A Wearable Machine Learning Solution for Internet Traffic Classification in Satellite Communications

Fannia Pacheco[1](✉), Ernesto Exposito[1], and Mathieu Gineste[2]

[1] Univ Pau & Pays Adour, E2S UPPA, LIUPPA, EA3000,
64600 Anglet, France
{f.pacheco,ernesto.exposito-garcia}@univ-pau.fr

[2] Business Line Telecommunication, R&D départment, Thales Alenia Space,
31100 Toulouse, France
mathieu.gineste@thalesaleniaspace.com

Abstract. In this paper, we present an architectural framework to perform Internet traffic classification in Satellite Communications for QoS management. Such framework is based on Machine Learning techniques. We propose the elements that the framework should include, as well as an implementation proposal. We define and validate some of its elements by evaluating an Internet dataset generated on an emulated Satellite Architecture. We also outline some discussions and future works that should be addressed in order to have an accurate Internet classification system.

Keywords: Internet traffic classification · Machine Learning · Satellite Communications · Deep packet inspection

1 Introduction

Internet traffic classification is a group of strategies 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 is important in Satellite communication principally to manage bandwidth resources and to ensure Quality of Service (QoS) requirements.

Traffic classification is widely implemented by Deep Parquet Inspection(DPI) solutions. Most of the commercial solutions use this technology for traffic management. DPI performs a matching 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. In order to overcome the former issues, Machine Learning (ML) emerged as a suitable solution, not only for the traffic classification task, but also for prediction and new knowledge discovery, among other things. In this context, 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.

S. Yangui et al. (Eds.): ICSOC 2019, LNCS 11895, pp. 202–215, 2019.
https://doi.org/10.1007/978-3-030-33702-5_15

In satellite networks, Internet traffic management is a key task due to it allows improving the QoS. Commonly, 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 [6,12,14]. The principle is to deploy passive monitoring points in order to perform traffic classification. These monitoring points can be at routers [6] or points of presence (PoPs) [12] 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 [1,8].

The authors outlined the complete process to achieve Internet traffic classification in the survey paper [10]. Therefore, this approach focuses its attention on developing a framework that can be deployed in a Satellite architecture. Such a framework comprises all the necessary elements to achieve the goal, as well as, additional components that should be integrated to assure a robust classification tool. We propose a hierarchical classification system based on ML, which treats encryption and flow patterns differently. We deploy the solution in a low level language that allows having an efficient and fast classification output. We also compare our approach with a well-known DPI solution called nDPI [2]. Finally, we set discussions about some important components that are in development; for instance, the treatment of tunneled connections and the evolution of the Internet network.

2 QoS Management in Satellite Communications

At this point, we start by introducing the general reference model to provide Satellite Communications. This model will serve us as guidance to find the requirements to integrate ML in such architecture. A common reference model of a multi-gateway Satellite architecture is shown in Fig. 1 [3]. This model is divided into two main blocks: Satellite access network and Satellite core network. On one hand, in the Satellite access network, a variety of network typologies can be used to the connectivity of the elements; these included the Satellite gateways and terminals. On the other hand, in the Satellite core network, an aggregate network allows interconnecting with other operators, corporations and Internet Service Providers (ISPs) through Points of Presence (PoPs).

Two main components of such model are described below:

- Satellite Terminal (ST): its function is to deliver broadband access to end-user equipment through IP routers and/or Ethernet switches.
- Satellite Gateway (GW): this component is in charge of deploying user plane functions such as packet routing and forwarding, interconnection to the data network, policy enforcement and data buffering. These functionalities are coordinated by the control and management systems of the Satellite network. The GW is composed of forwarding and returning link (FL and RL) subsystems, and a set of network functions. These network functions include the Performance Enhancing Proxy (PEP), switching and routing interfaces for the interconnection with the Satellite core network.

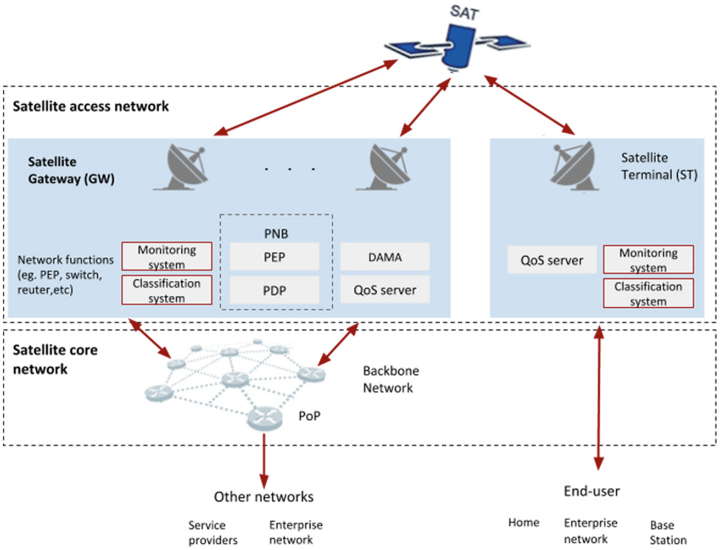


Fig. 1. Reference model of a multi-gateway Satellite network architecture.

One of the main objectives of this architecture is to provide a reliable communication system between different entities. However, improving the Quality of Service (QoS) and Quality of Experience (QoE) of their users is of paramount importance for network administrators. In principle, these last objectives can be achieved by manipulating the network functions. More specifically, a Policy Based Network (PBN) Architecture is deployed at this stage to perform traffic management [7]. In order 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 [5,13]. This idea parts from the assumption that some Internet traffic is more sensitive to information loss and delay such as Internet calling or video conference. In contrast, Internet browsing or file downloads are less pruned to be affected by these error conditions.

Following this idea, the main goal of our proposal is to correctly profile the Internet communications, to later transmit this information to a PBN that will take the necessary actions for QoS management. Hence, in Fig. 2, we add two new elements to allow Internet traffic classification: Monitoring and Classification system. The resulting classification is forwarded to the PBN. In the figure above, we also show two basic components comprised by the PBN: (i) A Policy Decision Point (PDP) that takes decisions for itself and for other network elements. These decisions imply actions for enforcement when the conditions of a policy rule are met [15], and (ii) Policy Enforcement Point (PEP) which is a logical entity that enforces policy decisions [15]. Marked Internet traffic can be forwarded to the PDP, which in turn will identify the associated GWs or STs and determine if more bandwidth should be assigned. This last decision is sent to the PEP for its execution. In addition to this, a QoS server can be deployed to enforcing QoS for different flows directly, and not to the GWs and STs as the PEP does.

3 Architecture Design

Making an abstraction of the elements in a real Satellite network distribution, the primary steps to achieve Internet traffic classification in a Satellite Architecture are:

1. Intercept Internet traffic in the GW and ST through passive monitoring points.
2. Compute statistical features that define the Internet flows.
3. Send the extracted features to the Classification system and mark the flows with their QoS classes.
4. Forward the classification to the PDP that will take decisions in order to improve the QoS. Then the PEP and QoS server will execute those decisions.

In order to design the system, we use a software engineering tool called Capella[1]. This tool provides methodological guidance, intuitive model editing, and viewing capabilities for Systems, Software and Hardware Architects. In Capella, the Operational analysis and System analysis help finding and defining the requirements of the system. Whereas, the Logical and Physical architectures aim at developing the solution. Figure 2 shows a System Analysis viewpoint, focused on the GW actor, developing the requirement: Provide Internet traffic classification in Satellite Communications for QoS management. We will discuss as follows the functions associated to this system analysis.

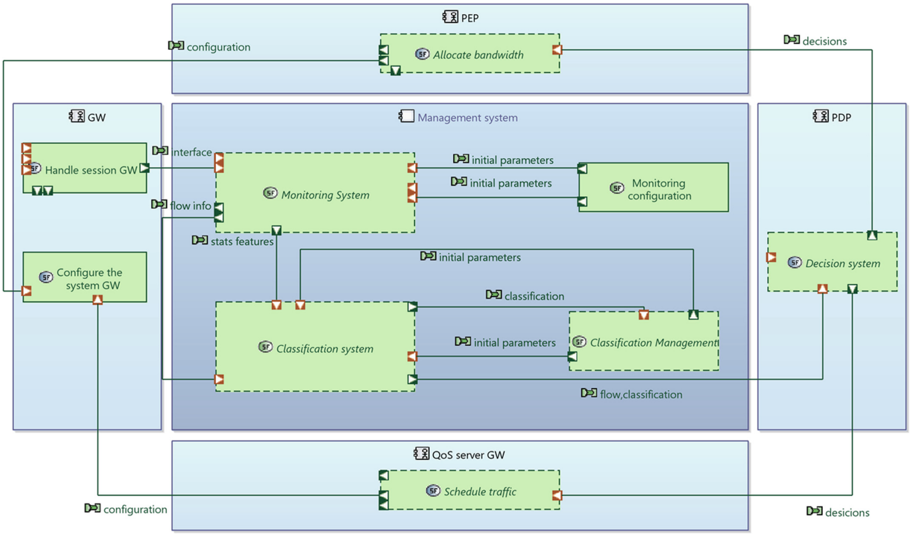


Fig. 2. System Analysis in the GW.

[1] https://www.polarsys.org/capella/.

3.1 Classification System

Particularly, this system proposes an automatic and logic process to analyze traffic in a hierarchical manner. The classification system is displayed in Fig. 3. Briefly speaking, the process starts performing the *Offline configuration* process in order to initialize the whole classification system (training process). In an online manner, the flow features pass through a *Flow discriminator 1 (D1)* that will be in charge of disjointing the non-encrypted/Encrypted flows from the tunneled flows. This separation will allow us to treat each technology differently. For instance, for the non-encrypted/Encrypted flows, classical ML models or DPI solutions (denoted as $Cl1$) can label the flows. Whereas, the tunneled flows will pass through another *Flow discriminator 2 (D2)* that separates the unitary (only one application within the tunnel) and the multiple (several applications at the same time in the tunnel). Finally, once the classifiers are actively working the *Online configuration* component is receiving information that can induce to change or to add models in the *Model repository*.

3.2 Monitoring System

Internet packets are captured to be organized into flows F. The construction of the flow is given in Fig. 4. In principle, all the flows are built matching the packet's headers, source (src) and destination (dst) IPs and ports. However, when $D1$ detected a multiplexed connection, the flow is broken into chunks of flows within a time interval, as seen in Fig. 4. Then, statistical based features are

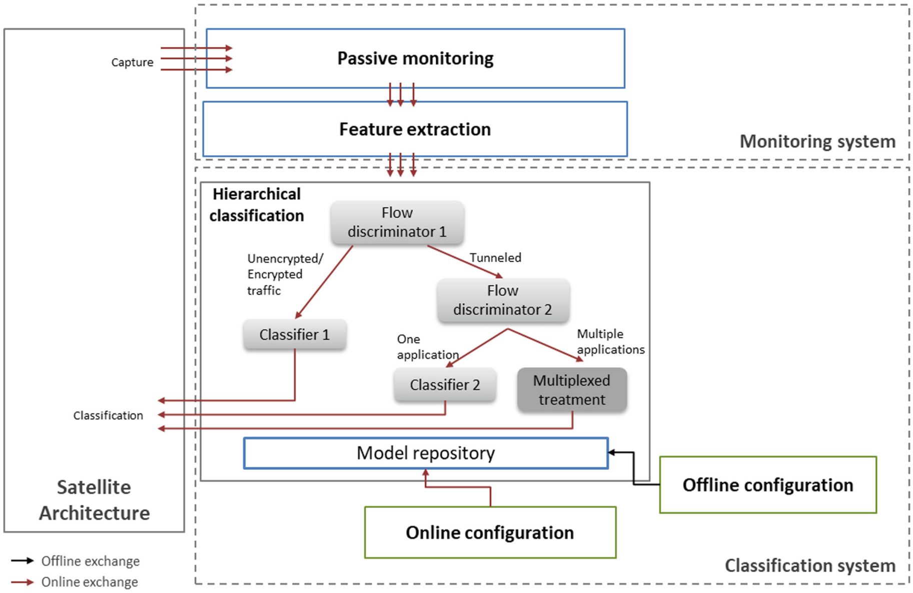


Fig. 3. Classification framework

computed for each flow in order to describe the communications. In brief, the properties computed are listed in Table 1. The passive monitoring and feature extraction processes were studied by the authors in [9,11]. The categorization of the packets (A, B, C, D, E and F) in Table 1 is obtained by studying the packet length distributions per class in the dataset.

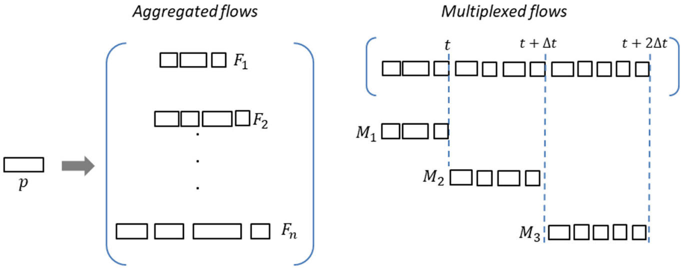


Fig. 4. Flow reconstruction.

Table 1. Result of the feature extraction process

Feature	Metric	Additional Information	Flow direction	Total
pktlen_[m]	[m] of the packet lengths	"m" refers to the metric Mean, Std, Min and Max	F,F_{src} and F_{dst}	12
iat_[m]	[m] of the inter-arrival time(iat)	-	F,F_{src} and F_{dst}	12
pktlen_[cat]_[m]	[m] of the packet lengths per [cat]	"cat" refers to the type of packet[a]	F,F_{src} and F_{dst}	72
iat_[cat]_[m]	[m] of the iat per [cat]		F,F_{src} and F_{dst}	72
bytes_[Δt]	bytes per [Δt]	"Δt" is the time windows	F,F_{src} and F_{dst}	3
pkt_[Δt]	packets counts per [Δt]	-	F,F_{src} and F_{dst}	3
Total				174

[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

3.3 Classification Management

This component implements the offline and online reconfiguration. Regarding the *Online reconfiguration* component, this element will be in charge of evaluating the predictions performed by the classifiers. This is deployed in order to cope with the evolution of the network. Therefore, in an online manner, this component will evaluate if the traffic observed belongs to an existing QoS class; if so the classifier will "evolve" to offer more accurate predictions. This approach can be translated to a retraining process when new data is generated; nonetheless, there are another approaches based on clustering that could detect class evolution.

As a final note, the current investigation does not treat the *Online configuration* and *Multiplexed treatment* due to they involve more complex tasks that will be presented in future works.

4 Implementation Design

The implementation proposal is presented in Fig. 5, with the operational and physical architecture in the same viewpoint. The subsystems proposed in Fig. 5 will define the way in which the components of the *QoS management system* work. For instance, the *Offline configuration* will be developed by the *Training process* and *Historical data manager* components, the *Online configuration* by the *Model manager* and the *Incremental Learning Model(ILM) manager* components. In addition to this, we define two new physical components that will be necessary for the implementation: A *GW server* that will be in charge of taking the Internet traffic for its further classification, and a *Management Server* that will handle offline and online configurations.

It is worth mentioning that the functions of the *GW server* and the *Management Server* can be comprised in the GW entity. This is modifiable according to the resources available in the real Satellite Architecture. On the other hand, all the functions concerning the *Classification system* are comprised in *Framework*: which in turn is a library developed for this aim. For what concerns the *sniffer*, we use existing solutions such as Libcap[2] for performing the sniffing. Then, we add the *Flow reconstruction* and *Feature Extraction* behaviors. The ML models $D1$, $Cl1$, $D2$ and $Cl2$ will be selected in the experimental section.

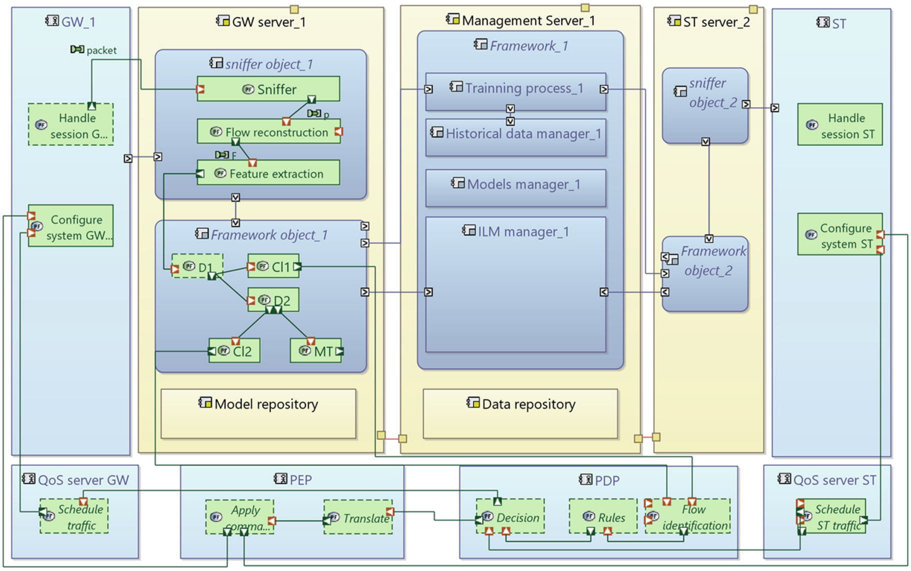


Fig. 5. System analysis in the GW.

[2] https://www.tcpdump.org/.

As additional comment, the reader can notice that the proposed implementation can be easily replicated in the ST component, as well as in different network components where packet monitoring is feasible.

5 Emulated Satellite Internet Traffic

This data set is a private dataset called SAT data. The model of a multi-gateway Satellite network in Fig. 6 with one ST and one GW was set over OpenSAND[3], which is a platform to emulate Satellite Communications. In addition to this, a VPN configuration is disposed between the ST and the GW, with the objective to emulate tunneled communications. 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.

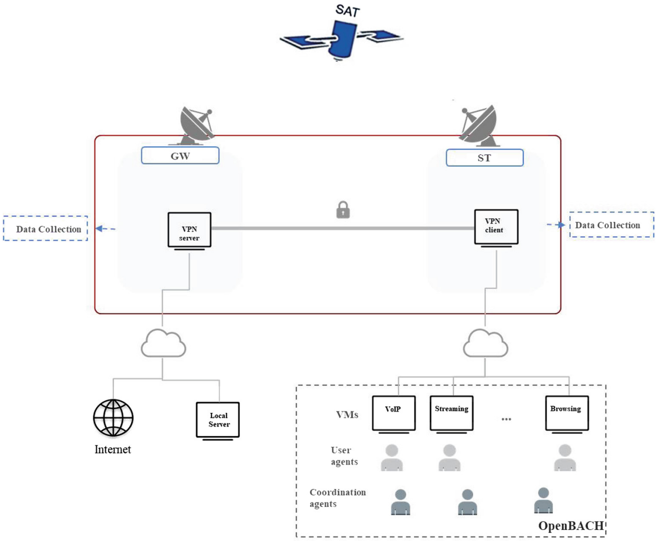


Fig. 6. Traffic emulation platform proposed in a Satellite Architecture.

The applications are launched 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 (iii) Multiple scenarios with the VPN: several applications are launched at the same time. Additionally, some network configurations were imposed on OpenSand. For each scenario, the data collection process was performed in the GW and ST, before and after the VPN. In

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

this sense, all the possible transformations that the data perceived is recorded. The labeling process is performed per file and application launched. However, for the VPN tunnel, a special treatment was performed: for each packet getting into the tunnel a flag was used to mark the application launched. Therefore, the multiplexed connections are correctly labeled. This dataset is still in development. In this particular work, we used only the data captured in the GW with the applications in Table 2. These 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 2, we show the flows captured per application and the amount of packets with and without the VPN. It is important to mention that the duration varies from 5 min up to 15 min. In addition to this, the experiments over the VPN were carried by using UDP as transport protocol.

Table 2. Class, packet and flow distribution of the SAT data in the GW.

		Without VPN		With VPN	
QoS class	Application	Flows	Packets	Packets: Unitary	Packets: Multiple
VoIP	facebook_voip	302	227997	74904	522275
	skype_voip	565	315281	60764	673780
	twinkle_voip	69	141663	26144	276995
Video	skype_video	579	925391	318335	2235781
	facebook_video	357	558880	162822	1000071
Streaming	youtube_video_streaming	760	158177	19619	486141
Browsing	web_browsing	6852	749979	91705	1824852
Unknown	unknown	58	2860	1080	2334

6 Experimental Evaluation

The training process was deployed by dividing the data as in Table 3. The complete data is used to build $D1$, while for the rest of classifiers the data is adapted according to their objectives. First at all, in order to build $Cl2$, we evaluate different time windows Δt to find the most adequate. Afterwards, we build the rest of the classifiers with different ML approaches. The best approaches are selected, and their average response time and accuracy are compared with nDPI.

Table 3. Data settings for building the classifiers.

Classifier	All data			
D1	Without VPN		With VPN	
Cl1	Unencrypted	Encrypted		
D2			unitary	multiple
Cl2			unitary	
MT				multiple

6.1 Classification System Results

Table 4 shows the results after evaluating different time windows for the unitary tunneled connections. The accuracy increase as Δt does; therefore, we compare the average number of packets evaluated for each application in Fig. 7. We can notice that for 5 ms and 10ms, the amount of packets is very low. To avoid this, the new window will be adjustable in the sense that $\Delta t = 10$ ms, but we wait until we have at least 20 packets to process.

Table 4. Accuracy results for *Cl*2 varying Δt

Δt	Num. flow	Cl2
5 ms	167097	0.8982
10 ms	120395	0.9647
100 ms	26634	0.9673

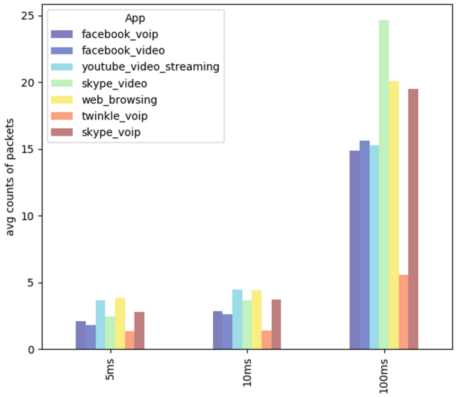


Fig. 7. Average counts of packets for each Δt

On the other hand, the results in Table 5 show a comparison between several classifiers: Decision Tree (DT), Random Forest (RF), K Nearest Neighbors (KNN), Ada Boost, Voting and Extra Trees (ETs). The best performance is standing up in bold. We picked DTs for the flow discrimination tasks, while RF for the traffic classification task.

Table 5. Accuracy scores of several ML classifiers.

	DT	RF	KNN	AdaBoost	Voting	ETs
D1	**0.9999**	0.9999	0.9999	0.9999	0.9999	0.9999
Cl1	0.8876	**0.9186**	0.8617	0.7986	0.8941	0.8938
D2	0.9588	**0.9646**	0.9526	0.9584	0.9636	0.9638
Cl2	0.9321	**0.9401**	0.9209	0.8333	0.9358	0.9304

Following, the complete framework was implemented in C. The tree based models are built in sklearn[6] and parsed to C for faster Internet classifications,

[6] https://scikit-learn.org.

inspired by the work in [4]. These tests were performed on a PC with an i7-6700HQ CPU and 32 Gb RAM. The response time and accuracy are measured over the test set. We also evaluate nDPI for traffic classification.

In Table 6, we can notice that the C implemented models maintain their accuracy. In the unencrypted case, ML outperforms nDPI; while, for the encrypted case nDPI is unable to detect the class of an unitary session as $Cl2$ does. Regarding the response time of the classifiers, in Table 7, we can notice that fast Internet classifications are possible. 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. In addition to this, the packet processing and flow metering response time varies from 5 ms to 15 ms.

Table 6. Accuracy (Acc) evaluating the test data

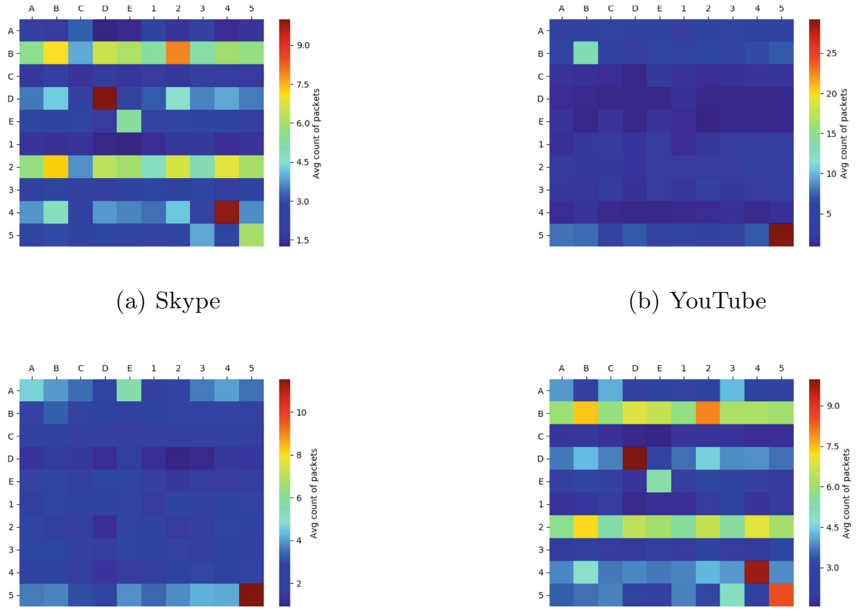
		Acc	
		ML	nDPI
Unencrypted	D1	0.9999	1
	Cl1	0.9186	0.5830
Encrypted	D2	0.9588	X
	Cl2	0.9401	X

Table 7. Average response time in μs

		Time(µs)	
		ML	nDPI
Unencrypted	D1	2.867	1
	Cl1	5	6.6460
Encrypted	D2	2.717	X
	Cl2	5	X

6.2 About the Multiplexed Connections

We were able to divide the multiplexed connections between unitary and non unitary scenarios. We saw that the unitary scenarios can be classified by classical ML approaches. The scenario with multiple applications within a tunnel is challenge in this field. In order to illustrate the problem, we take the unitary tunneled flows of Skype, YouTube and Browsing; and its equivalent mixed tunneled flow. We represent them as a combination of types of packets (A:E from the source and 1:5 from the destination, using the packet lengths described in Table 1). We count the average number of packets for each combination within a time windows of 100ms and plot it into a heatmap. For instance, the flow "AAB1CAA" has AA:2, AB:1, B1:1 and CA:1. This representation is in Fig. 8. We can notice that the unitary tunneled connections have distinctively sequence of patterns that are merged in the mixed tunneled flow. It is important to say that the Skype pattern is maintained and might be identified. This illustration gives us an idea of how to decrypt the behavior within the tunneled connections by looking at the packet's patterns. However, the complexity grows when more than three applications are multiplexed in the tunnel.



(a) Skype

(b) YouTube

(c) Browsing

(d) Skype,YouTube and Browsing mixed

Fig. 8. Heatmap representation of the flows with $\Delta t = 100\,\text{ms}$.

6.3 About the Evolution of Internet Traffic

Most of the publicly available datasets do not comprise all the existing applications on the Internet; in addition, the data collection process is tedious and expensive as remarked in [10]. One of the main deficiencies of ML in this field is handling with the evolution of the Internet traffic applications. If we consider some important QoS classes such as YouTube, NetFlix, Skype or Facebook video; as new incoming behavior, the classification accuracy might decrease considerably. Our architectural proposal comprises a component that should schedule retrainings of the models when the network administrators demand it. But also, an automatic approach can be set to continuously modify the trees of the RFs in the *Model repository* component. Such approach can be based on unsupervised methods for detecting the Internet evolution.

6.4 About the QoS Management

As we previously mentioned, it suffice to place the classification system over a network appliance that permits traffic monitoring. For instance, in the GW component, the classification output is forwarded to the PDP in order to perform the QoS management task. Depending on the classification output, QoS rules will be applied to trigger actions that will manage the Satellite resources. If a QoS rule is satisfied the traffic will be shaped as follows:

- Aggregate flows: the QoS rule is applied over all the incoming packets sharing the same tuple $(IP_{src}, IP_{dst}, port_{src}, port_{dst}, proto)$.
- Unitary tunneled flows: all the incoming packets of the unitary tunneled communications will be prioritized. However, this may be updated when the classification prediction of $D2$ or $Cl2$ changes in Δt.
- Multiplexed tunneled flows: we can think about prioritizing the tunnel as the unitary case. Nevertheless, in parallel, other less sensitive applications will be also benefited from this action. To avoid this, a classification per packet task should be designed.

In addition to this, we need to be sure that the QoS requirements are satisfied on time. For instance, according to [5], VoIP and Interactive video applications are very sensitive to delivery delays, to be specific they can tolerate around 100ms; whereas, another important class such as Video streaming around 10 s. We notice that the classification task can be achieved in 15 ms, giving sufficient time to treat those sensitive classes.

7 Conclusion

This work presented a ML system that can be integrated to Internet traffic architectures, being the Satellite Architecture our main interest. The proposal can be comparable with an existing DPI solution, which offers a portable software solution for Internet traffic inspection. We tested our approach in the GW component, with data captured from an emulated Satellite platform. This approach outperformed in accuracy and time a well-known DPI solution. We displayed the needs of having components that can deal with the evolution of the Internet network and the multiplexed connections, these last aspects are in development. Future works also include implementing the approach in the emulated Satellite platform, and tuning the framework proposed given different network conditions.

Acknowledgment. We 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. Bertaux, L., et al.: Software defined networking and virtualization for broadband satellite networks. IEEE Commun. Mag. **53**(3), 54–60 (2015)
2. Deri, L., Martinelli, M., Bujlow, T., Cardigliano, A.: nDPI: Open-source high-speed deep packet inspection. In: 2014 International Wireless Communications and Mobile Computing Conference (IWCMC), pp. 617–622 (2014)
3. Ferrús, R., Koumaras, H., et al.: Sdn/nfv-enabled satellite communications networks: opportunities, scenarios and challenges. Phys. Commun. **18**, 95–112 (2016). special Issue on Radio Access Network Architectures and Resource Management for 5G

4. Garcia, J., Korhonen, T., Andersson, R., Västlund, F.: Towards video flow classification at a million encrypted flows per second. In: 2018 IEEE 32nd International Conference on Advanced Information Networking and Applications (AINA), pp. 358–365, May 2018
5. ITU-T: End-user multimedia qos categories. Technical report, TELECOMMUNICATION STANDARDIZATION SECTOR OF ITU (2001)
6. Jin, Y., Duffield, N., Erman, J., Haffner, P., Sen, S., Zhang, Z.L.: A modular machine learning system for flow-level traffic classification in large networks. ACM Trans. Knowl. Discov. Data **6**(1), 4:1–4:34 (2012)
7. Moore, B., Ellesson, E., Strassner, J., Westerinen, A.: Policy core information model - version 1 specification, internet Engineering Task Force (IETF). https://tools.ietf.org/html/rfc3060
8. Ng, B., Hayes, M., Seah, W.K.G.: Developing a traffic classification platform for enterprise networks with SDN: Experiences & lessons learned. In: 2015 IFIP Networking Conference (IFIP Networking), pp. 1–9, May 2015
9. Pacheco, F., Exposito, E., Aguilar, J., Gineste, M., Baudoin, C.: A novel statistical based feature extraction approach for the inner-class feature estimation using linear regression. In: 2018 International Joint Conference on Neural Networks (IJCNN), pp. 1–8, July 2018
10. Pacheco, F., Exposito, E., Gineste, M., Baudoin, C., Aguilar, J.: Towards the deployment of machine learning solutions in network traffic classification: a systematic survey. IEEE Communications Surveys Tutorials, p. 1 (2018)
11. Pacheco, F., Exposito, E., Gineste, M., Budoin, C.: An autonomic traffic analysis proposal using machine learning techniques. In: Proceedings of the 9th International Conference on Management of Digital EcoSystems, MEDES 2017, pp. 273–280 (2017)
12. Pietrzyk, M., Costeux, J.L., Urvoy-Keller, G., En-Najjary, T.: Challenging statistical classification for operational usage: the adsl case. In: Proceedings of the 9th ACM SIGCOMM Conference on Internet Measurement, IMC 2009, pp. 122–135 (2009)
13. Siller, M., Woods, J.C.: QoS arbitration for improving the QoE in multimedia transmission. In: 2003 International Conference on Visual Information Engineering VIE 2003 (2003)
14. Trestian, I., Ranjan, S., Kuzmanovic, A., Nucci, A.: Googling the internet: profiling internet endpoints via the world wide web. IEEE/ACM Trans. Networking **18**(2), 666–679 (2010)
15. Yavatkar, R., Pendarakis, D., Guerin, R.: A framework for policy-based admission control, internet Engineering Task Force (IETF). https://tools.ietf.org/html/rfc2753