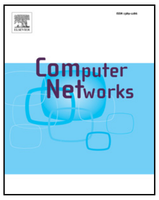
Computer Networks 127 (2017) 68–80

Contents lists available at ScienceDirect

Computer Networks

journal homepage: www.elsevier.com/locate/comnet



Ensemble network traffic classification: Algorithm comparison and novel ensemble scheme proposal

Santiago Egea Gómez*, Belén Carro Martínez, Antonio J. Sánchez-Esguevillas, Luis Hernández Callejo

Escuela Técnica Superior de Ingenieros de Telecomunicación, Universidad de Valladolid, Campus Miguel Delibes, Valladolid 47011, Spain

ARTICLE INFO

Article history:
Received 11 March 2017
Revised 19 July 2017
Accepted 25 July 2017
Available online 9 August 2017

ABSTRACT

Network Traffic Classification (NTC) is a key piece for network monitoring, Quality-of-Service management and network security. Machine Learning algorithms have drawn the attention of many researchers during the last few years as a promising solution for network traffic classification. In Machine Learning, ensemble algorithms are classifiers formed by a set of base estimators that cooperate to build more complex models according to given training and classification strategies. Resulting models normally exhibit significant accuracy improvements compared to single estimators, but also extra time cost, which may obstruct the application of these methods to online NTC. This paper studies and compares the performance of seven popular ensemble algorithms based on Decision Trees, focusing on model accuracy, byte accuracy, and latency to determine whether ensemble learning can be properly applied to this modeling task. We show that some of the studied algorithms overcome single Decision Tree in terms of model accuracy and byte accuracy. However, the notable latency increase hinders the application of these methods in real time contexts. Additionally, we introduce a novel ensemble classifier that exploits the imbalanced populations presented in traffic networks datasets to achieve faster classifications. The experimental results show that our scheme retains the accuracy improvements of ensemble methods but with low latency punishment, enhancing the prospect of ensembles methods for online network traffic classification.

1. Introduction

In the age of the Internet, vast amounts of devices are interconnected continuously exchanging information through data networks. The exponential growth of network traffic hinders Internet Services Providers (ISPs) to manage their infrastructures efficiently and Network Traffic Classification (NTC) plays a crucial role for this task. Traffic monitoring has attracted the attention of many researchers, and Machine Learning (ML) has shown to provide successful solutions in this area [1,2]. NTC allows network administrators to reallocate resources (e.g. underutilizing links capacity) and reconfigure network parameters (e.g. disable or enable firewall ports) to prevent Quality of Services (QoS) decays or to react to malicious behaviors [1]. As inspecting all connection flows manually is certainly unfeasible, many researchers have endeavored to develop techniques for effective NTC [2]. Network traffic classifiers aim to automatically identify traffic applications that are being used at a given instant.

NTC has to be carried out accounting for several requirements, traffic classifiers must accurately identify connection flows but, in real time conditions, there are other crucial aspects as:

- **Scalability.** Traffic classifiers will be implemented in network devices where huge amounts of packets from different users go through, so scalable classifiers are needed to manage these amounts of information [2].
- **Memory resources.** Due to memory limitations in network nodes, classifiers must only store the most relevant information to classify applications correctly and drop variables that are not useful [2].
- **Latency.** Identification process must be as fast as possible to determine applications that correspond to each flow before it ends or an anomaly event causes QoS flaws in the network [1].
- **Privacy.** Privacy policies force network traffic classifiers not to use sensitive information obtained from users. This fact limits considerably the available information for NTC [1,2].

Many research lines have arisen in NTC since this discipline emerged. The earliest traffic classifiers were based on the port number used by each application [1,2]. Since port-based tools only observe port numbers used by each connection flow without any information storage, they are the simplest and fastest classifiers.

* Corresponding author.

E-mail addresses: santiago.egea@alumnos.uva.es (S.E. Gómez), belcar@tel.uva.es (B.C. Martínez), antoniojavier.sanchez@uva.es (A.J. Sánchez-Esguevillas), luis.hernandez.callejo@uva.es (L. Hernández Callejo).

http://dx.doi.org/10.1016/j.comnet.2017.07.018
1389-1286/© 2017 Elsevier B.V. All rights reserved.

However, emerging applications have no fixed ports or use different port numbers while they are running, deteriorating the accuracy of port-based classifiers. Deep Packet Inspection (DPI) tools have appeared to overcome the former limitations [2]. DPI tools inspect packet payloads in order to check byte strings for matches with prefixed patterns. DPI based approaches give accurate results, but they also have critical limitations. DPI tools need to store packet contents and inspect them, thereby their memory consumption and latency increase excessively. Additionally, databases, which contain patterns associated with each application, must be maintained and updated with zero-day applications. The maintenance of these databases is quite arduous owing to the vertiginous increase in the number of Internet protocols and applications, and encrypted traffic also complicates pattern inspections. Finally, privacy policies constrain capacity of third parties to carry out lawful deep packet inspection [1]. In this line, ML is opening the ways to develop sophisticated network traffic classifiers, which achieve an acceptable tradeoff between computation complexity and accuracy respecting users' privacy.

In ML, ensemble algorithms are complex structures formed by sets of single estimators, called base estimators, which cooperate with each other according to training and classification strategies. A large number of studies have revealed the advantages of these methods in many diverse areas and this paper aims to assess the suitability of these algorithms for NTC. Since Decision Tree algorithms are one of the most suitable learning algorithm for online NTC [1,3,4], this work focuses on ensemble algorithms based on Decision Trees (DTs). Despite of their high computational complexity compared to single estimators, ensemble methods may provide more accurate predictive models. As no study of clear ensemble learning for online NTC has been provided yet, seven of the most popular ensemble algorithms are compared focusing on their capabilities to be applied to this issue in this paper. We evaluate classification accuracy metrics, but also assess the computational load of each candidate. The experimental results show that ensemble algorithms exhibit higher training and classification times than a single DT, which could obstruct their implementation in real time classifiers. As possible solution, we introduce a novel ensemble scheme that consists of a sequential chain of DTs, each DT acts as connection flow filter of its successor avoiding unnecessary and repetitive classifications to decrease training and classification times.

The remainder of the article is organized as follows. Section 2 reviews relevant previous works in NTC. The methodology followed to perform our experiments and our ensemble algorithm are described in Section 3. We present and discuss the results obtained in each experiment in Section 4. Finally, the relevant conclusions of this work are presented in Section 5.

2. Related work

The last trendy applications in NTC are based on ML algorithms. The fast-paced developments in ML have encouraged to research on these techniques in a wide number of research areas, an illustrative case is the use of clustering algorithms and Neural Networks for forecasting electricity demands [5,6]. Although several challenging issues must be overcome yet to accomplish efficient network traffic classifiers [2], ML provides promising results for online NTC. Internet traffic identification based on ML consists of various processes, learning algorithms are trained using knowledge, which is previously acquired from captured network traces and recorded on a dataset. Dataset construction is a complicated process that dramatically affects the accuracy of traffic classifiers and their computational complexity. In online contexts, packet acquisition, training and classification times are prominent to get feasible classifiers. The main reasons why ML algorithms are excellent candidates as core of modern NTC systems are their capability of identifying network traffic respecting userśprivacy rights, their ability to handle encrypted traffic and also their capacity to be less computationally weighted than DPI tools retaining acceptable accuracies [1,2].

Two leading learning approaches are distinguished in ML: supervised and unsupervised learning. The main difference between both approaches is that supervised learning requires a labeling process using prior knowledge about the problem in order to establish a ground truths for each connection flow; whereas training unsupervised algorithms do not need to assign application labels to each flow, and they are able to cluster classes automatically. Some researchers have adopted one of these perspectives for NTC, but hybrid techniques, known as semi-supervised approaches, have been also applied showing interesting results. This work focuses on supervised learning, namely in DTs-based algorithms.

The first relevant works in NTC [7,8] demonstrated that application flows can be accurately identified by computing statistical attributes using few packets when connection flows start, introducing the concept of early stage classification. The authors used the first packets of TCP flows to compute instances, and they trained classifiers based on clustering methods. Although promising results were reported in terms of accuracy, they did not assess training and classification speed of their proposals. More recently, [9] has studied the efficient number of packets to perform early application identification. The authors used packet-size-based features extracted from bidirectional flows to train standard ML algorithms, including some ensemble algorithms also considered in this work (ADA Boosting and Bagging algorithm). They concluded that the optimal number of packets to correctly classify TCP flows is 5–7 and it depends on network environments. Also [4] studied how many packets could be considered to classify internet applications. They used 12 features to identify encrypted flows and compared C4.5 DT to ADA Boosting algorithm with C4.5 DT as base estimator, only one ensemble algorithm is considered in this work.

The earliest comparison among supervised learning algorithms for NTC was carried out by [3]. They compared performances between standard algorithms: Bayesian Network, C4.5 decision tree, Naïve Bayes and Naïve Bayes Trees. Furthermore, they showed that Feature Selection (FS) algorithms based on correlation measures, such as Consistency-based FS, are more suitable for NTC datasets than other approaches. Also, they found that C4.5 Decision Trees exhibited the best performances in accuracy and classification speed. [3] is one of the earliest studies that compares computational costs of ML algorithms for NTC. Later, [10] evaluates classifier performances focusing on Accuracy and Recall scores, and also on classification rate and build time (or training time). Additionally, [10] observed the influence of the composition of training data and the effect of configuring dynamic-port applications on algorithm accuracies. They trained Bayesian Networks and DT algorithms showing that sample composition of training dataset affects considerably classifier performances. Finally, they discussed the importance of labeling correctly connection flows.

Many authors have developed sophisticated ML classifiers to solve open issues in NTC. In [11], the authors combined weak learning algorithms to get more accurate predictive models, this classifier is a clear example of ensemble algorithm. Furthermore, they showed that differences among network scenarios affect classifier performances (type of applications and protocols detected, traffic distributions, link capacities and so on). Another example of ensemble algorithm is presented in [12] exploiting Sub-Space Clustering, Evidence Accumulation and Hierarchical Clustering concepts. In this work a semi-supervised approach is presented to create applications groups using clustering algorithms and assign network services labels to unknown connections in a supervised fashion. In order to create robust application groups the authors combined several clustering models using different partitions of the same dataset. A Flow-level ML classifier scheme is presented in [13], the authors designed a modular architecture for High-speed links traffic classification using *m* ensemble classifiers. Namely, they used OneVsRest strategy, which is also considered in this paper, but they did not assess latency of their proposal. In [14], a Robust Network Traffic classifier is presented whose main goal is to identify zero-day applications. The authors provided a parameter optimization process and compared their algorithm to Random Forest, correlation-based classification, semi-supervised clustering and Support Vector Machines, showing that the Robust Traffic Classifier overcomes other approaches. In [15], the authors proposed a self-learning classifier that starts with small number of training instances and retrains itself to improve the model performances. They implemented a decision maker to extend the number of samples in training datasets using Random Forest algorithm. Accuracy improvements were reported in each retraining iteration. Other important open topic in NTC is the effect of subflow sampling over classifier performances, which is discussed in [16] and [17]. Additionally, Naïves Bayes, Bayesian Neural Networks and Support Vector Machines algorithms are independently studied in [18,19] and [20]. For a more general literature revision of Internet Classification area we suggest [21] and [22]. In [21] the authors review operational aspect of traffic analysis and its state-of-the-art, finally they discuss and compare relevant contributions for internet application identification, including DPI and port-based techniques; and several classification approaches are reviewed and compared using seven different network traces in [22].

Although some previous works evaluated ensemble classifiers for NTC ([4,11,14,15]), none of them has compared standard ensemble methods focusing on both, accuracy and latency performances. This work tries to fill this gap by comparing ensemble schemes through several experiments assessing accuracy and latency. Additionally to ensemble algorithms comparison, a novel ensemble scheme is presented to reduce training and classification times while retaining accuracy improvements of ensemble learning respect to single DTs. For more generality, our experiments have been performed using network traffic captured in two quite different environments, three traces were captured in an ISP backbone network and the other three were captured in host computers simulating human behaviors. Below we present the methodology followed in the experiments.

3. Methodology

This section describes the methodology used in the experiments and presents the ensemble algorithms considered (Section 3.4), as well as our ensemble scheme (Section 3.5). All programs used in this paper were developed using *Python2.7*, the library *Scikit-Learn* implements the ML algorithms studied and the network traffic traces were processed using *Scapy. Scikit-Learn* is a well-known ML library maintained by hundreds of users and whose usage is spreading over numerous research communities. Although other tools are preferred in production due to their lesser computational complexity, this library is a suitable choice for experimental and prototyping tasks as ours.

3.1. Datasets

For our experiments we have collected six network traces captured in two quite different environments. The Internet traffic that goes through different networks differs notably between environments in the type of applications found and their distribution, it depends on the usage of network services by users and on type of entity that is serviced (enterprises, educational institutions, private houses and so on). Imbalanced label distributions in datasets significantly affect the performances of learning algorithms [23].

Table 1
Network traffic traces information.

	Start date	Duration	Datasize	# Packets	# Flows
ISP-1	17/01/2017	298 seconds	12.12 GB	8,863,530	231,137
ISP-2	25/01/2017	259 seconds	16.96 GB	12,293,836	266,165
ISP-3	25/01/2017	280 seconds	17.64 GB	12,966,391	307,605
HOST-1	25/02/2013	~59 days	9438 MB	5,062,825	121,293
HOST-2	25/02/2013	~32 days	22 GB	21,000,000	245,627
HOST-3	25/02/2013	~65 days	7113 MB	7,203,000	744,814

Thus, using several traffic traces extracted from different network environments helps to get a better understanding of the performances of traffic classifiers. Next, we introduce the network traces employed.

3.1.1. ISP traces

The ISP traces were shared by an organization that provides Internet connection to more than two millions of users across Spain. The network traffic was captured at a node in the ISP network backbone where traffic rates of 7 GB/s are supported at high load hours. Tcpdump was employed for capturing data through a port mirror for redirecting network packets and each trace lasts approximately five minutes. The processing of these traces was performed respecting privacy rights of users in a server enabled for this purpose. These traces have been captured recently, thus the presence of encrypted applications and the latest protocols is ensured. At the request of the traces providers, the name does not appear explicitly in this work due to privacy concerns.

3.1.2. HOST traces

Privacy policies obstruct the possibility of sharing network traces with the application layer from institutions or ISPs, and traces without application layers can be labeled exclusively using Port-Based tools with low trust. Due to the difficulty in getting appropriate network traffic traces, we have used three network traces manually generated in three different hosts under a controlled environment. These traces have already been used in others works to validate DPI tools [24]. The information about the network captures is shown in Table 1.

3.1.3. Attributes generation, feature selection and labeling process

An ad-hoc developed tool of our own was developed to extract the datasets to feed the ML algorithms from the network traces. Our software takes as input network captures stored in pcap files and the number of packets to be considered to compute the statistical attributes. Our tool is able to split initial pcap files in traces that contain packets associated with each bidirectional connection flow. Once each flow is completely stored in its corresponding trace file, they are processed to compute instances with their associated application label. The output is a dataset that contains 77 statistics regarding number of packets, packets sizes, inter-arrival packet times, TCP windows and so. The whole collection of attributes is presented at the end of this paper in Annex 1 and it includes statistics accounting for outgoing, ingoing and both directions of flows. In our experiments only the first five packets at the beginning of each flow were used to compute all statistical attributes. The datasets employed for our experiments are publicly accessible via emailing to the authors. Because correlation-based filters have proven to be a proper FS algorithms in NTC, we have applied one of these algorithms in order to reduce the attribute space for all datasets.

For label assignment, we used a DPI tool called nDPI [25], publicly available in [26]. NDPI is able to handle encrypted traffic and is one of the most accurate open source DPI applications [27]. This tool identifies web services, as YouTube or Google, along with an

Table 2
Datasets information. %I denotes the percentage of instances in the dataset and %B the percentage of Bytes in the network capture.

	P2P		WWW		DNS		INT		S/C		Bulk		Media		NTP		DB	
	%I	%B	%I	%B	%I	%B	%I	%B	%I	%B	%I	%B	%I	%B	%I	%B	%I	%B
ISP-1	0.17	<0.01	80.46	99.60	16.28	0.08	1.99	0.10	0.73	0.24	–	–	–	–	0.37	< 0.01	–	–
ISP-2	–	–	75.30	99.60	21.50	0.10	2.52	0.12	0.35	0.18	–	–	–	–	0.34	<0.01	–	–
ISP-3	–	–	71.70	99.60	24.98	0.11	2.66	0.12	0.37	0.22	–	–	–	–	0.29	< 0.01	–	–
HOST-1	33.00	15.90	32.83	27.61	9.12	0.09	10.30	2.73	5.96	0.06	5.72	23.71	3.07	29.9	–	–	–	–
HOST-2	14.30	7.90	17.10	11.80	7.21	0.04	55.40	67.1	1.06	0.01	3.43	6.22	1.50	6.93	–	–	–	–
HOST-3	2.01	38.10	8.31	39.06	79.81	4.05	4.94	2.58	–	–	0.68	6.01	0.42	9.38	3.73	0.79	0.10	0.03

extended number of protocols. However nDPI was not capable of labeling all flows in our traces and some flows were labeled as unknown. Unknown flows were depreciated in this work, since applications marked as unknown could not be determined with certainty. In other cases, some encrypted flows were identified as SSL, and port-based information was examined to distinguish between HTTPS traffic and others, as encrypted SSH connections. Both, UDP and TCP flows, were employed in our experiments. Finally, different applications and protocols were detected among the six network traces; and each application was mapped to an application group according to its protocol properties and purposes, except DNS and NTP. The application grouping was carried out according to the following protocol types: P2P includes applications as eMule, BitTorrent or eDonkey; WWW includes all HTTP and HTTPS queries to Google, Facebook, GMail and other websites; INT (INTeractive) includes protocols as SSH, Telnet, RDP and so on; Services & Control (S/C) includes network control protocols and other services as NetBios, Radius, Kerberos and so forth; Bulk includes FTP and similar protocols; Media traffic includes RTP, Skype and so on; and DB includes MsSQL, MySQL and more database applications. Other applications, as email protocols, were detected and also depreciated from this study due to their low populations.

Table 2 shows the traffic distributions found in our datasets after label assignment. Note that the network traffics are highly imbalanced according to the percentage of instances per class (%I) for the two network environments under study. In the instances of ISP-1,2 and 3, more than 70% of the samples belong to WWW traffic, and the absence of P2P, Media and Bulk traffics may be due to its restricted used at educational environments. Additionally, lower accuracy performances are expected for ISP traces owing to the point of capture. In the middle of networks, packets statistics suffer from degradation due to multipath routing, packet loss and packet duplication. Also HOST datasets are highly imbalanced, note that P2P, INT and DNS are the predominant application flows in ISP-1,2 and 3 respectively. Non-uniform sample distributions, which generally characterized NTC datasets, are exploited by the novel ensemble scheme proposed in this work to achieve faster classification and training times. Finally, note that the number of samples corresponding to Bulk and Media flows are scarce in HOST datasets, nevertheless they have an important impact on network resources due to the bytes they produce to run.

The initial datasets were subjected to a Feature Selection (FS) process. Since correlation-based FS algorithms have found effective for NTC [3], three versions of the Fast Correlation-Based Feature selection algorithm [28,29] have been implemented for this process. As output the FS algorithms return rankings of attributes ordered according its relevance for the modeling task. These algorithms have been made publicly available in at [30]. Namely, we employed the FCBFiP version since it yielded the most accurate models in the preliminary results using the lesser number of attributes.

As final step, the datasets generated were split in training and validation subsets via a stratified technique to preserve the percentages of class populations in the training and validation phase. The 70% of the samples were used for training and the rest for validation. Next, parameters of base estimators were set using 10-fold stratified cross validation excluding the validation samples from this process, and no resampling techniques were employed avoiding to alter the class populations. This process was repeated for each iteration of the experiments.

3.2. Evaluation environment

All experiments were performed in a workstation with 12 GB of memory RAM and CPU AMD A10 6800 K (4.1 Ghz). Although the CPU has four cores and *Scikit-Learn* allows to train models in parallel, we used only one for our experiments in order to isolate each experiment in a unique processing core. As decision trees are sensitive to random initializations at the beginning of their learning phase, they suffer from variance when they are trained. To diminish the effects of models variance, the experiments were repeated ten times and the mean of resulted metrics is reported in Results section.

3.3. Performance metrics & statistical validation

We have studied several metrics to compare the proposed algorithms and applied a statistical validation procedure over the results as we describe below. Model performances were assessed isolating completely the validation datasets from the training processes.

3.3.1. Overall accuracy

Overall Accuracy (OA) is the percentage of samples labeled correctly. In this way, OA is defined as

$$OA = \frac{\sum TP_i}{\#Samples} \tag{1}$$

Where TP_i denotes True Positives associated with the class i and #Samples denotes the number of samples contained in the datasets.

3.3.2. Class accuracies

Because OA is the percentage of samples correctly labeled and network traffic is highly imbalanced, great precisions over high populated traffic will hide errors on application flows with low populations in the datasets (Table 2). Therefore we included the individual accuracy for each class. Thus, we define the Accuracy for a given class i as

$$A_i = \frac{TP_i}{\#Samples\ of\ class\ i} \tag{2}$$

where $\#Samples\ of\ class\ i$ is the number of samples associated with the class i.

3.3.3. Byte accuracy

OA and Class Accuracies alone could be insufficient to assess model performances, it is interesting to study how many bytes have been accurately labeled to appreciate more clearly algorithm reliabilities. Thus, as it is an insightful metric in NTC, we have included the Byte Accuracy metric (BA) in our comparison, BA is defined as

$$BA = \frac{Bytes\ labeled\ correctly}{Total\ bytes\ captured} \tag{3}$$

3.3.4. Number of features used in the models

As scalability and latency are important properties for online NTC classifiers, we have included in our results the number of statistical attributes used by each algorithm. Furthermore, we provide a model complexity evaluation to determine if ensemble algorithms are able to equal or overcome a single estimator performances using more reduced subsets, and thus, bringing computational benefits in the attribute computing phase.

3.3.5. Training and classification times

Finally, we measured Training and Classification times to quantify computational punishment of using different ensemble schemes. Although Classification time is more prominent in online NTC, novel classifiers include retraining phases, therefore, Training times are also relevant in our comparison.

3.3.6. Statistical validation: Friedman's test

Since the average of measurements obtained from experiments using different datasets might sometimes be insufficient to validate general observations, we conducted a statistical validation process to make our results more rigorous [31,32]. After measuring the previous properties for each algorithm over the six datasets studied, we have applied a well-known statistical method to compare multiple algorithms, the Friedmanś test. The Friedmanś test is a non-parametric statistical method for detecting differences amongst more than two related experiments. This procedure ranks the compared algorithms according to their results obtained for each dataset. The best scored algorithm is assigned the value 1 and the worst scored gets the value k, being k the number of compared algorithms. The Friedmanś test is computed by Eq. (4), where R_j is the average score for the algorithm j over the N datasets, with $N = 6$ for this case.

$$\chi^2{}_F = \frac{12N}{k*(k+1)}\left[\sum_j R_j^2 - 0.25k*(k+1)^2\right] \tag{4}$$

Once $\chi^2{}_F$ is computed, the p-value is obtained from a chi-squared random distribution with $k-1$ degrees of freedom. We have set the significance threshold at $\alpha = 0.05$. Thereby if the p-value is lesser than α, the null hypothesis, which states that statistical difference amongst candidates does not exist, is rejected.

In addition to the Friedmanś test, we have applied a post-hoc correction method called the Holmś procedure [32]. This method uses the adjusted p-values (APVs) to compare the performance of a control algorithm with respect to the rest, normally the control algorithm is the best scored in the Friedmanś ranking. The algorithms are sorted according to their average scores R_j, and the associated APVs are computed as $APV_j = \alpha/(k-p)$ where p is the position of the algorithm j in the ordered ranking. The value z_j is computed for each algorithm using Eq. (5), where R_i is the average score of the control algorithm in the Friedmanś test. The value z_j follows a normal distribution and its associated probability p_j can be obtained evaluating $Z(z_j)$. If $p_j < APV_J$, we conclude that a significant difference exists among the algorithm j and the control algorithm.

$$z_j = \left(R_i - R_j\right)/\sqrt{\frac{k(k+1)}{6n}} \tag{5}$$

This statistical validation procedure is similar to the methods applied in [9]. For more information about these methods read [31] and [32].

3.4. Ensemble classifiers

Ensemble classifiers are learning algorithms composed by multiple base estimators along with training and classification strategies to make final decisions [33–39]**.** Since DTs yield satisfactory results in NTC ([1,3,4]), we have selected the CART DT algorithm, provided by the *Scikit-learn* library, as base estimator for the ensemble structures. Ensemble algorithms can be distinguished according to the training and classification strategies they employ. *Scikit-learn* library contains a wide number of popular ensemble algorithms, some of these algorithms have been considered for the experiments presented in this paper. Below, we briefly describe the ensemble algorithms selected.

- **OneVsRest.** One classifier is built per class to distinguish one class from the rest [38], thereby one dataset is generated for each class to train each base estimator. Finally, unknown samples are classified according to the estimate of the posterior probability for each class: given an unknown sample, the class whose posterior probability is maximum is assigned to that sample.
- **OneVsOne.** One base estimator is trained to distinguish between two different classes excluding the rest from the training. Therefore, $n(n-1)/2$ datasets and classifiers are built for n classes. The final label is assigned by majority voting: the most voted class amongst all classifiers is the class associated with the unknown sample [38].
- **Error-Correcting Output-code (OutputCode).** One binary code is associated with each class and one classifier is trained in parallel per each bit. In classification, a new instance generates a code that is projected onto the binary space, and the closest label to the projected point is assigned to the unknown sample [36]. The code size is a design parameter that determines the number of classifiers in the model, 12 base estimators were used in this work.
- **Adaboost classifier (ADA).** This algorithm is composed by a set of weak estimators that are trained sequentially and a set of weights associated with each class [33,34]. In each training iteration, misclassified classes are awarded by increasing their associated weights. In contrast, classes with less error rate are punished decreasing their weights. Adaboost implements training and reweighting phases in its training process that speed it down considerably. Finally, Label assignment is performed via weighted majority voting. The number of estimators were set to 20 for Adaboost in the experiments to reach a right tradeoff between latency and accuracy.
- **Bagging algorithm.** In this instance a large number of base estimators are trained in parallel using different datasets [33,40]. Each dataset is generated applying bootstrap resampling and is used to train only one classifier. Majority voting strategy is used for label assignment. We set the number of base estimators to 20, since including many estimators leads to low classification and training speeds, whereas a low number of base estimators could lead to poor accuracy.
- **Random Forest (RF).** RF is a combination of several DTs, whose training process is based on the generation of random subsets from the original dataset to feed each DT [34,37]. Unlike Bagging, each subset is built by random selection of samples and attributes. The final label assignment is based on majority voting. Such as ADA and Bagging, the number of trees in the forest were set to 20.
- **Extremely Randomized Trees (ExtraTrees).** This algorithm is very similar to Random Forest but with two differences: ExtraTrees does not generate new training datasets but instead it uses the initial one; and also it does not choose the best splits, but chooses the split randomly [39]. Such as the former algorithms, the number of trees were set to 20.

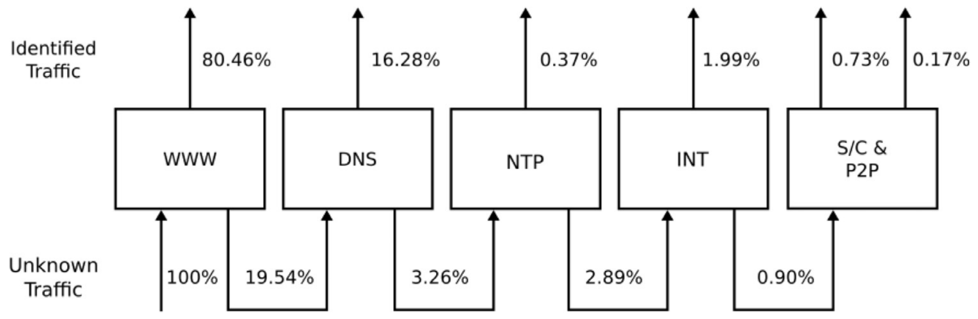


Fig. 1. Tailored decission tree chain structure for ISP-1 once the classifiers were ordered.

Finally, our proposed ensemble algorithm was considered for the comparison. We describe this novel proposal, called Tailored Decision Tree Chain (T-DTC), in the following section.

3.5. *Our proposal: tailored decision tree chain*

As it is discussed in Section 3.1, network traffic is highly imbalanced (Table 2), e.g. DNS traffic is quite more populated than Bulk, S/C and others in our datasets. This fact is not exclusive of the datasets used in this study, network traffic has been studied by several researchers showing that traffic distributions are highly imbalanced in many environments [41,42]. Furthermore, some type of flows are easier to identify than others as our experimental results show. These facts could be exploited to reach more efficient ensemble classifiers for online NTC. Next, a novel ensemble algorithm, called Tailored Decision Tree Chain (T-DTC), is introduced.

3.5.1. *Ensemble scheme & classification process*

Our proposal is based on the use of the fastest and most accurate DTs to classify and filter out samples avoiding repetitive classification of instances that are easily identified. In our scheme, a set of classifiers are sequentially ordered as a chain and trained to distinguish one traffic application from the rest, so that when T-DTC assigns an application label to an unknown sample, the connection flow is filtered out from the classification process and it is not classified in later stages. On the contrary, when T-DTC assigns the label “other” to an unknown sample, the instance passes to the ensuing classification stage to check if the sample corresponds to other application flow in the chain. This process is redundant until an application is assigned to the unknown sample and, immediately after the flow is identified, it is output from the classification process. Fig. 1 depicts this idea for the classes contained in ISP-1 once an appropriate order of classifiers was determined by the procedure described in next section**.** Note that more than 80% of the network traffic is classified by the first stage requiring being identified only by one DT; above 95% of flows are identified in the following two stages and roughly the 2% of the instances reach the two last classifiers passing through all classification stages. Note also that this scheme requires only $n-1$ classifiers (where n is the number of application flows to identify), since the last two classes share the same DT. This approach requires less classifiers than other strategies that are included in this paper (e.g. OneVsRest is composed by n classifiers, and OneVsOne uses $((n-1)*n/2))$ consuming less memory resources than other ensemble schemes. Next, we present the procedure followed to determine the proper order of DTs into T-DTC.

3.5.2. *Ordering the classifiers*

The order of the classifiers in the chain is crucial, since if inaccurate classifiers are put at the beginning of the chain, misclassified samples will not reach their corresponding DT and accuracy performances will diminish drastically. Thus, the fastest and most accurate DT must be put in the first classification stages. As the number of combinations grows exponentially as more type of applications to identify and testing all combinations is computationally weighted, we have studied the error metrics amongst classes to correctly order the classifiers in our structure. For that purpose, we trained a single DT and inspected the confusion matrix using only the training dataset for each network capture. This process considerably reduces the number of combinations considered as proper orders resulting in a bound set of choices. Finally, the order of the classifiers was chosen by assessing OA among the possibilities via cross validation.

The best order for the six datasets using this procedure were: WWW-DNS-NTP-INT-S/C-P2P for ISP-1 (as Fig. 1 shows); WWW-DNS-NTP-INT-S/C for ISP-2; WWW-DNS-INT-S/C-NTP for ISP-3; P2P-S/C-WWW-DNS-INT-Bulk-Media for HOST-1; INT-P2P-DNS-WWW-Bulk-Media-S/C for HOST-2; and NTP-DNS-INT-P2P-WWW-DB-Bulk-Media for HOST-3.

3.5.3. *Training process*

Considering n traffic applications, our algorithm generates $n-1$ datasets from the initial one to train each DT. For example, according to Fig. 1 T-DTC needs to input the whole training dataset to the first classifier, but reassigning the labels different from WWW to “other”. In the instance of the second classifier in the chain, as WWW traffic has been identified in the previous stage, the samples belonging to this traffic application have to be removed from the dataset; and samples that do not belong to DNS traffic are labeled as “other”. This process is repeated until reaching the last classifier, which do not need label reassignments. Once all datasets are generated, classifiers in chain were trained in similar way to other schemes, as OneVsOne, OneVsRest or OutputCode.

4. Results

In this section, we discuss the experimental results obtained from comparing the ensemble algorithms, including our proposal, to a single DT. In order to show a clearer comparison of the ensemble algorithms, we have remarked the model that provides the best OA score for each algorithm varying the size of the subsets according to the attribute ranking provided by the FS algorithm. We present and discuss accuracy metrics in Section 4.1, and computational time during training and classification phases in Section 4.2**;** and later, the results presented are undergone to a statistical validation procedure in Section 4.3. Below, we discuss model complexity in terms of number of statistical attributes that each classifier has to compute to accurately classify Internet traffic. In Section 4.4, we evaluate how many attributes at least each ensemble algorithm need including in its training phase to outperform or equal the best DT models. Through this experiment, we find out the models that provide better performances than DT using the less number of statistical attributes. Finally, we provide a summary of our results in Section 4.5.

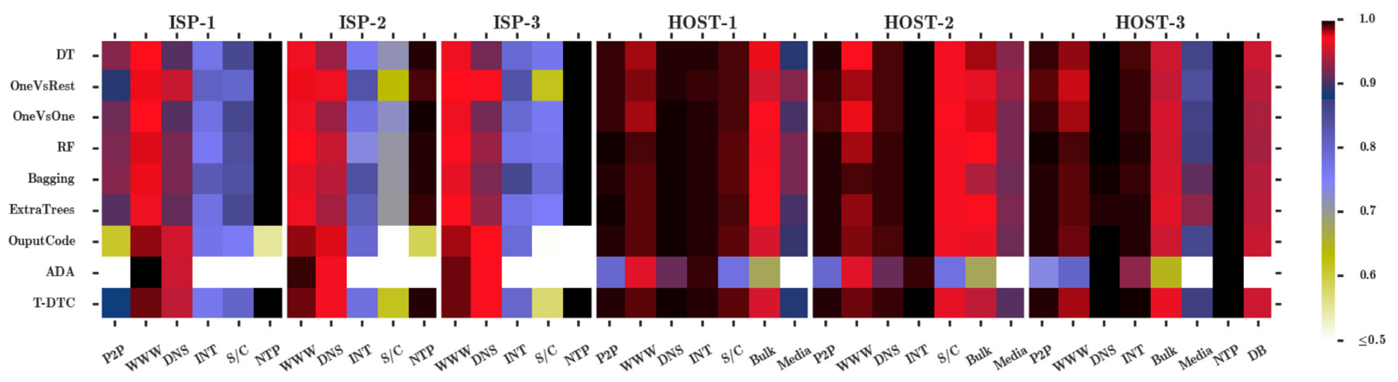


Fig. 2. Class accuracies for the algorithms and network traces studied.

Table 3
General performances for ISP traces after applying feature selection.

	OA	BA	# Features
ISP-1			
DT	0,95,685	0,99,417	9
OneVsRest	*0,96,502*	0,99,850	11
OneVsOne	0,95,680	0,99,445	8
RF	0,96,081	0,99,586	8
Bagging	0,96,122	0,99,897	10
ExtraTrees	0,95,249	0,99,336	27
OuputCode	0,97,011	*0,99,873*	19
ADA	0,95,915	0,99,698	6
T-DTC	**0,97,400**	**0,99,903**	14
ISP-2			
DT	0,95,310	0,93,077	32
OneVsRest	*0,96,681*	*0,95,995*	22
OneVsOne	0,95,384	0,95,169	30
RF	0,95,962	0,95,390	20
Bagging	0,95,467	0,95,002	12
ExtraTrees	0,95,492	0,94,860	25
OuputCode	0,97,396	0,96,047	28
ADA	0,95,587	0,94,488	13
T-DTC	**0,97,604**	**0,97,317**	22
ISP-3			
DT	0,94,817	0,95,078	22
OneVsRest	*0,96,697*	0,97,340	22
OneVsOne	0,94,845	0,95,943	21
RF	0,95,434	0,96,342	18
Bagging	0,94,889	0,95,867	13
ExtraTrees	0,95,476	0,92,762	14
OuputCode	0,97,003	*0,97,126*	23
ADA	0,95,055	0,96,480	13
T-DTC	**0,97,603**	**0,97,936**	18

Table 4
General performances for HOST traces after applying feature selection.

	OA	BA	# Features
HOST-1			
DT	0,98,652	0,94,162	30
OneVsRest	0,98,718	0,94,407	22
OneVsOne	0,98,646	**0,95,824**	30
RF	**0,99,041**	*0,95,611*	30
Bagging	0,99,005	0,95,502	29
ExtraTrees	*0,98,986*	0,95,709	30
OuputCode	0,98,840	0,94,910	30
ADA	0,92,842	0,59,624	8
T-DTC	0,98,790	0,91,688	43
HOST-2			
DT	0,99,192	**0,99,452**	32
OneVsRest	0,99,258	0,97,969	20
OneVsOne	0,99,194	*0,99,357*	16
RF	0,99,346	0,98,962	12
Bagging	0,99,330	0,98,419	12
ExtraTrees	**0,99,348**	0,98,556	14
OuputCode	*0,99,336*	0,98,118	22
ADA	0,92,842	0,89,946	9
T-DTC	0,99,319	0,99,447	38
HOST-3			
DT	0,99,742	0,95,806	38
OneVsRest	0,99,702	0,95,518	42
OneVsOne	0,99,750	0,95,263	42
RF	**0,99,813**	0,96,934	42
Bagging	0,99,590	0,96,492	43
ExtraTrees	0,99,363	0,97,793	42
OuputCode	0,99,795	*0,97,057*	30
ADA	0,97,526	0,87,591	35
T-DTC	*0,99,792*	**0,97,832**	21

4.1. Overall accuracy and byte accuracy evaluation

Table 3 contains the results obtained using the three datasets captured in the ISP backbone and Fig. 2 depicts graphically the class accuracies obtained for all network traces. Fig. 2 is a colormap that represents the accuracies obtained for each traffic class detected in the six network traces. The horizontal axis contains the applications found in each network traces, meanwhile each row of the vertical axis corresponds to each algorithm. In the case of ISP-1, we observe that the best results in terms of Overall Accuracy (OA) were provided by T-DTC, followed by OutputCode and OneVsRest. These three algorithms improve the accuracy for high populated traffic, WWW and DNS, resulting in higher OA scores. Ensemble algorithms generally overcome a single DT with the exception of OneVsOne and ExtraTrees. Although ADA yields high accuracy scores for WWW and DNS traffics, its performances over the rest of traffic applications are very poor affecting negatively the OA score. Examining the Byte Accuracy scores (BA), we find that the highest performances are provided by T-DTC, Bagging and OutputCode. In general, the learning algorithms yield similar results for the three ISP datasets, the three highest OA and BA for ISP-2 were provided by T-DTC, OutputCode and OneVsRest. Finally, the most accurate models in terms of OA for ISP-3 were T-DTC, Outputcode and OneVsRest; and, in terms of BA, OneVsRest outperforms OutputCode and T-DTC remains as the best score.

Table 4 shows the accuracy scores obtained for the network traffic captured simulating host activity artificially. The best algorithm for HOST-1 in terms of OA was Random Forest, followed by Bagging and Extremely Randomized Trees. Observing Table 4 and Fig. 2 we find that the greater accuracies identifying the high populated classes (P2P, WWW and INT) compared to a single DT result in OA improvements for these three algorithms. In general, the OA using a single DT is improved by most of ensemble algorithms, only ADA Boosting and OneVsOne got worse OA for HOST-1. If we focus on BA the observations change, the winner algorithm is OneVsOne nearly followed by ExtraTrees and RF. Finally, ADA and T-DTC yielded poor byte accuracies due to accuracy diminishing for Bulk and Media flows. In the instance of T-DTC, the accuracy loss was caused by error propagation between classifiers in the chain, when an application flow is misclassified in the first stages, T-DTC will yield low accuracy for that; error propagation is discussed below in this section. For HOST-2, the three most accurate models in terms of OA were provided by ExtraTrees, RF and OutputCode, respectively; while ADA yielded very poor results as in the former network trace. The accuracy improvements for the predominant application flows, especially WWW, result in the most of ensemble algorithms overcoming single estimator OA. Observing Byte Accuracy, none of the ensemble methods overcomes a single DT, the only ensemble algorithm that provides Byte Accuracy near to DTś are T-DTC and OneVsOne for HOST-2. This BA reduction is owing to the fact that most of the ensemble methods yield worse results than a DT for Bulk and Media traffic, whose impact over the byte distribution is decisive (see Table 2). In the case of T-DTC, the accuracy loss over Bulk and Media is offset by a better interactive (INT) traffic identification. In the instance of HOST-3, we can observe that the greatest OAs were provided by RF, OutputCode and T-DTC. Whereas the best BAs were resulted from training T-DTC, ExtraTrees and OutputCode, respectively. Although in terms of OA the ensemble models do not seem to provide significant improvements for HOST-3, the BA for these algorithms is notably better than DTś.

Comparing the results obtained in the two network environments, we can observe that ensemble learning generally provides better results in terms of Overall Accuracy and Byte Accuracy than a single DT for ISP and HOST traces. Only ADA algorithm exhibited worse performances than a single DT for most of cases studied making its application to NTC not recommended. Also, we can observe that perfect ensemble algorithm that performs clearly better than other candidates for all traces is not found. However, T-DTC performances are the best or very close to the best for all traces except HOST-1. The excellent performances achieved in most cases is due to when the OA and Class Accuracies are higher for a given dataset, less errors propagate from the first classification stages of T-DTC to following stages, meaning that consistency of labels contained in datasets is a determinant fact for T-DTC. Unlike HOST-2,3 and ISP-1,2,3, T-DTC suffers from deterioration of BA when it is trained with the dataset HOST-1. This performance decay is due to the poor Bulk and Media accuracies for this dataset, note that Bulk and Media traffic populate an important percentage of Bytes in this network capture (see Table 2), and thus the errors committed over these traffic flows have a major influence on the general performances of classifiers. Other remarkable observation is that when an application flow is more populated in a NTC dataset, classifiers normally exhibit better performances on it than other traffics, being class distributions an important fact for NTC. Finally, a substantial difference is found between the datasets obtained from a host (HOST-1,2,3) and datasets obtained from ISP (ISP-1,2,3). Due to the fact that ISP-1,2,3 were captured at a point placed in the middle of the backbone, the statistical attributes have higher variance hindering the general performances in terms of accuracy metrics, especially for P2P, DNS, S/C and INT flows. Note that high populated classes in HOST and ISP traces, as WWW and DNS, are more resilient to high variance of the statistical attributes. Furthermore, ISP traces are more cutting edge, and consequently, the presence of encrypted connections is higher than in HOST traces complicating the identification of application flows, as WWW or INT, which permit the use of encryption protocols.

4.2. Time performance comparison

Table 5 contains the results obtained from measuring the computational times in both, Training and Classification phases, for the models included in Tables 3 and 4. Since ADA yielded poor performances for OA and BA, it has been depreciated from this discussion. As expected, the fastest algorithm is a single DT compared to ensemble classifiers for all traces due to its lesser complexity.

In the case of ISP datasets, T-DTC provided the fastest ensemble model in classification and training, even T-DTC exhibited lesser training time than a DT for ISP-2. OneVsRest and OutputCode are the second and third fastest ensemble algorithms in classification, although their training phases are much longer than T-DTCś. OneVsOne, RF, Bagging and ExtraTrees exhibited quite long classification times being more than ten times slower than a single DT.

Focusing on HOST traces, the fastest ensemble algorithm is anew T-DTC for Training and Classification phases for the datasets HOST-1 and HOST-3. In the instance of HOST-2, T-DTC is overcome by ExtraTrees in Training, although T-TDC remains providing the best Classification Time. Although ExtraTrees retains a reasonable Training Time for HOST traces, its classification phase is quite more complex resulting in long Classification Times. OneVsRest yielded the second fastest Classification Time for all HOST traces, however its training phase is longer than other classifiers that provide more accurate models (see Table 4), as Random Forest, ExtraTrees and T-DTC. The slowest algorithms in Training are Bagging and OutputCode, but in Classification they spend less time than OneVsOne and ExtraTrees.

The cost of employing ensemble algorithms in NTC is clear as it is shown in Table 5, all ensemble algorithms suffer a latency increase in Training and Classification. Although the time punishment is higher if the network capture contains more connection flows, it is very different among ensemble algorithms. Algorithms formed by a huge number of classifiers exhibited a significant increase in their times, this is the case of OneVsOne, RF, Bagging and ExtraTrees; unlike them, OneVsRest and T-DTC do not suffer from huge time increases, since they are formed by a fewer number of classifiers. Because they are composed by a similar number of base estimators (for n classes OneVsRest trains n estimators and T-DTC trains $n-1$), they spend similar times in the classification task, being T-DTC always faster due to its classification strategy and more accurate for the majority of the analyzed traces.

4.3. Statistical validation

In this Section we present the results obtained from assessing the statistical significance of the performances presented in Tables 4 and 5. Table 6 shows the Friedmanś test scores along with the p-values and APVs obtained by applying the Holmś procedure. Note that statistical differences exist between algorithms for almost all performances with less than 0.05 level of significance. Only Byte Accuracy obtained a p-value greater than α. Although BA p-value is greater than 0.05, it is lesser than 0.1 thus retaining high relevant differences among algorithms.

For the Overall Accuracy, the three best performances are obtained by OutputCode, T-DTC and RF respectively. Setting OutputCode as control algorithm for the Holmś procedure, we find that no relevant statistical differences exist for OneVsRest, RF, Bagging, ExtraTrees and T-DTC; meanwhile OutputCode OA differs considerably from DT, OnevsOne and ADA. Focusing on Byte Accuracy we observe that T-DTC is clearly the best algorithm followed by OutputCode and RF, and that there are not big statistical differences between ensemble methods with the exception of ADA algorithm. For the instance of Training Time, we note that DT is the fastest algorithm, as expected, and T-DTC ties with ADA and ExtraTrees. Being DT the control algorithm, we can say that there are no differences between DT and RF, ExtraTrees, ADA or T-DTC. Finally, DT is the fastest in classification, and T-DTC and OnevsRest are the second and third fastest algorithms. According to the Holmś procedure there are not relevant differences between the previous three algorithms and DT in classification.

Table 5
Training and classification time for the traces studied (in seconds).

	ISP-1		ISP-2		ISP-3		HOST-1		HOST-2		HOST-3	
	Training Time	Classification time	Training Time	Classification time	Training Time	Classification time	Training Time	Classification time	Training Time	Classification time	Training Time	Classification time
DT	**0.94330**	**0.02095**	4.33370	**0.03834**	**2.85369**	**0.03146**	**1.38806**	**0.01901**	**1.64008**	**0.03180**	**4.31513**	**0.08269**
OneVsRest	5.55971	0.08686	10.55774	0.11443	10.99850	0.11432	4.65841	0.08323	5.36138	0.13081	37.53189	0.54066
OneVsOne	2.54106	0.37229	12.78923	0.52626	7.44010	0.36590	6.00742	0.52563	3.03271	0.74541	23.16181	3.45232
RF	2.51751	0.29148	7.81902	0.49980	5.71650	0.38325	3.47846	0.30827	2.63806	0.40228	13.88464	1.17011
Bagging	11.04407	0.35490	14.49463	0.42822	15.90904	0.39981	15.01821	0.37088	6.53809	0.40831	54.44654	2.02192
ExtraTrees	3.14286	0.52690	4.26417	0.61417	3.20274	0.64590	2.24429	0.53985	1.74953	0.51918	9.03618	1.76928
OuputCode	17.09089	0.18528	23.63508	0.20806	20.17624	0.19767	16.95315	0.18262	10.46566	0.23858	35.89734	0.82363
T-DTC	1.52025	0.04099	**3.87237**	0.06602	3.21352	0.05421	1.95967	0.04057	1.76991	0.06958	4.38929	0.48791

Table 6
Friedmanś test and Holmś procedure results.

	Overall Accuracy (OA)			Byte Accuracy (BA)			Training Time			Classification Time		
	Ranking	p-values	APVs	Ranking	p-values	APVs	Ranking	p-values	APVs	Ranking	p-values	APVs
DT	7.50	0.001	0.006	6.50	0.008	0.007	**1.66**	–	–	**1.00**	–	–
OneVsRest	4.50	**0.205**	0.016	5.00	**0.091**	0.009	6.83	0.001	0.008	3.00	**0.205**	0.025
OneVsOne	7.17	0.003	0.008	5.00	**0.091**	0.009	6.00	0.006	0.010	7.83	< 0.001	0.007
RF	3.00	**0.752**	0.025	4.33	**0.206**	0.025	4.50	**0.073**	0.0125	5.83	0.002	0.012
Bagging	5.17	**0.091**	0.011	5.00	**0.091**	0.009	8.17	<0.001	0.007	6.67	< 0.001	0.008
ExtraTrees	5.17	**0.091**	0.011	5.67	**0.035**	0.008	3.33	**0.292**	0.020	8.17	<0.001	0.006
OuputCode	**2.50**	–	–	3.83	**0.348**	0.050	8.67	< 0.001	0.006	4.00	**0.058**	0.017
ADA	7.33	0.002	0.007	7.33	0.002	0.006	3.33	**0.291**	0.02	6.50	<0.001	0.010
T-DTC	2.67	**0.916**	0.050	**2.33**	–	–	3.33	**0.598**	0.050	2.00	**0.527**	0.050
Friedmanś	25.91	0.001	–	13.64	0.091	–	40.80	< 0.001	–	43.02	<0.001	–

Table 7
First models in accomplishing the same Overall Accuracy (OA) as the best DT model. (BA = Byte Accuracy and # is the number of features used).

	ISP-1			ISP-2			ISP-3			HOST-1			HOST-2			HOST-3		
	OA	BA	#	OA	BA	#	OA	BA	#	OA	BA	#	OA	BA	#	OA	BA	#
DT	0.95685	0.99417	9	0.95310	0.93077	32	0.94817	0.95078	22	0.98652	0.94162	30	0.99192	**0.99452**	32	0.99742	0.95806	38
OneVsRest	**0.96339**	0.99456	5	0.96284	0.95744	6	0.95651	0.96294	**4**	0.98682	0.94251	19	0.99207	0.97957	19	–	–	–
OneVsOne	–	–	–	0.95370	0.93042	28	–	–	–	–	–	–	–	–	–	0.99750	0.96185	42
RF	0.95859	0.99467	5	0.95632	0.93149	6	0.95204	0.96229	7	0.98656	**0.95407**	16	**0.99260**	0.97829	**8**	**0.99799**	0.97158	38
Bagging	0.95779	**0.99502**	5	0.95467	0.94032	12	0.94888	0.95901	12	0.98667	0.95639	20	0.99234	0.98068	9	–	–	–
ExtraTrees	–	–	–	0.95335	0.90746	17	0.94883	0.89486	10	0.98696	0.95333	16	0.99286	0.97556	9	–	–	–
OuputCode	0.96162	0.99487	5	0.96549	**0.96035**	6	0.95542	0.96053	5	**0.98745**	0.94172	17	0.99243	0.97917	10	0.99753	0.96852	13
ADA	–	–	–	–	–	–	0.94982	0.96480	12	–	–	–	–	–	–	–	–	–
T-DTC	0.95808	0.95615	**3**	**0.97302**	0.93900	**6**	**0.96712**	**0.97109**	**4**	0.98662	0.94648	**11**	0.99240	0.98093	12	0.99750	**0.99205**	**12**

4.4. Model complexity evaluation

We have already assessed the best models built by each ensemble algorithm. Before a connection flow is classified, the classifier has to compute the statistics associated with it, and also it may be interesting to assess how many attributes we can drop to equal or overcome the best DT performances for each ensemble algorithm. Although ensemble algorithms yield longer training and classification times, they could offer other advantages by reducing the number of statistical attributes used for training predictive models. Table 7 contains the results obtained using the best DT OA as baseline for each dataset. From Table 7, we can observe that RF, OutputCode and T-DTC provided models that overcome the baseline OA using a reduced number of attributes for all dataset. RF got the best OA for HOST-1,2 in spite of using quite reduced subsets. Although T-DTC yielded worse BA than the baseline for ISP-1 and HOST-2, its accuracy metrics were the best or almost the best using a lesser number of attributes for ISP-2-3 and HOST-1-3. Bagging also exhibited good scores with a high subset reduction, and DT only got higher BA than its competitors for ISP-2 using many more features. These experimental results show that ensemble learning algorithms are able to get similar performances to DTś using less statistical attributes. The most of ensemble algorithms equaled or overcame single DT performances for almost all traces, only, ADA and OneVsOne achieved worse results than DTś in four or more of the network traces. The subset reduction could offset the training and classification time punishment discussed in Section 4.2.

Finally, we provide a rank of the most relevant features resulting from applying feature selection to our datasets. To appreciate the features that more contribute to the predictive models, we have counted the number of times the features appeared in the models with less number of attributes shown in Table 7; and Table 8 ranks the statistical attributes that are included in the different models at least two times. As Table 8 shows, none of the attributes in the datasets is used for the six analyzed datasets. The most employed attributes are maxTCPWin0 and NPKT_128. Although three of the most relevant attributes depend on TCP windows, packet-size based attributes have a remarkable presence as informative features. In Internet networks, Packet-size based features are more resilient to the operational status of networks (e.g. if the network is congested) than TCP-windows based features, therefore packet-size based attributes are more interesting for NTC. More research must be conducted to determine the optimal set of statistical attributes independently to network environments and operational status of networks.

4.5. Summary

Finally, Table 9 contains a summary of the results discussed in previous sections. Since we intend to compare only ensemble algorithm, we exclude DT from this summary; and also ADA algorithm was excluded due to its poor performances.

Table 8
Ranking of the most relevant features used for all the datasets included in the experiments.

Feature name	Description	# of times used
maxTCPWin0	Maximum TCP Windows used in flow packets considering outgoing direction.	4
NPKT_128	Number of packets whose applications bytes is higher than 64 and lesser than 128 bytes considering both directions.	4
maxBytes0	Maximum number of bytes transferred in flow packets considering outgoing direction.	3
%Bytes0	Percentage of bytes transferred in flow packets considering outgoing direction over the total number of bytes in whole flows.	3
minBytes0	Minimum number of bytes transferred in flow packets considering outgoing direction.	3
NPKT_64	Number of packets whose application bytes are higher than 32 and lesser than 64 bytes.	3
minTCPWin0	Minimum TCP Windows used in flow packets considering outgoing direction.	3
varBytes1	Variance of the bytes transferred in flow packets considering ingoing direction.	3
%Packets0	Percentage of the number packets transferred considering outgoing direction over the total number of packets in whole flows.	2
varTCPWinT	Variance of TCP Windows used in flow packets considering both directions.	2
maxBytesT	Maximum number of bytes transferred in flow packets considering both directions.	2

Table 9
Summary of results.

	ISP-1	ISP-2	ISP-3	HOST-1	HOST-2	HOST-3
Overall Accuracy (OA)						
Highest	T-DTC	T-DTC	T-DTC	RF	ExtraTrees	RF
Lowest	ExtraTrees	OneVsOne	OneVsOne	OneVsOne	OneVsOne	ExtraTrees
Byte Accuracy (BA)						
Highest	T-DTC	T-DTC	T-DTC	OneVsOne	T-DTC	T-DTC
Lowest	ExtraTrees	ExtraTrees	ExtraTrees	T-DTC	OneVsRest	OneVsOne
Training Time						
Fastest	ExtraTrees	T-DTC	T-DTC	T-DTC	ExtraTrees	T-DTC
Slowest	OuputCode	OutputCode	OutputCode	OuputCode	OuputCode	Bagging
Classification Time						
Fastest	T-DTC	T-DTC	T-DTC	T-DTC	T-DTC	T-DTC
Slowest	ExtraTrees	ExtraTrees	ExtraTrees	ExtraTrees	OneVsOne	OneVsOne

As Table 9 reveals, T-DTC is the fastest ensemble scheme in classification for all datasets studied. Also T-DTC speeds up notably the training phase getting the fastest times for four datasets and the second fastest for the others. Finally, T-DTC exhibits the highest BA for all ISP traces and two of the three HOST traces; and, in terms of OA, T-DTC yielded the best or very close to the best scores. The experimental results obtained validate our proposal for our datasets, improving the prospects of ensemble learning in real time NTC.

5. Conclusions

This work compares the performance of seven popular DT-based ensemble algorithms along with a single base estimator (DT) and a novel proposed ensemble scheme in order to assess the suitability of ensemble learning in real time NTC. Although some ML algorithms have previously been studied in related works for this problem, they lack of a clear and detailed comparison among ensemble algorithms assessing both accuracy metrics and computational costs. Our experimental results show that most of the ensemble algorithms improve the performance metrics of a single estimator for our datasets, but, as expected, extra time costs are found in classification and training phases. All ensemble algorithms analyzed in this paper exhibit a notable classification-time increase that hinders the use of these techniques in real time contexts. Also we have shown that some ensemble schemes are able to equal or overcome a single DT using a reduced subset of attributes, leading to computational savings, which could offset classification and training time punishments. Therefore, the performance improvements and attribute reduction of some ensemble algorithms could justify their implementation in some contexts, such as small networks or environments where the computational capacity of network devices is not a crucial limitation.

With the intention of boosting ensemble learning in online NTC, a novel ensemble method, called T-DTC, is proposed. T-DTC exploits imbalanced traffic distributions in NTC datasets to significantly speed up Training and Classification phases. T-DTC achieves time savings by ordering CART Decision Trees in a sequential chain in which each base estimator is trained to distinguish only one traffic application from the rest. Thereby, each classifier filters out samples that are assigned to an application group and feeds following classifiers with samples whose application is not detected, avoiding repetitive classifications of classes that are easily and accurately identified.

In this paper, we show that our ensemble scheme clearly outperforms other ensemble algorithms in terms of latency, but also retaining the essential performance metrics (such as Overall Accuracy and Byte Accuracy) improvements of ensemble learning respect to a single Decision Tree. Our proposal has been evaluated for two contexts with quite different operational features, showing that T-DTC is one of the best algorithm for both network environments. In conclusion, our proposed algorithm definitely enhances the prospects of ensemble learning to be applied to real time NTC. We have made the code of T-DTC public in [43], we encourage other researchers to assess our ensemble algorithm for their ML modeling tasks.

Acknowledgments

This work has been partially funded by the Ministerio de Economía y Competitividad (Grant number TIN2014) del Gobierno de España and the Fondo de Desarrollo Regional (FEDER) within the project “Inteligencia distribuida para el control y adaptación de redes dinámicas definidas por software, Ref: TIN2014-57991-C3-2-P”, in the Programa Estatal de Fomento de la Investigación Científica y Técnica de Excelencia, Subprograma Estatal de Generación de Conocimiento. Additionally, we would like to thank the Broadband Communications Research Group, especially Valentín Carela-Español for providing the network traces we have used in our work. Finally, we would like to thank the ISP for the real network traffic captures and the resources shared with us for this work.

Annex 1. Collection of Statistical Attributes

Table A1 contains the whole collection of statistical attributes used in this work along with a brief description of each one. Note

Table A1
Collection of statistics employed for the experiments presented in this work.

Feature Name	Description
%Packet0	Percentage of packets transferred in the direction 0
%Packet1	Percentage of packets transferred in the direction 1
%Bytes0	Percentage of bytes transferred over number of packets in the direction 0
%Bytes1	Percentage of bytes transferred over number of packets in the direction 1
meanBytes0	Mean of bytes transferred over number of packets in the direction 0
meanBytes1	Mean of bytes transferred over number of packets in the direction 1
meanBytesT	Mean of bytes transferred over number of packets in both directions
varBytes0	Variance of bytes transferred over number of packets in the direction 0
varBytes1	Variance of bytes transferred over number of packets in the direction 1
varBytesT	Variance of bytes transferred over number of packets in both directions
rmsBytes0	Root mean square of bytes transferred over number of packets in the direction 0
rmsBytes1	Root mean square of bytes transferred over number of packets in the direction 1
rmsBytesT	Root mean square of bytes transferred over number of packets in both directions
maxBytes0	Maximum number of bytes transferred in the direction 0
maxBytes1	Maximum number of bytes transferred in the direction 1
maxBytesT	Maximum number of bytes transferred in both directions
minBytes0	Minimum number of bytes transferred in the direction 0
minBytes1	Minimum number of bytes transferred in the direction 1
minBytesT	Minimum number of bytes transferred in both directions
meanInterArrivalTime0	Mean of interarrival time over number of packets in the direction 0
meanInterArrivalTime1	Mean of interarrival time over number of packets in the direction 1
meanInterArrivalTimeT	Mean of interarrival time over number of packets in both directions
varInterArrivalTime0	Variance of interarrival time over number of packets in the direction 0
varInterArrivalTime1	Variance of interarrival time over number of packets in the direction 1
varInterArrivalTimeT	Variance of interarrival time over number of packets in both directions
rmsInterArrivalTime0	Root mean square of interarrival time over number of packets in the direction 0
rmsInterArrivalTime1	Root mean square of interarrival time over number of packets in the direction 1
rmsInterArrivalTimeT	Root mean square of interarrival time over number of packets in both directions
maxInterArrivalTime0	Maximum number of interarrival time in one packet in the direction 0
maxInterArrivalTime1	Maximum number of interarrival time in one packet in the direction 1
maxInterArrivalTimeT	Maximum number of interarrival time in one packet in both directions
minInterArrivalTime0	Minimum number of interarrival time in one packet in the direction 0
minInterArrivalTime1	Minimum number of interarrival time in one packet in the direction 1
minInterArrivalTimeT	Minimum number of interarrival time in one packet in both directions
meanTCPWin0	Mean of TCP window sizes over number of packets in the direction 0
meanTCPWin1	Mean of TCP window sizes over number of packets in the direction 1
meanTCPWinT	Mean of TCP window sizes over number of packets in both directions
varTCPWin0	Variance of TCP window sizes over number of packets in the direction 0
varTCPWin1	Variance of TCP window sizes over number of packets in the direction 1
varTCPWinT	Variance of TCP window sizes over number of packets in both directions
rmsTCPWin0	Root mean square of TCP window sizes over number of packets in the direction 0
rmsTCPWin1	Root mean square of TCP window sizes over number of packets in the direction 1
rmsTCPWinT	Root mean square of TCP window sizes over number of packets in both directions
maxTCPWin0	Maximum number of TCP window in one packet in the direction 0
maxTCPWin1	Maximum number of TCP window sizes in one packet in the direction 1
maxTCPWinT	Maximum number of TCP window sizes in one packet in both directions
minTCPWin0	Minimum number of TCP window sizes in one packet in the direction 0
minTCPWin1	Minimum number of TCP window sizes in one packet in the direction 1
minTCPWinT	Minimum number of TCP window sizes in one packet in both directions
flowDurationT	Flow duration accounting for packets in both directions
flowDuration0	Flow duration accounting for packets in the direction 0
flowDuration1	Flow duration accounting for packets in the direction 1
flowSize	Total flow size in bytes accounting for bytes transferred in both directions
meanBytes/Time0	Mean of bytes transferred over flow duration in the direction 0
meanBytes/Time1	Mean of bytes transferred over flow duration in the direction 1
meanBytes/TimeT	Mean of bytes transferred over flow duration in both directions
varBytes/Time0	Variance of bytes transferred over flow duration in the direction 0
varBytes/Time1	Variance of bytes transferred over flow duration in the direction 1
varBytes/TimeT	Variance of bytes transferred over flow duration in both directions
rmsBytes/Time0	Root mean square of bytes transferred over flow duration in the direction 0
rmsBytes/Time1	Root mean square of bytes transferred over flow duration in the direction 1
rmsBytes/TimeT	Root mean square of bytes transferred over flow duration in both directions
meanTCPWin/Time0	Mean of TCP window sizes over flow duration in the direction 0
meanTCPWin/Time1	Mean of TCP window sizes over flow duration in the direction 1
meanTCPWin/TimeT	Mean of TCP window sizes over flow duration in both directions
varTCPWin/Time0	Variance of TCP window sizes over flow duration in the direction 0
varTCPWin/Time1	Variance of TCP window sizes over flow duration in the direction 1
varTCPWin/TimeT	Variance of TCP window sizes over flow duration in both directions
rmsTCPWin/Time0	Root mean square of TCP window sizes over flow duration in the direction 0
rmsTCPWin/Time1	Root mean square of TCP window sizes over flow duration in the direction 1
rmsTCPWin/TimeT	Root mean square of TCP window sizes over flow duration in both directions
NPKT_64	Number of packets with equal or less than 64 bytes in both directions
NPKT_128	Number of packets with equal or less than 128 bytes in both directions
NPKT_256	Number of packets with equal or less than 256 bytes in both directions
NPKT_512	Number of packets with equal or less than 512 bytes in both directions
NPKT_1024	Number of packets with equal or less than 1024 bytes in both directions
NPKT_MORE	Number of packets with more than 1024 bytes in both directions

that many statistics have been computed accounting for only one flow direction or both, “direction 0” denotes the way in which first packets of flow connections were detected and vice versa for “direction 1”. The datasets are publicly available via emailing the authors. This collection of statistics attributes could be improved adding new and more informative attributes extracted from IP and Transport layers.

References

[1] T. Nguyen, G. Armitage, A survey of techniques for internet traffic classification using machine learning, IEEE Commun. Surv. Tutorials 10 (4) (2008) 56–76.

[2] A. Dainotti, A. Pescape, K. Claffy, Issues and future directions in traffic classification, IEEE Netw. 26 (January 1) (2012) 35–40.

[3] N. Williams, S. Zander, G. Armitage, A preliminary performance comparison of five machine learning algorithms for practical IP traffic flow classification, ACM SIGCOMM Comput. Commun. Rev. 36 (October 5) (2006) 5.

[4] W. Li, A.W. Moore, A machine learning approach for efficient traffic classification, in: 2007 15th International Symposium on Modeling, Analysis, and Simulation of Computer and Telecommunication Systems, 2007, pp. 310–317.

[5] L. Hernández, C. Baladrón, J. Aguiar, B. Carro, A. Sánchez-Esguevillas, Classification and clustering of electricity demand patterns in industrial parks, Energies 5 (December 12) (2012) 5215–5228.

[6] L. Hernández, C. Baladrón, J.M. Aguiar, B. Carro, A. Sánchez-Esguevillas, J. Lloret, Artificial neural networks for short-term load forecasting in microgrids environment, Energy 75 (October) (2014) 252–264.

[7] L. Bernaille, R. Teixeira, I. Akodjenou, A. Soule, K. Salamatian, Traffic classification on the fly, ACM SIGCOMM Comput. Commun. Rev. 36 (2) (2006) 23–26.

[8] L. Bernaille, R. Teixeira, K. Salamatian, Early application identification, in: Proc. 2006 ACM Conex. Conf., 2006 6:1–6:12.

[9] L. Peng, B. Yang, Y. Chen, Effective packet number for early stage internet traffic identification, Neurocomputing 156 (2015) 252–267.

[10] M. Soysal, E.G. Schmidt, Machine learning algorithms for accurate flow-based network traffic classification: evaluation and comparison, Perform. Eval. 67 (June 6) (2010) 451–467.

[11] A. Callado, J. Kelner, D. Sadok, C. Alberto Kamienski, S. Fernandes, Better network traffic identification through the independent combination of techniques, J. Netw. Comput. Appl. 33 (July 4) (2010) 433–446.

[12] P. Casas, J. Mazel, and P. Owezarski, “MINETRAC: Mining Flows for Unsupervised Analysis & Semi-Supervised Classification.” (2011).

[13] Y. Jin, N. Duffield, J. Erman, P. Haffner, S. Sen, Z.-L. Zhang, A modular machine learning system for flow-level traffic classification in large networks, ACM Trans. Knowl. Discov. Data 6 (March 1) (2012) 1–34.

[14] J. Zhang, X. Chen, Y. Xiang, W. Zhou, J. Wu, Robust network traffic classification, IEEE/ACM Trans. Netw. 23 (4) (2015) 1257–1270.

[15] D.M. Divakaran, L. Su, Y.S. Liau, V.L. Vrizlynn, SLIC: self-learning intelligent classifier for network traffic, Comput. Netw. 91 (2015) 283–297.

[16] T.T.T. Nguyen, G. Armitage, P. Branch, S. Zander, Timely and continuous machine-learning-based classification for interactive IP traffic, IEEE/ACM Trans. Netw. 20 (December 6) (2012) 1880–1894.

[17] V. Carela-Español, P. Barlet-Ros, A. Cabellos-Aparicio, J. Solé-Pareta, Analysis of the impact of sampling on NetFlow traffic classification, Comput. Netw. 55 (April 5) (2011) 1083–1099.

[18] A.W. Moore, D. Zuev, Internet traffic classification using Bayesian analysis techniques, ACM SIGMETRICS Perform. Eval. Rev. 33 (June 1) (2005) 50.

[19] T. Auld, A.W. Moore, S.F. Gull, Bayesian neural networks for internet traffic classification, IEEE Trans. Neural Netw. 18 (January 1) (2007) 223–239.

[20] A. Este, F. Gringoli, L. Salgarelli, Support vector machines for TCP traffic classification, Comput. Netw. 53 (14) (Sep. 2009) 2476–2490.

[21] A. Callado, et al., A survey on internet traffic identification, IEEE Commun. Surv. Tutorials 11 (3) (2009) 37–52.

[22] H. Kim, K. Claffy, M. Fomenkov, D. Barman, M. Faloutsos, K.Y. Lee, Internet traffic classification demystified: myths, caveats, and the best practices, Traffic 50 (4) (2008) 1–12.

[23] R. Barandela, J.S. Sánchez, V. García, E. Rangel, Strategies for learning in class imbalance problems, Pattern Recognit. 36 (2003) 849–851.

[24] V. Carela-Español, T. Bujlow, P. Barlet-Ros, Is Our Ground-Truth for Traffic Classification Reliable?, 2014, pp. 98–108.

[25] L. Deri, M. Martinelli, T. Bujlow, A. Cardigliano, nDPI: Open-source high-speed deep packet inspection, in: 2014 International Wireless Communications and Mobile Computing Conference (IWCMC), 2014, pp. 617–622.

[26] “nDPI.”.

[27] T. Bujlow, V. Carela-Español, P. Barlet-Ros, Independent comparison of popular DPI tools for traffic classification, Comput. Netw. 76 (2015) 75–89.

[28] L. Yu, H. Liu, Feature selection for high-dimensional data: a fast correlation-based filter solution, in: Int. Conf. Mach. Learn., 2003, pp. 1–8.

[29] B. Senliol, G. Gulgezen, L. Yu, Z. Cataltepe, Fast Correlation Based Filter (FCBF) with a different search strategy, 2008 23rd Int. Symp. Comput. Inf. Sci. Isc. 2008, 2008.

[30] S.E. Gómez, 2016. “FCBF_module,” [Online]. Available: https://github.com/SantiagoEG/FCBF_module.

[31] J. Demšar, Statistical comparisons of classifiers over multiple data sets, J. Mach. Learn. Res. 7 (2006) 1–30.

[32] S. García, A. Fernández, J. Luengo, F. Herrera, Advanced nonparametric tests for multiple comparisons in the design of experiments in computational intelligence and data mining: experimental analysis of power, Inf. Sci. 180 (10) (2010) 2044–2064.

[33] E. Bauer, R. Kohavi, An empirical comparison of voting classification algorithms: bagging, boosting, and variants, Mach. Learn. 36 (1/2) (1999) 105–139.

[34] T.G. Dietterich, An experimental comparison of three methods for constructing ensembles of decision trees: bagging, boosting, and randomization, Mach. Learn. 40 (2) (2000) 139–157.

[35] R.E. Banfield, L.O. Hall, K.W. Bowyer, W.P. Kegelmeyer, A comparison of decision tree ensemble creation techniques, IEEE Trans. Pattern Anal. Mach. Intell. 29 (January 1) (2007) 173–180.

[36] T.G. Dietterich, G. Bakiri, Solving multiclass learning problems via error-correcting output codes, J. Artif. Intell. Res. 2 (1995) 263–286.

[37] L. Breiman, Random forests, Mach. Learn. 45 (1) (2001) 5–32.

[38] J. Milgram, M. Cheriet, R. Sabourin, One against One' or ‘One against All': which one is better for handwriting recognition with SVMs? in: Tenth Int. Work. Front. Handwrit. Recognit., 2006, pp. 1–6.

[39] P. Geurts, D. Ernst, L. Wehenkel, Extremely randomized trees, Mach. Learn. 63 (1) (2006) 3–42.

[40] T.G. Dietterich, Ensemble Methods in Machine Learning, 2000, pp. 1–15.

[41] K.C. Lan, J. Heidemann, A measurement study of correlations of internet flow characteristics, Comput. Netw. 50 (1) (2006) 46–62.

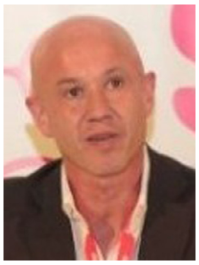
[42] P. Carvalho, P. Solis, B. Queiroz, B. Carneiro, M. Deus, A traffic analysis per application in real IP/MPLS service provider network, in: 2007 2nd IEEE/IFIP International Workshop on Broadband Convergence Networks, 2007, pp. 1–5.

[43] S.E. Gómez, 2017. “FCBF_module,” [Online]. Available: https://github.com/SantiagoEG/TEC_module.

Santiago Egea Gómez received the Telecommunication Engineering Degree from Polythenique University of Cartagena, Murcia, Spain. He is a PhD student at University of Valladolid, Valladolid, Spain. He is member of the Communications Systems and Networks (SRC) laboratory. His research interests include Signal Processing and Machine Learning specifically applied to telecommunication networks.

Belén Carro Martínez received the Ph.D. degree in the field of broadband access networks from the University of Valladolid, Valladolid, Spain, in 2001. She is a Professor at the Department of Signal Theory and Communications and Telematics Engineering, University of Valladolid. She is the Director of the Communications Systems and Networks (SRC) laboratory, working as a Technical Researcher and Research Manager in European and national projects in the areas of service engineering and SOA systems, IP broadband communications, NGN/IMS, VoIP/QoS and machine learning. She has supervised several Ph.D. students on topics related to personal communications, IMS and machine learning. She has extensive research publications experience, as author, reviewer and editor.

Antonio J. Sánchez-Esguevillas (Senior Member, IEEE) received the Ph.D. degree (with honors) in the field of QoS for real time multimedia services over IP networks from the University of Valladolid, Valladolid, Spain, in 2004. He has been managing innovation at Telefonica (both at Telefonica I+D-Services line and at Telefonica Corporation), Madrid, Spain. He has also been Adjunct Professor and Honorary Collaborator at the University of Valladolid, supervising several Ph.D. students. He has coordinated very large (in excess of 100 million) international R&D projects in the field of personal communication services, particularly related to voice over IP (VoIP) and Internet protocol (IP) multimedia subsystem (IMS). He has more than 50 international publications and several patents. His current research interests are in the area of digital services, including machine learning. Dr. Sánchez-Esguevillas is a member of the Editorial Board of IEEE COMMUNICATIONS MAGAZINE among others.



Luis Hernández-Callejo received the Ph.D. degree in Demand Forecast in Microgrids from the University of Valladolid, Valladolid, Spain, in 2014. He is wide-experienced engineer in sectors as Energy Efficiency, Distributed Generation, Smart Grids, Smart Metering and Microgrids. He is also a Professor at University of Valladolid. He has supervised several Ph.D. students on topics related to: Artificial Intelligence, Photovoltaic Hot Spot Thermography and Microgrids. He has remarkable research publications experiences as author and reviewer.