuals for GA-SPID (Scenario 1: k_p=1, T_i=2, T_d=5 and Scenario 2: k_p=1, T_i=2, T_d=2). In order to simplify the number of graphs we only represent SLA_1 and SLA_2, although the performance is similar for SLA_0. As it can be noticed in the graph, NN-SPID shows lower

oscillations than GA-SPID for both profiles, minimizing every abrupt fluctuation that appears in GA-SPID. This performance demonstrates that NN-SPID provides a more stable and robust response in the bandwidth allocation process than GA-SPID since the neural network applies a real time reconfiguration of the tuning parameters according to the network state. In contrast, the genetic algorithm has to be periodically launched to update the tuning parameters according to the last network conditions and this process can take a long time (around 800 s).

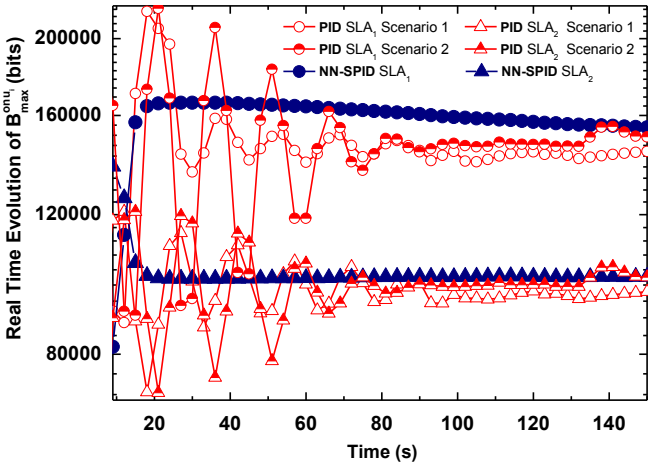


Figure 10. Real time evolution of the maximum permitted bandwidth for SLA_1 and SLA_2 comparing NN-SPID and GA-SPID.

Finally, the speed and stability of NN-SPID and GA-SPID become very important when service providers require real time changes in the guaranteed bandwidth levels. To demonstrate the performance of both algorithms in this situation, simulations have been carried out changing the guaranteed bandwidth levels at 150 seconds (Table III). Hence, Fig. 11 shows the evolution of the maximum permitted bandwidth along the time for the considered guaranteed bandwidth levels. As it can be noticed, both algorithms dynamically modify the maximum permitted bandwidth so that values can evolve to the new guaranteed bandwidth levels. However, the stability of both algorithms is quite different since GA-SPID shows high fluctuations in the allocation process when the guaranteed bandwidth levels change at 150 seconds, especially for the lowest priority profile SLA_2. On the contrary, NN-SPID exhibits a more stable response in the evolution of the maximum permitted bandwidth than GA-SPID, since it makes a real time adaptation of the tuning parameters according to the real time bandwidth requirements.

TABLE III

DIFFERENT GUARANTEED BANDWIDTH LEVELS FOR EVERY PROFILE ALONG THE TIME

Time (s)	Guaranteed Bandwidth SLA_0	Guaranteed Bandwidth SLA_1	Guaranteed Bandwidth SLA_2
0-150 s	100 Mbps	70 Mbps	50 Mbps
Higher than 150s	70 Mbps	40 Mbps	60 Mbps

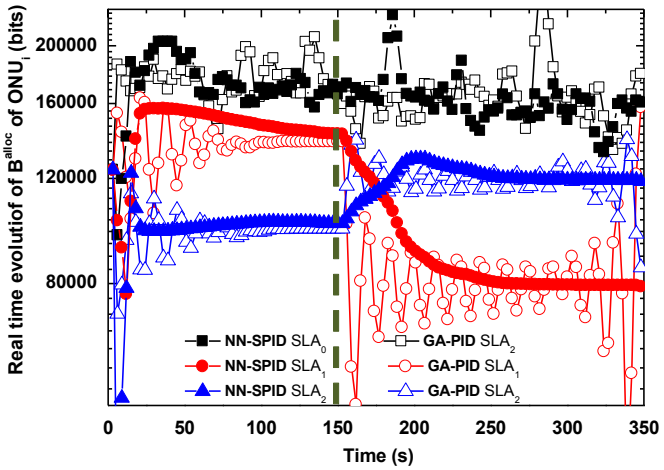


Figure 11. Variation of the mean allocated bandwidth of every profile when the guaranteed bandwidth levels vary along the time.

On theotherhand,DaSPIDisperfectlyapplicableto PONs ofdifferent coverages. Todemonstrateit, Fig. 15(a) and (b) shows the meanpacketdelayofDaSPIDforeach SLA (SLA0, SLA1 and SLA2) forP0 and P1 respectively,when considering severalendtoenddistances(25,50,75,and 100km).Asitcanbenoticed,forbothclassesofservice the meanpacketdelaysarealwaysbelowtheestablished bound foreachSLA,independentlyoftheconsidered distance. Thus,thepresentedalgorithmcanbeapplied in PONsofdifferent coverages without gettingworse performance.

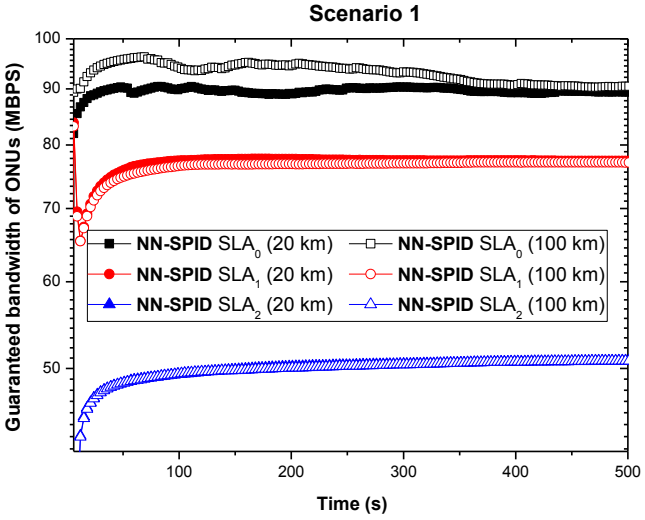


Figure 12. Variation of the mean allocated bandwidth of every profile when the guaranteed bandwidth levels vary along the time.

IV. CONCLUSION

In this research a PID control strategy based on an auto-adaptive neural network to control QoS requirements has been presented. The developed algorithm, called NN-SPID, has been designed to control the bandwidth allocation process so that different profiles are provided with

minimum bandwidth levels according to their contracted SLA. The control of quality of service in access network has never been implemented using a neural network integrated in a PID. The proposed algorithm has been compared with GA-SPID, a published algorithm designed for the same purpose than NN-SPID but it uses a genetic algorithm to automatically tune the PID controller. In this way, GA-SPID is an offline technique as it launches the genetic algorithm to calculate the best individuals (tuning parameters) for the controller and after that the PID starts the allocation process using these tuning parameters. In contrast, NN-SPID applies an online strategy to tune the controller since the values of the tuning parameters are constantly evolving to the best ones based on the last network state. Furthermore, the designed neural network does not use a learning technique in which the network is provided a priori with a set of good examples to learn and work, but NN-SPID applies an incremental learning technique so it adaptively learns from the real time network conditions modifying the associated weights of every layer of the neural network, in contrast to traditional neural networks that keep constant these weights along the time.

The results of the simulation study have demonstrated that NN-SPID ensures the minimum guarantee bandwidth levels to every profile faster than GA-SPID and independently from the QoS requirements. In fact, depending on the network scenario, NN-SPID is able to reduce up to ten times the required time to achieve the stipulated guaranteed bandwidth requirements. Furthermore, NN-SPID periodically adapts the tuning parameters, so it modifies the control signal of the PID to face real time changes in the network or traffic conditions. On the contrary, GA-SPID does not adapt the tuning parameters and it cannot efficiently react when changes in the network state happens. This performance was demonstrated when the guaranteed bandwidth levels were changed by service providers along the time. Results exhibited how GA-SPID shown higher oscillations than NN-SPID in the bandwidth allocation process especially just after the change appeared in the network. As a consequence, it can be stated that NN-SPID shows more stability and robustness than GA-SPID in case service providers require real time changes in the guaranteed bandwidth levels without interruptions in the offered services.

The integration of PID controllers and neural networks has never been proposed to provide QoS in PONs networks. Hence, in this paper we have developed an automatic and online PID tuning technique to efficiently ensure bandwidth requirements to different priority profiles using an auto-adaptive neural network. One interesting future research is focus on controlling other QoS parameters using this technique, such as the jitter, the packet delay of sensitive traffic or the packet loss ratio. On the other hand, we are also working in the implementation of this auto-adaptive algorithm to provide QoS requirements in the Second Generation PONs (NG-PON2), called TWDM-PON, which combine a more complex architecture based on a simultaneous time and wavelength division multiplexing [49-50].

ACKNOWLEDGMENT

This work has been funded by Spanish Ministry of Science and Innovation (TEC2014-53071-C3-2-P and TEC2015-71932-REDT).

REFERENCES

[1] K. Ahl, FTTH Council Europe. Creating Connected Continents. Bringing people together through FTTH. http://www.ftthcouncil.eu/documents/Publications/TLA6.pdf, Accessed May 2016.

[2] D. Murayama, N. Oota, K. Suzuki and N. Yoshimoto, "Low Latency Dynamic Bandwidth Allocation for 100 km Long-Reach EPONs," *OSA Journal of Optical Communications and Networking*, vol. 1, pp. 48-55, January 2013.

[3] L. Jiunn-Ru and C. Wen-Ping, "High utilization dynamic bandwidth allocation algorithm based on sorting report messages with additive-polling thresholds in EPONs," *Optical Switching and Networking*, vol. 18, pp. 81-95, November 2015.

[4] Ö.Can Turna, A. Muhammed Ali, A. Halim Zaim and T. Atmaca, "A new dynamic bandwidth allocation algorithm based on online–offline mode for EPON," *Optical Switching and Networking*, vol. 152, pp. 29-43, June 2014.

[5] Y. Yin and G.-S. Poo, "User-oriented hierarchical bandwidth scheduling for ethernet passive optical networks," *Computer Communications*, vol. 33, pp. 965-975, May 2010.

[6] B. Kantarci and H. Mouftah, "Two-stage report generation in long-reach EPON for enhanced delay performance," *Computer Communications*, vol. 36, pp. 1570-1580, August 2013.

[7] S. Herrería-Alonso, M. Rodriguez Pérez, M. Fernández Veiga and C. López García, "On the Use of the Doze Mode to Reduce Power Consumption in EPON System," *IEEE/OSA Journal of Lightwave Technology*, vol. 32, pp. 285-292, January 2014.

[8] H. I-Shyan, S. Zen-Der, K. Liang-Yu and C. Chun-Che, "A novel early DBA mechanism with prediction-based fair excessive bandwidth allocation scheme in EPON," *Computer Communications*, vol. 31, pp. 1814-1823, April 2007.

[9] M. Mcgarry, M. Reisslein and M. Maier, "Ethernet passive optical network architectures and dynamic bandwidth allocation algorithms," *IEEE Communication Survey Tutorials*, vol. 10, pp. 46-60, October 2008.

[10] A. Nikoukar, I. Hwang, A.T. Liem and C. Wang, "QoS-aware energy-efficient mechanism for sleeping mode ONUs in enhanced EPON," *Photonic Network Communications*, vol. 30, pp. 59-70, August 2015.

[11] A. Dixit, B. Lanoo, G. Das, D. Colle, M. Pickavet and P. Demeester, "Dynamic Bandwidth AllocationWith SLA Awareness for QoS in Ethernet Passive Optical Network," *IEEE/OSA Journal of Optical Communications and Networking*, vol. 5, pp. 240-253, March 2013.

[12] T. Berisa, A. Bazant and V. Mikac, "Bandwidth and delay guaranteed polling with adaptive cycle time (BDGPACT): a scheme for providing bandwidth and delay guarantees in passive optical networks," *OSA Journal of Optical Communications and Networking*, vol. 8, pp. 337-345, April 2009.

[13] M. Ma, Y. Zhu and T.H. Cheng, "A bandwidth guaranteed polling MAC protocol for Ethernet Passive optical networks," *In Proceedings of the IEEE INFOCOM'03*, 2003, USA, pp. 22-31.

[14] F.T. An, H. Bae, Y. Hsueh, M. Rogge, L. Kazovsky and K. Kim, "A new media access control protocol guaranteeing fairness among users in Ethernet-based passive optical networks," *In Proceedings of the IEEE OFC'03*, 2003, USA, pp. 134-135.

[15] C. Assi, Y. Ye, S. Sudhir, S. Dixit and M. Ali, "Dynamic bandwidth allocation for quality-of-service over Ethernet PONs," *IEEE Journal on Selected Areas Communications*, vol. 21, pp. 1467-1477, November 2003.

[16] C.-H. Chang, N Merayo, P. Kourtessis, J. Senior and R.M. Lorenzo, "Full-service MAC protocol for metro-reach GPONs," *IEEE/OSA Journal of Lightwave Technology*, vol. 28, pp. 1016-1022, April 2010.

[17] K.H. Ang, G. Chong and Y. Li, "PID Control System Analysis, Design and Technology," *IEEE Transactions on Control Systems Technology*, vol. 13, pp. 559-576, July 2005.

[18] K.J. Aström and T. Hägglund. Advanced PID control. Research Triangle Park, NC: ISA-The Instrumentation, Systems, and Automation Society, 2006.

[19] M.I. Menhas, I. Wang, M. Fei and H. Pan, "Comparative performance analysis of various binary coded PSO algorithms in multivariable PID controller design," *Expert Systems with Applications*, vol. 39, pp. 4390-4401, March 2012.

[20] V. Hultmann and L. dos Santos Coelho, "Tuning of PID controller based on a multiobjective genetic algorithm applied to a robotic manipulator," *Expert Systems with Applications*, vol. 39, pp. 8968-8974, April 2012.

[21] A. Bianciotto, B.J. Puttnam, B. Thomsen and P. Bayvel, "Optimization of Wavelength-Locking Loops for Fast Tunable Laser Stabilization in Dynamic Optical Networks," *IEEE/OSA Journal of Lightwave Technology*, vol. 27, pp. 2117-2124, June 2009.

[22] T. Tachibana, K. Kogiso and K. Sugimoto, "Dynamic Management of Computing and Network Resources with PID Control in Optical Grid Networks," *In Proceedings of the IEEE International Conference on Communications (ICC'08)*, 2008, Pekin, China, pp. 396-400.

[23] T. Tachibana and K. Sugimoto, "Light-path Establishment with PID control for Effective Wavelength utilization in all-optical Wavelength-Division Multiplexing Networks," *IEEE/OSA Journal of Optical Communications and Networking*, vol. 8, pp. 383-392, December 2008.

[24] C.K. Chen, H. Kuo and J. Yan, "GA-based PID active queue management control design for a class of TCP communication networks," *Expert Systems with Applications*, vol. 36, pp. 1903-1913, March 2009.

[25] T. Jiménez, N. Merayo, P. Fernández, R.J. Durán, I. de Miguel, R.M. Lorenzo and E.J. Abril, "Implementation of a PID Controller for the Bandwidth Assignment in Long-Reach PONs," *OSA Journal of Optical Communications and Networking*, vol. 4, pp. 392-401, May 2012.

[26] T. Jiménez, N. Merayo, P. Fernández, R.J. Durán, I. de Miguel, R.M. Lorenzo and E.J. Abril, "A PID-based algorithm to guarantee QoS delay requirements in LR-PONs," *Optical Switching and Networking*, vol. 14, pp. 78-92, August 2014.

[27] R. Rodriguez, A. Rodrigues and L. dos Santos, "Computational intelligence approach to PID controller design using the universal model," *Information Sciences*, vol. 180, pp. 3980–3991, October 2010.

[28] M.A. Keshari and A. Mehetab, Study of the design and tuning methods of pid controller based on fuzzy logic and genetic algorithm, Department of Electronics and Communication Engineering National Institute of Technology, Thesis, Bachelor in Electronics, 2012.

[29] C.J. Lakhmi and N.M. Martin, *Fusion of Neural Networks, Fuzzy Sets, and Genetic Algorithms: Industrial Applications, 1st*, CRC Press, 1998.

[30] Y. Yuan, C. Changuan, Z. Zhou, X. Huang and X. Yang, "Design of a single-input fuzzy PID controller based on genetic optimization scheme for DC-DC buck converter," *In Proceedings of the International Symposion on Next-Generation Electronics (ISNE)* , 2015, Taiwan, pp. 1-4.

[31] K.M. Hussain, R.A. Rajendran, M.S. Kumar and S.M. Giriraj Kumar, "Comparison of PID Controller Tuning Methods with Genetic Algorithm for FOPTD System," *International Journal of Engineering Research and Applications*, vol. 4, pp. 308-314, February 2014.

[32] H. Ali and S. Wadhwani, "Intelligent PID Controller Tuning for Higher Order Process system," *International Journal of u- and e-Service, Science and Technology*, vol. 8, pp. 323-330, 2015.

[33] M.J. Neath, A.K. Swain, U.K. Madawala and D.J. Thrimawithana, "An optimal PID controller for a bidirectional inductive power transfer system using multiobjective genetic algorithm," *IEEE Transactions on Power Electronics*, vol. 29, pp. 1523-1530, May 2013.

[34] V. Kozeny, "Genetic algorithms for credit scoring: Alternative fitness function performance comparison," *Expert Systems with Applications*, vol. 42, pp. 2998-3004, April 2015.

[35] M. Rajabi-Bahaabadi, A. Shariat-Mohayman, M. Babaei and C. Wook, "Multi-objective path finding in stochastic time-dependent road networks using non-dominated sorting genetic algorithm," *Expert Systems with Applications*, vol. 42, pp. 5056-5064, July 2015.

[36] T. Jiménez, N. Merayo, P. Fernández, R.J. Durán, I. de Miguel, R.M. Lorenzo and E.J. Abril, "An auto-tuning PID control system based on genetic algorithms to provide delay guarantees in Passive Optical Networks," *Expert Systems with Applications*, vol. 42, pp. 9211-9220, December 2015.

[37] H. Xu, J. Lai, Z. Yu and J. Liu," Based on Neural Network PID Controller Design and Simulation," *In Proceedings of the 2012 2nd International Conference on Computer and Information Application (ICCIA 2012)*, Taiyuan, China, pp. 844-866.

[38] L. Luoren and L. Jinling, "Research of PID Control Algorithm Based on Neural Network," *In Proceedings of Energy Procedia*, 2011, pp. 6988-6993.

[39] W. Lu, "The PID Controller Based on the Artificial Neural Network and the Differential Evolution Algorithm," *Journal of Computers*, vol. 7, pp. 2368-2375, October 2012.

[40] B. Yang, J. Chang, H. Su, J. Peng, S. Zhang, S. Guo, L. Zhang, C. Srinivasakannan, Z. Liu, Z. Li and Z. Cao, "Self-Adaptive PID Controller Integrated with RBFNN Identification Applied to Microwave Drying Process," *Journal of Convergence Information Technology*, vol. 8, pp. 779-783, January 2013.

[41] S. Shoujun and L. Weiguo, *Application of Improved PID Controller in Motor Drive System, PID Control, Implementation and Tuning*, InTech, 2011, Available from: http://www.intechopen.com/books/pid-control-implementation-and-tuning/application-of-improved-pid-controller-in-motor-drive-system.

[42] K. Vikas, G. Prerna and A.P. Mittal, "ANN based self-tuned PID like adaptive controller design for high performance PMSM position control," *Expert Systems with Applications*, vol. 41, pp. 7995-8002, December 2014.

[43] J. Peng and R. Dubay, "Identification and adaptive neural network control of a DC motor system with dead-zone characteristics," *ISA Transactions*, vol. 50, pp. 588-598, July 2011.

[44] S. Haykin, *Neural Networks and Learning Machines (3rd Edition)*, Prentice Hall, 2009.

[45] K.J. Aström and T. Hägglund, *PID Controllers: Theory, Design and Tuning*. 2nd Ed. Research Triangle Park, NC, Instrumet Soc. Amer, 1995.

[46] G. Kramer, B. Mukherjee and G. Pesavento, "Interleaved Polling with Adaptive Cycle Time (IPACT): A Dynamic Bandwidth Distribution Scheme in an Optical Access Network," *Photonic Network Communications*, vol. 4, pp. 89-107, January 2002.

[47] Available: OMNeT++ Discrete Event Simulator-Home. (2015). http:// www.omnet.com, Accessed.

[48] G. Kramer (2001). [Tutorial] Synthetic self-similar traffic generation, Available: http://glenkramer.com/ucdavis/trf_research.html.

[49] F.G. Effenberger, "PON Resilience," *OSA Journal of Optical Communications and Networking*, vol. 7, pp. 547-552, February 2015.

[50] N. Nakamura (2013). [Tutorial] NG-PON2 Technologies. NTT Access Network Service Systems Laboratories, NTT Corporation.

Noemi Merayo received the Telecommunication Engineer degree in engineering from the Valladolid University, Spain, in February of 2004 and the Ph.D. degree in the Optical Communication Group at the University of Valladolid, in July 2009. Since 2005 she has worked as a Junior Lecturer at the University of Valladolid. He has also been a Visiting Research Fellow at the University of Hertfordshire (London), working in the Optical Networks Group, Science and Technology Research Institute (STRI). Recently, she has been a postdoctoral research in the group of "Transmisiones Ópticas de Banda Ancha (TOyBA)" of the University of Zaragoza. Her doctoral research focused on the design and performance evaluation of optical networks, especially passive optical networks.

D. Juárez studied Telecommunication Engineering at the University of Valladolid, Spain. He was working for the Optical Communications Group of the University of Valladolid from 2015 to 2016, where he has developed algorithms for resource management in PON and NGPON2 networks.

Juan Carlos Aguado received the Telecommunication Engineer and Ph.D. degrees from the University of Valladolid, Spain, in 1997 and 2005, respectively. He has worked as a Junior Lecturer at the University of Valladolid since 1998. His current research interests include the design and evaluation of cognitive methods applied to physical-layer modeling and traffic routing in heterogeneous optical networks.

I. de Miguel received his Telecommunication Engineer degree in 1997, and his Ph.D. degree in 2002, both from the University of Valladolid, Spain. Since 1997 he has worked as a Junior Lecturer at the University of Valladolid. He has also been a Visiting Research Fellow at University College London (UCL), working in the Optical Networks Group. His research interests are the design and performance evaluation of optical networks, especially hybrid optical networks, as well as IP over WDM. Dr. de Miguel is the recipient of the Nortel Networks Prize to the best PhD Thesis on Optical Internet in 2002, awarded by the Spanish Institute and Association of Telecommunication Engineers (COIT/AEIT). He also received the 1997 Innovation and Development Regional Prize for his Graduation Project.

Ramón J. Durán was born in Cáceres, Spain, in 1978. He received his Telecommunication Engineer degree in 2002 and his PhD degree in 2008, both from the University of Valladolid, Spain. Since 2002, he has worked as a Junior Lecturer at the University of Valladolid and currently he is also Deputy Director of the Faculty of Telecommunication Engineering. His current research focuses on the design and performance evaluation of cognitive heterogeneous optical networks. Dr. Durán is the author of more than 60 papers in international journals and conferences.

P. Fernández obtained the Telecommunication Engineer degree from Universidad Politécnica de Cataluña, Barcelona, Spain, in 1997 and the PhD degree in 2004 from University of Valladolid. Since 1999 she has worked as a Junior Lecturer at the University of Valladolid. Her research interests are passive optical networks and fiber-optic communications components. Dr. Fernández is the author of more than 40 papers in international journals and conferences. Currently, she is a Professor and Head of the Faculty of Telecommunication Engineering at the University of Valladolid.

R. M. Lorenzo received his Telecommunication Engineer and PhD degrees from the University of Valladolid, Spain, in 1996 and 1999, respectively. From 1996 to 2000, he was a Junior Lecturer at the University of Valladolid, and joined the Optical Communications Group. Since 2000, he has been a Lecturer. His research interests include Integrated Optics, Optical Communication Systems and Optical Networks.

E. J. Abril received his Telecommunication Engineer and PhD degrees from Universidad Politécnica de Madrid, Spain, in 1985 and 1987, respectively. From 1984 to 1986, he was a Research Assistant at Universidad Politécnica de Madrid, becoming Lecturer in 1987. Since 1995, he has been Full Professor at University of Valladolid, Spain, where he founded the Optical Communications Group. His research interests include Integrated Optics, Optical Communication Systems and Optical Networks. Prof. Abril is the author of more than 100 papers in international journals and conferences.