A framework to classify heterogeneous Internet traffic with Machine Learning and Deep Learning techniques for satellite communications

Fannia Pacheco, Ernesto Expósito, Mathieu Gineste

A framework to classify heterogeneous Internet traffic with Machine Learning and Deep Learning techniques for Satellite Communications

Fannia Pacheco[a], Ernesto Exposito[a], Mathieu Gineste[b]

{*f.pacheco,ernesto.exposito-garcia*}*@univ-pau.fr, mathieu.gineste@thalesaleniaspace.com*

[a] *Univ Pau & Pays Adour, E2S UPPA, LIUPPA, EA3000, Anglet, 64600, France*
[b] *Département : Business Line Telecommunication, R&D department, Thales Alenia Space, TOULOUSE, 31100, France.*

Abstract

Nowadays, the Internet network system serves as a platform for communication, transaction, and entertainment, among others. This communication system is characterized by terrestrial and Satellite components that interact between themselves to provide transmission paths of information between endpoints. Particularly, Satellite Communication providers' interest is to improve customer satisfaction by optimally exploiting on demand available resources and offering Quality of Service (QoS). Improving the QoS implies to reduce errors linked to information loss and delays of Internet packets in Satellite Communications. In this sense, according to Internet traffic (Streaming, VoIP, Browsing, etc.) and those error conditions, the Internet flows can be classified into different sensitive and non-sensitive classes. Following this idea, this work aims at finding new Internet traffic classification approaches to improving the QoS. Machine Learning (ML) and Deep Learning (DL) techniques will be studied and deployed to classify Internet traffic. All the necessary elements to couple an ML or DL solution over a well-known Satellite Communication and QoS management architecture will be evaluated. To develop this solution, a rich and complete set of Internet traffic is required. In this context, an emulated Satellite Communication platform will serve as a data generation environment in which different Internet communications will be launched and captured. The proposed classification system will deal with different Internet communications (encrypted, unencrypted, and tunneled). This system will process the incoming traffic hierarchically to achieve a high classification performance. Finally, some experiments on a cloud emulated platform validates our proposal and set guidelines for its deployment over a Satellite architecture.

Keywords:
Internet traffic classification, Machine Learning, Satellite Communications, QoS management, Encrypted traffic.

1. Introduction

The Internet is an intricate network system where all its elements are organized and synchronized to work in harmony. This network system is characterized by terrestrial and Satellite components that interact between themselves to provide transmission paths of information between endpoints. The main goal of this configuration is to offer communication services. This work will refer to this network system as Satellite Communications all along with this document. Satellite Communications are conceived by architectural configurations that are continually evolving to offer better services. Satellite communication services are in constant search for: a) optimizing their infrastructure and network resources, and b) improving the customer satisfaction by exploding optimally available resources so that offering a wide range of services such as bandwidth and Quality of Service (QoS) on-demand [1].

This work aims at finding new approaches to improving the QoS, but what is the idea behind improving the QoS? The primary purpose is to provide an adequate Quality of Experience (QoE) from the user's point of view. The QoE is a very subjective value that can be quantitatively and qualitatively measured [2]. The metrics to measure the QoE range from cost, reliability, efficiency, privacy, security, interface user-friendliness, and user confidence. However, different fields of computer science define these metrics differently [3]. For Satellite Communications, the QoE might be translated to the delay and information loss perceived by the customer in real-time communications. Based on this, improving the QoS is highly correlated with avoiding information loss and delays; hence, the QoE can be indirectly improved.

To improve the QoS, one of the most common and accepted actions is to fulfill a set of requirements that can be executed by profiling Internet traffic [4, 5]. The former idea considers that some Internet traffic can be more sensitive to information loss and delay such as Internet calling (commonly called Voice over IP) or video conferences. In contrast, Internet browsing or file downloads are less pruned to be affected by these error conditions. Following this idea, a new field emerges in this area called traffic analysis.

Traffic analysis is the complete process that starts from intercepting traffic data to find relationships, patterns, anomalies, and misconfigurations, among other things, in the Internet network. Mainly, traffic classification is a subgroup of strategies in this field that aims at classifying the Internet traffic into predefined categories, such as normal or abnormal traffic, the type of application (streaming, web browsing, VoIP, etc.) or the name of the application (YouTube, Netflix, Facebook, etc.). Network traffic classification has become a crucial task for QoS management.

In the past, traffic classification relied on a port-based approach where its registered and known port identified each application, defined by the Internet Assigned Numbers Authority (IANA) [6]. This approach became unreliable and inaccurate due to, among other factors, the proliferation of new applications with unregistered or random generated ports. Another method that gained a lot of popularity in this field is called Deep Packet Inspection (DPI) [7]. DPI performs matchings between the packet payload and a set of stored signatures to classify network traffic. However, DPI fails when privacy policies and laws prevent accessing the packet content, as well as the case of protocol obfuscation or encapsulation. To overcome the previous issues, Machine Learning (ML) has emerged as a suitable solution, not only for the traffic classification task but also for prediction and new knowledge discovery, among other things [8]. In this context, the statistical features of IP flows are commonly extracted and stored from network traces to generate historical data. In this way, different ML models can be trained with this historical data, and new incoming flows can be analyzed with such models.

To summarize, this work will be focused on improving the QoS over Satellite Communications through Internet classification based on ML. Along with this document, we will study all the elements needed to achieve this objective and propose an ML-based Internet traffic classification solution. Before continuing, we will briefly review some works related to this subject.

1.1. Related works

With the exponential growth of Internet communications, more and more efforts to adequately optimize and manage Satellite resources have been accomplished. Traffic classification helps resource managers to state some guidelines for the administration of their assets. In this matter, the widely deployed classification approach is DPI. Even though, DPI tools have particular deficiencies, they are still widely used for traffic classification thanks to their accuracy for non-encrypted traffic [9, 7]. However, the increase of encrypted communications is unstoppable, impulsing the use of ML as one potential alternative. Some works already proposed the implementation of ML-based solutions in different network components or structures such as in cellular networks, WiFi networks, and Satellite networks.

In cellular networks, the mobile IP traffic classification can be performed at different levels either using the port, the packet payload [10, 11], or the statistical flow distribution [12]. For instance, [13] collected IP traffic extracted from mobile networks in fixed time windows. Statistical based features from normal and abnormal traffic are computed, and a classifier is trained for the analysis of the massive network users' traffic behaviors. The work in [14] presents an approach to collect and label mobile IP network traces correctly. The work in [15] exposed a generic architecture of a cellular network, and the possible positions where traffic monitoring can be deployed, such as in a Packet Switched (PS) Core. Finally, [16] presents a complete study of how to use Deep Learning (DL) models for Mobile encrypted traffic. This work shows that DL approaches present some drawbacks regarding to the classical, mainly due to its novelty in this field.

Similarly, in WiFi networks, IP-data can be extracted to apply ML approaches for the traffic classification. For instance, the work in [17] collected network traces from Wi-Fi controllers at a large university campus. These controllers connected access points to the campus backbone network, allowing wireless devices to access the Internet. The traces come from network traffic to/from malicious and benign domains, and statistical-based features were computed over these traces. A binary ML classifier was trained for detecting malicious domains. Similar approaches can be found in [18, 19, 20]. The difference between cellular/mobile networks and WiFi network resides in the technology used for the data exchange that might affect the speed, cost, and security.

Finally, in Satellite networks for improving the QoS, traffic data is captured from Satellite Internet Service Providers (ISPs). The works in this area aim to classify and to analyze Internet traffic in large networks [21, 22, 23, 24].

Network structure	Work	Data	Classification technique	Features	Advantages & Disadvantages			
					ground truth	encryption	evolution	implementation
ISP	[29]	CAIDA [a]	Statistical	Statistical				X
	[30]	WIDE [b] and ISP	Statistical/ML	Statistical				
	[21]	Private	ML	Statistical	X			X
	[24]	Private	ML	Statistical	X		X	X
Enterprise	[31]	Private	ML	Statistical				
Satellite	[23]	Private	ML	Statistical				X
Mobile	[14]	Private	ML	Statistical	X			X
	[13]	WIDE, CAIDA	ML	Statistical				
Wifi/Mobile	[17]	Private	ML	bag-of-words/ Statistical	X			
SDN	[25]	Private	Statistical/ Payload	Statistical				

Table 1: Related works.

[a] http://www.caida.org/data/
[b] http://mawi.wide.ad.jp/mawi/

The principle is the same as the previous cases, Internet traffic monitoring is deployed to perform traffic classification. These monitoring points can be at routers [21, 22] or point of presence (PoP) [23] of large ISP networks. Another emerging approach is the use of Software-defined networks(SDNs) in Satellite-terrestrial networks. In SDNs, traffic classification can be easily deployed in the SDN' master controllers as it is exposed in [25, 26].

The principal challenge of this research work is to find a well-suited classification system that can be implemented over a Satellite Architecture. As a consequence, we will carry the problems that imply using ML for traffic classification. In our survey paper [8], we identify several of these challenges summarized as follows: i) the data available with their ground truth is limited and hard to collect, ii) the scalability of traffic classification solutions is a challenge, iii) adaptive solutions are required due to the dynamism and evolution of the network, and iv) the solution applied require a correct validation. Furthermore, the ML approaches require to fulfill different challenges, such as a provided performance, the management of the increasing amount of traffic and transmission rates, and the reconfiguration capabilities, as it is exposed in [27], and similarly in [28].

In general, procedures such as Feature extraction and construction of ML solutions on Internet traffic classification are equivalent in different network conditions (either in WiFi, cellular and Satellite networks), what differs is the data collected that serves as knowledge to build such solution. The previous implies that for each case, a dedicated classification solution must respond to particular demands. For instance, in Table 1, we show some of the most important attributes of selected works such as the network structure, data, classification technique, and features used. In addition to this, four additional attributes act as advantages when filled with an "X", and as disadvantages when they do not. Those attributes are selected from the main challenges found previously, e.g., ground truth and encryption of the Internet flows, evolution, and implementation of the ML models. From the table, we can notice that private data is standard in this application, with some of them correctly labeled with their ground truth. Statistical based features stand as the most used along with ML models. We can notice that most of the approaches do not treat encrypted data and the evolution of the Internet network.

To conclude this section, we remark the need for counting with labeled historical data with a diversity of Internet communication protocols (encrypted or not encrypted). Moreover, evolving approaches are necessaries; otherwise, ML model implementation efforts are diminished. In the following section, we outline the scope of this investigation.

1.2. Contributions

The challenges in this field have guided the contributions of this work. Therefore, we will list each of them that fall in the area of Internet traffic classification for Satellite communications.

i The first contribution presents an architectural proposal that couples a classification solution within a Satellite Architecture for QoS management. This proposal will treat some communication protocols, such as tunneled communications differently.

ii Regarding the Internet classification solution, the contributions within this task are listed below,

- Hierarchical classification: a set of classifiers is organized into classification levels, where the traffic will pass through. These levels are characterized by discriminator classifiers that will divide Internet protocols technologies, such as tunneled vs. not tunneled.

- Encrypted traffic classification: we use the multi-label classification approach for treating tunneled connections. Traditional ML models will classify Internet streams with only one application within the tunnel. Finally, we propose a classification per packet in tunneled connections.

iii The historical data produced on a Satellite emulated platform will be correctly treated, and statistical-based features will represent the Internet data streams. In this context, we propose new features that might improve the performance of the classifiers. Besides, a particular feature extraction process for tunneled connections will be designed.

iv Finally, we study the reliability of implementing our ML solution on an emulated Internet network placed on a cloud platform. Its implementation and tests will serve as a guide for future works on the emulated and real Satellite platforms.

In the next section, we show the organization of this paper.

1.3. Organization

The remainder of this paper is organized as follows. Section 2 briefly presents the problem and techniques of traffic classification. Section 3 presents the general framework proposed. In this section, we also describe all the theoretical steps to achieve monitoring, feature extraction, labeling, and QoS management of Internet flows in a Satellite Architecture. The data used by this investigation is presented in Section 4.2. The evaluations of our solution will be presented in two parts: i) evaluation of the hierarchical classification system in Section 5, and ii) the evaluation of the implementation in Section 6. Some perspectives and conclusions will be outlined in Section 7.

2. Background

To better understand the Internet traffic classification process, we present as follows how to monitor Internet traffic in network appliances in Section 2.1. The monitored traffic can be encrypted or unencrypted; the difference between these types of traffic will be set in Section 2.2. Classical Internet classification approaches will be described in Section 2.3. Finally, in Section 2.4, we formally define two ML classification paradigms that will be used by our framework.

2.1. Internet traffic monitoring

The standard approach in network traffic monitoring is based on the extraction of a set of packets within a time window. The work in [32] presents a comprehensive procedure for Internet flow extraction using NetFlow and IPFIX. More precisely, the work defines the steps for data measurement as i) packet capturing, ii) flow metering and exportation, and iii) data collection. The packet capturing action refers to the procedure of extracting the binary data from monitoring points; in this step, each packet is considered as a single independent entity. Following, the flow metering process aims at aggregating the packets into Internet flows or streams. The exportation process occurs when it is considered that a flow is culminated, meaning that a communication was finished. The metering and exportation processes are related and can be merged. Finally, the data collection is in charge of storing the flows exported.

Regarding the data measurement steps, several research works try to improve each of their deficiencies separately. For instance, [33] presents a taxonomy to categorize the packet sampling techniques; this work aims at giving guidelines to select an adequate method according to the objectives to achieve. The works in [34, 35, 36] present packet capture engines running on commodity hardware, which are useful for reducing the response time in traffic classification. The work in [37] reports the most common implementations of widespread network monitoring approaches for packet capture (Tcpdumb, Wireshark, etc.), flow metering (nProbe, YAF, QoF, etc.), and data collecting (nProbe, flowd, nfdumb, etc.).

Against that background, a correct monitoring system has to be provided for Internet traffic classification. However, suffice to say that there already exist well-defined traffic monitoring tools that cover this matter. To conclude this section, what it is essential to retain is the information captured into Internet flows. As complementary information, we introduce how these flows are represented for different protocols and encrypted technologies in the following section.

2.2. *Encrypted vs Non-encrypted traffic*

Monitored packets create data streams between client-server exchanges (also denoted as source-to-destination, backward-to-forward exchanges). These exchanges are defined by communication protocols, where the Transport Control Protocol (TCP) and the User Datagram Protocol (UDP) are the most popular worldwide. However, the packet content is accessible after monitoring points due to these protocols only define a way to send and to receive information in the transport layer. We refer non-encrypted data to the Internet flows where their packet content can be inspected. In other words, we can access to the packet header and payload transmitted. Usually, these fields are examined to offer services such as traffic classification for QoS management, anomaly, and cyber-attack detection, among others. However, more and more protocols offer header and payload encryption to provide confidentiality of the packets navigating in the Internet network.

The encryption comes along with a structured authentication between peers, data integrity, and replay protection. Most of the encryption protocols follow two steps, i.e., initialization and transport [38]. Even though these steps are classic when using connection-oriented services, they vary in the way they are performed. Normally, for encryption protocols, the initialization is divided into an initial handshake, authentication, and a shared secret establishment. In the first phase, when the authentication is confirmed between peers, secret keys are established, and the transferred data is encrypted with such keys. Some classic encryption protocols are IPsec [38], TCPcryp [39], and TLS/SSL [40, 41]. These encryption protocols encode the packets' content to assure data integrity. In addition to this, they also provide an attractive feature that modifies the packet's IP header allowing the navigation over a virtual tunnel. In this context, it is impossible to avoid mentioning Virtual Private Networks (VPNs), which tunnel multiple flows into only one flow by using encryption protocols such as IPsec [42]. VPNs are deployed to allow securing communications on the Internet and accessing services with geographical constraints, among others. Enterprises all over the world, secure their Internet communications by using VPNs. However, this technology is also accessible to end-users.

For the objectives of this investigation, it suffices to show the difference that the non-encrypted, encrypted, and tunneled communications will carry for the data processing steps. In Figure 1, we can graphically place these differences in a general way. In summary, we observe:

- Non-encrypted data: Header and payload are available. Packet streams are identifiable meaning that we can differentiate and separate the flows sent by the client and those sent by the server. Figure 1(a) denotes the non-encrypted packets streams as white blocks: the blue ones coming from the client (also called source or forward traffic), and the red ones coming from the server (also called destination or backward traffic).

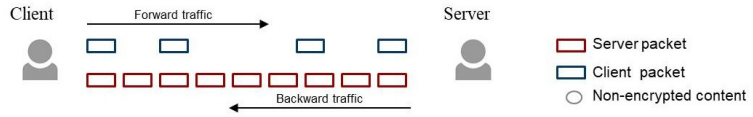
- Encrypted data: A layer of encryption is added, making the packet payload inaccessible. In addition to this, the packet's header is changed by the encryption protocol hiding the original source and destination IP addresses. In Figure 1(b), the packets with payload encryption are filled with gray color. Even though the header is transformed, we can still separate the flows between the client and the server as seen in the figure.

- Tunneled data: Stream content is encrypted, and also, another layer hides the IP directions of clients and servers. In this particular case, the data streams are not separable. In Figure 1(c), we can notice that several clients/servers can be using the same tunnel, mixing in this way the Internet traffic.

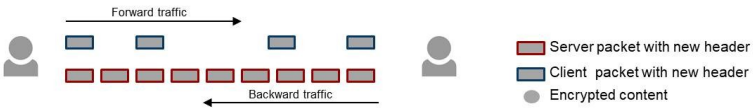
We will notice that the description above will be a crucial factor for discriminating and classifying Internet traffic.

2.3. *Traffic classification approaches*

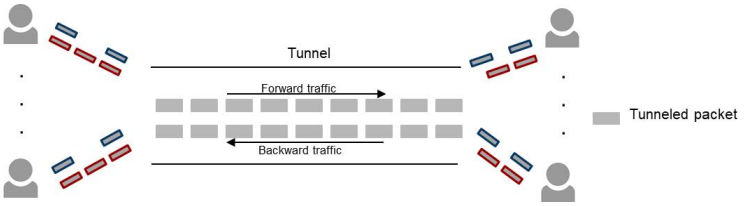
Several trends can be found to classify, comprehend, diagnose, or observe the status of the network such as Payload inspection, ML-based, Statistical-based, and behavioral techniques. Payload inspection commonly denoted Deep Packet Inspection (DPI) is found as the typical approach to perform traffic analysis [9, 7]. This technique analyzes the content of the Internet packets, i.e., the IP header and payload. DPI compares the information extracted from the packets with a set of signatures (previously defined and known) to identify different application protocols. Some of the DPI tools are nDPI, Libprotoident, PACE, L7-filter, and NBAR, among others. Recently, DPI tools have shown several drawbacks due to the growing number of new applications and protocols. Particularly, when a new protocol is created, the DPI tools must be updated. Otherwise, they will fail in their prediction getting, as a result, an unknown or an erroneous signature. As a consequence, the list of the tools' signatures has to be continuously updated.



(a) Non-encrypted Internet streams



(b) Encrypted Internet streams



(c) Tunneled Internet streams

Figure 1: Illustrative example of the difference between non-encrypted, encrypted and tunneled data

On the other hand, DPI is not adequate when a) packet encryption is used to protect the content in communication sessions, b) HTTP2 is deployed for multiplexing the packet content, c) NAT networks are utilized because they unable to differentiate between communication sessions, and d) Virtual private networks (VPNs) are deployed for data privacy and integrity, among others.

The statistical-based techniques try to find statistical differences between flows, communicating end systems and network configurations, among others. Such differences can be the result of two or more different applications or behaviors, characterized by statistical properties. In some contexts, statistical distributions can be used to model the network traffic patterns [43, 44, 45]. On the other hand, behavioral techniques aim at finding patterns among end-to-end communications in a network. It also studies community patterns where the communities are conformed of hosts at different points [46, 47, 48, 49]. The most common representation of behavioral patterns in the network is through graph modeling. Graph theory is used to find highly connected nodes (hosts), number of connections, and opened ports, among others [46].

Finally, ML-based techniques try to classify traffic based on the status of the Internet network. For such a case, IP flows are reported as the most common representation of Internet communications, where representative features (e.g. packet length and Inter-arrival time) can be extracted and used for traffic classification [50]. One of the main strengths of the ML approach is that the feature extraction can be performed without inspecting the packets' content of IP flows. Hence, these features are suitable for creating classification models for encrypted communications.

In the next section, we cover two main ML strategies that will be deployed for our solution.

2.4. Multi-class and multi-label classification

This section will explain two paradigms used by this investigation: multi-class and multi-label classification. If we take a standard classification approach, we know that the problem is to assign a sample to a single class (multi-class classification). In contrast, the multi-label classification problem states that a sample belongs to more than one class at the same time [51]. This new way to see the classification problem has become very useful in different domains; for instance, the more representative cases are found in image and text classification. For example, a text can belong to different categories at the same time. Whereas, an image can be classified into different classes and can have embedded different sub-images at the same time.

To formalize the classical multi-class classification problem, we present the description below.

- Training data $\{X, Y\}$ with $X \in \mathbb{R}^{n \times m}$ and $Y =\in \mathbb{R}^{n}$

- The label of a sample $x_i \in X$ is $y_i \in Y = \{1, 2, ..., L\}$.
- Learn a relation: $f : X \rightarrow Y$
- Each sample x_i is associated with a single label y_i

A simple and effective multi-class classification approach is the Random Forest (RF), which is a method derived from decision trees. This algorithm is based on a bagging strategy where the results of several classifiers are combined to get a better precision. In RF, k decision trees are built and trained with bootstrap samples versions of the original training data. Then, the final result is given by a combination of each tree output [52].

On the other hand, for the multi-label classification problem, we have transformed the labels Y using a binary alphabet $\{0, 1\}^L$. This means that each sample will be related to a vector class where the columns with a value equal to one represent the membership to the class of the column that they represent.

- Training data $\{X, Y\}$ with $X \in \mathbb{R}^{n \times m}$ and $Y =\in \mathbb{R}^{n \times L}$
- The label of a sample $x_i \in X$ is $y_i \in \{0, 1\}^L$
- Learn a relation: $g : X \rightarrow \{0, 1\}^L$
- Each sample x_i is associated with more than one label by the vector y_i

The simplest way to approach a multi-label problem is to divide it into multiple independent binary classification problems (one per class). Nonetheless, relationships between the labels are not considered, which can cause inconsistencies [53]. To overcome those issues, several ML algorithms include the multi-label learning task. For instance, an RF acting as a multi-label classifier changes in the way that the node splitting cost function handles multi-label problems. The label entropy or the Gini index of the child nodes suffer some modifications that lead to consider the node as a bag of positive labels with a certain probability [54].

At this stage, we can conclude that the flows in figures 1(a) and 1(b) can be treated by multi-class classifiers; whereas, the tunneled flows as in Figure 1(c) might be treated by a multi-label classifier. In the following section, we show how an ML-based solution based on multi-class and multi-label classifiers can be placed in a Satellite architecture.

3. Framework

In Figure 2, a general framework to treat Internet traffic is presented. In this figure, an abstraction of a Satellite communication is displayed. This configuration is a simplification of an operational architecture already studied in one of our works [55]. This architecture takes the concerned network functions of a Satellite Architecture and the Policy-Based Network (PBN) architecture integrated with a classification system. In the figure, we have added three levels to provide Internet traffic classification. Level 0 (L0) performs *Data Collection* and *Feature extraction* tasks. Level 1 (L1) and Level 2 (L2) is conformed by a *Hierarchical Classification* that marks the Internet traffic. Level 3 (L3) contains incremental learning approaches for updating the classification solution. Finally, Level 4 (L4) is in charge of performing QoS management.

In the proposed framework, the flow of activities of interest are:

1. Intercept Internet traffic either in the Gateway (GW) or the Satellite Terminal (ST) through passive monitoring points in the *Data Collection* block.
2. Perform feature extraction over the Internet flows.
3. Send the extracted features to the *Hierarchical classification* block and mark the flows with their QoS classes.
4. Finally, the classification is signaled to the Policy Decision Point (PDP) of the GW that will make decisions to improve the QoS.

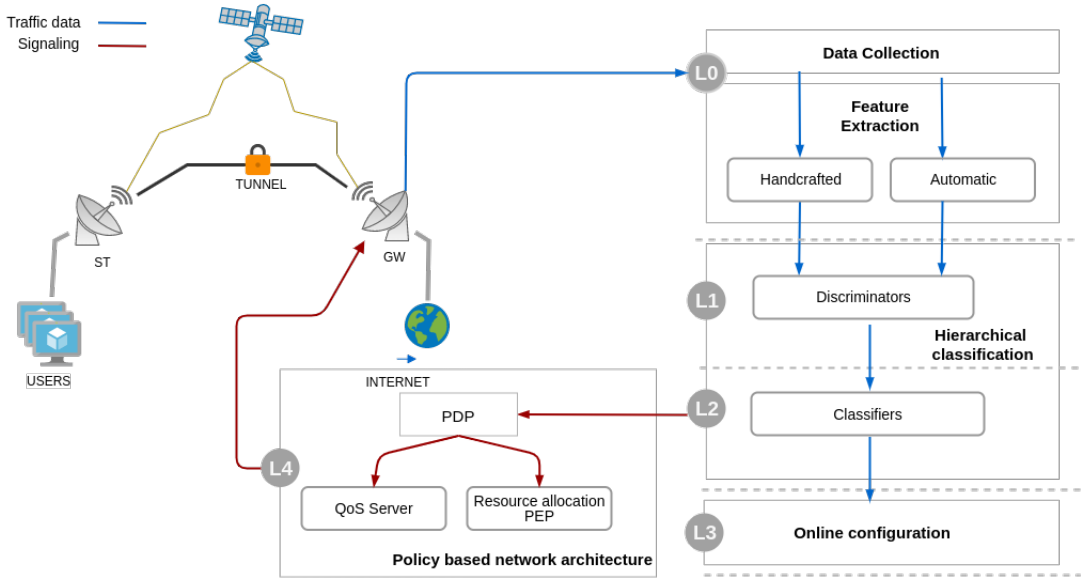


Figure 2: Hierarchical classification system.

In Figure 2, the Internet traffic will be captured from both the GW and ST components, and decentralized classification could be performed. However, another option will be to capture and to treat the traffic only in the GW. These modifications depend on the architecture choices, and they will not affect the functional operations of our ML-based classification solution. The authors recently presented more details about the Satellite architecture, coupled with this framework in [55]. In that work, more formal implementation guidelines were established. Therefore, in this paper, we will focus only on the theoretical basement of the levels 0-3 of the framework as follows.

3.1. Data Collection

Internet packets are captured to be organized into traffic objects, corresponding to traffic aggregates sharing some peculiarities. This action comprises steps such as packet capture, flow metering, and exportation. Passive monitoring points will be in charge of collecting packets of a communication session from the beginning of the connection until the end (for example, when the timeout is reached). In traffic classification, it is commonly used the term "flow" to describe communications between peers. An aggregated flow, according to [56], is a set of packets or frames in the network intercepted in a monitoring point during a time interval. The packets belonging to the same flow share several common properties. That is a) transport or application header fields (e.g., the destination IP address and the destination port number, among others), b) characteristics of the packets such as the number of MPLS labels, c) additional fields, such as the next-hop IP address, the output interface, etc.

Therefore, we can outline the following classical definition for an unidirectional flow F_i:

Definition 1. *A flow F_i is described by the set,*

$$F_i = \{H_i, P_i\} \tag{1}$$

where H_i is the header of the flow, and $P_i = \{p_{i1}, ..., p_{in}\}$ is a set of packets belonging to the flow.

Definition 2. *A flow header H_i is described by the tuple,*

$$H_i = (IP_{src}, IP_{dest}, port_{src}, port_{dest}, proto) \tag{2}$$

where IP_{src} and IP_{dest} are the IP source and destination addresses. $port_{src}$ and $port_{dest}$ are the source and destination transport port, respectively; and proto is the transport protocol.

A bidirectional flow $F = F_{scr} \cup F_{dst}$ is composed of a source flow F_{src} and a destination flow F_{dst}, both normally identified when some elements of the headers H_{src} and H_{dst} match.

In the next section, we show how the feature extraction is performed over the Internet flows.

3.2. Feature Extraction

The feature extraction can be divided into two main branches called in this work handcrafted and automatic. The handcrafted feature extraction consists of manual computation of statistical features performed over Internet packets, as it will be described in Section 3.2.1. On the other hand, the automatic branch only needs the binary information of the packets that will be fed to DL architectures to perform automatic feature extraction (refers to Section 3.2.2).

3.2.1. Handcrafted

Statistical based features are computed for each flow to describe the communications. The feature extraction process is applied over the packets that are captured in the monitoring step for each flow. However, when a tunneled connection is detected, the flow is broken into chunks of flows within a time interval, as seen in Figure 3. Then, statistical-based features are computed for each chunk of flows to describe the communications.

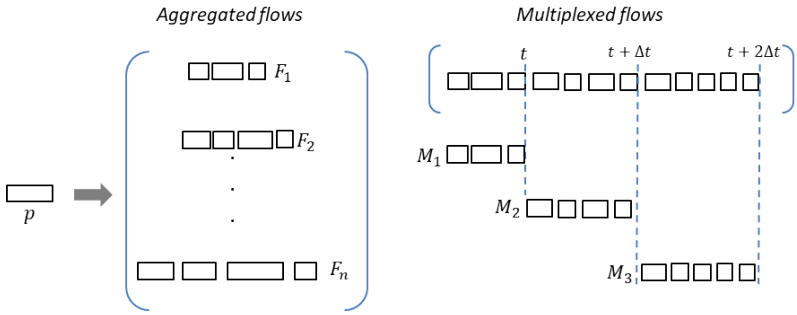


Figure 3: Flow reconstruction.

We will describe as follows the feature extraction procedures for both the aggregate and the tunneled flows.

- Feature extraction for aggregate flows

  The features extracted from packet flows are mainly statistical-based features, which are defined under the assumption that traffic at the network layer has statistical properties (such as the distribution of the flow duration, flow idle time, packet inter-arrival times and packet lengths). These properties are unique for certain types of applications and enable their identification. Under this assumption, the work in [50] proposes 249 statistical features, which can be extracted from flow network traffic. Properties such as inter-arrival time (IAT) and packet lengths are the most important characteristics considered, with their metrics, such as maximum, minimum, mean, and standard deviation, among others.

- Feature extraction for tunneled flows

  Intuitively, the Internet traffic over a Virtual Private Network (VPN) session changes the definition of an aggregate flow, in the sense that all the client flows are embedded in only one flow as it is shown in Figure 3.

  Definition 3. *Formally speaking, given a set of flows F, each of their packets is tunneled after passing through the VPN client by adding a new header. This new header will transform the set of flows into only one flow TF defined as,*

$$TF = \{TH, F\} \tag{3}$$

  where $F = \{f_i, ..., f_n\}$ is a set of flows, which in turn are conformed by an ordered sequence of packets, and TH is the shared tunneled header.

It is important to remark that within the tunnel, the flows are multiplexed, and in consequence, the order of the packets is unknown. The server takes this tunneled flow and demultiplexes them to reconstruct the same flow $\hat{F}$ with the difference that the IP addresses are changed. The main objective of our investigation is describing what types of applications are within the tunneled flow of TF.

It is worth noting that a TF can be an endless flow with no recollection of the opened or closed sessions and the number of clients within the tunnel, among others. For ML-based solutions, modeling this behavior might be complicated due to the statistical properties might be different from tunnel to tunnel, which might decrease the classification performance and generalization. What we propose is to continually cut this TF in chunks of sub-flows that will be partly "independent" of the global behavior of the tunnel, as it is illustrated in Figure 4.

We proposed to compute a set of statistical-based features over two flows: a small sub-flow SF and a bigger sub-flow BF. Taking into account that: i) the classification is performed over SF, and ii) BF serves as a memory to register past behaviors in the tunneled connection. Figure 4 illustrates the creation of such sub-flows, which are defined utilizing a time window. In the figure, we can notice that Δt_1 defines SF_1, while Δt_2 gathers the statistical information about BF . In further experiments, we will determine the adequate values for Δt_1 and Δt_2. It is important to mention that in an online manner, the construction of these flows is performed by sliding the time windows.

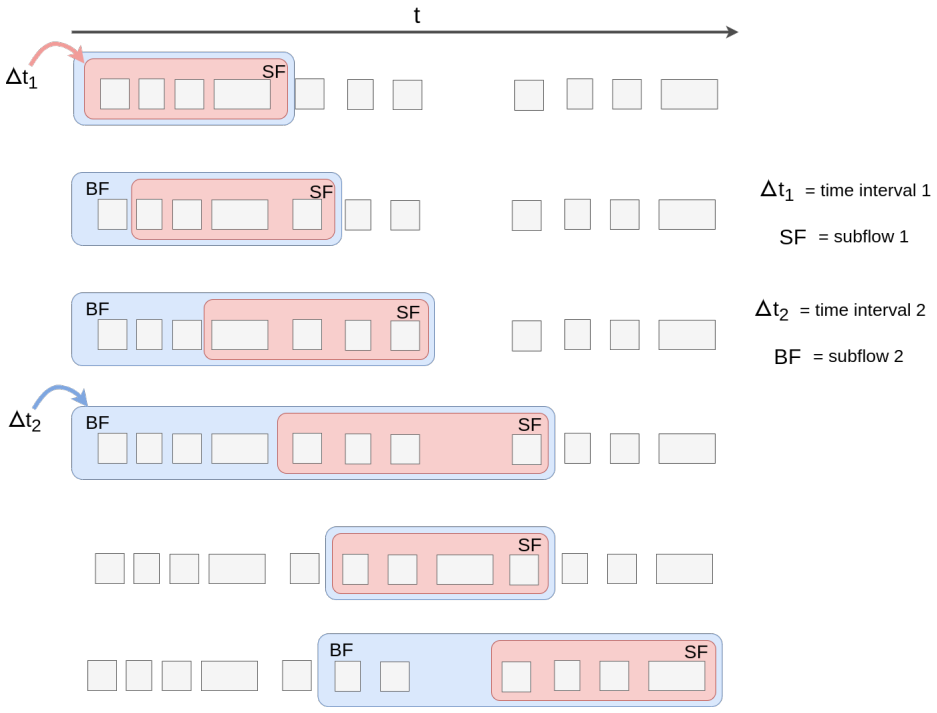


Figure 4: Flow construction for a VPN communication

Once the flow construction is finished, we used the same statistical based features than the aggregated flows for the SF and BF in F,F_{src} and F_{dst}.

The results of the handcrafted feature extraction process are presented as follows. In Table 2, the features computed for aggregated flows are listed along with their description and the flow direction used. Whereas in Table 3 shows the features for the tunneled connections. In addition, to compute these features, the most important parameters to gather per-flow in the monitoring process are:

- ID of the flow f represented by a tuple $t(f)$
- Packet flow p in both directions f, in the source and destination f_{src} and f_{dst}

- Arrival of the last seen packet for f, f_{src} and f_{dst}
- Statistical features for f, f_{src} and f_{dst}
- ML classification value $c(f)$
- Type of flow tf flag, tunneled or not

Feature	Metric	Additional Information	Total of flows	Total
pktlen_[m]	[m] of the packet lengths	"m" refers to the metric Mean, Std, Min and Max	3	12
iat_[m]	[m] of the inter-arrival time(iat)	-	3	12
bytes_[Δt]	bytes per [Δt]	"Δt" is the time windows	3	3
pkt_[Δt]	packets counts per [Δt]	-	3	3
		Total		38

Table 2: Result of the feature extraction process for aggregated flows

Feature	Metric	Additional Information	Total of flows	Total
pktlen_[m]	[m] of the packet lengths	"m" refers to the metric Mean, Std, Min and Max	6	24
iat_[m]	[m] of the inter-arrival time(iat)	-	6	24
pktlen_[cat]_[m]	[m] of the packet lengths per [cat]	"cat" refers to the type of packet[a]	6	144
iat_[cat]_[m]	[m] of the iat per [cat]		6	144
		Total		336

[a] A: pktlen <= 170, B: pktlen > 170 and pktlen <= 902, C: pktlen > 902 and pktlen <= 1314,D: pktlen > 1314 and pktlen <= 1426,E: pktlen > 1426 and pktlen <= 1500, F: pktlen > 1500

Table 3: Result of the feature extraction process for tunneled flows

3.2.2. Automatic

The automatic feature extraction is inspired by the works in [57, 16, 58]. In particular, the work in [57] proposes the use of Convolutional Neural Networks (CNN) for Internet traffic classification. An automatic feature extraction process is achieved by capturing the n bytes of the binary packets. These bytes are directly fed to a CNN to perform automatic feature extraction and classification. The work used 784 bytes of Internet sessions and flows; besides different layers of the OSI model were considered for the study. The resulting samples were reshaped into 28x28 images to make an analogy to the MINIST dataset. After this process, the results demonstrated that good classification performance is achieved and that the use of all layers/session contributes to this result. In a more extensive work, the authors in [16] tested the same approach under different DL architectures.

At this stage, the framework could be adapted to offer an automatic feature extraction process. Therefore, we took the packet processing steps presented in [57] and adapt it to our needs. For this process, we take into account the following aspects:

- The Internet communications will be divided by sessions (or bidirectional flows as in Section 4.2.3).
- All the layers are considered for this approach due to it was demonstrated by the previous works that they offered the best results.
- The bytes of the first 20 packets are considered for aggregated and multiplexed flows. The flows will be split as in Figure 3 for the handcrafted approach.

- The hexadecimal string of the first 20 packets is retained. This string is then converted to a vector of decimal values. This vector is resized to the desired input to the DL approach.
- For the size of the CNN's input, we selected the same dimension as the MINIST dataset ($n = 28$); but we also compute a new dimension resulting from the average bytes from the first 20 packets.
- Finally, the vector might be completed with zeros if it does not reach the size $n * n$. The result will be reshaped into an image format and normalized.
- For the classification, we evaluate the 20 first packet of the session over a CNN model. Each packet will provide a classification for the session until the threshold is reached. Different strategies can be adapted to define the final classification of the session, such as a voting process regarding the class of the first 20 packets or a classification per packet that considers all the incoming packets of a session.

To culminate this section, Table 4 shows the procedure to achieve the first level of *Data Collection* and *Feature extraction*. In short, the header of sniffed packets p allows getting the tuple t that serves as the flow's identifier. If the packet belongs to an existing flow, the statistical features of such flow are updated, and the hexadecimal stream is retained. The tunneled flows have the peculiarity that once the elapsed time $t_e(f)$ is upper than Δt, the flow's parameters are reset. Ending flows are always notified to the GW to keep track of the closed sessions. The handcrafted or automatic features will be fed to a classification system that is detailed as follows.

Macro algorithm
Input: Network interface N **Procedure:** For p sniffed in N, 1. Get the tuple $t(p) = (IP_{src}, IP_{dst}, proto)$ from p 2. If p belongs to an existing flow $f \in F \vert t(p) = t(f)$ 2.1. If tf is False (f is not tunneled), 2.1.1. Compute statistical features SF over f % HANDCRAFTED 2.1.2. Retain the hexadecimal stream of p % AUTOMATIC 2.1.3. Update $t-1$ parameters of the flow 2.2. Else, 2.2.1. if $t_e(f)$ ¡ Δt, do the steps from 2.1.1. to 2.1.3. 2.2.2. Else, do the steps from 3.1. to 3.2. 3. Else, 3.1. Create a new flow with $t(p)$ 3.2. Update $t-1$ parameters of the flow 4. Check for flows in idle timeout end for Output: $SF, t(p)$

Table 4: Macro algorithm for passive monitoring and feature extraction

3.3. Hierarchical classification

We propose a classification per level as it is depicted in Figure 5. In summary, after inline traffic is processed, Internet traffic features are evaluated by several classifiers disposed hierarchically. That is to say, that the flow of information is bifurcated depending on its properties. The levels of this system are detailed as follows.

- In L1, session discriminators are disposed. *Discriminator 1* separates tunneled from not tunneled sessions. Whereas, *Discriminator 2* detects if a tunneled session has unitary or multiple sessions.

- In the second level L2, a standard classification is performed. *Classifier 1 (Cl1)* marks the not tunneled/aggregated flows, and *Classifier 2 (Cl2)* marks the unitary tunneled connections. A *Multi-Label Classifier (MLC)* and *Packet Classifier (PC)* treat tunneled traffic with multiple sessions.
- Finally, in the third level L3, class evolution might be detected by an *Incremental Learning Model (ILM)* that will induce reconfigurations over the $Classifiers$.

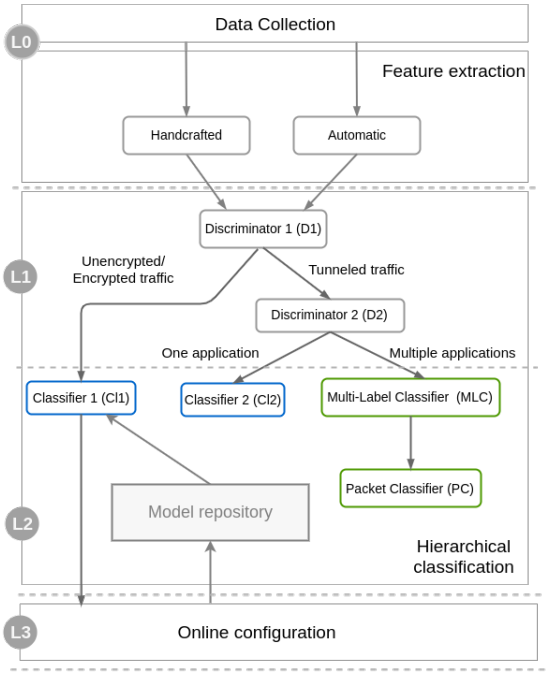


Figure 5: Hierarchical classification system.

It is important to mention that there are already works that aim at hierarchically treating Internet traffic. For instance, the work in [59] proposes to classify the traffic into main classes such as P2P, HTTPS, MSN, SSL/TLS, etc. Afterward, the categories are organized into Internet applications. This approach gets good granularity results; however, tunneled applications are not studied. A similar work is presented by [60], where the objective is to know at first what Anonymity Tools (ATs) are used in the Internet communication, and then to get the name of the Internet application. The first work differs to our approach in the sense that the class/application granularity is not our interest; instead, we are looking for dividing Internet communication protocols (e.g., tunneled, encrypted and non-encrypted). The second work shares similar aspects to our approach and demonstrates that traffic using different ATs can be separated; however, tunneled traffic with multiple sessions is not treated.

We provide more details about the main elements of the framework in the next sections.

3.3.1. Discriminators and Classifiers

The discriminators are in charge of separating different types of traffic, as it is illustrated in Figure 5. We will evaluate different classical ML classifiers to perform this task with handcrafted features. In principle, based on our experience with this application [55], we found out that the most accurate ML approach is the tree-based one. Particularity, Random Forest (RF) highlights thanks to its capabilities for dealing with class imbalance; this ML model is also suitable for multi-label tasks. We will test other ML algorithms such as the K-nearest neighbors (KNN), Decision tree (DT), Extra Trees, Voting, and AdaBoost to build the discriminators.

For the automatic feature extraction, we will deploy two Convolution Neural Networks (CNNs) architectures: 1 dimensional (1D-CNN) and 2 dimensional (2D-CNN). The DL architecture is the same as defined by [16, 57]. In summary, a convolution layer with 32 kernels of size 5 transforms the data at first. A max pooling operation with size

2 is performed. The result is passed through another convolution operation with 64 channels and size 5. Another max pooling layer is added with the same size as the first one. Following flatten and dropout operations are added to the architecture to transmit the features to a fully connected layer with size 1024. A final fully connected layer uses a softmax function to obtain the probability of each class.

Finally, the classifiers belong to the second level of our hierarchical classification, as in Figure 5. They are in charge of defining the class of the traffic. This case will be handled similarly as the *Discriminators*.

3.3.2. Tunneled traffic treatment

This branch aims at detecting the classes within a tunnel with multiple sessions. Given the characteristics of the problem, the multi-label classification task seems to be a suitable path. The work in [61] already demonstrated that multi-label classification could be achieved by a Neural Network-based approach to detect Streaming connections in tunnels. Therefore, this approach will allow us to determine the QoS classes in the multiplexed sessions; however, another interest is to detect explicitly to which class belongs to each packet. In Figure 6, we propose two approaches to deal with multiple applications within a session:

- **Flow classification**: a multi-label classifier defines the classes of the flow SF. In this case, with a multi-label classifier, we expect to have a set of positive labels with a certain probability. In Figure 6, we define the positive class those with a probability higher than 0.6, for example, the VoIP and Browsing classes.
- **Packet classification**: an incoming packet p from the flow SF is classified into only one class by using its length and the statistical-based features of SF and BF. The packet classification task will be treated as a multi-class classification problem.

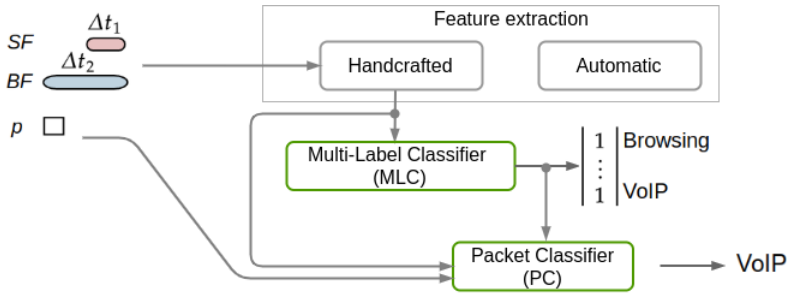


Figure 6: Classification process for multiplexed flows.

These two strategies will be leveraged to find the most adequate for QoS management, given specific protocols.

3.3.3. Online Configuration

Regarding the evolution of the Internet, the main objective behind is to offer a self-learning and self-configuration approach to i) detect new traffic behaviors (new Internet applications), ii) learn or complete the knowledge of minority classes, iii) improve the classification performance, and iv) cope with the evolution of Internet network. The objective will be to have a *Online configuration* block that continuously interacts with the *Model Repository* to induce configurations when required. This idea is not developed in this paper; however, it is considered for the modeling of our framework.

In the next section, the data used by this investigation is described.

4. Internet traffic data

We present Internet traffic data to test and validate our framework. In Section 4.1, a common public dataset is described. Whereas, in Section 4.2, we present a dataset developed in an emulated Satellite architecture.

4.1. VPN-NonVPN

The authors in [62] launched the most common applications in the Internet network and reproduced the same experiments encrypting the data under a VPN connection. The data counts with six QoS classes: Chat, Streaming, VoIP, Browsing, P2P, File Transfer, and Email. The authors captured the complete set of flows for each class, and the sessions were saved in binary files with their respective file names.

The labeling was done using the file identifier. A summary of this dataset is presented in Table 5. For this dataset, the tunneled connections were broken into windows of $\Delta t = 300ms$. Some flow sessions were discarded; for instance, short sessions with less than ten packets and Tor applications.

Applications	Class	Flows
Skype, Facebook, Hangouts, ICQ, AIM,	Chat	612
	VPN-Chat	13580
Skype, FTPS, SFTP	FT	899
	VPN-FT	23832
Bittorrent, uTorrent	P2P	48
	VPN-P2P	1859
Hangouts, Facebook, Skype, Voipbuster	VoIP	1933
	VPN-VoIP	87792
Youtube, Netflix, Spotify, Vimeo	Streaming	448
	VPN-Streaming	12395
Email, Gmail (SMPT, POP3, IMAP)	Mail	286
	VPN-Mail	3141

Table 5: Flow and class distribution of the public dataset VPN-NonVPN.

4.2. Emulated Satellite Internet Traffic

The SAT dataset was created in a collaborative project where the authors actively participated, and this dataset is publicly available [1]. The model of a multi-gateway Satellite network with one ST and one GW was set over a virtual environment, as in Figure 7. In Section 4.2.1, we will explain the network environment set in this platform. Section 4.2.2 lists the applications launched, and Section 4.2.3 details the data collection process performed.

4.2.1. Internet Network construction

Figure 7 shows the emulated environment to capture Internet data. We can notice that different workstations are available in the ST and GW subnet. The satellite segment was emulated by OpenSAND [2], which is a platform to emulate Satellite Communications. In addition to this, a VPN configuration is disposed between the ST and the GW to emulate tunneled communications. The Internet access is done via the GW, and all the traffic from or to the Internet passes over the satellite link through the VPN tunnel. The same is valid for traffic between the two subnets. Finally, two network conditions are proposed:

- SAT ethernet (SAT_eth): an Ethernet connection is disposed between the GW and ST. A delay of $50ms$ is imposed on all the communications.
- SAT OpenSAND (SAT_os): a satellite connection is disposed between the GW and the ST with OpenSAND. The emulated satellite configurations are a constant delay of 250ms, emission of Digital Video Broadcasting Satellite (DVB-S) from the GW to the ST 40Mbps, and from the ST to the GW 5Mbps.

Finally, the VPN technology selected to create the tunnel was OpenVPN [3]. To configure the VPN, four protocols were used: tcp and udp use TCP and UDP as transport protocols; in this case, the packets are only tunneled. This configuration is commonly used to accelerate the response time. Whereas, tcp_sec and udp_sec use TCP and UDP as transport protocols with the packets encrypted and tunneled.

[1] http://eexposit.perso.univ-pau.fr/content/SAT_dataset/
[2] http://opensand.org/
[3] https://openvpn.net/

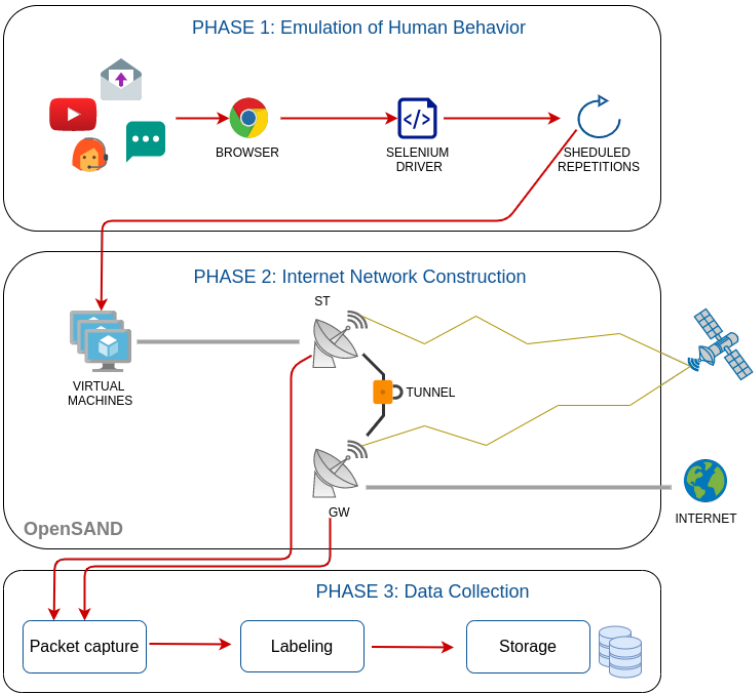


Figure 7: Traffic emulation platform proposed in a Satellite Architecture.

4.2.2. *Emulation of Human Behavior*

Several applications were launched and captured by OpenBACH [4]. The user behavior was mimicked by using Selenium [5], which is a tool to test web applications. It is important to mention that different scripts using selenium were customized to open/close/remain in Internet websites as similar as possible as the human behavior. The applications were launched differently to get a heterogeneous dataset; for instance, different codecs and websites were used for the VoIP and browsing applications, respectively. In Table 6, we show the applications launched with their variations; their duration varies from 5min up to 15min. In addition to those applications, the service mail with "smtp" was launched only for the SAT_os.

QoS class	Application	Parameter	Value
VoIP- Voice	Skype Facebook Twinkle	Audio track Audio track codecs	3 tracks randomly launched 3 tracks randomly launched G.711, G.726 and GSM randomly launched
VoIP- Video	Skype Facebook	Video track Video track	3 tracks randomly launched 3 tracks randomly launched
Video Streaming	YouTube	Content	Random list of videos
Browsing	HTTP/HTTPS sites	Duration Website launched	From 10s to 60s randomly selected Randomly selected

Table 6: Description of the applications launched for the SAT data.

This launching process was performed in three main scenarios on the platform: i) Internet traffic without the tunnel ii) Unitary scenarios with the VPN: only one application at a time is launched, and ii) Multiple scenarios with the VPN: several applications are launched at the same time.

[4] https://www.openbach.org/
[5] https://www.seleniumhq.org/

4.2.3. Data collection

For each scenario, the data collection process was performed in the GW, ST, and within the tunnel. In this sense, all the possible transformation that the data perceived is recorded. The labeling process is performed per file and packet. However, for the VPN tunnel, special treatment was performed. For each packet getting into the VPN tunnel, a flag was used to denote the application launched. Therefore, the multiplexed connections are correctly labeled.

Figure 9 shows the structure of the resulting SAT data. The applications were launched without a VPN (denoted in Figure 9 as None) and with a VPN. In the VPN branch, four protocols were used: *tcp*, *udp*, *tcp_sec* and *udp_sec*. Following, for each protocol, the unitary and multiplexed scenarios were launched, and traffic was captured at the same time in the ST and the GW. This dataset is still in development. In Table 7, we show the flows captured per application and the number of packets with and without the VPN.

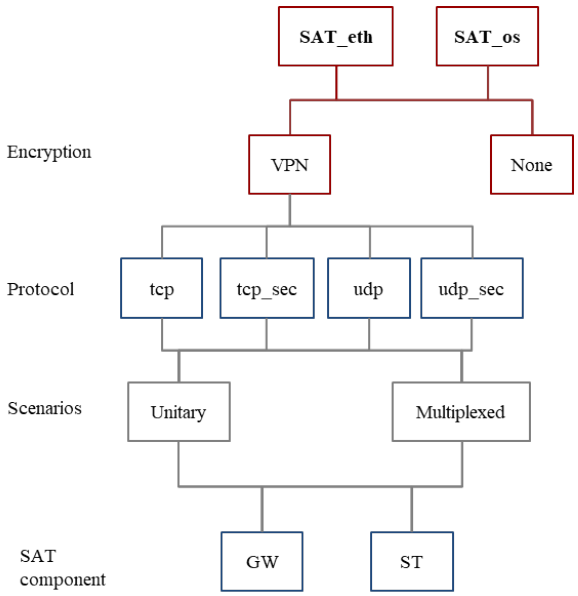


Figure 8: Illustration of the experiment launched in the emulated SAT plaform.

		SAT_eth				SAT_os			
		without VPN		**with VPN**		**without VPN**		**with VPN**	
QoS class	Application	Flows	Packets	Packets: Unitary	Packets: Multiple	Flows	Packets	Packets: Unitary	Packets: Multiple
VoIP	facebook	302	227997	74904	522275	895	392383	35624	1046296
	skype	565	315281	60764	673780	2258	1701348	52895	3718250
	twinkle	69	141663	26144	276995	432	179855	16465	673556
Video	skype	579	925391	318335	2235781	1972	828860	176840	3718250
	facebook	357	558880	162822	1000071	895		74883	2156023
Streaming	youtube	760	158177	19619	486141	2370	392598	25977	1207393
Browsing	web_browsing	6852	749979	91705	1824852	10550	800853	58265	2597338
Unknown	unknown	58	2860	1080	2334	356	1219	51	778516
smtp	smtp	-	-	-	-	134	18672	3391	59066

Table 7: Class, packet and flow distribution of the SAT data in the GW.

5. Hierarchical classification evaluation

In this section, we present the results after evaluating each component of our hierarchical classification system. To start, we show the features that will be used for each of our proposed classifiers. Table 8 shows the flows needed for each classifier, depending on the feature extraction process chosen. For instance, for an unencrypted communication, the handcrafted feature extraction will use the bidirectional flows of a session F, F_{dst} and F_{src}. Whereas, the automatic feature extraction does not take into account the direction of the flow, only the packets of the session (e.g., only the packets of F no matter if it comes from the source of the destination).

To obtain early Internet traffic classification, we considered only the first 20 packets of the sessions. For the automatic feature extraction, the flows are targeted with the class of the first seen packets. The scope of this work will not treat the MLC with the automatic feature extraction; these tasks will be envisaged for future works.

		Handcrafted statistical features				**Automatic feature extraction**		
		Aggregate flows	Tunneled flow	Tunneled sub-flow	Packet length	Aggregate flows	Tunneled flow	Packet bytes
Classifier	Description	F,F_{src} and F_{dst}	SF,SF_{src} and SF_{dst}	BF,BF_{src} and BF_{dst}	-	F	SF	-
D1	Flow discriminator							
Cl1	Classifier of unencrypted traffic							
D2	Flow discriminator							
Cl2	Classifier of unitary tunneled traffic							
MLC1	Multilabel classifier of tunneled traffic					N\A	N\A	N\A
PC1	Packet classifier of tunneled traffic							

Table 8: Flows used for each classifier and feature extraction process.

The handcrafted approach will be used to train a Random Forest (RF), a Decision tree (DT), an Extra Trees (ET), a Voting (V), and an AdaBoost (AB) model. The classifiers' settings were modified to obtain their overall best performance. On the other hand, the automatic feature extraction will be performed by 1-Dimensional Convolutional Neural Networks (1D_CNN) and 2-Dimensional Convolutional Neural Network (2D_CNN) with matrix dimensions of $28 \times 28 = 784$ and $\mu \times \mu$. μ is the average bytes of the 20 first packets of the session, and this value is computed for each classifier.

In the following sections, we present the performance of the discriminators, classifiers, multi-label classifiers, and packet classifiers in the same order.

5.1. Discriminators

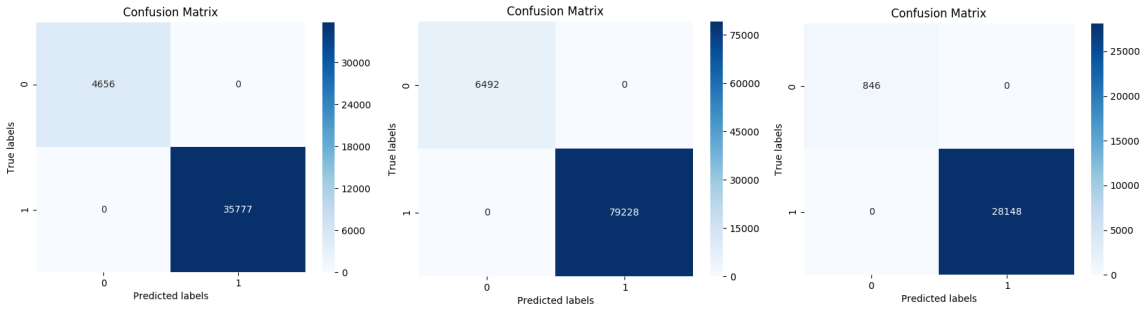
We need adequate classifiers *Discriminator 1 (D1)* and *Discriminator 2 (D2)* that will perform session separation tasks. For this experiment, we trained several ML-based classifiers and compared their performance. The performance will be evaluated in terms of accuracy and G-mean over 5-folds.

For $D1$, we need to know if the input flows are multiplexed or not. To do so, we partition the tunneled flows into windows of $\Delta t = 300ms$ and labeled as "multiplexed", while the rest of flows are "non-multiplexed". From this data, 66% of the samples were used for the training process, and the rest for the test. We train several classifiers and evaluate the test set. Table 9 shows that discrimination capabilities are achieved by using statistical-based techniques. However, we notice that the CNNs are also adequate for this task.

We selected DT as the most suitable discriminator due to its simplicity, low-cost construction, evaluation, and time response. The confusion matrices of the best DT classifier for each dataset are shown in Figure 9. We can notice from the figures that the class identification is appropriately made, with zero misclassifications.

		SAT_eth-GW		SAT_os-GW		VPN-NonVPN	
Features	Classifier	Acc	G-mean	Acc	G-mean	Acc	G-mean
Hand	DT	100	100	100	100	100	100
	RF	100	100	100	100 (± 0.01)	100	100
	KNN	99.98 (± 0.04)	99.96 (± 0.09)	99.98 (± 0.01)	99.95 (± 0.05)	99.91 (± 0.05)	99.07 (± 0.49)
	ET	100	100	100	100	100	100
	V	100 (± 0.01)	100	100	100 (± 0.01)	100	100
	AB	100	100	100	100	100	100
Automatic	1D_CNN-28	99.99	99.99	100	100 (± 0.01)	99.76 (± 0.29)	99.41 (± 0.66)
	1D_CNN-μ	99.99	99.99	100	100 (± 0.01)	99.61 (± 0.86)	99.46 (± 0.32)
	2D_CNN-28	99.99	99.99	100	100 (± 0.01)	98.44 (± 0.97)	96.69 (± 2.38)
	2D_CNN-μ	99.99	99.99	100	100 (± 0.01)	99.11 (± 0.44)	98.29 (± 1.30)

Table 9: Accuracy (Acc) and G-mean results in percentage after testing the $D1$ classifiers. These results are in $avg(\pm std)$ format obtained over 5-folds.



(a) SAT_eth-GW (b) SAT_os-GW (c) VPN-NonVPN

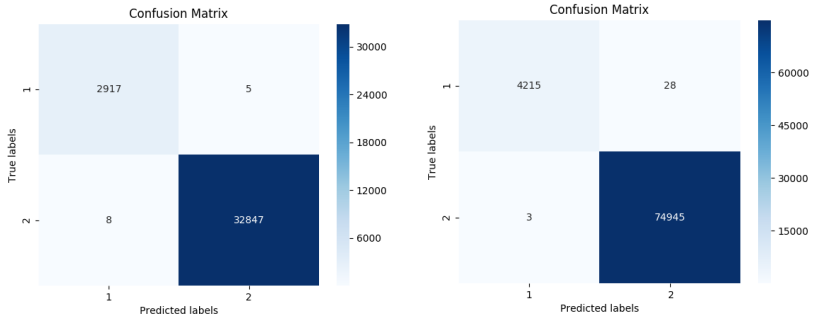
Figure 9: Confusion matrix with the best DT classifier acting as $D1$.

For $D2$, we want to separate the unitary tunneled connections from the multiple ones. We took all the multiplexed connections and divided them into windows of $300ms$, labeling each window with its respective class (Unitary = 1 or Multiple = 2). The VPN-NonVPN dataset is excluded from this study due to it only contains unitary tunneled connections. The results in Table 10 are similar to those for $D1$, allowing us to conclude that a simple DT is also suitable for this task.

		SAT_eth-GW		SAT_os-GW	
Features	Classifier	acc	G-mean	acc	G-mean
Hand	DT	99.92 (± 0.06)	99.73 (± 0.27)	99.95 (± 0.01)	99.72 (± 0.11)
	RF	99.94 (± 0.02)	99.71 (± 0.21)	99.96 (± 0.01)	99.63 (± 0.08)
	KNN	99.46 (± 0.04)	97.78 (± 0.32)	99.55 (± 0.04)	96.68 (± 0.52)
	ET	99.92 (± 0.02)	99.67 (± 0.20))	99.94 (± 0.02)	99.49 (± 0.14)
	V	99.91 (± 0.03)	99.53 (± 0.20)	99.94 (± 0.02)	99.42 (± 0.11)
	AB	99.96 (± 0.01)	99.85 (± 0.06)	99.96 (± 0.01)	99.76 (± 0.16)
Automatic	1D_CNN-28	99.97(± 0.03)	99.60 (± 0.30)	99.75 (± 0.10)	99.86 (± 0.45)
	1D_CNN-μ	99.99 (± 0.01)	99.92 (± 0.22)	99.60 (± 0.03)	99.32 (± 0.52)
	2D_CNN-28	99.75(± 0.05)	99.30 (± 0.45)	99.62 (± 0.01)	98.86 (± 0.15)
	2D_CNN-μ	99.99 (± 0.01)	99.92 (± 0.22)	99.60 (± 0.30)	97.92 (± 0.32)

Table 10: Accuracy (Acc) and G-mean results in percentage after testing the $D2$ classifiers. These results are in $avg(\pm std)$ format obtained over 5-folds.

Finally, the confusion matrices of $D2$ with DT are given in Figure 10. We can notice that some of the multiple tunneled connections can be classified as unitary , while fewer flows are misclassified as multiple.



(a) SAT_eth-GW (b) SAT_os-GW

Figure 10: Confusion matrix with the best classifier of DT acting as $D2$.

The flow discrimination tasks allow us to divide the problem into several sub-classification problems. Therefore, once the traffic types are differentiated, we can apply a different classification technique for each type, as it is detailed as follows.

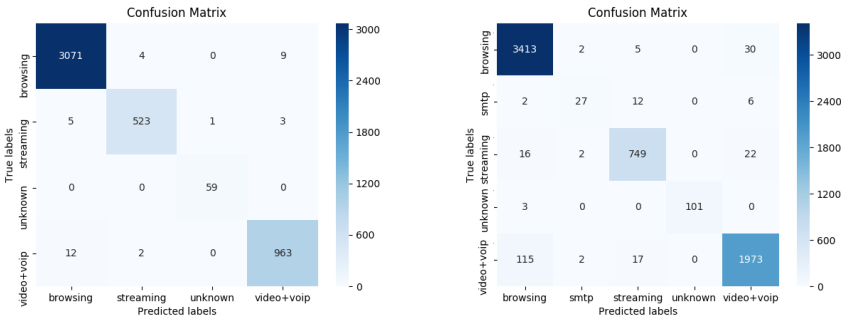
5.2. Classifiers

The discriminator tests were repeated in this section, with the difference that the multi-class classification objective is to obtain the QoS classes from Internet sessions. In Table 11, the results are depicted for $Cl1$ using non tunneled traffic. From the table, we can notice that good classifiers are built for the SAT datasets. However, for the VPN-NonVPN, we encounter lower performances.

		SAT_eth-GW		SAT_os-GW		VPN-NonVPN	
Features	Classifier	acc	G-mean	acc	G-mean	acc	G-mean
Hand	DT	98.03 (± 0.38)	97.57 (± 0.48)	94.24 (± 0.95)	82.74 (± 0.84)	71.04 (± 3.72)	60.31 (± 13.53)
	RF	99.00 (± 0.31)	98.64 (± 0.58)	96.77 (± 0.22)	86.72 (± 0.36)	77.61 (± 2.24)	65.00 (± 10.95)
	KNN	95.98 (± 0.47)	94.28 (± 2.16)	91.34 (± 0.80)	80.42 (± 1.03)	65.10 (± 3.45)	9.24 (± 36.97)
	ET	98.98 (± 0.29)	98.51 (± 0.44)	96.69 (± 0.32)	86.66 (± 0.39)	76.62 (± 1.68)	62.67 (± 16.79)
	V	98.73 (± 0.12)	98.23 (± 0.43)	96.03 (± 0.64)	85.86 (± 0.48)	75.04 (± 1.62)	49.71 (± 50.84)
	AB	63.76 (± 13.69)	49.40 (± 19.33)	50.13 (± 22.89)	46.49 (± 13.89)	38.38 (± 6.17)	17.06 (± 28.25)
Automatic	1D_CNN-28	97.74 (± 0.69)	95.78 (± 3.28)	95.41 (± 0.37)	84.41 (± 1.26)	46.07 (± 0.61)	19.14 (± 27.60)
	1D_CNN-μ	97.86 (± 0.41)	92.62 (± 4.16)	95.08 (± 0.28)	81.07 (± 2.41)	43.99 (± 2.94)	0.00 (± 0.00)
	2D_CNN-28	96.89 (± 0.45)	91.99 (± 3.32)	96.15 (± 0.26)	89.79 (± 4.01)	44.99 (± 2.42)	18.63 (± 31.01)
	2D_CNN-μ	96.23 (+/- 0.46)	84.45 (+/- 3.07)	96.15 (± 0.26)	89.79 (± 4.01)	49.70 (± 4.60)	0.00 (± 0.00)

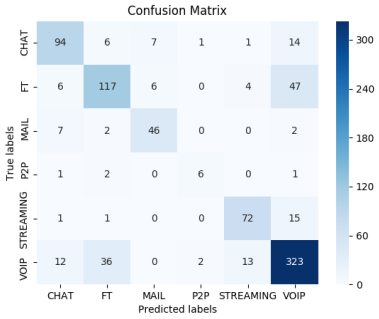
Table 11: Accuracy (Acc), G-mean and F-score results in percentage after testing the $Cl1$ classifiers. These results are in $avg(\pm std)$ format obtained over 5-folds.

The confusion matrices of the best RF classifiers are depicted in Figure 11. In this figure, we remark the class-imbalance presented by the VPN-nonVPN dataset that might deteriorate the performance of the classifiers. Whereas, for the SAT dataset, there are lower misclassifications even though there is a class-imbalance. Another point that can contribute to the performance in both datasets is the labeling process. The SAT dataset was built labeling each packet and marking the unknown packets generated in Internet sessions, while, in the VPN-NonVPN, the labeling process is given by file.



(a) SAT_eth-GW

(b) SAT_os-GW



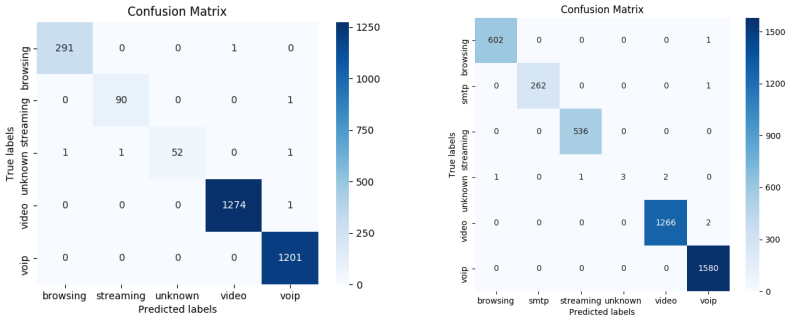
(c) VPN-NonVPN

Figure 11: Confusion matrix with the best classifier of RF acting as $Cl1$.

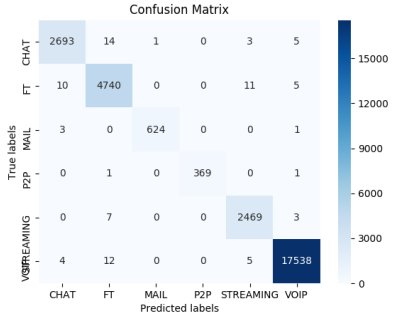
For the classifier $Cl2$, the results are shown in Table 12. In this particular case, class discrimination can be achieved with high performance for all the datasets either with handcrafted or automatic feature extraction. The misclassifications are lower even in the presence of class-imbalance, as seen in the confusion matrices in Figure 12. Regarding the performance of the CNNs, the 1D-CNNs with images of size 28×28 perform better than using the size μ and the 2D-CNNs.

		SAT_eth-GW		SAT_os-GW		VPN-NonVPN	
Features	Classifier	acc	G-mean	acc	G-mean	acc	G-mean
Hand	DT	99.66 (± 0.19)	99.28 (± 0.87)	99.65 (± 0.16)	70.42 (± 71.98)	99.32 (± 0.20)	98.67 (± 0.32)
	RF	99.65 (± 0.13)	99.30 (± 0.64)	99.80 (± 0.13)	70.26 (± 70.80)	99.60 (± 0.07)	99.02 (± 0.26)
	KNN	96.81 (± 0.48)	96.38 (± 1.55)	97.78 (± 0.73)	0.00	97.36 (± 0.35)	94.81 (± 0.76)
	ET	99.59 (± 0.21)	99.16 (± 0.79)	99.75 (± 0.16)	67.85 (± 68.43)	99.53 (± 0.10)	98.93 (± 0.34)
	V	99.61 (± 0.26)	99.30 (± 0.89)	99.73 (± 0.18)	67.87 (± 68.45)	99.47 (± 0.08)	98.82 (± 0.33)
	AB	85.94 (± 12.83)	81.60 (± 14.05)	90.31 (± 3.30)	81.14 (± 18.38)	76.54 (± 8.61)	62.90 (± 13.52)
Automatic	1D_CNN-28	95.48 (± 0.73)	63.62 (± 9.46)	97.47 (± 1.70)	91.14 (± 9.02)	98.80 (± 0.17)	95.99 (± 1.09)
	1D_CNN-μ	92.18 (± 0.83)	23.83 (± 40.37)	87.37 (± 0.74)	41.19 (± 6.42)	90.71 (+/- 0.32)	88.79 (+/- 5.44)
	2D_CNN-28	71.60 (± 3.56)	51.00 (± 52.55)	88.50 (± 0.83)	48.30 (± 6.74)	97.03 (± 0.46)	91.26 (± 1.38)
	2D_CNN-μ	90.57 (± 1.00)	44.87 (± 6.69)	86.19 (± 0.32)	34.72 (± 7.62)	89.44 (± 0.52)	84.17 (± 8.26)

Table 12: Accuracy (Acc) and G-mean results in percentage after testing the $Cl2$ classifiers. These results are in $avg(\pm std)$ format obtained over 5-folds.



(a) SAT_eth-GW (b) SAT_os-GW



(c) VPN-NonVPN

Figure 12: Confusion matrix with the best classifier of RF acting as $Cl2$.

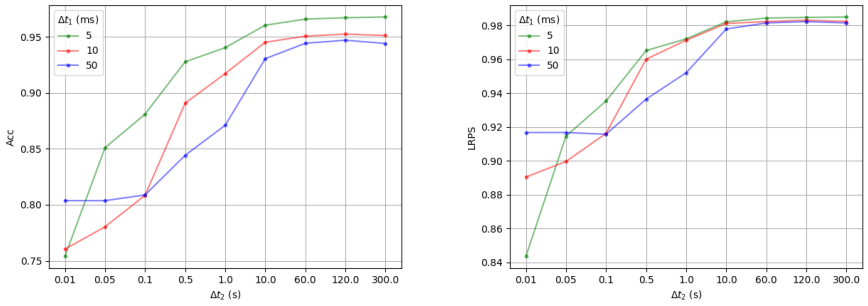
We demonstrated that $Cl1$ and $Cl2$ could be built with adequate accuracy. In the next section, the tunneled connections with multiple applications will be treated by a multi-label classifier and a packet classifier.

5.3. Multi-label classifier: Experimental results

For this experiment, we take the data under udp and udp_sec as the transport protocol. First of all, we analyze the effect caused by the feature extraction process (proposed in Section 3.2.1) to an RF multi-label classifier. Metrics such as accuracy (Acc), Label ranking average precision score (LRPS), and F-score will allow us to interpret the results. Following, we train several multi-label classifiers and evaluate their performance to select the most accurate. Finally, we introduce the packet classifier results in tunneled flows.

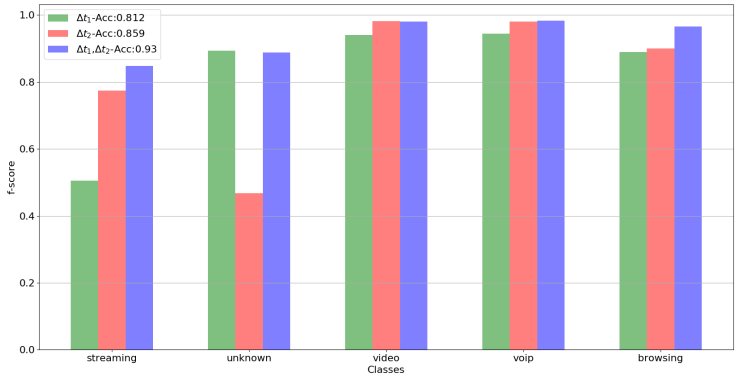
5.3.1. Feature extraction window

The feature extraction over multiplexed connections is performed over two-time windows Δt_1 and Δt_2. Two flows are derived from this construction: SF for Δt_1 and BF for Δt_2. The question that arises is how to define these window values. One can consider the minimum amount of packets needed to perform an accurate classification as classically done for aggregate flows [63]. However, in a tunneled connection, this assumption is not valid due to the variety of scenarios found in a tunnel. For such a case, we opted to empirically propose three possible values of $\Delta t_1 = \{5ms, 10ms, 50ms\}$ to build SF that will represent the statistical behavior of a local stream zone where the classification is performed. The values of Δt_1 are defined according to the maximum delivery delay ($100ms$) allowed by classes such as VoIP, Interactive video, and Video streaming. On the other hand, Δt_2 does not have any time constraints due to it builds a flow BF that serves as additional information. In this case, we vary Δt_2's value from Δt_1 to $300ms$. An RF was trained with all the time windows proposed, and the accuracy and LRPS were computed to select the most suitable combination.



(a) Accuracy

(b) LRPS



(c) F-score

Figure 13: (a)-(b)Accuracy and LRPS of a RF with $\Delta t_1 = \{5ms, 10ms, 50ms\}$, and Δt_2 varied from 0.01s to 300s.(c) F-score of a RF using: only $\Delta t_1 = 10ms$,only $\Delta t_2 = 300ms$ and both $\{\Delta t_1 = 10ms, \Delta t_2 = 300ms\}$.

In the figures 13(a) and 13(b), it is noticeable that the most suitable values of Δt_2 are upper than $10s$ due to they help to increase the accuracy and the LRPS. While for Δt_1, we can conclude that while the smaller its value is the better. The reasoning of this is that a QoS management decision is performed over the local flow SF implemented as a sliding window. This decision can be to prioritize the complete tunnel when some classes of interest are seen. This decision might be updated every Δt_1. We found out that for $\Delta t_1 = 5ms$, the global accuracy was better than the other cases. However, we have to be aware of the functional constraints such as response time when monitoring and classifying time-constrained interactive applications. On the other hand, Figure 13(c) shows the F-scores of each class using three variations of the windows to compute the handcrafted features. We can notice that if we only use SF with $\Delta t_1 = 10ms$, the capability to detect streaming applications is diminished. In the same manner, if we only take BF with $\Delta t_2 = 300ms$, the unknown applications won't be detected. Finally, with the combination of both flows SF and BF, the performance is higher, and the F-score for all the classes is upper than 0.8. This last study gives us hints about the benefits of using two time windows for classifying multiplexed applications.

5.3.2. Multi-label classifiers comparison

In this part, we study in more detail the multi-label classification performance. It is logical to find that the RF multi-label classifier is accurate for this type of application. In Table 13, we validate the use of RF as the multi-label classifier by comparing it with close related approaches.

For the *tcp* protocol the same experiments were launched getting similar results. For instance, the accuracy and LRPS with $\Delta t_1 = 10ms$ and $\Delta t_2 = 60s$ is 0.9261 and 0.9703, respectively.

To conclude this section, we found that our approach to compute the statistical-based feature over two sub-flows in the VPN tunnel allows us to improve the accuracy of the classifier. We also demonstrate that the RF multi-label

		SAT_eth-GW			SAT_os-GW		
Features	Classifier	acc	LRPS	F-score	acc	LRPS	F-score
Hand	DT	88.52 (± 0.22)	95.36 (± 0.12)	95.21 (± 0.07)	91.86 (± 0.04)	96.65 (± 0.05)	96.53 (± 0.03)
	RF	91.51 (± 0.22)	96.82 (± 0.09)	96.51 (± 0.11)	94.11 (± 0.07)	97.69 (± 0.01)	97.51 (± 0.02)
	ET	90.94 (± 0.26)	96.50 (± 0.10)	96.25 (± 0.10)	93.59 (± 0.29)	97.38 (± 0.10)	97.24 (± 0.12)
	KNN	80.49 (± 0.15)	92.04 (± 0.02)	91.72 (± 0.08)	87.43 (± 0.23)	93.97 (± 0.07)	93.70 (± 0.04)
	OnevsALL	32.85 (± 0.81)	61.21 (± 0.65)	60.50 (± 0.65)	27.81 (± 0.05)	56.62 (± 0.13)	51.16 (± 0.22)

Table 13: Accuracy (Acc), LRPS and F-score results in percentage after testing the *MLC* classifiers for the *udp* protocol. These results are in *avg*(±*std*) format obtained over 5-folds.

classifier is the most adequate for this problem. In the next section, we show a small variation of this approach.

5.4. Packet classifier: Experimental results

The multi-label classification results allowed us to validate the construction and discrimination capabilities of our statistical features for tunneled applications. By using those results, we envisage predicting the class of unitary packets instead of flows.

To build this classifier, we consider:

- Only the packets that fall into a flow SF with more than one application at a time are considered. In consequence, we are going to filter all the packets where only one class is predicted for a flow SF.
- The statistical features are created with $\Delta t_1 = 10ms$ and $\Delta t_2 = 60s$ and the current packet length for the handcrafted approach while the automatic feature extraction uses the bytes of each packet.
- With the filtered data, we build a packet classifier modeled by an RF with the handcrafted features and a 1D-CNN for the automatic features.

The dataset characteristics are depicted in Table 14. In this table, we can notice that the total number of samples is equivalent to the total number of packets in a tunneled communication. Whereas, the filtered samples represent the packets that fall into a window Δt_1 with more than one class within the tunnel.

Dataset	Protocol	# samples	# filtered samples (% total)
SAT_eth	*udp*	3527278	196778 (5.58%)
	udp_sec	3425113	216210 (6.31%)
SAT_os	*udp*	4637915	326839 (14.19%)
	udp_sec	5504400	414003 (13.30 %)

Table 14: Data description for the packet classifier

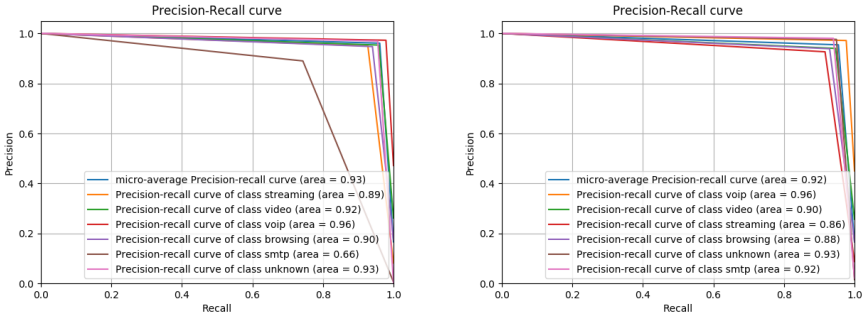
In Table 15, we show the results after building the packet classifiers for each protocol. We notice than the RF models are adequate for this problem; however, the CNNs present competitive results.

		SAT_eth-GW		SAT_os-GW	
Data	Classifier	Acc	G-mean	Acc	G-mean
udp	RF	95.32 (± 0.31)	92.76 (± 1.05)	95.35 (± 0.23)	87.22 (± 2.25)
	1D_CNN-28	99.66 (± 0.22)	98.65 (± 1.37)	99.18 (± 0.49)	93.04 (± 9.51)
udp_sec	RF	95.60 (± 0.16)	92.38 (± 0.85)	94.65 (± 0.17)	92.79 (± 0.33)
	1D_CNN-28	82.62 (± 20.75)	62.63 (± 77.21)	94.22 (± 14.12)	96.61 (± 6.34)

Table 15: Accuracy (Acc) and G-mean results in percentage after testing the *PC* classifiers. These results are in *avg*(±*std*) format obtained over 5-folds.

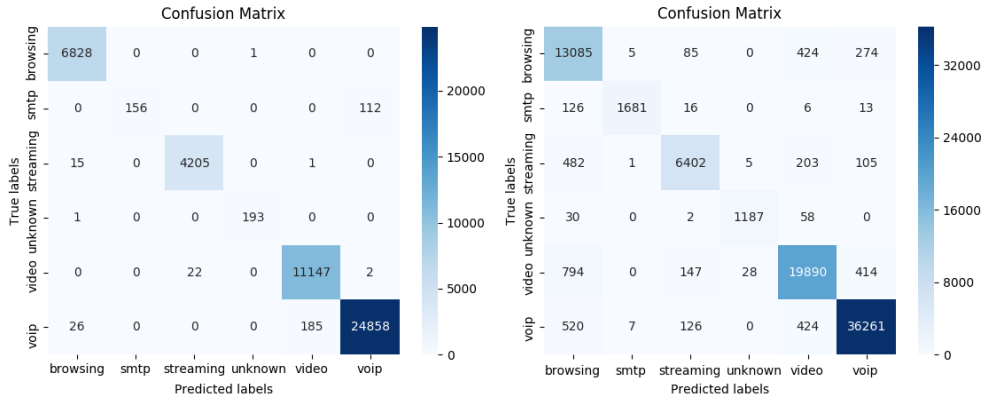
In addition, we show the performance per class expressed by a Precision-Recall curve for SAT_os-GW in Figure 14 for *udp* and *udp_sec*, respectively. We observe that for all the cases, the AUC is upper than 0.95. On the other

hand, Precision-Recall tendencies indicate that the classifiers have good performance levels. However, we can notice that a small decrease is presented for the class "streaming" and "smtp" in OpenSAND which in turn are minority classes. The confusion matrices for the best classifiers using SAT_os-GW are given in Figure 15.



(a) Precision-Recall for class using *udp* (b) Precision-Recall for class using *udp_sec*

Figure 14: Precision-Recall curves for the data SAT_os with *udp* and *udp_sec* as transport protocols.



(a) Confusion matrix of 1D_CNN-28 using *udp* (b) Confusion matrix of RF using *udp_sec*

Figure 15: Confusion matrix with the best classifier acting as *PC*.

From these figures, we can notice that adequate performance per class is achieved. However, when developing these experiments, we discover that there are few samples with more than two applications in parallel with our window configuration (lower than 10 % as shown in Table 14). This characteristic can be attributed to the construction of the data and the flows. For instance, how the applications were launched in parallel and for how long they are running, among others. Therefore, more fine experiments should be addressed for other cases.

6. Implementation evaluation

We need to submit our system to different scenarios to evaluate its performance. It would be ideal to use the emulated Satellite platform (found in Section 4.2) to execute performance tests; however, this will be done for future works. In our case, we set the guidelines to test the classification system in a testbed environment placed on the Proxmox Virtual Environment [6]. We will establish a simple communication between peers acting as source (Internet

[6]https://www.proxmox.com/en/

client) and destination (Internet server) in Section 6.1. A router will be placed between source and destination to intercept and to classify the Internet traffic. Finally, in Section 6.2, we compare the ML-based solution with nDPI in terms of accuracy and response time.

6.1. Experimental setup

Using the Proxmox Virtual Environment, we create a testbed on the cloud (OVH cloud provider). We set the components in Figure 16 with an *User agent* that will act as a real Internet client. This agent will be in charge of emulating real Internet connections. All the traffic generated by this agent will pass through a router before going to the Internet. In this router, we will place our monitoring and classification system. This router might represent the GW server proposed in Figure 7. Connected to this router, we placed a management server that will hold offline processes, such as the training process and the ILM. The configurations established for our router and management server were Virtual Machines (VMs) with the following configuration 4*1.2Ghz CPU and 10Gb RAM. Indeed, these values might vary according to real Satellite resources.

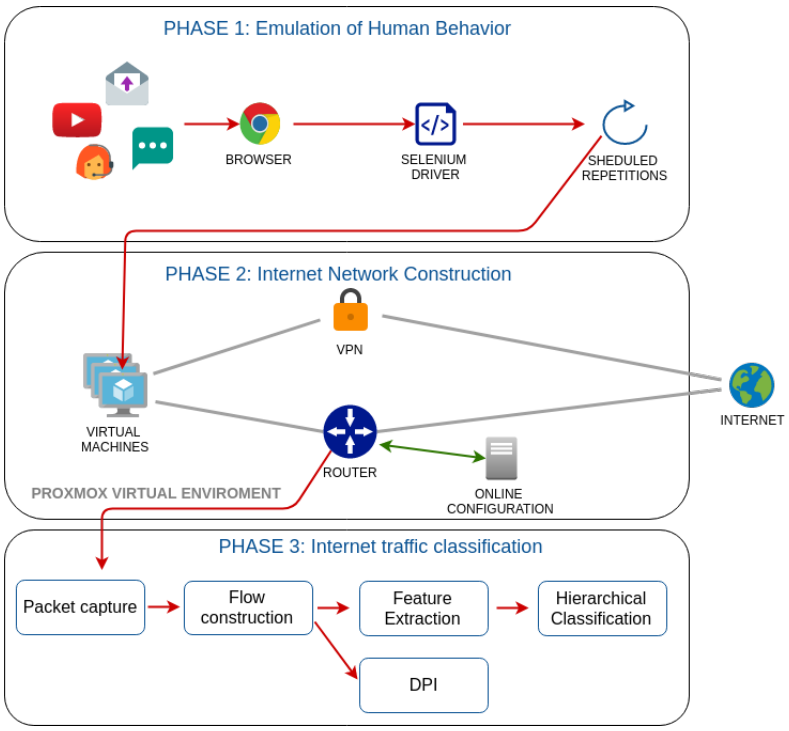


Figure 16: Implementation proposal on Proxmox.

6.2. nDPI vs hierarchical classification results

In this section, we measure the classification capabilities of our implemented prototype against its more close competitor. The complete framework was implemented in C considering only the handcrafted feature extraction process. We use libcap[7] for the monitoring process, while the ML models (trained with scikit-learn[8]) were parsed from Python to C. We measure the average accuracy of the classifiers ($D1$, $Cl1$, $D2$ and $Cl2$) for the SAT_eth data. In Table 16, we can notice that the C models maintain their accuracy. In the unencrypted case, ML outperforms nDPI; while, for the encrypted example, nDPI is unable to detect the class of a unitary session as $Cl2$ does.

[7] https://www.tcpdump.org/pcap.html
[8] https://scikit-learn.org/stable/

Regarding the response time of the classifiers, in Table 17, we show the number of packets per second processed by each approach. We can notice that fast Internet classifications are possible with an ML-based classifier. It is important to mention that the model response time differs for each entry depending on how deep they go into the tree's branches until a leaf is reached.

		ML	nDPI
Unencrypted	D1	0.9999	1
	Cl1	0.9186	0.5830
Encrypted	D2	0.9588	N/A
	Cl2	0.9401	N/A

Table 16: Accuracy evaluating the test data

		ML	nDPI
Unencrypted	D1	348796	1000000
	Cl1	200000	150466
Encrypted	D2	368053	N/A
	Cl2	200000	N/A

Table 17: Accuracy evaluating the test data

On the other hand, we would like to show in Figure 17 the estimated time to perform the complete classification process. For this experiment, we sniff in inline mode the traffic generated by the agent during 60s. We compute the overall time that the packets stay in each activity. In this figure, we can notice that the sniffing process is the longest by using libpcap. This might be caused by the packet processing time in the kernel space, but also by libpcap and the threaded functions designed to deal with the packets. Regarding the classification process, we found out that the average is lower than 15 μs; where the flow metering process is around 5μs and the feature extraction 1 μs.

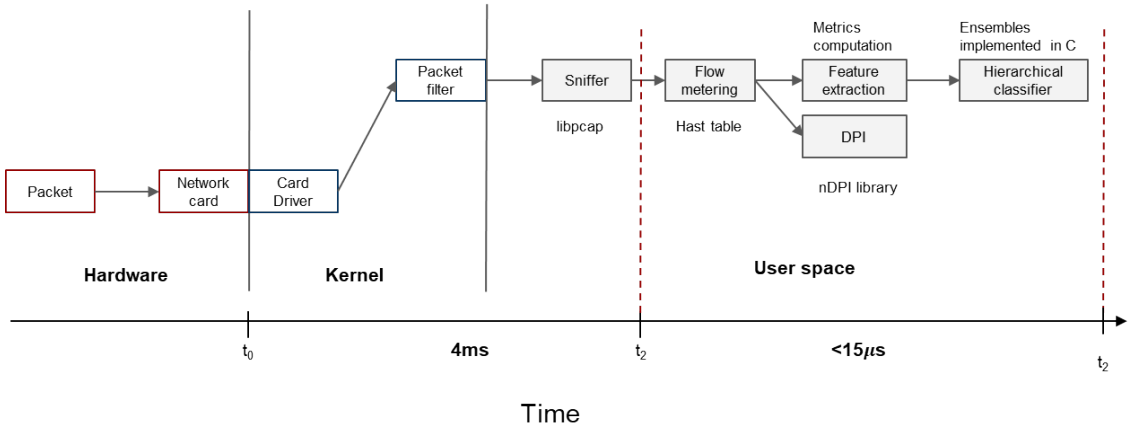


Figure 17: Time analysis of the classification process.

The response time obtained is acceptable principally for the Hierarchical classification task implemented in C. Regarding the packet processing task, the response time could be improved depending on the operational needs of the Satellite architecture. Emerging technologies are dedicated to minimizing packet processing time. For instance, a project called Data Plane Development Kit (DPDK) is a set of libraries devoted to enabling high-speed packet processing [9]. In our particular case, DPDK can be programmed as a kernel-bypass toolkit as it was already proposed by nDPI in its architecture. In addition to this, we need to be sure that the QoS requirements are satisfied on time. For instance, according to [4], VoIP, and Interactive video applications are susceptible to delivery delays. To be specific,

[9] https://www.ntop.org/ndpi/traffic-classification-using-ndpi-over-dpdk/

they can tolerate around 100ms of delay; whereas, another critical class such as Video streaming no more than 10s. We notice that the classification task can be achieved in 15ms, giving sufficient time to treat those sensitive classes.

Regarding the RAM used by our system, we can make a greedy approximation by considering using double variables (8 bytes), and compared with the memory used by nDPI. Therefore, with a total of 346 statistical features and other 160 necessary historical features, we get the approximate result in Table 18. From the table, we can conclude that the RAM consumption increases 4KB per-flow regarding nDPI.

	Memory
nDPI	1KB*F
Hierarchical classifier	(346+160)*8B*F + 1KB*F = (4KB+1KB)*F

Table 18: Memory consumed by our proposal and by nDPI

Future studies should be carried out to enhance the available resource usage (CPU and RAM consumption). It is important to mention that the ML-based solution provokes a negligible overhead over the router; all its operations are reduced to if-then evaluations with few loops involved. Following this line of ideas, future works would instance more tests to perform a more extensive assessment of the system. In the next section, we present the perspectives of our work.

7. Conclusion and Pespectives

Machine Learning has become a valuable tool for knowledge extraction in different domains of science. In our particular case, we intended to design Internet traffic classification techniques for QoS management in Satellite Communications. The structure of Satellite Communications presents a high complexity that yields us to study an abstraction of its components and interactions. We theoretically propose a Satellite communication architecture, as well as, all the necessary elements needed to perform QoS management and to integrate an ML or DL based classification system.

We built and validated all the components of our Hierarchical classification system, getting accurate results. It is important to remark that Internet traffic communications are carried out using different communication protocols. At the same time, Internet applications can change the way these protocols interact with the actors in the network to offer various services such as security and data integrity, among others. In light of this, the description of the Internet stream becomes challenging. Nonetheless, we found out that each combination of application-protocol has some indistinctively statistical properties that allow us to differentiate them. We offer a new way to compute the statistical-based features for tunneled connections to maximize the knowledge extracted from multiplexed flows. The special care and enrichment of the statistical features allow us to assure good discrimination in our classification system. In addition to this, the automatic feature extraction with DL provided promising results.

The theoretical and experimental analysis above-mentioned provides a system that can classify Internet traffic. However, the nature of Internet traffic leads us to deduce that our proposal needs to be extended. Other types of communications should be considered for the Classification System; for instance, different types of tunneled protocols such as IPsec and HTTP2, among others. In this sense, the hierarchical classification could grow in depth and shallowness as it is depicted in Figure 18. New historical behaviors captured from the Internet network might lead to design new *Discriminators* either by using ML, DL, or classical approaches (such as port and protocol identification). Following, a dedicated classifier might correctly analyze each type of communication.

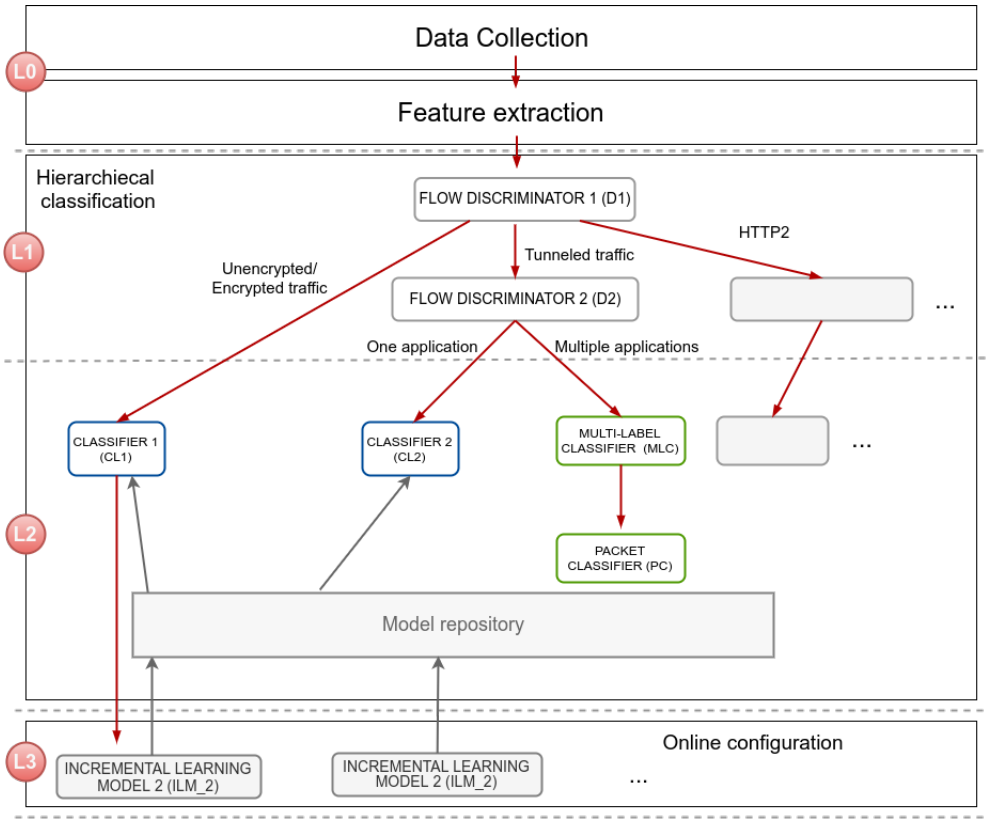


Figure 18: Graphical perspective of the Classification System

Moreover, the class evolution is prevalent for any communication technology. Therefore, it will be necessary to set evolving models to update the classifiers designed. In Figure 18, we integrate an Incremental Learning Model (ILM) for each classifier. This component should be able to deal with the evolution of Internet traffic. The main idea is to have a component that can somehow modify the model repository to update the ML-based classifiers.

To validate our implementation, we submitted our solution to several conditions in which the response was satisfactory regarding its close competitor, nDPI. We set the guidelines to perform the same experiments in an emulated Satellite architecture. Finally, we give access to the emulated Satellite dataset. It is important to mention that more complex interactions between actors in Satellite communications should be integrated into the data created; however, this data serves as a good start point to promote advancements on this field.

Acknowledgment

This work is sponsored by Thales Alenia Space, TOULOUSE, 31100, France. We want to thank the Département : Business Line Telecommunication, R&D department, for their assistance. We also want to thank the Centre National d'Études Spatiales (CNES), Toulouse, France for allowing us to use the SAT data, which is developed under the project R&T CNES: Application du Machine Learning au Satcom.

References

[1] C. Niephaus, M. Kretschmer, G. Ghinea, QoS Provisioning in Converged Satellite and Terrestrial Networks: A Survey of the State-of-the-Art, IEEE Communications Surveys Tutorials 18 (4) (2016) 2415–2441.

[2] B. Hestnes, P. Brooks, S. Heiestad, T. Ulseth, C. Aaby., Quality of Experience in real-time person-person communication - User based QoS expressed in technical network QoS terms, in: Proceedings of the 19thInternational Symposium on Human Factors in Telecommunication, 3–10, 2003.

[3] B. Corrie, H. yee Wong, T. Zimmerman, V. K, Towards quality of experience in advanced collaborative environments, in: in Third Annual Workshop on Advanced Collaborative Environments, 2003.
[4] ITU-T, End-user multimedia QoS categories, Tech. Rep., TELECOMMUNICATION STANDARDIZATION SECTOR OF ITU, 2001.
[5] M. Siller, J. C. Woods, QoS arbitration for improving the QoE in multimedia transmission, in: 2003 International Conference on Visual Information Engineering VIE 2003, 238–241, 2003.
[6] Internet Assigned Numbers Authority (IANA), https://www.iana.org/, accessed: 2019-09-27, ????
[7] T. Bujlow, V. Carela-Espanol, P. Barlet-Ros, Independent comparison of popular DPI tools for traffic classification, Computer Networks 76 (2015) 75 – 89.
[8] F. Pacheco, E. Exposito, M. Gineste, C. Baudoin, J. Aguilar, Towards the deployment of Machine Learning solutions in network traffic classification: A systematic survey, IEEE Communications Surveys Tutorials (2018) 1–1.
[9] R. Antonello, S. Fernandes, C. Kamienski, D. Sadok, J. Kelner, I. Gdor, G. Szab, T. Westholm, Deep packet inspection tools and techniques in commodity platforms: Challenges and trends, Journal of Network and Computer Applications 35 (6) (2012) 1863 – 1878.
[10] Z. Xu, L. Ma, J. Sun, Efficient tri-ary search tree based packet classification algorithm, IET Conference Proceedings (2007) 833–836(3).
[11] W. Pak, Y. Choi, High Performance and High Scalable Packet Classification Algorithm for Network Security Systems, IEEE Transactions on Dependable and Secure Computing 14 (1) (2017) 37–49.
[12] G. Aceto, D. Ciuonzo, A. Montieri, A. Pescap, Multi-classification approaches for classifying mobile app traffic, Journal of Network and Computer Applications 103 (2018) 131 – 145, ISSN 1084-8045.
[13] Y. Lai, Y. Chen, Z. Liu, Z. Yang, X. Li, On monitoring and predicting mobile network traffic abnormality, Simulation Modelling Practice and Theory 50 (2015) 176 – 188.
[14] Z. Liu, R. Wang, D. Tang, Extending labeled mobile network traffic data by three levels traffic identification fusion, Future Generation Computer Systems 88 (2018) 453 – 466, ISSN 0167-739X.
[15] D. Naboulsi, M. Fiore, S. Ribot, R. Stanica, Large-Scale Mobile Traffic Analysis: A Survey, IEEE Communications Surveys Tutorials 18 (1) (2016) 124–161.
[16] G. Aceto, D. Ciuonzo, A. Montieri, A. Pescap, Mobile Encrypted Traffic Classification Using Deep Learning: Experimental Evaluation, Lessons Learned, and Challenges, IEEE Transactions on Network and Service Management 16 (2) (2019) 445–458.
[17] A. Raghuramu, P. H. Pathak, H. Zang, J. Han, C. Liu, C.-N. Chuah, Uncovering the footprints of malicious traffic in wireless mobile networks, Computer Communications 95 (2016) 95 – 107.
[18] J. Riihijarvi, P. Mahonen, Machine Learning for Performance Prediction in Mobile Cellular Networks, IEEE Computational Intelligence Magazine 13 (1) (2018) 51–60.
[19] K. Lalitha, V. Josna, Traffic Verification for Network Anomaly Detection in Sensor Networks, Procedia Technology 24 (2016) 1400 – 1405, international Conference on Emerging Trends in Engineering, Science and Technology (ICETEST - 2015).
[20] C. Zhang, P. Patras, H. Haddadi, Deep Learning in Mobile and Wireless Networking: A Survey, CoRR abs/1803.04311, URL http://arxiv.org/abs/1803.04311.
[21] 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 (1) (2012) 4:1–4:34, ISSN 1556-4681.
[22] I. Trestian, S. Ranjan, A. Kuzmanovic, A. Nucci, Googling the Internet: Profiling Internet Endpoints via the World Wide Web, IEEE/ACM Transactions on Networking 18 (2) (2010) 666–679.
[23] M. Pietrzyk, J.-L. Costeux, G. Urvoy-Keller, T. En-Najjary, Challenging Statistical Classification for Operational Usage: The ADSL Case, in: Proceedings of the 9th ACM SIGCOMM Conference on Internet Measurement, IMC '09, ISBN 978-1-60558-771-4, 122–135, 2009.
[24] L. Grimaudo, M. Mellia, E. Baralis, R. Keralapura, SeLeCT: Self-Learning Classifier for Internet Traffic, IEEE Transactions on Network and Service Management 11 (2) (2014) 144–157.
[25] B. Ng, M. Hayes, W. K. G. Seah, Developing a traffic classification platform for enterprise networks with SDN: Experiences amp;amp; lessons learned, in: 2015 IFIP Networking Conference (IFIP Networking), 1–9, 2015.
[26] L. Bertaux, S. Medjiah, P. Berthou, S. Abdellatif, A. Hakiri, P. Gelard, F. Planchou, M. Bruyere, Software defined networking and virtualization for broadband satellite networks, IEEE Communications Magazine 53 (3) (2015) 54–60.
[27] N. A. Khater, R. E. Overill, Network traffic classification techniques and challenges, in: 2015 Tenth International Conference on Digital Information Management (ICDIM), 43–48, 2015.
[28] P. Foremski, On different ways to classify Internet traffic: a short review of selected publications, Theoretical and Applied Informatics 25 (2).
[29] P. M. Santiago del Rio, D. Rossi, F. Gringoli, L. Nava, L. Salgarelli, J. Aracil, Wire-speed Statistical Classification of Network Traffic on Commodity Hardware, in: Proceedings of the 2012 Internet Measurement Conference, IMC '12, ISBN 978-1-4503-1705-4, 65–72, 2012.
[30] J. Zhang, Y. Xiang, Y. Wang, W. Zhou, Y. Xiang, Y. Guan, Network Traffic Classification Using Correlation Information, IEEE Transactions on Parallel and Distributed Systems 24 (1) (2013) 104–117.
[31] T. En Najjary, G. Urvoy Keller, A first look at traffic classification in enterprise networks, in: TRAC 2010, 1st ACM International Workshop on Traffic Analysis and Classification, June 28th-July 2nd, 2010, Caen, France, Caen, FRANCE, 2010.
[32] R. Hofstede, P. Celeda, B. Trammell, I. Drago, R. Sadre, A. Sperotto, A. Pras, Flow Monitoring Explained: From Packet Capture to Data Analysis With NetFlow and IPFIX, IEEE Communications Surveys Tutorials 16 (4) (2014) 2037–2064.
[33] S. J. M. C., P. Carvalho, L. Solange Rito, Inside packet sampling techniques: exploring modularity to enhance network measurements, International Journal of Communication Systems 30 (6) (2017) e3135–n/a, ISSN 1099-1131.
[34] V. Moreno, J. Ramos, P. M. S. del Ro, J. L. Garca-Dorado, F. J. Gomez-Arribas, J. Aracil, Commodity Packet Capture Engines: Tutorial, Cookbook and Applicability, IEEE Communications Surveys Tutorials 17 (3) (2015) 1364–1390.
[35] P. Velan, V. Pus, High-density network flow monitoring, in: 2015 IFIP/IEEE International Symposium on Integrated Network Management (IM), 996–1001, 2015.
[36] T. Wellem, Y. K. Lai, C. Y. Huang, W. Y. Chung, A hardware-accelerated infrastructure for flexible sketch-based network traffic monitoring, in: 2016 IEEE 17th International Conference on High Performance Switching and Routing (HPSR), 162–167, 2016.
[37] I. Ghafir, V. Prenosil, J. Svoboda, M. Hammoudeh, A Survey on Network Security Monitoring Systems, in: 2016 IEEE 4th International

Conference on Future Internet of Things and Cloud Workshops (FiCloudW), 77–82, 2016.
[38] J. F. Kurose, K. W. Ross, Computer Networking: A Top-Down Approach (6th Edition), Pearson, 6th edn., ISBN 0132856204, 9780132856201, 2012.
[39] A. Bittau, D. Boneh, M. Hamburg, M. Handley, D. Mazieres, Q. Slack, Cryptographic protection of TCP Streams (tcpcrypt), https://datatracker.ietf.org/doc/rfc8548/, internet Engineering Task Force (IETF), ????
[40] A. Freier, P. Karlton, P. Kocher, The Secure Sockets Layer (SSL) Protocol Version 3.0, https://tools.ietf.org/html/rfc6101, internet Engineering Task Force (IETF), ????
[41] T. Dierks, E. Rescorla, The Transport Layer Security (TLS) Protocol, https://tools.ietf.org/html/rfc8446, internet Engineering Task Force (IETF), ????
[42] R. Stanton, Securing VPNs: comparing SSL and IPsec, Computer Fraud & Security 2005 (9) (2005) 17 – 19.
[43] E. Bocchi, L. Grimaudo, M. Mellia, E. Baralis, S. Saha, S. Miskovic, G. Modelo-Howard, S.-J. Lee, MAGMA network behavior classifier for malware traffic, Computer Networks 109, Part 2 (2016) 142 – 156.
[44] D. Muelas, M. Gordo, J. L. Garca-Dorado, J. E. L. de Vergara, Dictyogram: A statistical approach for the definition and visualization of network flow categories, in: 2015 11th International Conference on Network and Service Management (CNSM), 219–227, 2015.
[45] A. Tongaonkar, R. Torres, M. Iliofotou, Ram, Towards self adaptive network traffic classification, Computer Communications 56 (2015) 35 – 46.
[46] G. Levchuk, Function and activity classification in network traffic data: existing methods, their weaknesses, and a path forward, vol. 9850, 9850 – 9850 – 13, 2016.
[47] M. Iliofotou, H. chul Kim, M. Faloutsos, M. Mitzenmacher, P. Pappu, G. Varghese, Graption: A graph-based P2P traffic classification framework for the internet backbone, Computer Networks 55 (8) (2011) 1909 – 1920.
[48] J. Jusko, M. Rehak, Identifying peer-to-peer communities in the network by connection graph analysis, International Journal of Network Management 24 (4) (2014) 235–252.
[49] L. Akoglu, H. Tong, D. Koutra, Graph based anomaly detection and description: a survey, Data Mining and Knowledge Discovery 29 (3) (2015) 626–688.
[50] A. Moore, M. Crogan, A. W. Moore, Q. Mary, D. Zuev, D. Zuev, M. L. Crogan, Discriminators for use in flow-based classification, Tech. Rep., University of London, 2005.
[51] J. Read, B. Pfahringer, G. Holmes, E. Frank, Classifier chains for multi-label classification, Machine Learning 85 (3) (2011) 333.
[52] L. Breiman, Random forests, Machine Learning 45 (1) (2001) 5–32.
[53] G. Tsoumakas, I. Katakis, Multi-label classification: An overview, Int J Data Warehousing and Mining 2007 (2007) 1–13.
[54] R. Agrawal, A. Gupta, Y. Prabhu, M. Varma, Multi-label Learning with Millions of Labels: Recommending Advertiser Bid Phrases for Web Pages, in: Proceedings of the 22Nd International Conference on World Wide Web, WWW '13, ISBN 978-1-4503-2035-1, 13–24, 2013.
[55] F. Pacheco, E. Exposito, M. Gineste, A wearable Machine Learning solution for Internet traffic classification in Satellite Communications, in: The 17th International Conference on Service-Oriented Computing (ICSOC), –, 2019.
[56] B. Claise, B. Trammell, P. Aitken, Specification of the IP Flow Information Export (IPFIX) Protocol for the Exchange of Flow Information, Tech. Rep., Internet Engineering Task Force (IETF), 2013.
[57] W. Wang, M. Zhu, J. Wang, X. Zeng, Z. Yang, End-to-end encrypted traffic classification with one-dimensional convolution neural networks, in: 2017 IEEE International Conference on Intelligence and Security Informatics (ISI), 43–48, 2017.
[58] Wei Wang, Ming Zhu, Xuewen Zeng, Xiaozhou Ye, Yiqiang Sheng, Malware traffic classification using convolutional neural network for representation learning, in: 2017 International Conference on Information Networking (ICOIN), 712–717, 2017.
[59] L. Grimaudo, M. Mellia, E. Baralis, Hierarchical learning for fine grained internet traffic classification, in: 2012 8th International Wireless Communications and Mobile Computing Conference (IWCMC), 463–468, 2012.
[60] A. Montieri, D. Ciuonzo, G. Bovenzi, V. Persico, A. Pescap, A Dive into the Dark Web: Hierarchical Traffic Classification of Anonymity Tools, IEEE Transactions on Network Science and Engineering (2019) 1–1.
[61] Y. Shi, D. Feng, S. Biswas, A Natural Language-Inspired Multi-label Video Streaming Traffic Classification Method Based on Deep Neural Networks abs/1906.02679, URL https://arxiv.org/abs/1906.02679.
[62] G. Draper-Gil, A. H. Lashkari, M. S. I. Mamun, A. A. Ghorbani, Characterization of Encrypted and VPN Traffic using Time-related Features, in: Proceedings of the 2nd International Conference on Information Systems Security and Privacy - Volume 1: ICISSP,, 407–414, 2016.
[63] L. Peng, B. Yang, Y. Chen, Z. Chen, Effectiveness of Statistical Features for Early Stage Internet Traffic Identification, International Journal of Parallel Programming 44 (1) (2016) 181–197.