IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received November 19, 2020, accepted November 27, 2020, date of publication December 1, 2020,
date of current version December 14, 2020.

*Digital Object Identifier 10.1109/ACCESS.2020.3041690*

# Predictive Traffic Control and Differentiation on Smart Grid Neighborhood Area Networks

**JUAN PABLO ASTUDILLO LEÓN[1], FRANCISCO J. RICO-NOVELLA[2], AND LUIS J. DE LA CRUZ LLOPIS[2]**

[1]Coordinación de Investigación e Innovación, ABREC, Quito 170528, Ecuador
[2]Department of Network Engineering, Universitat Politècnica de Catalunya (UPC), 08034 Barcelona, Spain

Corresponding author: Juan Pablo Astudillo León (juan.astudillo@abrec.org)

This work was supported by the Spanish Government under Research Project "sMArt Grid using Open Source intelligence (MAGOS)" under Grant TEC2017-84197-C4-3-R.

**ABSTRACT** Smart Grid (SG) networks include an associated data network for the transmission and reception of control data related to the electric power supply service. A subset of this data network is the SG Neighborhood Area Network (SG NAN), whose objective is to interconnect the subscribers' homes with the supplier control center. The data flows transmitted through these SG NANs belong to different applications, giving rise to the need for different quality of service requirements. Additionally, other subscriber appliances could use this network to communicate over the Internet. To avoid network congestion, as well as to differentiate the quality of service (QoS) received by the different data flows, a congestion control mechanism with traffic differentiation capabilities is required. The main contribution of this work is the proposal of a new congestion control mechanism based on machine learning techniques to try to guarantee the different QoS requirements to the different data flows. A main problem when applying machine learning techniques is the need for datasets to be used in the training steps. In this sense, a second contribution of this article is the proposal of a method to generate such datasets by means of simulation techniques. The proposed mechanism is then evaluated in the context of a wireless SG NAN. The nodes of this network are the subscriber's smart meters, which in turn perform the function of concentrating the data traffic sent and received by the rest of the home appliances. Besides, different machine learning classification methods are taken into account. The evaluation carried out shows significant improvements in terms of network throughput, transit time, and quality of service differentiation. Finally, the computational cost of the algorithms used in this proposal has also been evaluated, using real low-cost IoT hardware platforms.

**INDEX TERMS** Smart grid, neighborhood area networks, machine learning, deep learning, congestion control.

## I. INTRODUCTION

The traditional electrical energy distribution networks have evolved to the so-called Smart Grids (SG), in which an associated data communication network is available to complement the traditional electrical infrastructure. On the one hand, the electricity distribution infrastructure is responsible for generating, transporting, and distributing this valuable resource to the subscribers [1]. On the other hand, the data communication network is used to improve the service offered to the subscribers, as well as to provide some feedback about the operation status to the control center [2]. The associated data network comprises three sub-networks: the Home Area Network (HAN), the Neighborhood Area Network (NAN), and the Wide Area Network (WAN). The HAN sub-network is implemented in the subscriber's home, interconnecting all the available appliances together with the smart meter, which in turn performs the functions of data concentrator towards the NAN sub-network. Thus, the NAN sub-network interconnects the subscribers' homes (within a limited geographical area), forwarding (receiving) the data to (from) the control center through a gateway connected to the WAN network. Different technologies can be used for each of these sub-networks.

The availability of this new infrastructure for data transmission gives rise to the possibility of offering new services to the subscribers of the electric companies. These services can be related to energy consumption, as well as to a wide set of new applications from which both the supplier companies and the subscribers could make a profit. Thereby, in an

The associate editor coordinating the review of this manuscript and approving it for publication was Giambattista Gruosso.

environment in which the number of IoT devices is constantly growing, many new home appliances will be able to transmit (and receive) their data flows across the NAN networks. However, an uncontrolled data transmission rate by such applications could lead to network congestion situations. Some effects of network congestion are packet losses and buffering delays. These effects can drastically reduce the network performance of the NAN, especially for those applications with a higher quality of service (QoS) requirements. Therefore, a congestion control mechanism is mandatory to achieve adequate QoS provision. Furthermore, knowing that traffic flows with different QoS requirements will coexist in the network, this mechanism should be able to distinguish between those flows, implementing a differentiated resource allocation strategy and thus guaranteeing the different degrees of quality demanded.

The goal and main contribution of this work is the proposal and evaluation of a new congestion control mechanism for SG NANs, based on machine learning techniques and able to differentiate between traffic flows with different QoS requirements. To achieve a good performance, most machine learning techniques require large datasets, which are used during the training phase of the selected algorithms. In fact, in many situations, the unavailability of these datasets makes the application of these techniques unfeasible. Thereby, a second contribution of this work is the proposal of a mechanism to obtain the required datasets by means of network simulations. Among the available machine learning techniques, in this article, we detail the results obtained with two of them: decision trees and feedforward neural networks. The complete system has been evaluated also through network simulations, using the ns-3 simulator [3]. The selected network technology is the IEEE 802.11s Wireless Mesh Network [4].

To deal with the different QoS levels, the data packets are classified into different categories (or traffic types), depending on the application to which they belong. A different priority is assigned to each category, giving preference to the transmission of the highest priority packets. Thus, data packets belonging to critical applications of the Smart Grid are assigned to higher categories, while those belonging to less relevant applications are assigned to lower categories.

The application of the mechanism is done in a distributed way. Every time a source node must transmit a packet belonging to a certain flow, it will apply a decision algorithm to predict, as a function of the current network status (utilization factors of the wireless channels and buffer occupancy of the network nodes), if the packet will be correctly delivered (on time) to its destination, without compromising the correct transmission of higher category packets. If these conditions are met, the packet will be transmitted. Otherwise, it will be discarded, avoiding a waste of network resources.

The rest of the paper is organized as follows. The following section presents some related work. The proposed congestion control mechanism is described in detail in Section III. Next, the performance evaluation is presented in the following two sections: section IV presents the obtained results with a decision tree classifier, while section V evaluates the performance of a neural network-based mechanism. Finally, Section VI concludes the paper.

## II. RELATED WORK

The improvement of the performance offered by wireless SG NANs is a field of work that has attracted the attention of several research groups. Some improvements have been proposed in relation to different fields, such as network topology, routing algorithms and protocols, channel allocation, and congestion control.

Regarding the network topology, in [5] a multigate communication network, based on IEEE 802.11s [6], together with a mechanism to implement the load balance between gateways is proposed. The possibility of multi-channel transmission is also taken into account, as well as real-time traffic scheduling.

Improving the capabilities offered by routing algorithms and protocols is the focus of multiple papers. For instance, a comparison between Optimized Link State Protocol (OLSR) [7] and Hybrid Wireless Mesh Protocol (HWMP) [6] is presented in [8]. For both protocols, different variations have been proposed to improve their performance, modifying some of their procedures or the metric used to obtain the cost of the channels [9]–[14]. A more detailed summary of these works, together with a proposal that combines multipath routing with multichannel assignment can be found in [15].

Congestion control involves a set of techniques to detect and correct a possible overuse of the network resources which leads to a performance degradation. Basically, an unregulated traffic generation rate by some source nodes, or even their geographical position or the network size, can give rise to a partial or complete congestion situation. Most Smart Grid applications have strong security and reliability requirements, and thus, congestion control mechanisms are mandatory to provide the necessary QoS. In this sense, a generic proposal for IEEE 802.11s networks, also based on the modification of the routing protocol, can be found in [16].

In the context of Wireless Mesh Networks, most of the congestion control mechanisms are focused on the Transmission Control Protocol (TCP) [16]. Taking into account that a part of the traffic to be transmitted over the SG NANs uses this transport protocol, all the obtained improvements are undoubtedly of extreme importance for the correct operation of the network. A major problem when using TCP is that it recognizes and handles all packet losses as network congestion [17]. Hence, performing an unjustified congestion control, when a packet loss is due to another reason, can degrade the end-to-end throughput. With this goal in mind, several researchers are focused on the design of learning algorithms that differentiate between network congestion and transmission errors [18]–[22]. On the other hand, a multitude of new and future IoT applications are and will be transmitted without the need for an end-to-end connection, such as those established by the TCP protocol. These applications will then make use of the lighter User Datagram Protocol (UDP).

Nowadays, the application of machine learning techniques to improve the performance in wireless communication networks is a hot topic. In this sense, numerous recent works can be found. A very good survey can be found in [23].

In [24] and [25], the authors present a new forwarding mechanism, based on a learning algorithm, by which each node dynamically selects its next hop with the highest potential bandwidth. In their proposal, when a node must forward a data packet, the node learns the historical changes of bandwidth and then predicts the possible future bandwidths of the links with its neighbor nodes. Besides, in order to avoid flooding problems, they also propose a geographical algorithm to let the source node to figure out the best forwarding region. They evaluate their mechanism in a simulated noiseless radio network environment using the MATLAB platform. Another recent work that deals with the routing mechanism, focused in this case on wireless sensors networks, is presented in [26], where authors propose a combination of algorithms which offers improvements in terms of network lifetime, coverage, and robustness, as well as a reduction in the energy consumption of the overall network system. On the other hand, a study of some parameters that could be of relevance for a congestion control mechanism, together with a first evaluation of some machine learning techniques, appears in [27]. The authors state that transmission energy, queue size, distance between the transmitter and the receiver, transmission rate, and channel cost are relevant parameters to design a congestion control decision mechanism. It is not proposed a method to obtain datasets for algorithms training, nor the operational congestion control mechanism, for a specific wireless network technology. Traffic differentiation is not taken into account either.

We have previously proposed two alternatives for congestion control on SG NANs. One of them is implemented on the intermediate nodes and takes into account the possibility of being in emergency situations [28]. The second one delves into the need to offer an equitable distribution of network resources among all traffic generating nodes [29]. In this work, a new congestion control mechanism that uses machine learning and deep learning techniques is presented. The contribution is focused on UDP based applications and takes into account the need of traffic differentiation, providing different levels of QoS to the traffic flows depending on their needs and relevance. We propose, implement, and evaluate some procedures to obtain useful datasets to train the machine learning algorithms when traffic differentiation is mandatory. Besides, we have built a complete framework to evaluate two different classification mechanisms (based on decision trees and neural networks techniques). A study of the computational cost of the proposed mechanism, and its implementation feasibility on current low-cost hardware platforms, is also included.

## III. PROPOSED SOLUTION

The application scenario for the proposed congestion control mechanism is shown in Figure 1, where the different subnets that make up the Smart Grid data network can be seen:

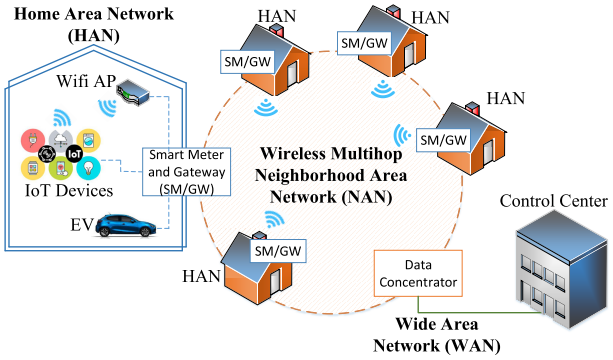



**FIGURE 1.** Smart Grid data network.

Home Area Network (HAN), Neighborhood Area Network (NAN), and Wide Area Network (WAN). Within each HAN, different data generators/receivers (computers, IoT sensors and actuators, electrical vehicles, ...) can coexist with the devices of the electrical network. The smart meter (SM-GW) can perform the gateway functions, to concentrate all these data flows to/from the NAN network. Thus, different data traffics will coexist within the NAN, some of them being of critical importance for the correct operation of the Smart Grid. Therefore, it is highly recommended to include traffic differentiation techniques in the congestion control mechanism.

The proposed mechanism is based on machine learning techniques, and therefore it is necessary to distinguish two main steps:

- Training of the selected algorithms: In this step, previous datasets are needed, in which the correct (or not) reception of the data packets, as a function of the network state (channels utilization, occupation of buffers, presence of packets with different priority), has been detected. As it has been previously mentioned, in this work a method to obtain these datasets, by means of network simulation techniques, is proposed.
- Operation phase: In this phase, the NAN source nodes (that is, the SM-GWs) are able to predict whether a new packet to be transmitted will be correctly (on time) received or not. If the packet will not be received correctly, or if its transmission will make it impossible to provide the required QoS to a higher priority packet, it will be directly discarded at the source node.

Figure 2 shows the road-map used for building our machine learning-based congestion control mechanism. The entire process can be separated into three steps. First, the necessary datasets are generated. Second, the training of the machine learning algorithms is carried out. Third, the mechanism can be put into operation on the SG NAN.

In the first step, data collection is mandatory since there is not an available dataset for NANs that contemplates different network loads and traffic differentiation. This way, a method has been designed to create a dataset according to our problem. To this end, numerous network simulations have been carried out with the help of the ns-3 simulator. In these

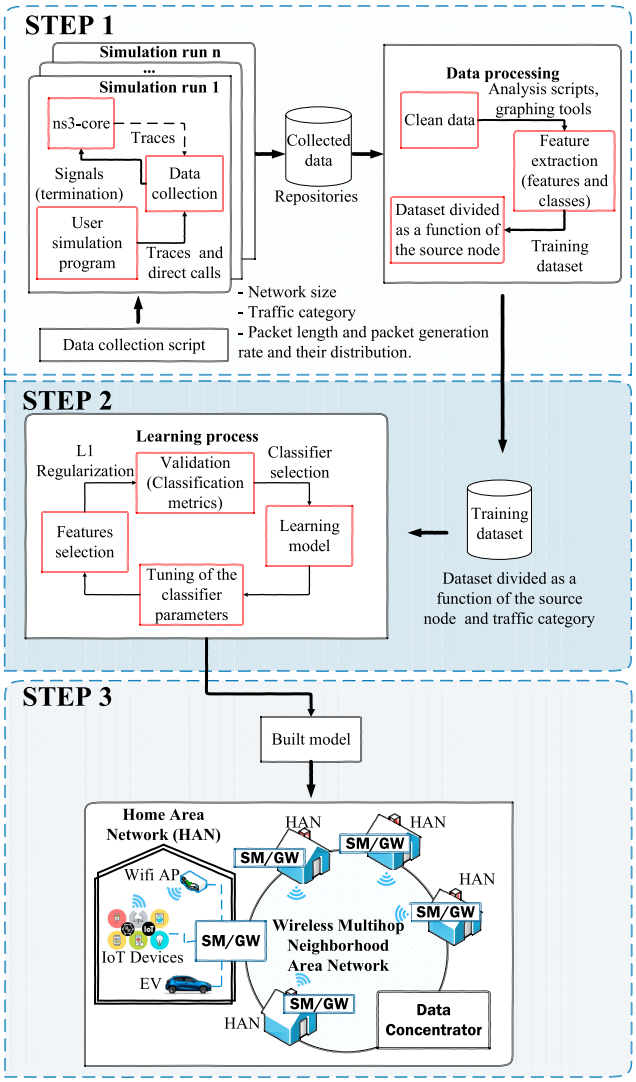



**FIGURE 2.** Development and operation of the congestion control mechanism.

simulations, the relevant network state parameters found by every data packet in its way from its source to its destination, together with the received service, in terms of network transit time, are captured and stored. For our purposes, the channel utilization factor and the buffer occupancy are good descriptors of the network load state. To get a complete description of the network behavior, different network load values are generated by the different SG NAN applications. Once the data have been collected, they must be processed. The objective here is to clean and organize those data in a structured way, which can be used to train the machine learning algorithms.

In the next two subsections, the data collection and processing steps, will be deeply covered.

### A. DATA COLLECTION

As previously stated, to build our proposal the first step we must do is to obtain a large amount of data that describes all possible network load situations, together with the service received by the data packets. In this work, the scenarios used for data collection consist of communication networks

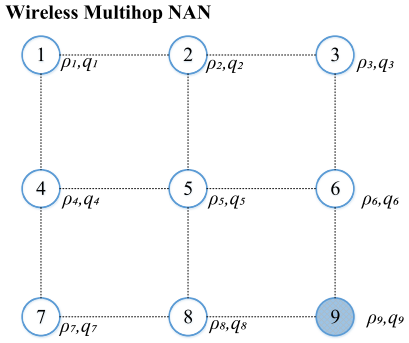



**FIGURE 3.** Data collection scenario.

**TABLE 1.** Main simulation parameters used for data collection.

| Description | Value |
|---|---|
| Network simulator | ns-3.28 |
| Distance between nodes | 80 m |
| Simulation time | 170 s |
| Transport layer | UDP |
| Routing protocol | HWMP |
| Routing metric | Airtime Link Metric (ALM) |
| Maximum queue size | 100 packets |
| Inter arrival time average and PDF | from 0.01s to 0.3 s (Exponential) |
| Packet length average and PDF | 200 Bytes (Exponential) |

arranged in a grid topology, in which each node represents a home connected to the SG NAN through the SM/GW. The data transmitted by all these nodes are destined to another data concentrator, which forwards those data packets to the control center through a WAN network. Figure 3 shows a SG NAN scenario consisting of eight SM/GWs (nodes 1 to 8) and one data concentrator (node 9).

The data collection process can be carried out in two different ways: offline or online. In our context, an offline data collection has been performed. Among others, the channel utilization factor and the number of queued packets ($\rho_i$ and $q_i$ respectively in Figure 3, with $i = 1 \ldots N$ and $N$ representing the total number of nodes) will be stored. Besides, the offline data collection can be done through active or passive monitoring. For our purposes, a passive monitoring can be performed, in order not to introduce additional control traffic such as probe packets [17].

Several simulation runs were performed to build the training dataset. The main simulation parameters are summarized in Table 1. As shown, a variable size (specifically, a truncated exponential distribution) has been chosen for the data packets. For the inter-arrival time, the same criteria has been followed, and so packets are generated with different and random time between each other. In order to obtain the full range of values of the channel utilization factor and buffer occupation, a variable packet inter-arrival time has been configured.

Figure 4 shows an example about the way the data collection has been done. The figure represents the simplest case, that is, when just one traffic type or category (*cat*) is considered. In this example, just one data packet that belongs to traffic category 1 is transmitted, from SM/GW 1 towards the data concentrator. As it can be seen, when the packet travels through the network hop by hop, the current channel

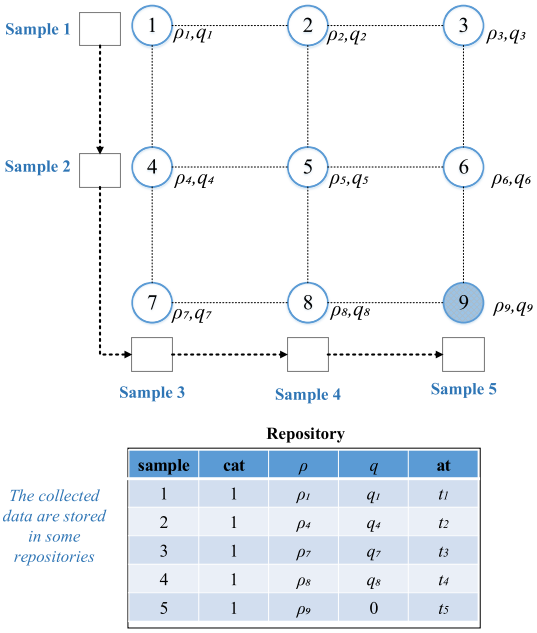



**FIGURE 4.** Data collection.

utilization factor and the buffer occupancy are stored in the repository. Besides, many more events are stored such as the arrival time (*at*), the packet category and a packet identifier (*pid*) among others.

### B. DATA PROCESSING AND FEATURE EXTRACTION

The collected data are stored in some repositories in an unstructured way. Therefore, they have to be processed in order to organize and to define the features and labeled classes, which will be the inputs and outputs to train the machine learning algorithms. This process is often called as feature extraction. For this purpose, the *pandas* tool [30]–[32] has been chosen to process, create, and organize the variables from the collected data in a structured way.

First, the data are processed to have one meaningful sample per packet. That is, each sample will contain the network parameters perceived by one packet in its way hop by hop through the whole network (see Table 2). Second, the features and classes are extracted from these variables. Figure 5 presents the resulting training dataset with the inputs and the outputs for the machine learning algorithms. Each training sample represents the parameters for a unique packet transmitted over the network. The features are the channel utilization factor and the buffer occupancy per node ($\rho_i$ and $q_i$), while the class represents whether the packet has been successfully received or not ($r$). To decide if a packet has been successfully received, its network transit time is considered. This way, if that time is lower than the maximum allowed according to the required quality of service, the packet is considered successfully received. Otherwise, the transmission is considered unsuccessful.

With these guidelines on how to build a representative dataset for our targeted problem, several data files have been generated from scratch. Besides, different traffic patterns

**TABLE 2.** Variables used to represent the unstructured data.

| Variable | Description | Value |
|---|---|---|
| pid | Packet identifier | unique integer value |
| source | Node that generates the packet | |
| destination | Destination node of the packet | |
| cat | Traffic type or category | [1, 2, 3, 4] |
| $\rho_i$ | Channel utilization factor (node $i$) | [0 to 1] |
| $q_i$ | Number of queued packets (node $i$) | [0 to 100] |
| ntt | Packet network transit time | |
| $r$ | Boolean value that represents if the packet was received or not in the destination | [0, 1] |

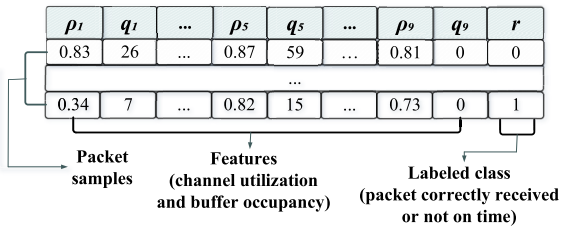



**FIGURE 5.** Features and labels for the training and testing dataset.

**TABLE 3.** SG applications transmitted over the smart grid [33].

| Traffic category | Applications |
|---|---|
| 1 | Demand Response, Outage Management |
| 2 | Video surveillance, Overhead Transmission Line Monitoring, Substation Automation systems (SASs) |
| 3 | Home Energy Managament (HEM), Electric Vehicles (EVs) Charging |
| 4 | Meter Data Management |

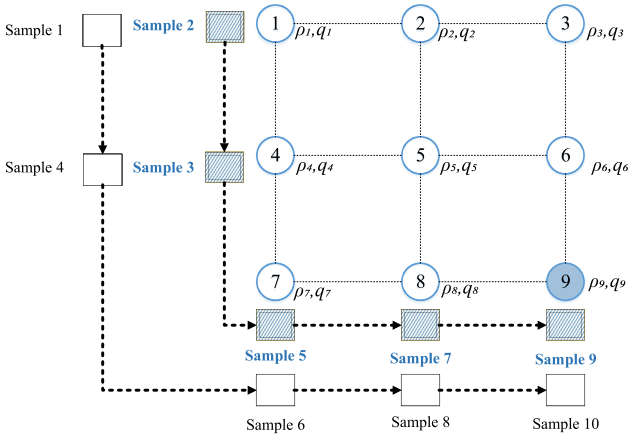
have been considered to generate the datasets, with the aim of collecting a large amount of historical data and with different values of the selected features.

### C. TRAFFIC DIFFERENTIATION

In this subsection, the previous procedures will be extended and generalized to be used when the QoS provided to different traffic flows must be differentiated. For instance, Table 3 shows some different Smart Grid applications which are grouped according to their relevance. Thus, here we consider different traffic priorities, assigning the higher priority to traffic category 1.

Therefore, a new dataset that includes different types of traffic has to be built. In order to explain the methodology used to create a dataset with different traffic types, a simple example will again be used, which is shown in Figure 6. In this example, two packets that belong to two different traffic categories (1 and 2) are considered.

The two packets are transmitted at the same time, and we have included a common packet identifier (*cpid*) in their header. Basically, the objective of having a common identifier is to analyze how the transmission of data belonging to traffic category 2 penalize the traffic category 1 in terms of successful packet reception and network packet transit time. In the example, once the two packets have arrived to node 9, we get the values of their network transit times ($t_9$ for the




**Repository**

| sample | cat | cpid | $\rho$ | $q$ | at |
|---|---|---|---|---|---|
| 1 | 1 | 1 | $\rho_1$ | $q_1$ | $t_1$ |
| 2 | 2 | 1 | $\rho_1$ | $q_1$ | $t_2$ |
| 3 | 2 | 1 | $\rho_4$ | $q_4$ | $t_3$ |
| 4 | 1 | 1 | $\rho_4$ | $q_4$ | $t_4$ |
| 5 | 2 | 1 | $\rho_7$ | $q_7$ | $t_5$ |
| 6 | 1 | 1 | $\rho_7$ | $q_7$ | $t_6$ |
| 7 | 2 | 1 | $\rho_8$ | $q_8$ | $t_7$ |
| 8 | 1 | 1 | $\rho_8$ | $q_8$ | $t_8$ |
| 9 | 2 | 1 | $\rho_9$ | 0 | $t_9$ |
| 10 | 1 | 1 | $\rho_9$ | 0 | $t_{10}$ |

**FIGURE 6.** Data collection used to build a training dataset considering different traffic types.

**TABLE 4.** Required QoS.

| Cat | Network transit time |
|---|---|
| 1 | 50 ms |
| 2 | 100 ms |

category 2 packet, and $t_{10}$ for the category 1 packet). If the value obtained for the packet belonging to category 1 is lower than the required (see an example in Table 4), the packet is considered successfully transmitted. Otherwise, the packet transmission is unsuccessful and so we consider that the transmission of the lower category packet has been detrimental, and therefore we mark it as not selectable for transmission. That is, in the next step of the process, the machine learning algorithms will be trained to not transmit the category 2 packets in this network situation. Note that this is an approximation, and maybe the transmission of the packet belonging to category 2 was not the only cause of the incorrect reception of the packet belonging to category 1.

The same approach is applied when transmitting more than two traffic categories. Besides, in the previous example, the two traffic types are transmitted at the same rate. However, in the real data collection implemented, different combinations of traffic patterns were considered. For instance, a combination consist of keeping constant the rate of traffic type 1, while increasing the rate of traffic type 2, and observing how the service offered to traffic type 1 is reduced. We always keep common identifiers to be able to infer the influence of the

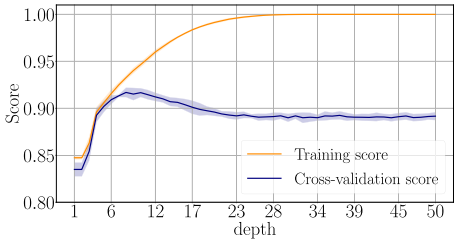



**FIGURE 7.** Validation curve.

lower categories traffics in the service received by the higher ones.

Once we have created the needed datasets, the next step is using them to train the classification algorithms and evaluate the resulting performance. In the next sections, two different classifiers will be considered: decision trees and neural networks. In order to avoid underfitting and overfitting problems, the size of the different datasets has been chosen by inspecting the *learning curve*, which relates the classification accuracy to the number of samples in the dataset.

## IV. DECISION TREE BASED CLASSIFICATION

### A. ONE TRAFFIC CATEGORY

Decision trees belong to the set of supervised learning algorithms. They are used to solve classification and regression problems [34]. Since the proposed congestion control mechanism is aimed to solve a binary classification problem, this type of classifier represents a very good option, with a low computational load, which is very interesting if the network nodes are based on low-cost hardware architectures. With this classifier, the prediction is obtained by means of a set of if-then-else decision rules [35].

To get good performance from machine learning algorithms, some hyperparameters must be adjusted. To this end, there are basic techniques such as Grid Search or Random Search, and more sophisticated techniques such as Bayesian Optimization and Evolutionary Optimization. In this sense, decision trees have different hyperparameters that must be adjusted. Among those parameters, one of the most important is the depth of the tree. In general, a greater depth provides a greater accuracy, although it also results in a greater number of if-then-else operations. Figure 7 shows the training and validation scores of a decision tree with different depth values. As it can be seen, if the depth value is very low, both the training score and the validation score are low (underfitting). Medium depth values (from 6 to 12) provide high values for both scores. It means that the classifier is performing fairly well. However, if the depth value is too high, the classifier will overfit, which means that the training score is good but the validation score is poor. In this work, a good trade-off has been found between the required number of operations and the obtained accuracy, by selecting a maximum depth value equal to 10. On the other hand, the *entropy* function has been selected to measure the quality of each split in the decision tree.

Once the hyper-parameters have been tuned, the resulting Receiver Operating Characteristic (ROC) curve has been

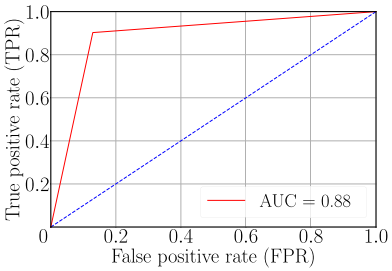



**FIGURE 8.** Decision Tree Receiver Operating Characteristic (ROC).

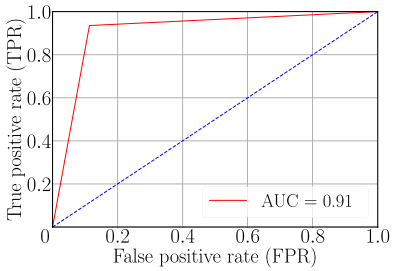



**FIGURE 9.** Decision Tree ROC (node 1 and divided dataset).

obtained as shown in Figure 8. For this purpose, we split the dataset into two sub-sets, reserving a 30% of the data to test the prediction capability. The ROC curve is one of the most well-known performance metrics for machine learning classifiers [36]. Basically, it is a two-dimensional graph which plots the *true positive rate* (TPR) against the *false positive rate* (FPR) for different values of the discrimination threshold:

- The TPR (also called *sensitivity* or *recall*) represents the proportion of positive samples that were correctly classified as positive. It can be calculated as follows:

$$TPR = \frac{TP}{TP + FN} \tag{1}$$

  where TP (*true positive*) is the number of positive samples correctly classified as positive, and FN (*false negative*) is the number of positive samples incorrectly classified as positive.
- The FPR (also called *fall-out*) represents the proportion of negative samples that were incorrectly classified as positive. It can be calculated as follows:

$$FPR = \frac{FP}{FP + TN} \tag{2}$$

  where FP (*false positive*) is the number of negative samples incorrectly classified as positive, and TN (*true negative*) is the number of negative samples correctly classified as negative.

Therefore, classifiers that produce curves closer to the top-left corner obtain a better performance. Obviously, the best classification is achieved in the point (0, 1). A uniform random prediction will provide points over the diagonal line. Points under that line represent a prediction worse than a random classification. The Area Under the Curve (AUC) is also used as a performance value, with a maximum value equal to 1. This value is also shown in the figure. As it can be seen, the decision tree classifier trained with our proposed dataset provides a ROC curve with a high level of prediction performance.

For the previous performance evaluation, the simplest case has been considered. In this case all the network nodes use the same classifier each time they have to make the decision of whether or not to transmit a data packet. In order to obtain a performance improvement, the dataset can be divided into $N$ (number of nodes) different subsets $SS_i$, $i = 1 \ldots N$. In each subset, only the samples corresponding to the packets generated from node $i$ will be included. This way, a different decision tree will be built for each node. Some examples will be shown later.

**TABLE 5.** Decision tree performance metrics [35], [38].

| Performance metric | Same dataset | Split dataset |
|---|---|---|
| *F-score* | 0.907 | 0.921 |
| *Cohen's kappa* | 0.772 | 0.824 |

Figure 9 shows the ROC curve for node number 1 when the dataset has been divided as a function of the source nodes. Comparing Figures 8 and 9, the second model exhibits a better prediction performance. Besides, the AUC value is higher because the second plot is nearer to the top-left corner. The cost to pay for this performance improvement is a higher computational load in the training phase and a slight increase in the network control traffic.

These two possibilities have also been evaluated in terms of the *F-Score* (harmonic mean of the *precision* and *recall*) and *Cohen's kappa* (agreement between two raters) metrics [37]. The results are shown in Table 5. Overall, it can be seen that better results are obtained when a different decision tree is built and used by each source node.

In the following, these two possibilities will be evaluated in a networking scenario with the help of the ns-3 simulator (Step 3 in Figure 2). For this purpose, the model is exported to be used in a real network. Thus, each network node will have available the parameters of its decision tree. Furthermore, as inputs for these trees, they will need the updated values of $\rho_i$ and $q_i$ of the rest of the network nodes. These values must be broadcast by all nodes at regular time intervals. Such broadcasting can be done by means of the inclusion of new dedicated fields in the HWMP routing protocol frames (PREQ and PREP), or by means of the generation and transmission of special management frames (Action subtype) [6].

### 1) NETWORK PERFORMANCE EVALUATION

The first two network experiments carried out are related to the evaluation of the generated training datasets. On the one hand, the complete dataset will be used to generate the model to be used by all the network nodes. Remember that in this case all of them will have the same decision rules. On the other hand, the training dataset will be divided as a function of the source nodes. That is, each node will have its own decision rules. For these two experiments, a Smart Grid NAN scenario made up of eight smart meters and one data concentrator (as already seen in Figure 3) has been considered. In this first evaluation, only one traffic type is

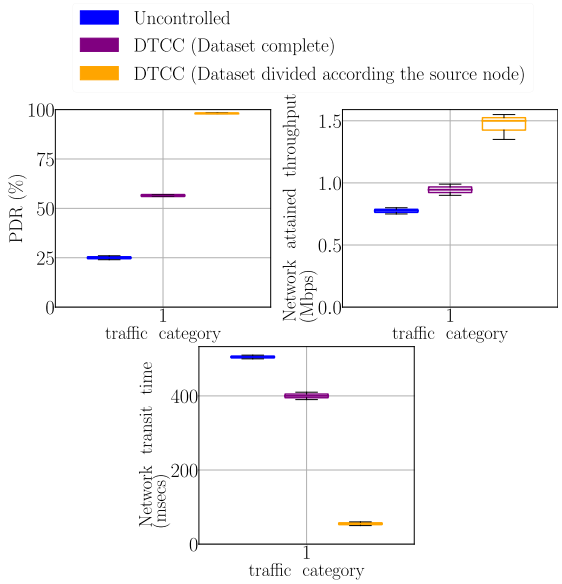
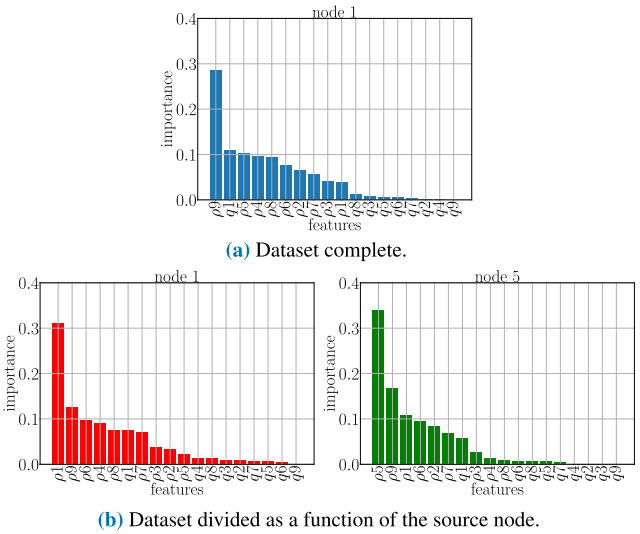



**FIGURE 10.** Packet delivery ratio, network attained throughput and transit time.




**(a)** Dataset complete.

**(b)** Dataset divided as a function of the source node.

**FIGURE 11.** Features importance.

transmitted upstream from the smart meters towards the data concentrator. The packet generation rate and packet length distributions have been considered as truncated exponentials, with average values 200 Bytes and 200 pkt/s respectively. With these values, the traffic generated by the applications is greater than the traffic the network is able to transport. This way, the congestion control mechanism must be activated by all the network nodes.

Figure 10 presents, by means of box-plots, the results of these two first sets of simulations in terms of packet delivery ratio (PDR), attained throughput, and network transit time. As it can be seen, the network performance is better for those three parameters when the Decision Tree based Congestion Control (DTCC) mechanism is applied. Besides, the results are even better when the training data set is divided according to the source node. For instance, in the absence of a congestion control mechanism, the PDR only reaches the value of 26%. However, when the proposed mechanism is applied, the PDR reaches a 50% and this value increases to 98% when the data set is divided by nodes. Similar results are obtained for the network throughput, where DTCC exhibits better results. That is, more information is correctly transmitted to the data concentrator. Finally, the network transit time is decreased more than the 50% with the proposed solution, and this value decreases even more when the dataset is divided according the source node.

From these preliminary results, it is possible to conclude first that the proposed mechanism will effectively improve the network performance. Second, dividing the dataset by nodes represents an even greater improvement, although it also implies a greater operational complexity in the network.

### 2) FEATURES SELECTION

The next step is to make a selection of the best features that should be taken into account. In the previous experiments, the decision trees have been built taking into account all the available features, that is, the values of $\rho_i$ and $q_i$ measured by all the network nodes. However, all these features will not have the same importance for all the nodes, having some of them more relevance than others. Thus, the number of used features could be reduced, giving rise to less complex models. To make this relevance analysis, the L1 regularization technique is commonly used [39]. By means of this technique, a ranking has been created where the different features of the dataset are sorted by their relative importance. Figures 11a and 11b present the features importance for the two cases previously considered (dataset complete and dataset divided according the source node respectively). Note that the feature importance plots are normalized. In the second case (Figure 11b) the features ranking is shown for nodes number 1 and 5. As it can be seen, the most relevant features at each node are their own channel utilization ($\rho_1$ and $\rho_5$ respectively). However, when the whole dataset is used by all the network nodes (Figure 11a) the most important feature is related to the data concentrator ($\rho_9$). Figure 12 shows a simplified example of the different decision trees obtained in both cases. The trees have been truncated with a depth value equal to three. At the final branches, the classification obtained by a greater number of samples at this level is indicated (the value "1" means a prediction of success in the packet transmission, and the value "0" means a prediction of not success).

On the other hand, Figure 13 shows the number of features needed by every network node to reach a certain accuracy score (80, 85, and 90%). As expected, the higher the desired accuracy score, the more features are needed.

In order to illustrate the benefits obtained by applying feature reduction, a new set of simulations has been carried out. In this case, the models (per node) are generated with the most important characteristics to obtain an accuracy score of 85%. At the beginning of the training phase, the model is trained just with the most relevant feature. If the model does

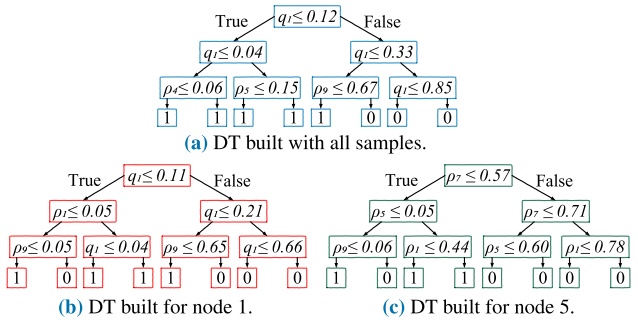



**FIGURE 12.** Decision trees examples.

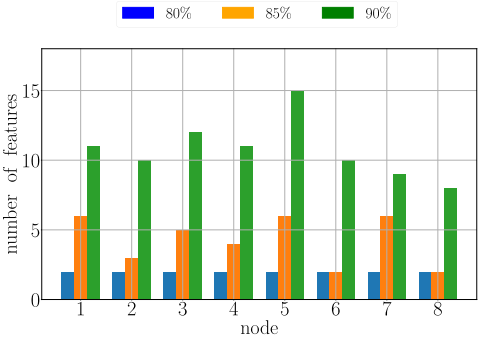



**FIGURE 13.** Number of features needed to reach different accuracy scores.

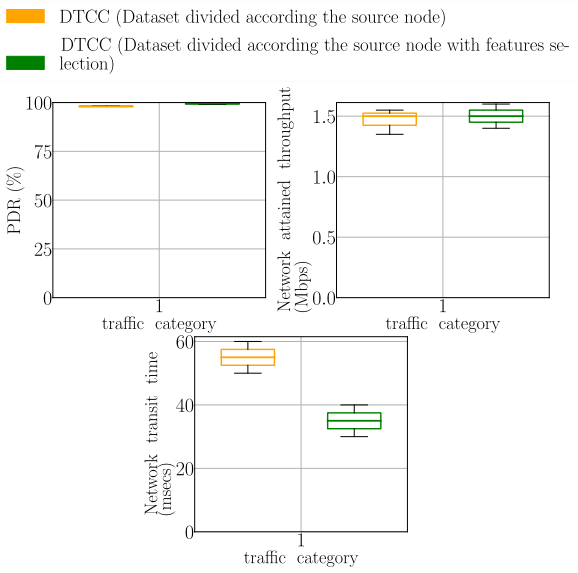



**FIGURE 14.** Packet delivery ratio, network attained throughput and network transit time (number of selected features computed for a target accuracy of 85 %).

not reach the desired testing score, the second most important feature is included in a new training, and so on. Moreover, a maximum number of features equal to 6 was set up as a trade-off between the accuracy and the complexity of the model.

Figure 14 shows the results in terms of packet delivery, throughput and transit time. As it can be seen, these results are pretty similar to those presented previously in Figure 10, showing even a slight improvement in terms of network transit time. Besides, thanks to the feature reduction, the model complexity, as well as the amount of network control traffic, have been reduced.

### B. TRAFFIC DIFFERENTIATION

As already mentioned, another requirement for the proposed congestion control mechanism is that it must allow traffic differentiation. With the huge increase in traffic expected for data networks due to the proliferation of IoT devices, we need to be able to distinguish the importance of the different traffic flows and assign different priorities to allocate the shared network resources.

In this sense, the classification algorithms must be "taught" to distinguish between different traffic types to decide whether or not a packet can be transmitted. As explained in subsection III-C, in this proposal this differentiation is carried out when building the datasets that will be used in the algorithms training phase. Thus, during the simulations carried out to generate the different datasets, the influence of some traffic types on others has been analyzed for all possible combinations of the network load. After this analysis, in the data processing phase, the labels assigned to the different samples are modified in order to favor the most priority traffic types, but without unnecessarily discarding those with lower priority.

First, we will evaluate a Smart Grid NAN scenario where two traffic types are transmitted. The previous network scenario is considered again, but now all the network nodes are transmitting packets belonging to the two traffic types. On the one hand, the system will provide congestion control in situations of high network load. On the other hand, the solution will provide higher transmission priority to traffic type 1.

Table 4 shows the maximum allowable network transit time values that have been selected for each traffic type. The dataset has been divided as a function of the source node and the traffic type. Therefore, each node have its own decision rules based on the traffic type to be transmitted. Besides, a features selection has also be done for a target accuracy of 85%.

With respect to the decision tree parameters, they have been chosen again after performing several evaluations with different combinations of possible values: the tree depth has been set to 6, and the selected decision criterion has been again the entropy.

The packet length and the packet generation rate distributions have been selected as truncated exponential for both data flows, and their average values are 200 Bytes and 100 pkt/s respectively. As previously done, we are operating with a high network load, and therefore, the congestion control mechanism is activated.

Figure 15 shows the obtained results in terms of packet delivery ratio, network throughput and transit time. As it can be seen, in the absence of a congestion control solution a high percentage of packets will be lost on their way towards the data concentrator. However, when the DTCC mechanism is

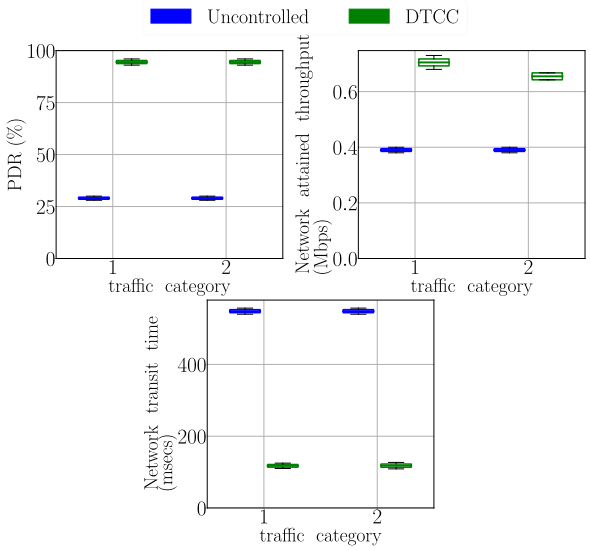



**FIGURE 15.** Packet delivery ratio, network attained throughput and network transit time (number of selected features computed for a target accuracy of 85 %).

applied, a 92% of the packets are correctly transmitted to the sink node. Regarding the attained throughput, the value for traffic type 1 is higher than for traffic class 2, and thus, different priorities are correctly provided to each traffic category. Besides, the attained throughput is higher compared to the uncontrolled case for both traffics. Finally, the figure shows also a significant improvement obtained in terms of network transit time with our solution.

Overall, the previous results indicate the benefits obtained with the proposed mechanism when two traffic types are transmitted on the network. Next, a last set of experiments will be carried out to verify if the mechanism works correctly for a higher number of traffics types (4) and a larger scenario. In a similar way, the traffic type 1 has the highest QoS needs, while the traffic type 4 has the lowest QoS requirements. For this purpose, a new data collection and data processing have been done as follows. First, if the QoS requirements of traffic type 1 are not met (packet is not received correctly on time), the lower priority traffics are not considered for transmission (that is, their classification labels are set to 0). The same will happen if the QoS needs of traffic 2 are not met. In this case, traffic types 3 and 4 will not be considered for transmission. As before, for the data collection a common identifier is included in the packet header, in order to be able to relate the four packets transmitted simultaneously, belonging each one to a different traffic type. The maximum allowed network transit times are presented in Table 6. As previously done, different combinations of traffic patterns were configured in the data collection process.

In this case, a network size of 25 nodes has been considered, with 24 SM/GW transmitting data packets towards the data concentrator. The packet length and the packet generation rate distributions have been selected as exponential for the four data flows, and their average values are 200 Bytes

**TABLE 6.** Maximum network transit time allowed for each category.

| Cat | Network transit time |
|---|---|
| 1 | 50 ms |
| 2 | 100 ms |
| 3 | 1000 ms |
| 4 | 2000 ms |

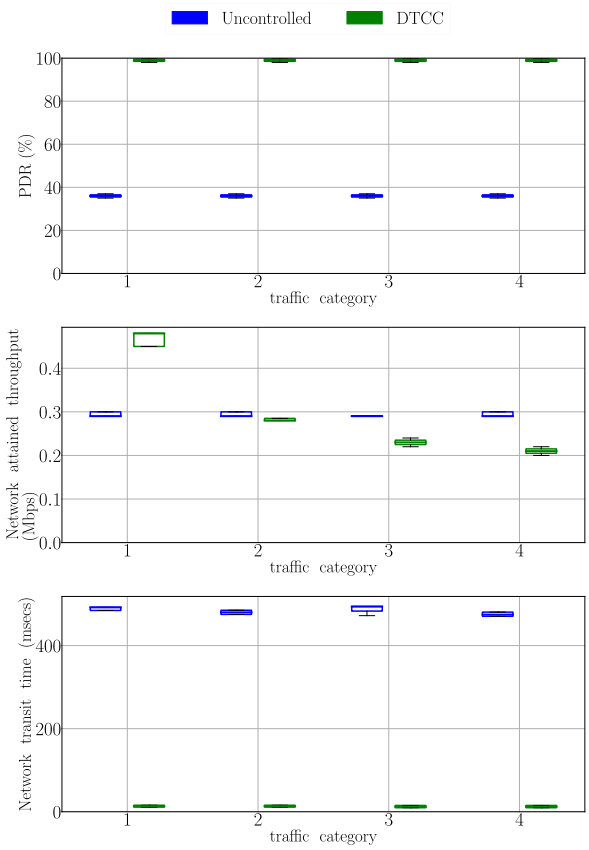


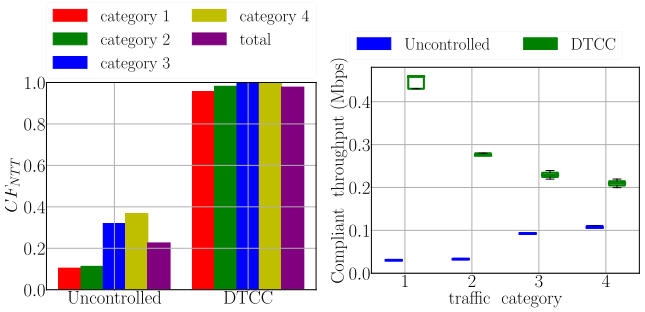

**FIGURE 16.** Packet delivery ratio, network attained throughput and network transit time (number of selected features computed for a target accuracy of 85 %).

and 20 pkt/s respectively. With these values, the NAN will operate again in a congested scenario.

Given that the current evaluation scenario consists of 25 nodes, the training dataset will contain 50 features (the values of $\rho_i$ and $q_i$ in every node). In order to reduce the dataset dimension as well as the model complexity, a features selection is performed again, with a target accuracy equal to 85%.

The obtained results are shown in Figure 16. As it can be seen, in the absence of a congestion control solution, a 75% of packets will be lost in their way towards the data concentrator. However, with the proposed solution, the PDR is 100% for all traffic types. Besides, the traffic type 1 is strongly favoured in terms of throughput, and thereby different priorities are provided to each traffic type. Finally, it is also shown the significant improvements obtained in term of network transit time with our solution.

Regarding the throughput, the previous figure shows a higher value for the traffic category with the highest priority,




(a) Network transit time compliant factor. (b) Compliant throughput.

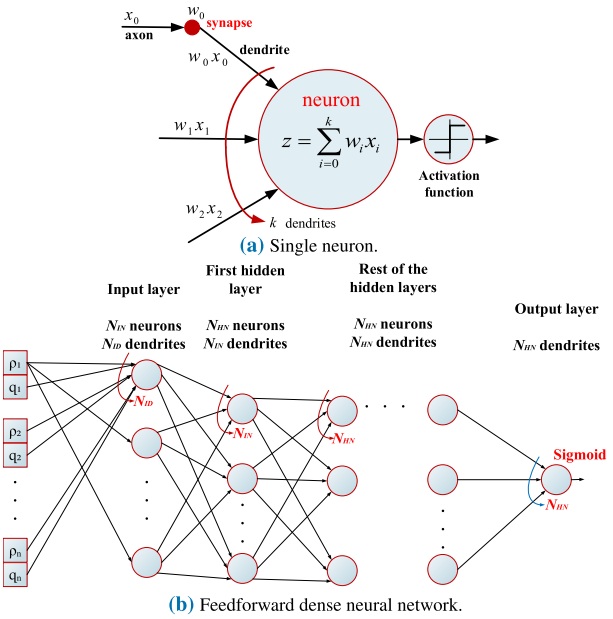
**FIGURE 17.** Network transit time compliant factor and complaint throughput.

but a lower throughput is reflected for the other three categories. On the one hand, as already mentioned, to get different values depending on the traffic type is a desired effect, since one of the requirements of the proposal is to differentiate the relevance of the traffic flows. On the other hand, it must also be taken into account that the received packets, in order to be useful, must meet their requirements regarding the maximum allowed network transit time, and this time is different for each category (see Table 6). In this sense, an interesting measure is the percentage of packets of each category that has been delivered to their destination on time. We refer to this percentage as the compliant factor and we represent its values in Figure 17a. In addition to the factor for each category, a global value is also offered, taking into account the proportion of packets of each traffic type. Our results indicate that this factor is considerably better in all cases when applying DTCC, reaching practically the maximum value. Taking this factor into account, the compliant throughput is represented in Figure 17b, where it can be seen how the obtained value is higher, for all traffic categories, when DTCC is applied.

## V. FEEDFORWARD NEURAL NETWORK BASED CLASSIFICATION

In this section, a neural network is used in the classification step. Figure 18 depicts a generic neural network together with the smallest part of this architecture, the neuron. On the one hand (Figure 18a), neurons consist of mathematical functions aiming to emulate a biological neuron. Basically, a neuron or perceptron is a computational unit that calculates the weighted average of its inputs, being the resulting value the input of an activation function, which finally generates the output of the neuron. On the other hand, the artificial neural network is a collection of interconnected neurons, organized into different layers (Figure 18b):

- In the first layer, there are a certain number of neurons ($N_{IN}$) and dendrites ($N_{ID}$) that represent the input of the neural network. For our purpose, these inputs are the $\rho_i$ and $q_i$ (as defined in the previous sections) values.
- Second, the network is made up of a set of (hidden) layers ($N_{HL}$), each one containing a certain number of neurons ($N_{HN}$). These are the two main hyperparameters that define the network architecture or topology. In the following sections, the selection of these values will be covered.
- Finally, the last layer consists of one neuron with $N_{HN}$ dendrites and one axon, which provides the network output.




(a) Single neuron. (b) Feedforward dense neural network.

**FIGURE 18.** Neural network.

**TABLE 7.** Neural network training parameters.

| | Parameter | Value |
|---|---|---|
| **Architecture** | Neural network type | Feedforward |
| | Layer type | Dense |
| | Input activation function | ReLU |
| | Intermediate activation function | ReLU |
| | Output activation function | sigmoid |
| **Learning process** | Loss function | mean square error |
| | Optimizer | adam |
| | metrics | accuracy |
| **Network training** | Validation split | 0.3 |
| | Testing split | 0.10 |
| | Number of epochs | 200 |
| | Validation-based | Early stopping |

### A. NEURAL NETWORK DESIGN

In the following subsections, the characteristics selected in the design of the neural network are specified, which are also summarized in Table 7. To build and to train the model, we have made use of the *keras* [40] set of tools inside our framework.

#### 1) NEURAL NETWORK ARCHITECTURE

In this work, a feedforward neural network [41] has been chosen to implement the desired classification mechanism. With this network topology, the information progresses in only one direction, that is, no feedback loops are built. Therefore, the computational cost is smaller. The next step is to chose how the different layers are interconnected. Among the different possibilities, we have used a fully-connected

network, also known as *dense layer type*. In a fully-connected network, the inputs of each neuron are the outputs of all the neurons in the previous layer.

#### 2) ACTIVATION FUNCTION

Another characteristic of the neural network that must be selected is the activation function of the neurons. There are different activation functions, mainly classified into three types: binary step, linear, and nonlinear activation functions. Among them, nonlinear activation functions are mainly preferred as they allow the nodes to learn more complex structures in the data. For the first and hidden layers, we have chosen the well-established Rectified Linear Activation (ReLU) function for, among other characteristics, its computational simplicity and linear behavior [42], [43]. Finally, the sigmoid activation function has been used in the output layer, due mainly to the fact that its output varies from 0 to 1, which fits our classification problem.

#### 3) LEARNING PROCESS

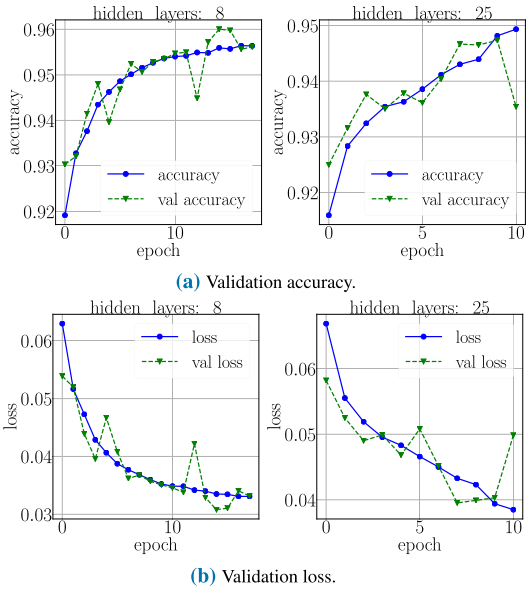
In the learning process, a suitable optimization algorithm has to be selected, which is in charge of computing the best set of weights that minimizes the prediction error (quantified by means of some loss function). Basically, the optimization algorithm computes the derivative of the loss function with respect to the neural network parameters. With this computation, the appropriate value to adjust the weights is obtained. For our purposes, the well-established adaptive moment estimation (adam) optimizer is used. This optimization algorithm has been proven to be a good choice compared to the classical stochastic gradient descent procedures [44]. On the other hand, the well-known mean square error (MSE) has been chosen as the loss function.

#### 4) MODEL TRAINING

In the training phase, the original datasets are divided again into training and validation subsets. Besides, we have to set up the number of epochs that the model will iterate through all the training samples. Generally speaking, the more epochs are run, the more accurate the model is. However, to avoid overfitting, the validation-based early stopping technique has been used [45]–[47]. With this technique, the training phase is finished when it is detected that increasing the number of epochs does not improve the model accuracy. It will be seen later when analyzing the obtained ROC curves and accuracy values, that the model is not showing overfitting problems.

#### 5) SELECTING THE NUMBER OF HIDDEN LAYERS AND NEURONS

As previously said, artificial neural networks have two main hyperparameters that control the architecture or topology of the network: the number of layers and the number of nodes in each of them. For the first (input) and last (output) layers, we have already selected the number of neurons (Figure 18) and their inputs. However, the number of hidden layers and neurons are still unknown parameters that must be configured. There are some heuristics methods on how to estimate




**(a)** Validation accuracy.

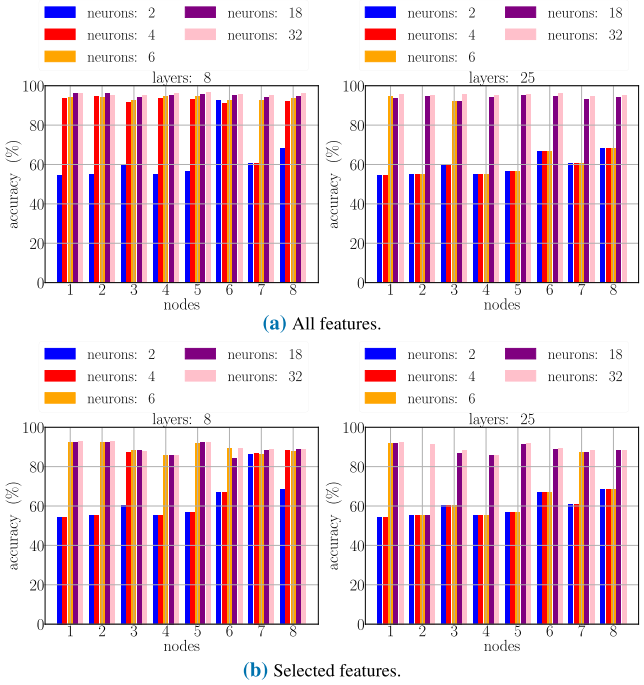
**(b)** Validation loss.

**FIGURE 19.** Model accuracy and loss.

those values [48]. However, in this work, we have carried out a systematic experimentation to discover the best configuration for our specific datasets. Thus, the first experiments are aimed to select the number of hidden layers and neurons.

For these first experiments, the neural network has been trained with all the available features, that is, without doing a features selection. A network size of 9 nodes and just one traffic category have been considered. Figure 19 shows the results obtained for the network node number 1, in terms of accuracy and loss in front of the number of epochs. The values obtained for the rest of the network nodes are very similar. For these two parameters, both the general value (obtained from the whole training dataset) and the validation value (obtained from the validation subset) are shown. As it can be seen, increasing the number of hidden layers from 8 to 25 does not represent a significant improvement. Thus, the configuration for 8 layers might be seen as a good selection if we want to reduce the model complexity. With the following experiments, more information will be obtained for the selection of this parameter.

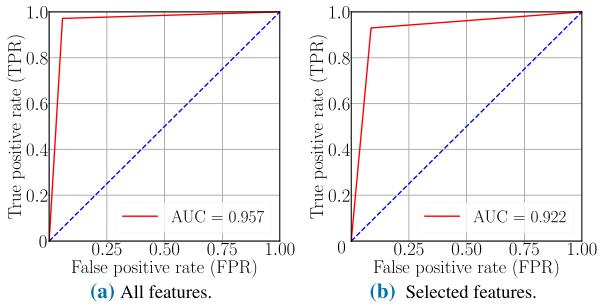
A new set of training runs were performed to select the number of neurons at each hidden layer. Here, we have obtained the accuracy value for two cases: using all the available features, or just the most relevant. From the previous results, the number of hidden layers has been configured to 8 and 25, while the number of neurons is a variable parameter. Figure 20 shows the obtained accuracy for every network node. As it can be seen, even though the results are very similar, a better performance is obtained when the network is built with 8 layers. Furthermore, almost the same results are obtained when the number of neurons is greater or equal than 6. Therefore, the values selected for the number of hidden layers and for the number of neurons will be 8 and 6 respectively. These values will be used in the rest of the experiments carried out in this section.




(a) All features.

(b) Selected features.

**FIGURE 20.** Neural network accuracy in each source node.




(a) All features. (b) Selected features.

**FIGURE 21.** Receiver Operating Characteristic (ROC) metric.

**TABLE 8.** Neural network performance [35], [38].

| Performance metric | All features | Selected features |
|---|---|---|
| *F-score* | 0.961 | 0.928 |
| *Cohen's kappa* | 0.915 | 0.844 |

### 6) VALIDATION OF THE NEURAL NETWORK WITH CLASSIFICATION METRICS

In the same way as previously done with the decision tree classifier, the neural network will also be evaluated by means of the ROC curve and other performance metrics. On the one hand, Figure 21 shows the ROC curve computed in the two considered cases: when the model is trained with all the features or just with the most relevant. As it can be seen, for both cases the neural network exhibits a higher prediction power than the decision tree classifier (Figures 8 and 9). Here, the plots are nearer to the top-left corner and so the AUC value is higher. On the other hand, Table 8 presents the resulting values of the *F-Score* and *Cohen's kappa* metrics. Comparing these results with those obtained for Decision Trees (Table 5), it can be seen again that better results are achieved with the neural network.

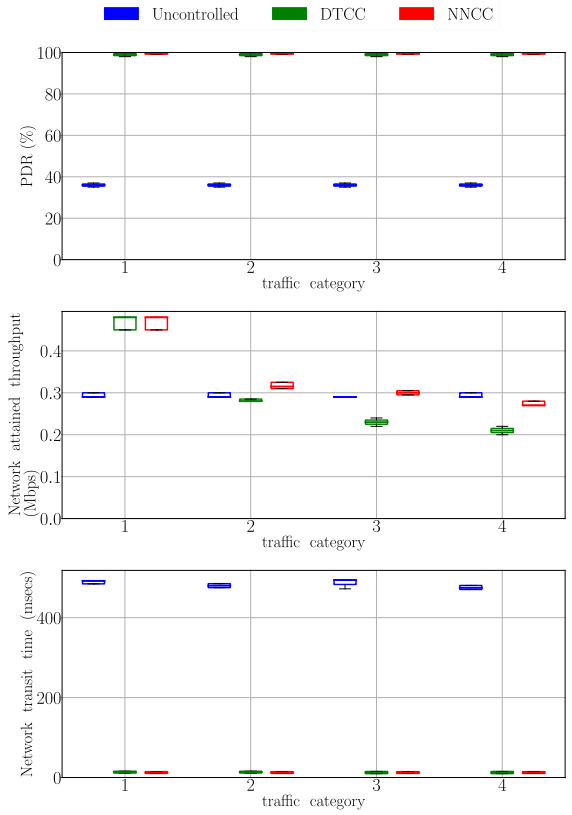


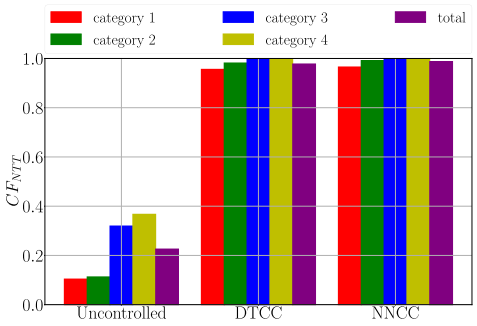

**FIGURE 22.** Packet delivery ratio (PDR), network attained throughput and transit time (network size 25 nodes, 4 Traffic categories).

## B. NETWORK PERFORMANCE EVALUATION

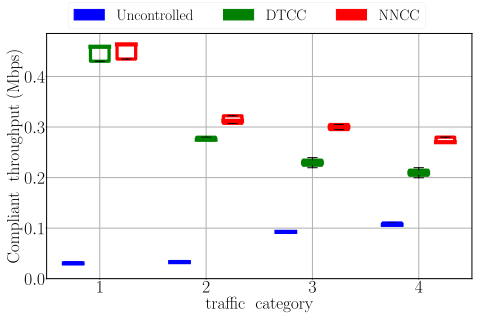
In this subsection, we will evaluate the trained neural network through network simulations. We have considered again a realistic SG NAN scenario made up of a set of smart meters and one data concentrator. Besides, we will comparethe decision tree (DTCC) and the neural network congestion control (NNCC) mechanisms. Finally, some considerations about the computational cost of both proposals will also be provided.

For this set of experiments, we have considered a large network size consisting of 24 smart meters transmitting upstream data traffic towards the data concentrator. Besides, we have considered 4 traffic categories and provided traffic differentiation based on the QoS needs of each traffic category. Similar to the previous experiments, the packet length and the packet generation rate distributions have been selected as truncated exponentials for the four data flows, and their average values are 200 Bytes and 20 pkt/s respectively.

Figure 22 shows again the significant improvements obtained in terms of PDR, network throughput, and transit time when both DTCC and NNCC mechanisms are applied. On the one hand, Figure 22 shows that in the absence of our congestion control solutions, approximately 75% of packets will be lost on their way towards the data concentrator. However, when the congestion control mechanisms are applied, the PDR is almost 100%. Besides, the PDR reached is slightly better with NNCC. On the other hand, the results vary when




(a) Network transit time compliant factor.




(b) Compliant throughput.

**FIGURE 23.** Network transit time compliant factor and compliant throughput.

the attained throughput is analyzed. For both cases, the delivered data is higher when the congestion control mechanisms are applied, and traffics with higher priorities are favored. Besides, the attained throughput for traffic type 1 is almost the same for both solutions, while for the rest of the traffics the NNCC proposal provides higher values. Regarding the network transit time, it is shown the significant improvements obtained for both proposals. As it can be seen, this value is approximately 30 times lower compared to the uncontrolled case.

As previously done, we have also computed the network transit time compliant factor. Remember that the received packets, to be useful, must meet their requirements regarding the maximum allowed network transit time, and this time is different for each category (see again Table 6). Figure 23a shows how the value of this factor is considerably better in all cases when applying the DTCC and NNCC mechanisms, reaching practically the maximum value. However, this factor is slightly better with the neural network classifier. Finally, taking this factor into account, the compliant throughput is represented in Figure 23b, where it can be seen again how the value is higher, for all the traffic categories, if DTCC or NNCC are applied.

As a summary of the presented results, we can conclude that both mechanisms perform well over the targeted Smart Grid NANs, but a better behavior is observed with the neural network. This improvement is higher if we consider a higher number of traffic categories and larger network sizes. In these cases, we have a higher number of features describing the overall network situation, and therefore the higher adaptation ability of the neural networks shows its advantages. On the other hand, this greater prediction capability does not come free, because it has a higher computational cost. This issue will be evaluated in the next subsection.

### C. COMPUTATIONAL COST

The proposed congestion control mechanism works on a per-packet basis, that is, the node must decide, for each new packet generated by the applications, whether or not it should be transmitted through the network. This means that all the computations needed by the mechanism must be performed every time a new packet is generated. Therefore, the computational complexity of the mechanism is an important factor to take into account when checking the possibility of implementing it in a real network environment, especially considering that the network nodes must be kept at the lowest possible cost.

Among the two possibilities presented in this work, neural networks represent a much higher computational cost. In fact, the computational cost of decision trees is the needed to make a number of comparisons equal to the depth of the tree minus one. This cost is affordable, without problems, by current low-cost hardware platforms.

In the case of neural networks, we must first consider the computational cost of a neuron, which have to perform in a first step the following calculation (the so-called net input):

$$z = \sum_{i=0}^{k} w_i x_i \tag{3}$$

where $k$ is the number of dendrites, $w_i$ the weights and $x_i$ the input values. In particular, $x_0 = 1$ and $w_0$ is the additive inverse of the neuron decision threshold [49]. This calculation represents a total of $k$ products and $k$ sums, or what is the same $k$ accumulated products. In a second step, the activation function is applied on z. As previously explained, in this work we have used the ReLU function in all neurons, except for the output neuron in which the sigmoid function has been chosen. Thus, for each neuron in the network, the computational cost can be considered the cost of $k$ accumulated products ($kC_{AP}$) plus the cost of the activation function ($C_{RL}$ or $C_{SIG}$).

Regarding the network layers, there are some differences with respect to the number of neurons and dendrites:

- In the input layer there are $N_{IN}$ neurons with $N_{ID}$ dendrites each, and so the computational cost is $N_{IN}(N_{ID}C_{AP} + C_{RL})$.
- In the first hidden layer there are $N_{HN}$ neurons with $N_{IN}$ dendrites in each neuron, and so the computational cost is $N_{HN}(N_{IN}C_{AP} + C_{RL})$.
- For the rest of the hidden layers, there are $N_{HN}$ neurons with $N_{HN}$ dendrites in each neuron, and so the computational cost is $N_{HN}(N_{HN}C_{AP} + C_{RL})$.
- Finally, in the last layer there are just one neuron with $N_{HN}$ dendrites, and so the computational cost is $N_{HN}C_{AP} + C_{SIG}$.

**TABLE 9. Computational cost evaluated on hardware platforms.**

| $N_{ID}$ | $N_{IN}$ | $N_{HL}$ | $N_{AP}$ | $N_{RL}$ | $N_{SIG}$ | **ESP-8266** | **ESP-32** |
|---|---|---|---|---|---|---|---|
| 6 | 6 | 8 | 330 | 54 | 1 | 2898 | 52397 |
| 6 | 6 | 25 | 942 | 156 | 1 | 1017 | 18959 |
| 18 | 6 | 25 | 1024 | 156 | 1 | 947 | 17794 |
| 18 | 18 | 25 | 8442 | 468 | 1 | 116 | 2349 |

Taken into account all the network layers, the global computational cost of the neural network ($C_{NN}$) is:

$$\begin{aligned} C_{NN} &= N_{IN}(N_{ID}C_{AP} + C_{RL}) + N_{HN}(N_{IN}C_{AP} + C_{RL}) \\ &\quad + (N_{HL} - 1)N_{HN}(N_{HN}C_{AP} + C_{RL}) \\ &\quad + N_{HN}C_{AP} + C_{SIG} \end{aligned} \tag{4}$$

This expression can be rewritten as:

$$\begin{aligned} C_{NN} = [N_{IN}N_{ID} + N_{HN}\,[N_{IN} + N_{HN}\,(N_{HL} - 1) + 1]]\,C_{AP} \\ + \,(N_{IN} + N_{HL}N_{HN})\,C_{RL} + C_{SIG} \end{aligned} \tag{5}$$

where we can clearly identify the number of necessary accumulated products, ReLU and sigmoid functions ($N_{AP}$, $N_{RL}$ and $N_{SIG}$ respectively), for every complete run of the neural network:

$$C_{NN} = N_{AP}C_{AP} + N_{RL}C_{RL} + N_{SIG}C_{SIG} \tag{6}$$

In the previous dimensioning experiments, we have considered $N_{IN} = N_{HN}$, and so:

$$N_{AP} = N_{IN}(N_{ID} + N_{HL}N_{IN} + 1) \tag{7}$$

$$N_{RL} = N_{IN}(N_{HL} + 1) \tag{8}$$

$$N_{SIG} = 1 \tag{9}$$

With the selected values ($N_{ID} = 6$, $N_{IN} = 6$, $N_{HL} = 8$) we get a value of 330 accumulated products, 54 ReLUs and 1 sigmoid needed for every packet transmission. To calibrate the magnitude of these values, we have performed a set of simulations on real low-cost hardware platforms commonly used nowadays in IoT device developments. Specifically, we have considered the platforms ESP-8266 [50] and ESP-32 [51]. The first one is provided with one 160MHz low-power 32-bit microprocessor and 80 KB of on-chip SRAM, while the second one has two 240MHz low-power 32-bit microprocessors and 520 KB of on-chip SRAM. Table 9 shows the results obtained for different combinations of the network dimensioning parameters. The values of $N_{AP}$, $N_{RL}$ and $N_{SIG}$ are computed following the previous expressions. For each of the platforms under study, it is shown the number of times per second that all the necessary computations can be run, that is, the number of packets per second that can be checked by the congeestion control mechanism. As it can be seen, for the selected design parameters, the computation cost is affordable even for the least expensive platform (ESP-8266). The first line of the table shows the results for the previously selected parameters. The ESP-8266 platform is able to perform all the necessary operations 2898 times per second, and so this is the number of packets per second that can be evaluated by the congestion control mechanism before they are transmitted or discarded. The values are of course higher if we make use of the ESP-32 platform. For parameter combinations that lead to more complex networks, the use of higher cost hardware platforms could be necessary. It is worth mentioning that in the tests carried out the microprocessors have been fully dedicated to this task, and the clock frequency (and therefore the power consumption) have been adjusted to the maximum possible value.

## VI. CONCLUSION AND FUTURE WORK

In this work, a congestion control mechanism for Smart Grid Neighborhood Area Networks (SG NANs), based on machine learning techniques, has been proposed, implemented and evaluated.

In the proposed mechanism, the source nodes must decide if they transmit or not each new data packet generated by the applications, depending on the state of the network (amount of traffic being currently transported). This state is characterized by the value of the channels utilization factor and by the occupation of the packet buffers. These values are measured by all the network nodes and broadcast to the source nodes. Besides, the separation of the traffic flows into different categories is taken into account, depending on the relevance of the data generating applications. Thereby, the provision of different QoS levels is also allowed.

In most applications of machine learning techniques, it is essential to previously have large datasets that make it possible to train the selected algorithms. Thus, the first contribution of this work has been the proposal of a mechanism for the generation of appropriate datasets. These datasets allow us to describe the behavior of the network and the influence of some traffic categories on others.

On the other hand, it has also been considered the division of the obtained datasets in various subsets. This way, an individualized training is carried out for each source node. This division results in a better performance with the cost of a small increase in the complexity of the mechanism during the initial phase. Besides, a feature selection has also been taken into account to reduce the complexity of the model.

Two different classification algorithms have been used and evaluated. The first one is based on decision trees, which present a very low computational cost. Its performance is evaluated in a first step through the Receiver Operating Characterisitic (ROC) curve, showing a very good accuracy value, which is even better when the dataset is divided according to the source node. Next, the benefits obtained in a SG NAN environments have been evaluated, in terms of packet delivery ratio (PDR), throughput and network transit time, obtaining significant improvements in the three parameters. On the other hand, the system behavior has been evaluated considering different traffic categories, observing how the mechanism adequately differentiates the QoS provided to each of them. At this point, in addition to the previously evaluated parameters, the concepts of compliant factor and compliant throughput have been introduced, to describe in a more clearly way the obtained improvements.

The second classification mechanism has been based on the use of neural networks. With this classifier, the benefits have improved the results obtained with the decision trees, although this is at the cost of a higher computational complexity.

As future lines of work, other interesting and relevant characteristics, such as emergency awareness and fairness in the distribution of network resources among all the network nodes, will be considered.

## REFERENCES

[1] N. Shaukat, S. M. Ali, C. A. Mehmood, B. Khan, M. Jawad, U. Farid, Z. Ullah, S. M. Anwar, and M. Majid, "A survey on consumers empowerment, communication technologies, and renewable generation penetration within smart grid," *Renew. Sustain. Energy Rev.*, vol. 81, pp. 1453–1475, Jan. 2018.

[2] V. C. Gungor, D. Sahin, T. Kocak, S. Ergut, C. Buccella, C. Cecati, and G. P. Hancke, "A survey on smart grid potential applications and communication requirements," *IEEE Trans. Ind. Informat.*, vol. 9, no. 1, pp. 28–42, Feb. 2013.

[3] NS-3 Consortium. *NS-3 is a Discrete-Event Network Simulator*. Accessed: Sep. 10, 2020. [Online]. Available: https://www.nsnam.org/

[4] *IEEE Standard for Information Technology—Local and Metropolitan Area Networks—Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications. Amendment 10: Mesh Networking*, IEEE Standard 802.11s, IEEE Computer Society, Mar. 2011.

[5] H. Gharavi and B. Hu, "Multigate communication network for smart grid," *Proc. IEEE*, vol. 99, no. 6, pp. 1028–1045, Jun. 2011.

[6] *IEEE Standard for Information technology—Local and metropolitan area networks. Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications*, IEEE Standard 802.11-2016, Dec. 2016.

[7] T. Clausen and P. Jacquet, "Optimized link state routing protocol (OLSR)," *Internet Engineering Task Force (IETF)*, vol. 4, p. 75, Oct. 2003.

[8] Y. Tsado, K. A. A. Gamage, D. Lund, and B. Adebisi, "Performance analysis of variable smart grid traffic over ad hoc wireless mesh networks," in *Proc. Int. Conf. Smart Syst. Technol. (SST)*, Oct. 2016, pp. 81–86.

[9] Y. Tsado, K. Gamage, B. Adebisi, D. Lund, K. Rabie, and A. Ikpehai, "Improving the reliability of optimised link state routing in a smart grid neighbour area network based wireless mesh network using multiple metrics," *Energies*, vol. 10, no. 3, p. 287, Feb. 2017.

[10] J.-S. Jung, K.-W. Lim, J.-B. Kim, Y.-B. Ko, Y. Kim, and S.-Y. Lee, "Improving IEEE 802.11s wireless mesh networks for reliable routing in the smart grid infrastructure," in *Proc. IEEE Int. Conf. Commun. Workshops (ICC)*, Jun. 2011, pp. 1–5.

[11] J. Kim, D. Kim, K.-W. Lim, Y.-B. Ko, and S.-Y. Lee, "Improving the reliability of IEEE 802.11s based wireless mesh networks for smart grid systems," *J. Commun. Netw.*, vol. 14, no. 6, pp. 629–639, Dec. 2012.

[12] X. Deng, L. He, C. Zhu, M. Dong, K. Ota, and L. Cai, "QoS-aware and load-balance routing for IEEE 802.11s based neighborhood area network in smart grid," *Wireless Pers. Commun.*, vol. 89, no. 4, pp. 1065–1088, Aug. 2016.

[13] X. Deng, Q. Peng, L. He, and T. He, "Interference-aware QoS routing for neighbourhood area network in smart grid," *IET Commun.*, vol. 11, no. 5, pp. 756–764, Mar. 2017.

[14] X. Deng, L. He, X. Li, Q. Liu, L. Cai, and Z. Chen, "A reliable QoS-aware routing scheme for neighbor area network in smart grid," *Peer-to-Peer Netw. Appl.*, vol. 9, no. 4, pp. 616–627, Jul. 2016.

[15] J. Astudillo León and L. de la Cruz Llopis, "A joint multi-path and multi-channel protocol for traffic routing in smart grid neighborhood area networks," *Sensors*, vol. 18, no. 11, p. 4052, Nov. 2018.

[16] K. A. Khaliq, A. Qayyum, and J. Pannek, "Performance analysis of proposed congestion avoiding protocol for IEEE 802.11s," *Int. J. Adv. Comput. Sci. Appl.*, vol. 8, no. 2, pp. 356–369, 2017.

[17] R. Boutaba, M. A. Salahuddin, N. Limam, S. Ayoubi, N. Shahriar, F. Estrada-Solano, and O. M. Caicedo, "A comprehensive survey on machine learning for networking: Evolution, applications and research opportunities," *J. Internet Services Appl.*, vol. 9, no. 1, p. 16, Dec. 2018.

[18] D. Barman and I. Matta, "Model-based loss inference by TCP over heterogeneous networks," in *Proc. WiOpt*, 2004, pp. 364–373.

[19] I. El Khayat, P. Geurts, and G. Leduc, "Improving TCP in wireless networks with an adaptive machine-learnt classifier of packet loss causes," in *Proc. Int. Conf. Res. Netw.* Berlin, Germany: Springer, 2005, pp. 549–560.

[20] I. El Khayat, P. Geurts, and G. Leduc, "Enhancement of TCP over wired/wireless networks with packet loss classifiers inferred by supervised learning," *Wireless Netw.*, vol. 16, no. 2, pp. 273–290, Feb. 2010.

[21] P. Geurts, I. El Khayat, and G. Leduc, "A machine learning approach to improve congestion control over wireless computer networks," in *Proc. 4th IEEE Int. Conf. Data Mining (ICDM)*, Nov. 2004, pp. 383–386.

[22] J. Liu, I. Matta, and M. Crovella, "End-to-end inference of loss nature in a hybrid wired/wireless environment," WiOpt, Model. Optim. Mobile, Ad Hoc Wireless Netw., Sophia Antipolis, France, Tech. Rep. nria-00466774, Mar. 2003, p. 9.

[23] Y. Sun, M. Peng, Y. Zhou, Y. Huang, and S. Mao, "Application of machine learning in wireless networks: Key techniques and open issues," *IEEE Commun. Surveys Tuts.*, vol. 21, no. 4, pp. 3072–3108, 4th Quart., 2019.

[24] J. Yang, J. Shen, and M. Ying, "Forwarding with prediction over machine learning based nodes in wireless mesh networks," *Trans. Netw. Commun.*, vol. 6, no. 6, p. 33, Dec. 2018.

[25] J. Yang, J. Shen, P. Guo, B. Payne, and T. Wei, "A machine learning based forwarding algorithm over cognitive radios in wireless mesh networks," in *Proc. Int. Conf. Mach. Learn. Intell. Commun.* Cham, Switzerland: Springer, 2016, pp. 228–234.

[26] M. Wang, S. Wang, and B. Zhang, "APTEEN routing protocol optimization in wireless sensor networks based on combination of genetic algorithms and fruit fly optimization algorithm," *Ad Hoc Netw.*, vol. 102, May 2020, Art. no. 102138.

[27] A. Sawhney, R. Bhatia, and P. Mahajan, "Congestion control in wireless communication network using fuzzy logic and machine learning techniques," *Int. J. Adv. Res. Electr., Electron. Instrum. Eng.*, vol. 3, no. 11, pp. 12825–12832, Nov. 2014.

[28] J. P. A. León and L. J. de la Cruz Llopis, "Emergency aware congestion control for smart grid neighborhood area networks," *Ad Hoc Netw.*, vol. 93, Oct. 2019, Art. no. 101898.

[29] J. P. A. L.ón, T. Begin, A. Busson, and L. J. D. L. C. Llopis, "A fair and distributed congestion control mechanism for smart grid neighborhood area networks," *Ad Hoc Netw.*, vol. 104, Jul. 2020, Art. no. 102169.

[30] W. McKinney, "Data structures for statistical computing in Python," in *Proc. 9th Python Sci. Conf. (SciPy)*, S. van der Walt and J. Millman, Eds. Austin, TX, USA: 2010, pp. 51–56.

[31] G. Van Rossum and F. L. Drake, *Python Reference Manual*. Amsterdam, The Netherlands: Centrum voor Wiskunde en Informatica, 1995.

[32] G. Van Rossum and F. L. Drake, *Python 3 Reference Manual*. Scotts Valley, CA, USA: CreateSpace, 2009.

[33] G. Dileep, "A survey on smart grid technologies and applications," *Renew. Energy*, vol. 146, pp. 2589–2625, Feb. 2020.

[34] L. Rokach and O. Z. Maimon, *Data Mining With Decision Trees: Theory and Applications*, vol. 69. Singapore: World Scientific, 2008.

[35] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, "Scikit-learn: Machine learning in Python," *J. Mach. Learn. Res.*, vol. 12, pp. 2825–2830, Oct. 2011.

[36] A. Tharwat, "Classification assessment methods," in *Applied Computing and Informatics*. Bingley, U.K.: Emerald Publishing Limited, Aug. 2018, pp. 1–25.

[37] J. Cohen, "A coefficient of agreement for nominal scales," *Educ. Psychol. Meas.*, vol. 20, no. 1, pp. 37–46, Apr. 1960.

[38] L. L. Cárdenas, J. P. A. León, P. A. B. Bautista, and M. A. Igartua, "Tips and tools to automate OMNeT++ simulations and to facilitate post data management tasks," UPCommons, Barcelona, Spain, Tech. Rep., 2020.

[39] J. Lokhorst, "The lasso and generalised linear models," Honors Project, Univ. Adelaide, Adelaide, SA, Australia, 1999.

[40] F. Chollet. (2015). *Keras*. [Online]. Available: https://keras.io

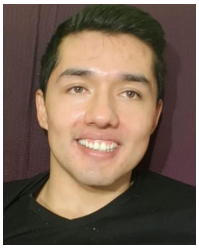
[41] R. Reed and R. J. MarksII, *Neural Smithing: Supervised Learning in Feedforward Artificial Neural Networks*. Cambridge, MA, USA: MIT Press, 1999.

[42] X. Glorot, A. Bordes, and Y. Bengio, "Deep sparse rectifier neural networks," in *Proc. 14th Int. Conf. Artif. Intell. Statist.*, 2011, pp. 315–323.

[43] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. Cambridge, MA, USA: MIT Press, 2016.

[44] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," 2014, *arXiv:1412.6980*. [Online]. Available: http://arxiv.org/abs/1412.6980

[45] Y. Yao, L. Rosasco, and A. Caponnetto, "On early stopping in gradient descent learning," *Constructive Approximation*, vol. 26, no. 2, pp. 289–315, Aug. 2007.
[46] L. Prechelt, *Early Stopping—But When?* Berlin, Germany: Springer, 2012, pp. 53–67.
[47] G. Raskutti, M. J. Wainwright, and B. Yu, "Early stopping and non-parametric regression: An optimal data-dependent stopping rule," *J. Mach. Learn. Res.*, vol. 15, pp. 335–366, Jan. 2014.
[48] D. Stathakis, "How many hidden layers and nodes?" *Int. J. Remote Sens.*, vol. 30, no. 8, pp. 2133–2147, Apr. 2009.
[49] R. Sebastian and M. Vahid, *Python Machine Learning Machine Learning and Deep Learning With Python Scikit-Learn and TensorFlow 2*. Birmingham, U.K.: Packt Publishing Ltd, 2019.
[50] Ai-Thinker Technology. *ESP-8266*. Accessed: Sep. 10, 2020. [Online]. Available: https://www.espressif.com/en/products/socs/esp8266ex/overview
[51] Espressif Systems. *ESP-32*. Accessed: Sep. 10, 2020. [Online]. Available: https://www.espressif.com/en/products/socs/esp32/overview



**JUAN PABLO ASTUDILLO LEÓN** received the degree in electronics engineering from the Universidad Politécnica Salesiana, Cuenca, Ecuador, in 2012, the degree in telecommunications technologies and the master's degree in telecommunications engineering from the Universidad de Buenos Aires, Buenos Aires, Argentina, in 2014 and 2015, respectively, and the Ph.D. degree from the Networking Department, Universitat Politècnica de Catalunya, Barcelona, Spain, in 2020. He was an Assistant Professor with the Universidad Politécnica Salesiana in 2016. His research interests include wireless mesh networks, wireless mobile ad hoc networks, smart grids, traffic engineering, machine learning, and deep learning.

**FRANCISCO J. RICO-NOVELLA** received the degree in telecommunication engineering and the Ph.D. degree in telecommunications engineering from the Universitat Politècnica de Catalunya (UPC), Barcelona, Spain, in 1989 and 1995, respectively. He is currently an Associate Professor with the Department of Network Engineering, UPC. His current research interests include cryptography and the IoT smart services.

**LUIS J. DE LA CRUZ LLOPIS** received the degree in telecommunication engineering and the Ph.D. degree in telecommunications engineering from the Universitat Politècnica de Catalunya (UPC), Barcelona, Spain, in 1994 and 1999, respectively. He is currently an Associate Professor with the Department of Network Engineering, UPC. His current research interests include wireless networks and the IoT smart services.