HAL
open science

Classification techniques for the management of the "Quality of Service" in Satellite Communication systems

Fannia Pacheco

To cite this version:

Fannia Pacheco. Classification techniques for the management of the "Quality of Service" in Satellite Communication systems. Cryptography and Security [cs.CR]. Université de Pau et des Pays de l'Adour, 2019. English. NNT : 2019PAUU3026 . tel-03066234

HAL Id: tel-03066234

https://theses.hal.science/tel-03066234v1

Submitted on 15 Dec 2020

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

École doctorale des sciences exactes et leurs applications

Classification techniques for the management of the "Quality of Service" in Satellite Communication systems

THÈSE

pour l'obtention du

Doctorat de l'Université de Pau et des Pays de l'Adour (France) (mention Informatique)

par

Fannia PACHECO

Sutenue le 22 Novembre 2019 devant le jury composé de:

Rapporteurs:

Isabelle GUERIN	Université Lyon 1
Urtzi AYESTA	Institut de Recherche en Informatique de Toulouse (IRIT)

Examinateurs:

María Teresa GOMEZ	University of Seville
Marie-Véronique LE LANN	INSA de Toulouse

Directeurs:

Ernesto EXPOSITO	Univ Pau & Pays Adour
Mathieu GINESTE	Thales Alenia Space

Laboratoire d'informatique de l'Université de Pau et des Pays de L'Adour (LIUPPA)

Acknowledgements

First, I would like to thank the company Thales Alenia Space for funding this Ph.D. program. I also wish to express my greatest thanks to my two supervisors Ernesto Exposito and Mathieu Gineste, for allowing me to do this Ph.D. and for guiding me during this work.

I am very grateful to Isabelle Guerin, Urtzi Ayesta, María Teresa Gomez, and Marie-Véronique Le Lann for accepting to be members of the jury. In particular, I would like to express my gratitude to Isabelle Guerin and Urtzi Ayesta for accepting to review this dissertation. In addition, I really appreciate the participation of Noëlle Bru and José Aguilar in mid-term evaluations of my work, for their interest and their useful remarks.

I would like to thank all my colleagues from the University of Pau, researchers, teachers, staff members and more specifically current and former PhD students.

I am very grateful with all my friends and the great people that I came across during the period of my thesis. In particular, I would like to thank my dear friends Matt Steinberg, Elena Garcia and Loreto Garcia for their unconditional support.

Also I want to thank my family who maintains my foot on the ground, and always reminds me that there are more important things in life than the academic research.

Last but not least, special thanks to my partner, Maxime, for always being there with his patience and love in good and bad times. Without him, this would not be possible.

Abstract

The Internet has become indispensable for the daily activities of human beings. Nowadays, this 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 thesis project aims at finding new Internet traffic classification approaches to improving customer satisfaction by improving the QoS.

Machine Learning (ML) algorithms will be studied and deployed to classify Internet traffic. All the necessary elements, to couple an ML solution over a well-known Satellite Communication and QoS management architecture, will be evaluated. In this architecture, one or more monitoring points will intercept Satellite Internet traffic, which in turn will be treated and marked with predefined classes by ML-based classification techniques. The marked traffic will be interpreted by a QoS management architecture that will take actions according to the class type.

To develop this ML-based solution, a rich and complete set of Internet traffic is required; however, historical labeled data is hardly publicly available. In this context, binary packets should be monitored and stored to generate historical data. To do so, an emulated cloud platform will serve as a data generation environment in which different Internet communications will be launched and captured. This study is escalated to a Satellite Communication architecture.

Moreover, statistical-based features are extracted from the packet flows. Some statistical-based computations will be adapted to achieve accurate Internet traffic classification for encrypted and unencrypted packets in the historical data. Afterward, a 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. Besides, to cope with the evolution of Internet applications, a new method is presented to induce updates over the original classification system. Finally, some experiments in the cloud emulated platform validate our proposal and set guidelines for its deployment over a Satellite architecture.

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

Résumé

De nos jours, Internet est devenu indispensable dans le quotidien des humains. Aujourd'hui, ce réseau sert entre autres de plate-forme de communication, de système transactions et de divertissement. Ces activités sont possibles grâce à des composants satellites qui gèrent les flux d'informations. Dans ce contexte, l'intérêt des fournisseurs de communications satellites est d'améliorer la satisfaction des clients à travers d'utilisation optimale de ces ressources. La qualité de service (QDS) est utilisée pour accomplir cet objectif. Améliorer la QDS permet la réduction des erreurs liées à la perte et la latence paquets; par conséquent "la qualité de service" (QDS) aide à optimiser le trafic internet. En fonction du trafic internet (Streaming, VoIP, Transfert de fichiers, etc.) et de ses erreurs, le flux de paquet peut être classé parmi plusieurs catégories. En suivant cette idée, ce projet de thèse vise à trouver des nouvelles approches de classification du trafic Internet pour améliorer la QDS.

Pour classifier le trafic Internet, l'apprentissage automatique sera étudié et déployé. Les composants qui permettront de coupler une solution d'apprentissage automatique avec une architecture satellite et de qualité de service seront évalués. Dans cette architecture, un ou plusieurs points de surveillance capteront le trafic Internet. Des techniques de classifications marqueront le trafic capté en classes qui seront interprétées par l'architecture de la qualité de service.

Pour développer notre solution, une base de données riche et complète sera requise; toutefois, les données historiques labellisées sont difficilement disponibles pour le public. Dans ce contexte, des paquets binaires seront extraits et stockées pour générer un historique de données. Par conséquent, une plate-forme d'émulation du trafic Internet sur le cloud pour générer des flux de communication a été proposée. Cela sera aussi implanté sur une plateforme d'émulation de communication Satellite.

En outre, des flux IP devront être construits avec les paquets et quelques caractéristiques statistiques pour discriminer et décrire le trafic Internet correctement seront présentées. Ensuite, un système de classification sera capable de gérer différentes communications sur Internet (cryptées, non cryptées et en tunnel). Ce système traitera le trafic entrant de manière hiérarchique pour atteindre une performance de classification élevée. Par ailleurs, pour faire face à l'évolution des applications Internet, une nouvelle méthode est présentée pour induire des mises à jour au système de classification initiale. Finalement, des expériences sur la plate-forme émulée dans le cloud seront mises en place pour valider notre proposition et définir des directives pour son déploiement sur l'architecture Satellite.

Mots-clés: trafic Internet, apprentissage automatique, communications satellites, qualité de service, trafic crypté.

Publications and projects

Journal

- F Pacheco, E Exposito, M Gineste, C Baudoin and J Aguilar. Towards the deployment of Machine Learning solutions in network traffic classification: A systematic survey. IEEE Communications Surveys & Tutorials (2018).

Conferences

- F. Pacheco, E. Exposito and M. Gineste, "A wearable Machine Learning solution for Internet traffic classification in Satellite Communications," Accepted for publication in The 17th International Conference on Service-Oriented Computing ICSOC 2019, held in Toulouse France, October 28-31, 2019.
- F. Pacheco, E. Exposito, J. Aguilar, M. Gineste and C. Baudoin,"A novel statistical based feature extraction approach for the inner-class feature estimation using linear regression," 2018 International Joint Conference on Neural Networks (IJCNN), Rio de Janeiro, 2018, pp. 1-8.
- F. Pacheco, E. Exposito, M. Gineste, and C. Budoin. 2017. "An autonomic traffic analysis proposal using Machine Learning techniques". In Proceedings of the 9th International Conference on Management of Digital EcoSystems (MEDES '17). ACM, New York, NY, USA, p.p. 273-280.

Projects with Thales

- Projet: «Application du machine learning au Satcom». R&D Département : Business Line Telecommunication. Thales Alenias Space, from Sep 2018.
- Projet: « Techniques d'optimisation des communications sécurisées ». R&D Département : Business Line Telecommunication. Thales Alenias Space, from Dec 2016 to Feb 2018.
- Projet: « Architecture de QoS basée sur les DPI », R&D Département : Business Line Telecommunication. Thales Alenias Space, from Dec 2016 to Dec 2017.

Contents

Acknowledgements **iii**

Abstract **v**

Résumé **vii**

1 Introduction **1**
1.1 Introduction . . . 1
1.2 Context . . . 2
1.3 Challenges . . . 4
1.4 Related works . . . 5
1.5 Positioning . . . 7
1.6 Contributions . . . 8
1.7 Structure of the Thesis . . . 9

2 Background and Related works **11**
2.1 Introduction . . . 11
2.2 QoS management in Satellite Communications . . . 12
2.2.1 General Satellite architecture . . . 12
2.2.2 Policy based network QoS management . . . 13
2.3 Traditional Internet traffic classification . . . 15
2.3.1 Internet traffic monitoring . . . 15
2.3.2 Encrypted vs Non-encrypted traffic . . . 16
2.3.3 Traffic classification approaches . . . 18
2.4 Machine Learning introduction . . . 20
2.4.1 Data collection . . . 21
2.4.2 Feature Engineering . . . 22
2.4.3 Algorithm selection and model deployment . . . 22
2.4.4 Validation of classification models . . . 23
2.5 Internet traffic classification with ML . . . 24
2.5.1 Data collection . . . 24
2.5.2 Feature engineering . . . 25
2.5.3 Classification with Machine Learning . . . 26
2.5.4 Model deployment . . . 26
2.6 State of the Art Analysis . . . 27
2.7 Summary . . . 29

3 QoS management in Satellite Communications with ML **31**
3.1 Introduction . . . 31
3.2 Capella . . . 32
3.2.1 Operational analysis . . . 33
3.2.2 System Analysis . . . 33
3.3 Reference high-level architecture: QoS management in Satellite Communications with ML . . . 34
3.3.1 Passive monitoring . . . 36
3.3.2 Feature Extraction . . . 36
3.3.3 Classification system . . . 37
Offline configuration component . . . 38
Hierarchical classification component . . . 38
Online reconfiguration component . . . 39
3.3.4 Policy based network architecture . . . 39
3.4 Requirement analysis with Capella . . . 40
3.4.1 Operational analysis . . . 41
Operational Entities/Actors and Capabilities . . . 41
Scenario 1: Communication GW/ST . . . 41
Scenario 2: QoS management in the GW/ST . . . 42
Operational context . . . 43
3.4.2 System analysis . . . 44
Missions and Capabilities . . . 44
System architecture composite . . . 45
Monitoring and Classification: system functions . . . 46
QoS management: system function . . . 47
Final System architecture . . . 48
3.5 Summary . . . 49

4 Characterizing and collecting Internet traffic **51**
4.1 Introduction . . . 51
4.2 QoS categories in Internet traffic . . . 53
4.3 Virtual Private Network . . . 54
4.4 Public available datasets . . . 55
4.5 Internet traffic from an emulated network on a cloud platform . . . 57
4.5.1 User agent . . . 58
4.5.2 Coordination agent . . . 59
4.5.3 Traffic collection . . . 60
4.6 Satellite dataset . . . 61
4.6.1 User agent . . . 62
4.6.2 Scenarios . . . 62
4.6.3 Data collection . . . 62
4.6.4 Results . . . 63
4.7 Preliminary analysis of the datasets . . . 64
4.8 Summary . . . 65

5 Feature extraction from communication streams **67**
5.1 Introduction . . . 67
5.2 Statistical features for data streams . . . 68
5.3 Feature extraction for aggregate flows . . . 69
5.3.1 Statistical based features . . . 70
5.3.2 Statistical feature generation with Linear Regression . . . 71
Create the feature models . . . 74
Feature extraction . . . 76
Result analysis . . . 77
5.4 Feature extraction for tunneled flows . . . 79
5.4.1 Slicing tunneled connections . . . 79
5.4.2 Categorizing tunneled packets . . . 81
5.5 Result . . . 82
5.6 Summary . . . 83

6 A Heterogeneous Classification System for Internet traffic **85**
6.1 Introduction . . . 85
6.2 Classification System overview . . . 87
6.3 Flow discriminator . . . 88
6.3.1 Experimental results . . . 89
6.4 Multi-label classifier . . . 91
6.4.1 Proposal . . . 92
6.4.2 Multi-label classifier: Experimental results . . . 94
Feature extraction window . . . 94
Multi-label classifiers comparison . . . 95
Analysis . . . 95
6.4.3 Packet classifier: Experimental results . . . 98
6.5 Classifiers . . . 100
6.5.1 Base classifiers . . . 101
6.5.2 Meta-classifier . . . 102
6.5.3 Reconfiguration process . . . 102
6.5.4 Experimental results . . . 104
6.6 Incremental learning model . . . 104
6.6.1 One-class classification . . . 106
KNN and one-class classification . . . 107
6.6.2 Proposal . . . 107
6.6.3 Model build . . . 108
6.6.4 Test . . . 109
6.6.5 Online evaluation . . . 111
6.6.6 Experimental results . . . 112
PAM results . . . 113
VPN-nonVPN results . . . 113
SAT results . . . 115
6.7 Classification System perspective . . . 118

6.8 Summary 119

7 Towards the implementation of the Classification System 121

7.1 Introduction 121
7.2 ARCADIA and Capella for the solution implementation 122
7.2.1 Logical architecture 122
7.2.2 Physical architecture 123
7.3 Implementation proposal 123
7.3.1 Logical architecture 123
Offline configuration: logical function 124
Online configuration: logical function 125
7.3.2 Physical architecture 127
Implementation details 128
7.4 Experimental results 129
7.4.1 Experimental setup 130
7.4.2 nDPI vs hierarchical classification results 131
7.4.3 Load/Stress performance 132
7.4.4 Online Display 134
7.5 Notes about the QoS management 135
7.6 Summary 136

8 Conclusions and perspectives 137

8.1 Introduction 137
8.2 Conclusions 137
Problem description: ML solution in a Satellite Architecture . 137
Data collection: Toward automatic Internet traffic generation and collection 138
Feature extraction: How to describe Internet communication streams 138
Classification System design and construction 139
Implementation approach 139
8.3 Perspectives 140

A Packet categorization 143

B Classification system settings 145

B.1 Offline training 145
B.2 ILM testbed 145

C Ensemble and ILM analysis 147

C.1 Random forests as base classifiers 147
C.2 Decision trees as base classifiers 148
Implementation results 150
C.3 Further analysis 151

D Tree based model implementation in C 153

References **155**

List of Figures

1.1 Context of the thesis. 4

2.1 Reference model of a multi-gateway satellite network architecture. . . 13
2.2 Policy Based Network Architecture. 14
2.3 Illustrative example of the difference between non-encrypted, encrypted and tunneled data 18
2.4 General view of the traffic classification approaches. 19
2.5 General steps to perform ML. 21
2.6 Trends of the selected papers for classifying traffic with ML. 29

3.1 Phases for modeling with Capella. 33
3.2 Example of a Satellite Network Architecture with traffic classification. 35
3.3 Flow reconstruction. 36
3.4 Classification framework 38
3.5 Operational Entities/Actors and Capabilities. 42
3.6 Operational activity interaction of a client-server communication in the GW/ST. 42
3.7 Operational activity iteration of the QoS management in the GW/ST. 43
3.8 Operational context. 44
3.9 Missions and capabilities of the system analysis. 44
3.10 System architecture. 45
3.11 Monitoring and Classification system function. 46
3.12 QoS management system function. 47
3.13 System architecture. 48

4.1 Main data collection components for traffic classification 52
4.2 QoS classes for Internet traffic classification. 53
4.3 Traffic emulation platform proposed. 58
4.4 Application and user emulation with Selenium. 59
4.5 The protocol to perform the data collection in the traffic emulation platform proposed. 60
4.6 Traffic emulation platform proposed in a Satellite Architecture [43]. . . 61
4.7 Illustration of the experiment launched in the emulated SAT plaform. 63
4.8 Class and application distribution of the PAM and VPN-nonVPN dataset. 64
4.9 Class and application distribution of our SAT dataset. 64

5.1 Feature extraction overview 71
5.2 Graphical description of the proposal. 73

5.3 (a) Statistical feature values computed for a packet sequence in the client with the Streaming class. (b) Statistical feature values computed for a packet sequence in the server with the Streaming class. 78
5.4 (a) Statistical feature values computed for a packet sequence in the client with the VoIP class. (b) Statistical feature values computed for a packet sequence in the client with the VoIP class. 78
5.5 Packet and flow tranformation in a VPN communication 79
5.6 Flow construction for a VPN communication 80
5.7 Packet distribution of the tunneled connections using udp and udp_sec. 82

6.1 Hierarchical classification system. 88
6.2 Flow discriminators of the hierarchical system. 89
6.3 Multi-label classifier of the hierarchical system. 91
6.4 Classification process for multiplexed flows. 93
6.5 Accuracy and LRPS of a RF with $\Delta t_1 = \{5ms, 10ms, 50ms\}$, and Δt_2 varied from 0.01s to 300s. 94
6.6 Confusion matrix for Multiplexed tcp sessions with $\Delta t_1 = 10ms$ and $\Delta t_2 = 60s$ 95
6.7 Cumulative mean value of the packet lenght of three applications Video-Streaming-Browsing, before and after the tunnel for the destination direction flow. 96
6.8 Cumulative mean value of the packet lenght for three applications Streaming-Streaming-Streaming, before and after the tunnel for the destination direction flow. 97
6.9 Cumulative mean value of the packet lenght for 8 applications, before and after the tunnel for the destination direction flow. 97
6.10 Cumulative mean value of the packet lenght for 8 applications, before and after the tunnel for the source direction flow. 97
6.11 Precision-Recall curves for the udp dataset. 99
6.12 Precision-Recall curves for the udp_sec dataset. 99
6.13 Precision-Recall curves for the tcp_sec dataset. 99
6.14 Classifiers of the hierarchical system. 100
6.15 Incremental learning model (ILM) of the hierarchical system. 105
6.16 Graphical description of the proposal. 108
6.17 Graphical description of the model build step. 109
6.18 Graphical description of the overlap detected between three one-class classifiers. 110
6.19 Graphical description of the micro clusters in overlap detected between three one-class classifiers. 110
6.20 Graphical ilustration after resampling the data. 110
6.21 Accuracy for two applications of the class Streaming and VoIP, with $B = 50$. 113
6.22 Accuracy for the application Wikipedia, $B = 50$ 114
6.23 Accuracy for the application eDonkey, $B = 50$ 114

6.24 Accuracy for the applications with $B = 50$ 115
6.25 Results for the applications Amazon, Yahoo and others, $B = 50$ . . . 115
6.26 Accuracy for the applications with $B = 50$ 116
6.27 Results for the application Skype, $B = 500$ 116
6.28 Results for the application Facebook, $B = 100$ 117
6.29 Graphical perspective of the Classification System 119

7.1 Transition from the system to the logical analysis. 124
7.2 Logical function of the Offline configuration system. 125
7.3 The Offline configuration system integrated in the Logical architecture. 125
7.4 Logical function of the Online configuration. 126
7.5 The Online configuration system integrated in the logical architecture. 127
7.6 Physical architecture of the ML solution in the GW. 128
7.7 Physical architecture of the ML solution in the GW and ST. 128
7.8 Implementation proposal on Proxmox. 130
7.9 Implementation proposal on the router. 131
7.10 Time analysis of the classification process. 132
7.11 Dashboard of the proposal. 134
7.12 QoS management for OpenVPN tunneled connections. 136

A.1 Decision tree udp and udp_sec. 143

C.1 Acc and ROC-AUC of the dynamic ensemble varying the number of base classifiers and the maximum depth of the DTs 149
C.2 F-score of the dynamic ensemble varying the number of base classifiers and the maximum depth of the DTs 149
C.3 F-score of the dynamic ensemble varying the number of base classifiers and the maximum depth of the DTs 149
C.4 Acc and ROC-AUC of the dynamic ensemble varying the number of base classifiers and the maximum depth of the DTs 150
C.5 F-score of the dynamic ensemble varying the number of base classifiers and the maximum depth of the DTs 150
C.6 F-score of the dynamic ensemble varying the number of base classifiers and the maximum depth of the DTs 151
C.7 Heat map varying the number of base classifiers and batch size 152

D.1 Schema to parse a tree based model from python to C. 153

List of Tables

1.1 Related works. 7

3.1 Macro algorithm for passive monitoring and feature extraction 37
3.2 Macro algorithm for the hierarchical classification component for a new flow 39
3.3 Macro algorithm for the *Online reconfiguration* component 40
3.4 Pseudo code for the *PBN* component 40

4.1 Description of the datasets 56
4.2 Flow and class distribution of the public datasets PAM and VPN-nonVPN. 57
4.3 Description of the applications launched for the SAT data. 62
4.4 Class, packet and flow distribution of the SAT data in the GW. 63

5.1 Features extracted from the bidirectional flows 72
5.2 Pseudo code of the proposal 74
5.3 Pseudo code to build the feature models for each class. 76
5.4 Pseudo code to extract the features from the raw data 77
5.5 Flows used for each classifier 82
5.6 Result of the feature extraction process for aggregated flows 83
5.7 Result of the feature extraction process for tunneled flows 83

6.1 Accuracy results after testing the $D1$ classifiers 90
6.2 Confusion matrix after evaluating $D1$ 90
6.3 Confusion matrix after evaluating $D2$ 90
6.4 Accuracy result after evaluating the multi-label classifier 95
6.5 Accuracy result after evaluating the multi-label classifier 98
6.6 Pseudo code for penalizing and adding a base classifier 103
6.7 Result after training the classifiers with all the flow sequence 104
6.8 Pseudo code of the model build 109
6.9 Pseudo code of the test 111
6.10 Pseudo code of the online evaluation 112
6.11 Accuracy result after evaluating the ILM and KNN models 112
6.12 Final accuracy after evaluating only one unknown application and a validation set. 117
6.13 Online modification with one unknown application. 117
6.14 Online modification with the validation set. 117

7.1 Data settings for building the classifiers. 131
7.2 Accuracy evaluating the test data 131
7.3 Average packets per second processed pkt/s 131
7.4 Load/stress performance 133
7.5 Memory consumed by our proposal and by nDPI 133

B.1 Setting defined to test the Incremental learning system 145

C.1 148
C.2 Accuracy of the evolutionary test 151

D.1 Pseudo code for building the model in python 154
D.2 Pseudo code for reconstructing the model in C 154

List of Abbreviations

IP	**I**nternet **P**rotocol
DPI	**D**eep **P**acket **I**nspection
VoIP	**V**oice **o**ver **IP**
VPN	**V**irtual **P**rivate **N**etwork
QoE	**Q**uality of **E**xperience
QoS	**Q**uality of **S**ervice
SAT	**Sat**ellite
ST	**S**atellite **T**erminal
GW	**G**ateway
ML	**M**achine **L**earning
FE	**F**eature **E**xtraction
RF	**R**andom **F**orest
DT	**D**ecision **T**ree
ILM	**I**ncremental **L**earning **M**odel
LR	**L**inear **R**egression

Chapter 1

Introduction

Contents

1.1 Introduction 1
1.2 Context 2
1.3 Challenges 4
1.4 Related works 5
1.5 Positioning 7
1.6 Contributions 8
1.7 Structure of the Thesis 9

1.1 Introduction

The Internet is an intricate network system that is organized and synchronized in such a way that all its elements 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 [71]. 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 [130].

This thesis project aims at finding new approaches to improving customer satisfaction by 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 [76]. 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 [35]. 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.

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 [84, 156]. This idea is based on the assumption 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 in order to find relationships, patterns, anomalies, and misconfigurations, among other things, in the Internet network. Particularly, 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 key 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) [81]. 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) [22]. 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 [135]. In this context, the statistical features of IP flows are commonly extracted from network traces, and they are stored 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 project's objective is focused on improving the QoS over Satellite Communications through Internet classification based on ML. Along with this thesis project, we will study all the elements needed to achieve this objective and propose an ML-based Internet traffic classification solution.

1.2 Context

The context of this thesis covers three interleaved aspects: 1) QoS management in Satellite Communications, 2) Internet Traffic classification, and 3) Machine Learning. QoS management in Satellite Communications can be defined by a set of policies or rules that permit to administer, manage, and control access to network resources. Internet traffic classification can be achieved, marking Internet flows with QoS classes (such as Streaming, VoIP, Chat, etc.) by using different well-known techniques. Finally, Machine Learning can be defined as a compound of strategies to perform classification, prediction, and clustering over vast amounts of datasets. The relationship

between these subjects is presented in Figure 1.1, where:

A **Satellite Architecture** holds all the necessary components to offer Satellite Communications. Among the elements of this architecture, it can be set another architectural structure that will manage and optimize the resources of the network system; we will denominated it at this stage as **QoS architecture**.

Monitoring points can be set along with the network appliances such as the Gateways (GWs), Satellite Terminals (STs), and end-points. The main objective is to capture inline Internet traffic.

Then, the traffic is evaluated by **Classification** techniques, that will label the Internet traffic by using different approaches. In our particular case, we will focus on **Machine Learning** techniques.

The result of the Internet traffic classification is used by the **QoS architecture** to manage/administer the resources in order to better perform and achieve their principal objective: offer an acceptable QoS to their clients.

In this context, different actors that interact in this architecture can be identified. In principle, end-points or client and servers actors will hold Internet communication, and the Satellite Architecture will serve as a supportive structure to perform this task. Traditionally, communication between two end-points is carried out by using communication protocols. These protocols are defined by a set of rules, standardized by the Internet Engineering Task Force (IETF) [82]. Internet applications use such protocols indistinctly allowing traffic profiling. In addition to this, the applications involved in a communication can also define particular behaviors for each type of traffic [185]. Given these characteristics, we can consider that Internet traffic modeling based on ML is a suitable path to follow. In order to build ML models for traffic classification, the common procedure is to have a representative number of interactions between client and server. Such interactions so-called historical data must reflect the real Internet communications passing through the Satellite architecture [135]. The most common Internet communications vary from applications of type Video Streaming (YouTube, Netflix, etc) to Voice over IP (Skype, Facebook calls, etc) and web browsing (Google, Baidu, etc), among others [33]. However, it is important to state that the success of the ML solution depends on how complete the historical data is. Moreover, the proliferation of new Internet applications demands a constant update of this historical data.

These facts impose essential challenges that must be considered before applying ML techniques for Internet traffic classification. Therefore, in the next section, we make an account of the most critical challenges faced by this investigation.

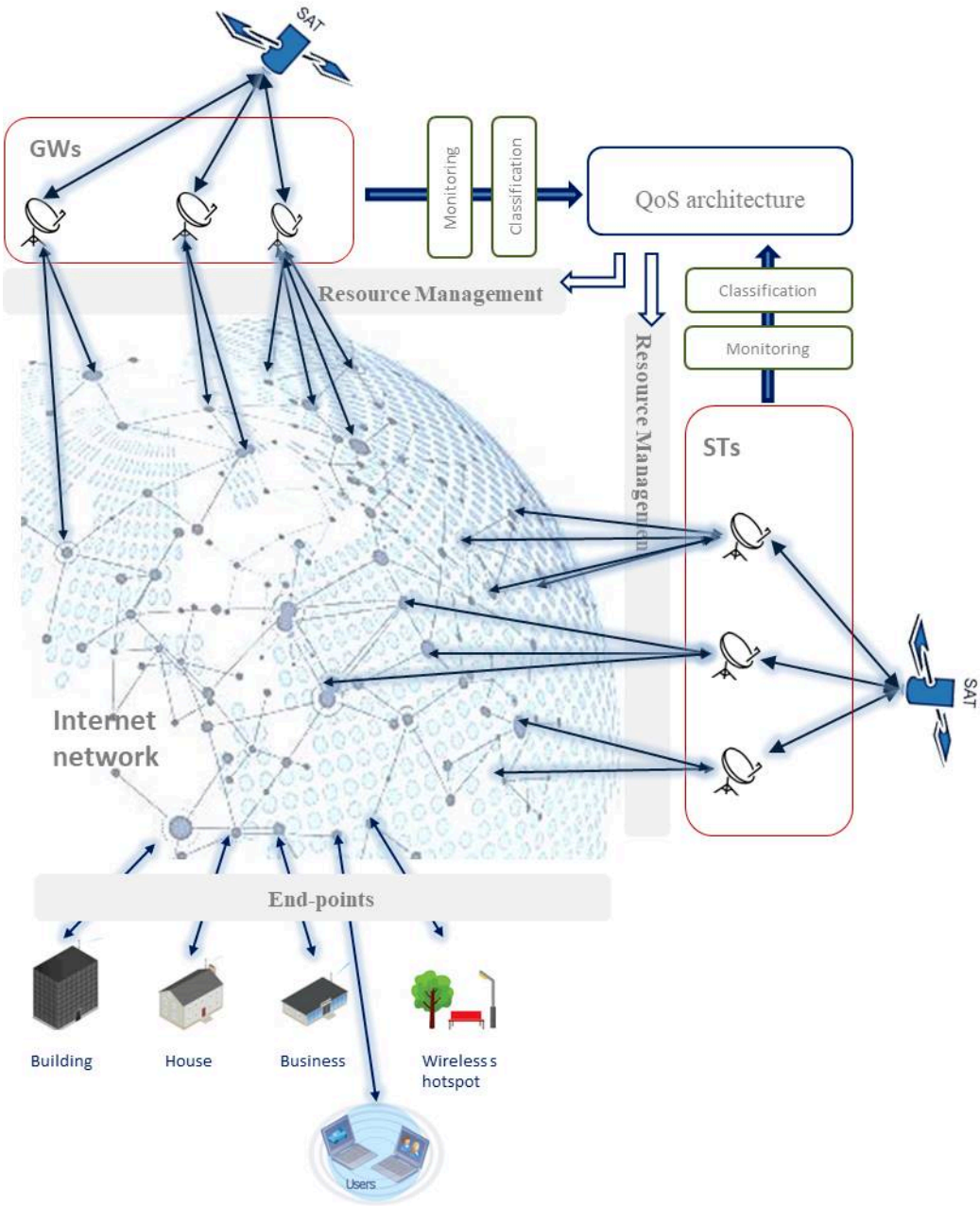


Figure 1.1: Context of the thesis.

1.3 Challenges

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. The work in [36] identifies 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 to require a

correct validation. Moreover, these future directions encourage the execution of more rigorous evaluations and comparisons of the ML approaches, to the development of tools for the ground truth definition, and the use of multiclassifier systems, among other things. 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 [93], and similarly in [60]. Notably, in our survey paper [135], we locate the challenges concerned by this investigation in each of the steps to achieve ML:

- **Data collection**: historical data is hardly publicly available. The production of a labeled historical data set is costly and time-demanding. One of the main concerns, in this field, is the lack of public data, which can be considered as a core resource for applying ML approaches. The difficulty of defining the ground truth values of the data collected also represents a challenge.

- **Feature extraction**: typically, statistical features are computed from streams of Internet data; however, different communication protocols disable the characterization of the stream. Encrypted traffic has become a new way to prevent intrusion into transmitted information in the Internet network. ML is highly suitable for analyzing these types of communications due to it does not intrude into the packet content; for instance, in some cases, the statistical behavior of the connection might be sufficient as exposed in [173]. However, new encryption techniques affect the observable features that can be extracted for traffic classification.

- **ML solution deployment**: in the field of Internet traffic classification, few implementations have been conceived due to the maintainability of this solution; which is mainly promoted by the evolution of Internet traffic demanding constant updates of the ML solutions. An update of the ML solution implies an update of the data sources.

- **Deployment architecture**: this item is related to where the ML can be deployed. Particularly, in Satellite Architectures, the resources allocated to the ML solution might be limited and a fast classification response is expected. In addition to this, the Internet network conditions might vary, making the classification results very sensitive.

1.4 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 [7, 22]. 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 [182, 137], or the statistical flow distribution. [102] 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 [111] presents an approach to collect and label mobile IP network traces correctly. For instance, the work in [126] 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.

Similarly, in WiFi networks, IP-data can be extracted to apply ML approaches for the traffic classification. For instance, the work in [146] collected network traces from Wi-Fi controllers at a large university campus. These controllers connected access points to the campus backbone network, allowing the 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 [148, 103, 187]. 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 [88, 166, 140, 68]. 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 [88, 166] or point of presence (PoP) [140] 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 [129, 16].

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 1.1, 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, fours 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 in the previous section, 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.

Network structure	Work	Data	Classification technique	Features	Advantages & Disadvantages			
					ground truth	encryption	evolution	implementation
ISP	[149]	CAIDA [24]	Statistical	Stats				X
	[189]	WIDE [32] and ISP	Statistical/ML	Stats				
	[88]	Private	ML	Stats	X			X
	[68]	Private	ML	Stats	X		X	X
Enterprise	[127]	Private	ML	Stats				
Satellite	[140]	Private	ML	Stats				X
Mobile	[111]	Private	ML	Stats	X			X
	[102]	WIDE [32], CAIDA, etc	ML	Stats				
Wifi/Mobile	[146]	Private	ML	bag-of-words/Stats	X			
SDN	[129]	Private	Statistical/Payload	Stats				

Table 1.1: Related works.

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.

1.5 Positioning

The main objective of this thesis project is to build a classification solution that can be coupled to a Satellite Architecture. In order to do so, we will navigate between three main fields.

- Satellite communication: Real Internet data from Satellite communications are not publicly available due to privacy matters. To tackle this problem, we will get historical data from an emulated Internet network platform. In this platform, we will study how Satellite communications are handled and what are the elements that allow performing traffic classification. This thesis project will not delve into the QoS management implementation. Nonetheless, we study in general what these components receive as inputs from an ML-based classification system. In the same manner, we give guidelines for deploying our proposal in a real or emulated Satellite architecture. It is important to mention that most of the tests will be performed on an emulated Internet network created for this aim on a cloud platform.

- Internet traffic: we need to consider encryption protocols and how they affect potential classification systems. Among the Internet traffic studied, we also include non-encrypted and tunneled connections. To have a knowledge base for building the ML models, we used well-known available datasets generated in small emulated Internet networks. In addition to this, data generated on an emulated Satellite platform will be also analyzed.

- Machine Learning: we will try to answer the principal challenges identified for this proposal. In this matter, Internet flows will be represented by data streams, where statistical-based features will describe such flows. The ML solution will deal with multiple types of traffic, such as non-encrypted and tunneled. Finally, we will propose an incremental learning system that copes with the evolution of the internet.

Once stated the areas touched by this thesis project, the following section lists the contributions achieved.

1.6 Contributions

The challenges found in this field guide the contributions of this thesis project. 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 an ML solution within a Satellite Architecture for QoS management. This proposal includes a heterogeneous Classification System that will treat some communication protocols such as tunneled communications differently. In this sense, we evaluate all the necessary elements that this Classification System must include.

ii Our second contribution is the design and implementation on an emulated Internet network on a cloud platform, that will serve for the production of data in a Satellite emulated platform. In this prototype, the nominal behavior of Internet users is mimicked to get human behaviors as realist as possible. In addition to this, Internet traffic will be correctly labeled.

iii The data produced 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 Classification System performance. Besides, a particular feature extraction process for encrypted connections will be designed.

iv Regarding the Internet classification system, 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 flow 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.
- Incremental learning model: We propose a dedicated algorithm to learn one class at a time. This method will try to find the evolution of one class to update the current classifiers.

v Finally, we study the reliability of implementing our ML solution on an emulated Internet network placed on a cloud platform. Its implementation and test will be later reproduced over the emulated Satellite platform.

It is important to mention that these main contributions cover the main challenges found in the state of art study. In the next section, we show the organization of this thesis project.

1.7 Structure of the Thesis

This thesis project is organized as follows,

Chapter 2. Background and Related works

This chapter presents the basis to understand how Internet traffic classification is performed in Satellite Communications. The presentation of this chapter follows the concepts presented in the context and challenges of this introduction. Firstly, a general Satellite architecture is studied along with the most common QoS management architecture. Secondly, traffic monitoring processes and tools are reviewed. Following, the difference between some encrypted technologies will be highlighted before entering in defining how different techniques perform the Internet traffic classification. An ML introduction is given to appreciate some of our contributions in this area better. Right after, it will be presented how ML has served as a key classification tool for Internet traffic classification. To conclude, a state of the art analysis is settled to remark the challenges and contributions in this field.

Chapter 3. QoS management in Satellite Communications with ML

This chapter presents how the ML solution should be coupled to a Satellite Architecture. Firstly, a theoretical proposition is presented. This proposition will define how the monitoring and classification processes will be set in a Satellite Architecture. Also, how all these elements need to communicate to perform QoS management is also presented. Then, to formalize such a proposal, a requirement analysis will be done by using a software engineering tool. This chapter concludes with an architecture identifying all the components to be developed in order to achieve the goals.

Chapter 4. Characterizing and collecting Internet traffic

Historical Internet traffic is needed to carry our investigation. This need forced us to search for publicly available historical data, as well as to create our data. In this chapter, it will be presented the Internet traffic categories to set their relative importance for QoS management. One encryption technology is commonly found in the historical data; therefore, we define some of its principles. Two public datasets, used to validate our proposal, are described. Following, we design a prototype to automatically generate Internet traces in a cloud-based

platform. The result of this platform is implemented in an emulated Satellite Communication platform where an important amount of Internet traffic is collected for its analysis. Finally, a study of the resulting data is shown.

Chapter 5. Feature extraction from communication streams

How Internet communications are processed in order to obtain something usable for the ML algorithms is studied in this chapter. It will be defined how the feature extraction is performed over common Internet streams as well for the encryption technologies concerned by this research work. In addition to this, a novel feature extraction approach is given in order to improve the classical approaches.

Chapter 6. A Heterogeneous Classification System for Internet traffic

The classification system, able to deal with different Internet communications, is detailed. This system will process in a hierarchical way the incoming traffic in order to achieve a high classification performance. In addition, in order to cope with the evolution of Internet applications, a new method is presented to induce updates over the original Classification System.

Chapter 7. Towards the implementation of the Classification System

A proposal to implement this solution in a Satellite Architecture is given. Some experiments in a cloud emulated platform are carried on in order to show an analog implementation. In this section, time response and performance are evaluated in order to validate the system proposed.

Chapter 8. Conclusion and Perspectives

This chapter concludes the thesis and outlines the perspectives of this work.

Chapter 2

Background and Related works

Contents

2.1 Introduction **11**
2.2 QoS management in Satellite Communications **12**
2.2.1 General Satellite architecture 12
2.2.2 Policy based network QoS management 13
2.3 Traditional Internet traffic classification **15**
2.3.1 Internet traffic monitoring 15
2.3.2 Encrypted vs Non-encrypted traffic 16
2.3.3 Traffic classification approaches 18
2.4 Machine Learning introduction **20**
2.4.1 Data collection 21
2.4.2 Feature Engineering 22
2.4.3 Algorithm selection and model deployment 22
2.4.4 Validation of classification models 23
2.5 Internet traffic classification with ML **24**
2.5.1 Data collection 24
2.5.2 Feature engineering 25
2.5.3 Classification with Machine Learning 26
2.5.4 Model deployment 26
2.6 State of the Art Analysis **27**
2.7 Summary **29**

2.1 Introduction

In this section, we present several topics that concern this investigation. These topics are presented in a vertical way, going from the most general to the most specific to define the elements that allow achieving QoS management in Satellite Communications. To recall the traffic classification process, in the first place, active Satellite Communications are held by components and actors interacting in this architecture. Internet traffic is then intercepted at one or more monitoring points; this process is normally called *Passive Monitoring*. The traffic is analyzed by different classification techniques which in turn will mark the traffic with predefined classes. The marked

traffic will be interpreted by a QoS management architecture that will take actions according to the class type.

In order to better understand the former process, we present in Section 2.2 a well-known Satellite Communication and QoS management architecture. We will place the main components that might interact with an ML-based solution. In Section 2.3, it is described how traditional Internet traffic classification is done by practitioners in the field. Within this section, we detailed the *Passive Monitoring* process, the structure of the Internet traffic data by establishing differences between encrypted and non-encrypted traffic, and finally the classical Internet traffic classification process. As this investigation is focused on the ML branch, an introduction of this field is given in Section 2.4. Following, a state of the art regarding Internet traffic classification with ML is exposed in Section 2.5. A complete analysis of the state of the art is presented in Section 2.6, where the challenges and possible research directions are discussed. Finally, Section 2.7 concludes this chapter.

2.2 QoS management in Satellite Communications

At this point, we start by introducing the general architecture of Satellite Communications in Section 2.2.1. Whereas in Section 2.2.2, we present a common approach to perform QoS management using a well-known architecture.

2.2.1 General Satellite architecture

A common reference model of a multi-gateway Satellite architecture is shown in Figure 2.1 [59, 167]. 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 topologies 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).

The main components in each block of this 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, data buffering, and transmission of data on the air interface, among others. These functionalities are coordinated by the control and management systems of the Satellite network. The GW is composed of forward and return 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.

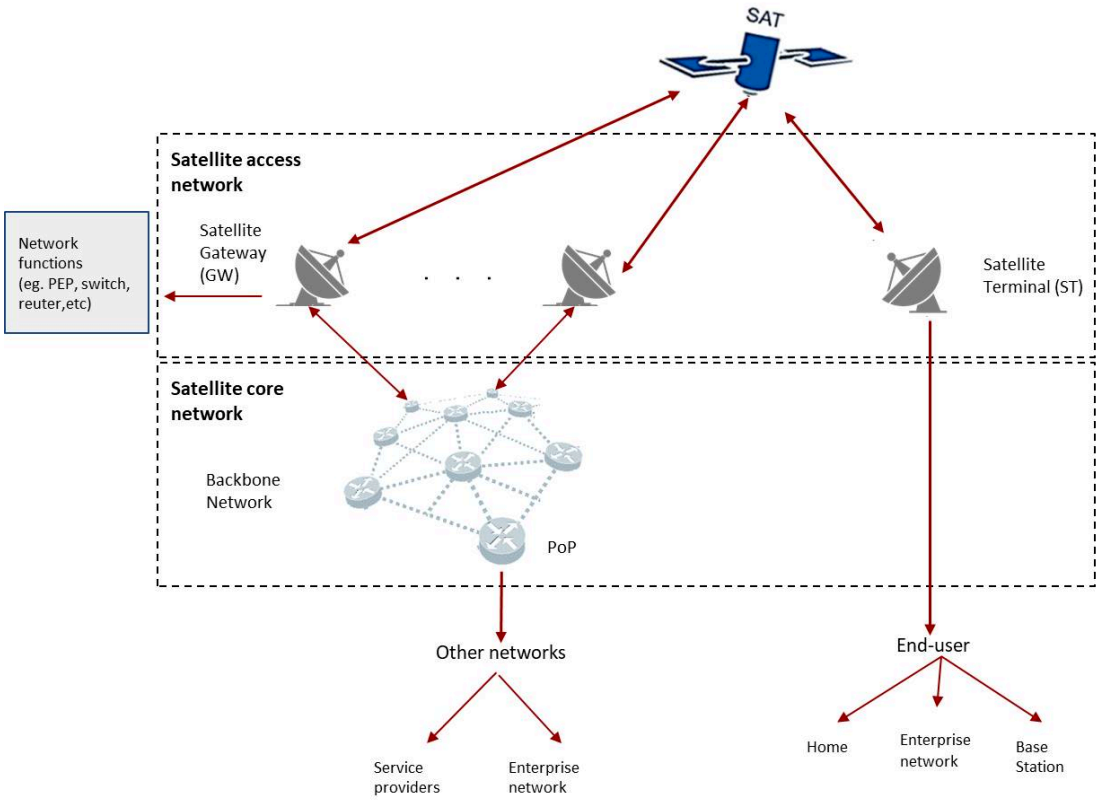


Figure 2.1: Reference model of a multi-gateway satellite network architecture.

- Backbone network: it interconnects network components to provide paths for information exchange.
- End users and other networks: final clients or users of the Satellite network.

One of the main objectives of this architecture is to provide a reliable satellite communication system between different entities. Moreover, 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 managing efficiently the network resources. More specifically, a Policy-Based Network Architecture is deployed at this stage to perform traffic management; this well-known reference model is described as follows.

2.2.2 Policy based network QoS management

This network architecture aims at providing and managing services in order to guarantee efficient Satellite communications. A Policy-Based Network Architecture permits disposing of the necessary elements to achieve QoS management. This architecture must be able to administer, manage and control access of network resources following a rule-based approach. These rules have a condition-action structure. In addition, different types of rules are defined for different administrative domains. For instance, the model Policy Core Information Model (PCIM) [123, 122] divided such policies

into: i) Abstraction of policy rules, defined at different levels, from business to network parameterization objectives, ii) Policy defining an action, define the actions to take in order to guarantee the policy when its conditions are met, and ii) Policy defining a condition, the requirements or events needed to execute the policy's actions.

The more basic architecture of a Policy-Based Network (PBN) is shown in Figure 2.2. The PBN is composed of four main components,

- Policy Decision Point (PDP): is a logical entity 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 [183].
- Policy Enforcement Point (PEP): is a logical entity that enforces policy decisions [183]. In this architecture, the PEPs initiate persistent TCP connections with the PDP. This connection will allow the exchange requests/decisions between themselves (PDP and PEPs).
- Common Open Policy Service (COPS): is a query/response protocol.
- Human network manager: its objective is to construct, deploy policies, and monitor the PBN status. In order to do so, the manager uses a policy management tool that serves as the interface between the components related (PDP and Policy repository).
- Policy repository: this element stores the policy rules and policy data.

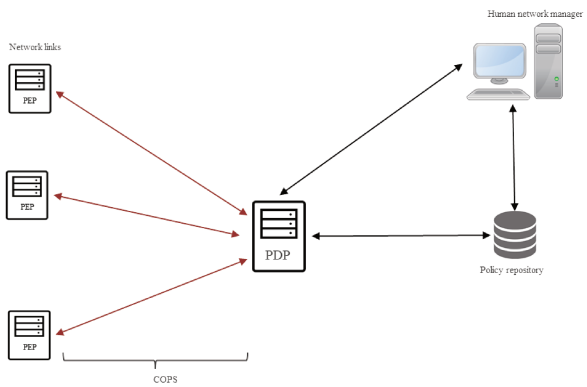


Figure 2.2: Policy Based Network Architecture.

As stated earlier, the PBN is a structure that allows assuring basic QoS requirements. Therefore, in Internet traffic classification, this architecture is widely used. Marked Internet traffic can be forwarded to the PDP. This last one can or cannot take decisions according to the former results to schedule the traffic.

Up until this point, we notice that Internet classification solutions should be a function coupled to the monitoring points and linked to the PDPs. As a consequence, such solutions are highly correlated to the monitoring point process and conditions. In the next section, we continue with the next fundamental level to this study: how traditional approaches perform passive monitoring and traffic classification in Internet networks.

2.3 Traditional Internet traffic classification

In order to better understand the Internet traffic classification process, we present as follows how to monitor Internet traffic in network appliances in Section 2.3.1. The monitored traffic can be encrypted or unencrypted; the difference between these types of traffic will be set in Section 2.3.2. Finally, classical approaches will be described in Section 2.3.3.

2.3.1 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 [78] 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 step refers to the procedure of extracting the binary data from monitoring points, in this step each packet is considered as a single independent entity[86]. 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 reserach works try to improve separately each of their deficiencies. For instance, [86] presents a taxonomy to categorize the packet sampling techniques, this work aims at giving guidelines to select the most adequate method according to the objectives to achieve. The works in [124, 172, 177] present packet capture engines running on commodity hardware, which are useful for reducing the time response in traffic classification. Some other implementations of traffic monitoring are found in the literature, such as the work in [128] over industrial networks, while [9, 96] apply it over home networks. The work in [65] reports the most common implementations of popular network monitoring approaches for packet capture (Tcpdumb, Wireshark, etc.), flow metering (nProbe, YAF, QoF, etc.), and data collecting (nProbe, flowd, nfdumb, etc.).

One of the main challenges regarding traffic monitoring relies on capturing packets in real-time. A large volume of data at high speed is involved. In order to deal with streaming data, classically batch-based methods are deployed; however, these methods do not offer fast responses in critical environments, such as multimedia traffic monitoring [151] or network threats. Classically, the batch-based method is deployed after the flow metering and exportation steps. Its main purpose is to manage and store the exported flows, usually within an interval of time, into binary files (e.g. pcap files). The batch-based method must fulfill certain requirements such as data processing performance and fault tolerance. In the case of the stream-based approach, the former requirements must be accomplished with more exigent demands, in particular, the data processing performance. For instance, the work in [89] analyzes the

traffic observation process in a streaming way to reduce delays for traffic classification. The authors propose a workflow that distributes the exported flows by using a messaging system. The IP flows are transformed into a data serialization format. The selected data serialization format was the Binary JavaScript Object Notation (BSON). The final aim is to offer a distributed data system that is more efficient than a batch-based method. The work in [29] evaluated the performance of one of the most used distributed stream processing systems for traffic monitoring. The same authors extended this work to compare three stream processing systems, in order to find the suitability of each one for real-time network flow processing [30].

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

2.3.2 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 as 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. Normally, these fields are inspected in order 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 [101]. 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 [101], TCPcryp [17], and TLS/SSL [62, 46]. 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 the Virtual Private Networks (VPNs) which tunnels multiple flows into only one flow by using encryption protocols such as IPsec [163]. VPNs are deployed to allow securing communications on the Internet, and accessing to 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.

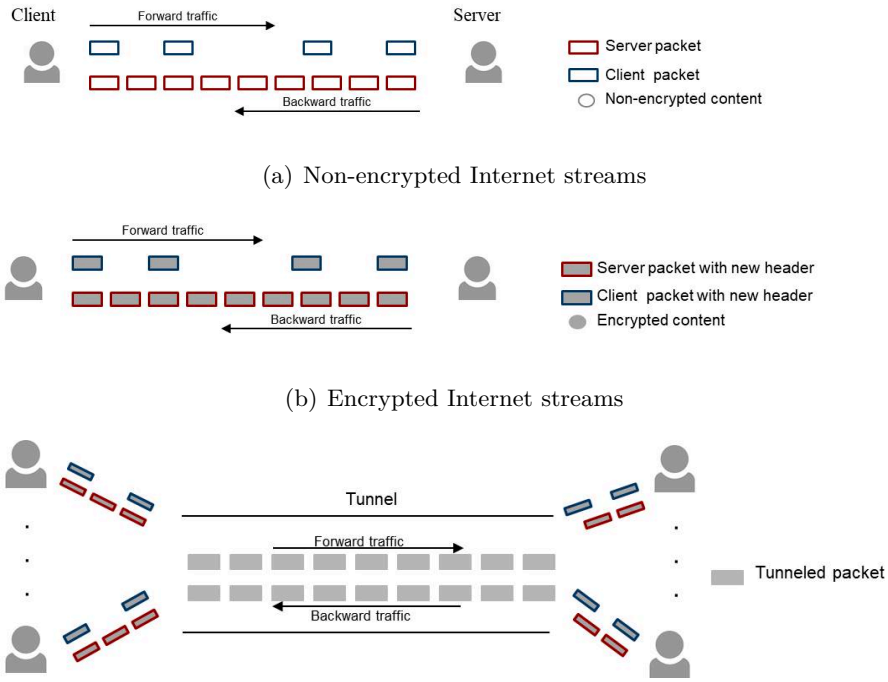
For the objectives of this investigation, it suffices to show the difference that the non-encrypted, encrypted and tunneled communications will carry for the data processing steps. In Figure 2.3, we can graphically place these differences in a general way. In summary, we observe:

- 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 2.3(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 2.3(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 server as seen in the figure.

- Tunneled data: Stream content is encrypted, and in addition, another layer hides the IP directions of clients and servers. In this particular case, the data streams are not separable. In Figure 2.3(c), we can notice that several clients/servers can be using the same tunnel, mixing in this way the Internet traffic.

Before continuing, let us remind that the classification objective is to label each data stream seen between client and server. Intuitively, in Figure 2.3, for the cases a) and b) the classification is easy to achieve, it suffices to model the data exchanges for each Internet application; contrary case occurs with case c) where a special treatment should be deployed. The knowledge provided, about the Internet flow streams, allows us to continue our scientific review with the classification techniques in the next section.



(a) Non-encrypted Internet streams

(b) Encrypted Internet streams

(c) Tunneled Internet streams

Figure 2.3: Illustrative example of the difference between non-encrypted, encrypted and tunneled data

2.3.3 Traffic classification approaches

Several trends can be found to classify, comprehend, diagnose or observe the status of the network. A taxonomy of traffic classification can be found in [92]. Nonetheless, a modification of this taxonomy is proposed in Figure 2.4. This figure counts with two main divisions: *Data* and *Techniques*. A brief description of each component is given as follows.

- *Data*: it refers to the type of input data used to create the traffic classification solution. It is noticeable that traffic data can be encrypted or non-encrypted. The traffic can be labeled either by DPI tools or real-time captures.
- *Techniques*: four main branches are detected, such as Machine Learning, Statistical based, Behavioral based and Payload inspection. In this section, these four approaches are briefly described in order to have a general idea of how they work.

Payload inspection, normally denoted Deep Packet Inspection(DPI), is found as the common approach to perform traffic analysis [7, 22]. 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

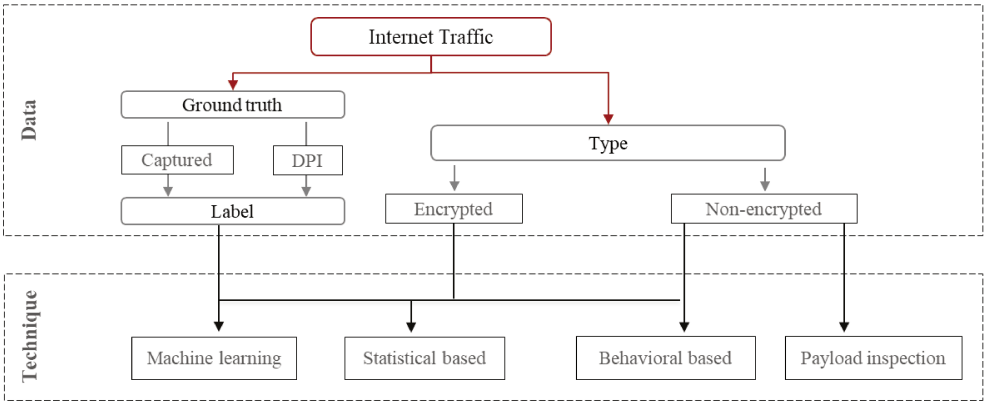


Figure 2.4: General view of the traffic classification approaches.

updated; otherwise, they will fail in their prediction getting as result an unknown or an erroneous signature. As a consequence, the list of the tools' signatures has to be constantly updated. 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 [18, 125, 164]. The work in [18] introduces a monitoring scenario with public and private IP addresses, and measures statistics for each profile, such as the number of TCP and UDP packets, as well as the number of failed flows. The objective is to construct different statistical distributions, such as negative exponential or Gaussian, to detect the reachability in P2P communications. In a similar manner, the work in [125] proposes the categorization of the flows using statistical distributions. The main deficiency of this approach is the static construction of statistical models that do not integrate learning processes. Like the previous approach, this disadvantage affects its performance in the presence of dynamic growing and evolution of the Internet traffic patterns. In addition, some of the statistical-based approaches are adapted and improved by the ML techniques.

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 [107, 80, 90, 3]. The most common representation of behavioral patterns in the network is through graph modeling, in which graph theory is used to find highly connected nodes (hosts), number of connections, and opened ports, among others [107]. As an example, [80] analyzes traffic behavior to identify P2P traffic. The first step is to cluster together similar flows through a k-means model. Following, the clusters are represented by a Traffic Dispersion Graph (TDG), where the nodes are represented by the IP addresses and the link between the nodes is the registered flows. Finally, a set of rules is applied

over the graphs to detect the name of the application. These rules take into account features, such as the percentage of nodes and the average node degree of the graph. The work in [90] proposes an approach to identify P2P communities, where the interactions in the network are represented by graphs. The nodes are formed by the tuple (IP, port), and the connections are given by the number of packets interchanged between nodes. P2P networks are identified by using the port distribution of known remote peers, in order to do so, a multinomial classifier is built to decide whether a graph represents one of the known networks. A different target that is normally studied by these techniques is identifying the traffic activity patterns, such as the works in [87, 181]. For instance, [87] presents the Traffic Analysis Graphs (TAGs) for visually unveiling the behavior of different types of applications. In a TAG, the nodes are the IP addresses and the edges are the flows of interest; the flows of interest are defined according to the purpose of the study, in order to build TAGs that capture relevant traffic activities among hosts. [181] builds bipartite graphs and compute their similarity matrix. This matrix will serve as input to a clustering algorithm (k-means) that will gather together similar nodes. This technique is highly expensive due to a lot of network information is required to build behavioral models; in addition to this, Internet traffic evolution and network configuration updates remain as a deficiency.

The classical traffic classification techniques share the same constraints deficiencies that we can sum them up as follows: a) Internet traffic evolution causes a decrease of the classifiers' performance; in consequence, constants updates of the classification solutions are needed, and b) encrypted traffic cannot be classified by DPI solution, while statistical and behavioral techniques do not cover this subject either. Conversely, ML could cope with these challenges becoming in this way one of the front-and-center tools used by this investigation. We will deeply explore it in the next two sections.

2.4 Machine Learning introduction

Machine Learning (ML) techniques are very popular approaches to identify and classify patterns in different domains. Its main objective is to give to the computers automatic learning capabilities, where the machines are able to extract knowledge from a process under certain conditions. ML tries to extract knowledge from a set of features or attributes, which represents the measurable properties of a process or observed phenomena. In this way, the learning process is performed by training different models, i.e., classification, prediction or clustering model; and their use depends on the problem characteristics. The knowledge extraction is handled by an ML model, which is built with historical experiences recorded from case studies.

In general, the steps to achieve knowledge discovery with ML techniques are shown in Figure 2.5. In this figure, a distinction between the blocks that can be performed in an offline (blue arrows) and online (gray arrows) manner is made.

Generally speaking, the offline procedures treat historical datasets stored by the *Data collection* block at one or more monitoring points in the network. In the offline

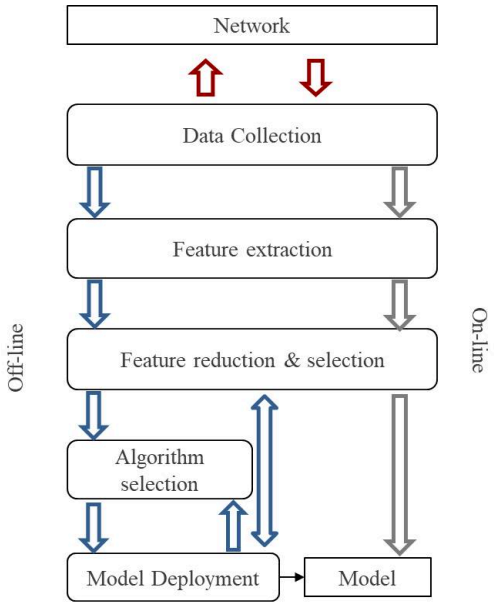


Figure 2.5: General steps to perform ML.

run, a historical dataset must be collected; on the contrary, in the online run, streams of packets are continuously processed.

Once the historical dataset that characterizes the problem is recorded, relevant features are extracted in the *Feature Extraction* block. At this point, the resulting features can be processed by either Feature Selection(FS) or Feature Reduction(FR) approaches, to obtain a reduced space or a set of new features. Following, the *Algorithm Selection* block refers to the procedures and methods intended to select the most adequate ML algorithms. Several works deploy different means of comparison to justify or validate their selection. Finally, the *Model Deployment* block is focused on the efforts for implementing such models. In Figure 2.5, the FR and FS procedure also have a connection with the *Algorithm Selection* and *Model Deployment* blocks, due to some approaches select the most relevant features based on the performance given by the ML models.

On the other hand, in an online run, data is collected to perform feature extraction and FS or FR, to later evaluate these features over the model built in the *Model Deployment* block. We can remark that manipulations in an online manner, over the *Algorithm Selection* and *Model Deployment* blocks, might envisage evolving or upgraded ML solutions. It is also important to mention that this workflow represents a guide that comprises the general steps to use ML for traffic classification. However, the order of these steps may vary and can be also found combined. We briefly overview these steps in the following sections.

2.4.1 Data collection

This step aims at gathering information regarding a case study. Measuring procedures are established, in order to capture data either from physical or digital sensors. Such data describes the current or historical status, which is used to define the experimental testbed. A testbed is composed of all the software, hardware, and networking

components, among others held by the process of interest. This testbed is necessary for building the model (learning and testing) with the ML techniques. Samples are captured and gathered from multiple scenarios set in the testbed.

2.4.2 Feature Engineering

It is one of the most important steps due to it allows measuring or computing features that might give information about the state of the process. It also treats the features to obtain a better representation of the process with feature selection and reduction procedures. We briefly detail the common procedures in this phase.

a) Feature extraction (FE)

In brief, an FE procedure computes different metrics that reflect specific properties in the data collected. The main aim is to obtain descriptors that better characterize the problem. The result of the FE process is a structured table formed by columns of attributes where each row is a sample, with an additional optional column with the current status of each sample (commonly called label or class). In case that the status is unknown, the samples are unlabeled.

Data processing procedures can be performed in order to delete unwanted missing values and to clean the data, among other things. This last one is related to outliers detection that might disrupt the ML solution performance. Also, the data can be transformed through normalization or aggregation operations over the values of the attributes. In the aggregation procedures, the features are combined into a single feature that would be more meaningful to the problem.

b) Feature Reduction (FR) and Selection (FS)

This is an optional step that allows selecting or reducing the number of extracted features. FR is to create new attributes using the original ones, while FS is to find a low set of attributes that better describes a process. FR and FS aim at decreasing time response and memory consumption, and increasing performance, among others. Surveys about the performance and comprehension of the FR and FS processes are presented by [110, 31]. These techniques are commonly divided into Filter, Wrapper and Embedded approaches, which in turn can be developed by supervised and unsupervised strategies. In the supervised strategy, the objective is to find the features that most contribute to defining the classification decision. In the unsupervised strategy, the main aim is to determine the features that allow a correct clustering of the data.

2.4.3 Algorithm selection and model deployment

Different ML algorithms have been developed and tested for solving tasks, such as classification, clustering, and regression. The selection of the ML algorithm is related to the problem to solve or the type of knowledge that the practitioners want to discover.

In ML, there are two classical types of learning, supervised and unsupervised learning. Most of the supervised learning algorithms adjust their model parameters

in order to minimize the error between the model output and the real expected output of an input. This means that historical data has to be labeled. On the other hand, unsupervised algorithms try to find relationships between the inputs without beforehand knowledge of the outputs. These relationships can be similarities, proximities, and statistical relationships, among others. As a derived consequence of the learning process, the supervised algorithms are commonly used to perform classification tasks, while the unsupervised ones are rather used to cluster inputs in order to find anomalous or similar behaviors between themselves. In general, the ML model and the type of learning are associated with the type of problem to solve.

Different categorizations of the ML algorithms can be found [52, 179, 73]. For instance, a general classification groups the classical supervised algorithms based on a statistical model, trees, rules, and neural networks (NNs), among others. In addition, several approaches do not necessarily belong to the groups mentioned above and can be grouped into parametric and non-parametric.

Nowadays, there is a big variety of ML algorithms not only based on supervised and unsupervised learning. For instance, semi-supervised algorithms take advantage of unlabeled data to train classifiers, either training a classifier with the labeled samples and then evaluating the unlabeled ones in the classifier, or using unsupervised approaches for the unlabeled samples. Hybrid approaches are also found through the combination of supervised and unsupervised learning, due to the presence of labeled and unlabeled samples in datasets [141]. Moreover, ensemble techniques use a variety of ML models (commonly classifiers) and combine their results through a combination strategy. Strategies such as Bagging and Boosting are widely used to build ensemble models [142].

2.4.4 Validation of classification models

This section overviews the most common validation approaches for classification solutions. Supervised learning requires the beforehand knowledge of the sample labels, which are key information to validate the ML models. The usual approach is to divide the dataset into a training and a test set. The ML models are built with the training set, while the resulting models are assessed by the test set in order to evaluate their prediction capabilities. Given the model predictions and the ground truth labels of the test set, several performance metrics can be deployed to quantify the classification capability of an ML solution. For instance, [159] presents a study of the classification performance metrics divided into the type of classification to achieve: binary, multi-class, multi-labeled and hierarchical. Binary classification occurs when an input sample can be classified into only one of two distinct classes. On the contrary, multi-class classification implies that the input can be classified into only one class within a pool of classes. Multi-labeled classification allows the classification of an input sample into more than one class in the pool of classes. Finally, hierarchical classification is similar to the multi-class classification but with more granularity, in the sense that the principal classes are divided into lower levels of subclasses.

In order to validate ML models, one of the most common approaches is to measure their performance in terms of the classification capabilities. Several relationships

can be found among the model predictions and the ground truth labels, such as the number of samples correctly and incorrectly assigned to a class, among others. These counts allow computing metrics, such as Accuracy, Precision, Recall, F-score, Receiver Operating Characteristic (ROC), etc. For instance, for binary classification with a positive and a negative class, the counts of true positives (TP), false negatives (FN), true negatives (TN), and false positives (FP) can be used to compute the performance of the classifier through the sensitivity and the specificity metrics. A combination of these metrics, such as the F-score and the ROC, offers more precise information about the performance of the classifiers for both classes. Particularly, the ROC curve is obtained by computing the sensitivity and specificity varying the classifier's discrimination threshold [58]. In the ROC curve, the ideal value represents a high sensitivity and specificity. The result is an interpretable figure that illustrates the performance of the classifier in different operating points; in addition, the Area Under the Curve (AUC) can be computed to obtain a compact measure from the ROC. The analysis above can be mapped to multiclass-problems by computing overall performance measures with micro or macro averages of the binary performances. Each of these metrics exposes different aspects of the model performance [143, 75].

Following the same scheme previously presented, in the next section, we envisage to describe how ML is applied over our application, Internet traffic classification.

2.5 Internet traffic classification with ML

For this particular domain, one of the main goals is 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 can be extracted and used for traffic classification [121]. 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. However, this approach can encounter problems with the use of the HTTP2 protocol and VPNs, due to the separation of communication sessions is not explicit.

In the following sections, we present an overview of the most important elements to consider for each step in Figure 2.5. This study will lead us to identify the challenges in this field.

2.5.1 Data collection

The *Data Collection* step allows measuring different scenarios on the Internet. This phase mainly collects IP flows within a time window. Additionally, this block carries several steps, such as packet management, flow reconstruction, and storage. Traditionally, historical data is a very important source of knowledge for building ML solutions. A rich and complete set of observations regarding a problem can improve the performance and generalization of the ML models. However, in the traffic classification domain, this aspect is critical due to: complexity and scalability of the Internet network, the constant evolution of the traffic, and privacy policies that do

not allow data collection, among others. In consequence, real Internet traffic data is hardly available for analysis and knowledge extraction. Two main aspects must be taken into account for the data collection, they will be explained as follows.

a) Network environment

In particular, three main trends were found in the field: real traffic acquisition, traffic generation, and emulation. Real traffic is normally collected from the network, obfuscating private information among communication entities (e.g. clients-servers). Traffic generation tries to simulate similar real traffic conditions by copying or modeling real interactions through scripts [19, 180, 37]. Finally, traffic emulation aims at setting scenarios as close as real ones, where one or more actors can intentionally emulate common interactions in the network [117, 120, 171, 131]. The optimal solution is to capture real traffic from the network in order to have a reliable source of realistic data. However, this solution is hard to conduct, mainly due to privacy matters; in addition to this, real traffic is hard and most of the time impossible to label with its ground true application. Even though some works manage to obtain realistic conditions for monitoring traffic, the data is hardly ever publicly available for the research community.

b) Data measurement

The process detailed in Section 2.3 is followed for data collection: i) packet capturing, ii) flow metering and exportation, and iii) storage. Moreover, a new procedure, called label assignment, is defined. The label assignment procedure refers to set an identifier for each flow; this identifier is related to particular patterns in a communication session, e.g., name or type of application. Associate ground truth information with traffic traces can be a tedious task due to the complexity of tracking the flows belonging to specific applications; moreover, this procedure is a key task for validating traffic classifiers. For the best of our knowledge, few works implement a reliable ground truth assignment, due to the most common approach is to use DPI tools for such task [69, 28].

2.5.2 Feature engineering

Once the IP flows are collected, the next step is to process this data to find representative features that describe the Internet communications.

a) Feature extraction

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 time and packet lengths) that are unique for certain types of applications, and enable different source applications to be distinguished from each other [41, 116, 139]. Under this assumption, the work in [121] proposes 249 statistical features, which can be extracted from flow network traffic.

b) Feature Selection and Reduction

ML processes might or might not count with an FR or FS process. Several studies state that a low amount of features is needed to wholly obtain patterns that differentiate an application to another. The works in [188, 56, 55] study the most relevant statistical features for traffic classification. In [56], several FS techniques are used to obtain the most important features, while a newly proposed method selects the smallest set. The results were crossed validated with three datasets measuring the goodness, similarity, and stability of each feature; giving, as a result, a small set, between 6 and 14 statistical features, that offers the best performance measured through the accuracy.

2.5.3 Classification with Machine Learning

It is common to find in this field different solutions using a variety of ML algorithms. Given the wide number of ML algorithms, finding the most adequate is very important in traffic classification. Particularly, most of the works have based their selection on building and testing several models until finding the one with the highest performance. In this section, selected works are detailed to outline the challenges involved in selecting an ML algorithm.

Along with our study, it was noticed that most of the works are focused on particular objectives to achieve: either to identify the type of application, the protocol application, anomalies or tunneled connections. The most popular task is Internet traffic classification, which aims at detecting the application name of IP flows. Labeled datasets are used to train supervised algorithms in order to select the best model, which in turn is obtained by measuring the classification performance [161, 139, 118, 186]. However, for more fine classification tasks, the combination of several classifiers might solve the generalization problem encountered by the classical classifiers. These types of solutions aimed at creating more specialized classifiers to solve the class-imbalance problem [25, 70, 1, 112]. One or more classes have more samples than another. This case is very harmful to the ML classifiers due to it will bias them to learn more from the majority class than the minority. It can also cause over-specification (overfitting) of the minority class. This behavior is commonly found in traffic classification.

Moreover, unsupervised techniques are deployed for anomalous detection, due to its capabilities to detect patterns that are not similar to normal or nominal conditions; but, also to perform classification tasks [10, 157, 175, 109, 50]. Finally, hybrids and advanced approaches combined supervised and unsupervised techniques to achieve a more accurate performance than the techniques above [190, 13, 14].

2.5.4 Model deployment

The main question that arises is how the ML solutions can be deployed into real Internet network scenarios. For most of the works studied in our survey paper [135], the implementation of the ML solution is not performed, and normally a proof of concept is presented. Firstly, we present some works give hints about how the ML

approach can be deployed. Finally, it is analyzed the importance of the ML solution reconfiguration; which plays an important role in the traffic classification task.

a) Online implementation

One of the most important features, that the Internet network has, is that transmission rates are normally very high and the dimension of the network is big. These main characteristics make the classifier implementation efforts challenging. The most common approach is to deploy the ML solutions over the traffic monitoring tools; hence, each time that a packet flow is observed, it is possible to perform the classification. For instance, [27] uses a NetFlow enabled router for monitoring the traces, which are forwarded in an online manner to an ML classifier. NetFlow is a Cisco protocol that aims at exporting IP flow information from routers and switches. Similarly, DBSstream [12] integrates the traffic classification solutions into its monitoring platform. Another approach is to implement the ML solution in a stand-alone classification module; for example, the work in [150] implements the ML solution into a Field-Programmable Gate Array (FPGA) based embedded system. The FPGA device uses the information at the network layer, such as the packet sizes and IATs.

b) Reconfiguration

One of the main issues, found in most of the ML solutions studied so far, is that when new patterns are appearing in the network (e.g. new codecs or applications), the ML models must be updated. The ML-based classification is pruned to rapidly be out-of-date due to the dynamism of the network. Therefore, self-learning, evolving or retraining strategies must be taken into account [91, 47, 176]. In general, for most of the ML techniques, an update implies a model retraining with a new historical dataset (commonly labeled). This means that the model has to be constantly retrained at any time that a new application or behavior appears on the network. The cost of performing this step can be significant; nonetheless, if this point is not considered then the model performance is at risk.

In the next section, we make a summary of the main trends in each subject discussed previously. This study will lead us to place the main challenges in this field.

2.6 State of the Art Analysis

More concisely stated, particular characteristics are taken from each phase in Figure 2.5. These characteristics are detailed in Figure 2.6, likewise they represent some paths that most of the works in this field follow for traffic classification. In this figure, it can be noticed that for the *Data collection* phase, it will be studied if the Internet traffic is: real or emulated, publicly available, encrypted and labeled (ground truth). These aspects will become very important for characterizing the problem. The *Feature Engineering* phase comprises the FE and FS approaches used by the

reviewed papers. In this sense, four FE approaches were found as the most common ones, such as statistical-based (STATsB), graph-based (GRAPH), time series based and hybrid approaches. In the FS phase, it will be denoted if the reviewed papers performed or not this procedure. Following, for the *Algorithm selection* phase, the ML approach used and the objective to achieve are defined. The trends studied are classical classification (CClass), Multi-classification and ensemble approach (MClass&E), clustering for classification and anomaly detection (Clust), and hybrids and advances techniques (H&A). Among the classification objectives studied were found application name (AppN), application category (AppC) and anomaly detection (AD). Also, other objectives are considered, such as user behavior detection and community search, among others. Finally, in the *Model Deployment* phase, it is differentiated the papers that implemented the ML solution (YES), and the ones that did not do it or do not specify it (NNS). The reconfiguration of the solution will be a key aspect to study, in this sense, it is verified if the reviewed papers offer either a retraining (RTraining), a self-learning or evolving (SLE), or other/non-specified (ONS) process.

The question that arises is: which is the best path to take? 49 papers were selected as they present the complete procedure in Figure 2.5 using a variety of strategies for traffic classification, compelling to show the challenges remarked in the previous sections. The results of the state of the art are summarized in Figure 2.6. A path is drawn in order to know the procedural trends commonly taken.

It is noticeable from Fig 2.6, that most of the works studied did not use encrypted data, and the ground truth establishment was not commonly performed. In the feature engineering process, few works applied FS for their solution. In terms of the ML approach used, the less explored are the unsupervised techniques, the multi-classification, and the ensemble approaches. Finally, for the model deployment phase, there are few works that implemented the solution in real-world scenarios. Moreover, reconfiguration processes do not handle evolving or autonomic reconfiguration. Given these scenarios and the discussions of the previous section, the following future trends have been identified.

- The proposition of emulated traffic architectures, oriented to create ML solutions, can allow the generation of encrypted and non-encrypted traffic, as well as a reliable flow labeling process.

- Given the efficiency of the statistical-based approach, an exploration to improve its computation will provide robust classification solutions. In the case of the FS, an efficient dynamic selection of the features might offer a higher classification performance.

- In the algorithm selection block, the future trends to exploit are the deployment of multi-classification and ensemble approaches, which are very promising path given the characteristics of the Internet data.

- For the ML solutions, more experimental tests are needed to measure the performance in terms of response time and complexity. In this particular case, it is

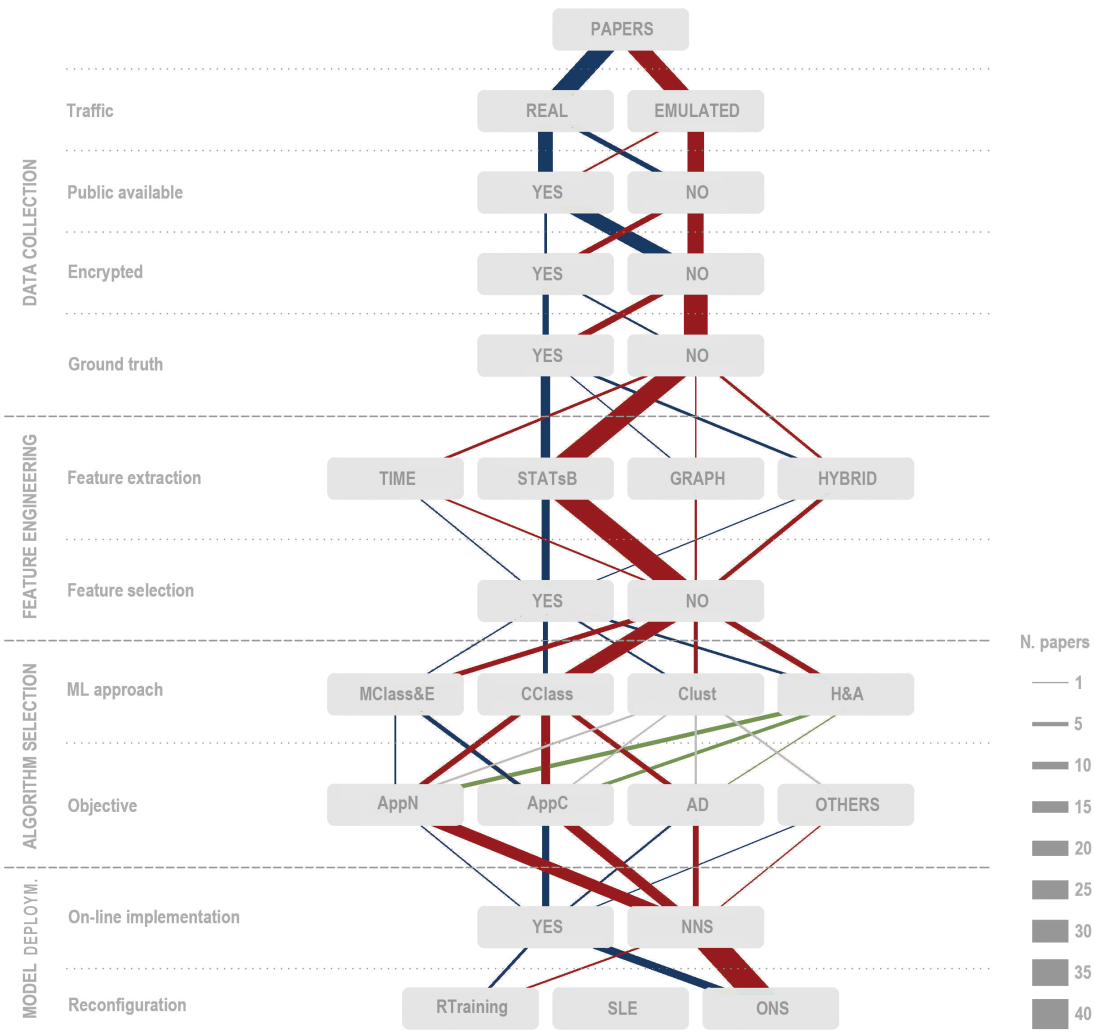


Figure 2.6: Trends of the selected papers for classifying traffic with ML.

necessary to propose distributed and scalable ML solutions that can deal with the dimension of the Internet network.

- In terms of the Model reconfiguration, an alternative solution will be to deploy the concept of incremental learning, in order to deal with the dynamism of the Internet network.

2.7 Summary

This section presented the common procedures to achieve traffic classification through ML techniques in Satellite Communications. In this way, the general Satellite Architecture was studied; as well, as the network functions that allow placing an Internet traffic classification module. We notice that in a PBN architecture, the ML solution can send the classification results to the PDP component, which in turn will act on the Satellite Architecture for resource provisioning.

The procedures to achieve traffic classification were analyzed, where ML stands as a promising path in this field. In particular, the ML approach can deal with

encrypted and non-encrypted data thanks to its non-intrusion to the content payload. In general, working with encrypted traffic is challenging, where the type of features should lead to improve classifications in the absence of the packet content. The current publicly labeled data is scarce, which makes it hard to compare ML solutions. Furthermore, for knowledge extraction with ML, the classification task was found as the most popular. About this subject, multi-classification and ensemble approaches present some advantages that make them compelling for dealing with some problems in traffic classification, such as class-imbalance and generalization. However, the clustering approach can help to find new or anomalous behaviors in Internet traffic; therefore, its study in this field should be extended. Finally, the implementation of such solutions remains an important task to achieve, due to different factors, mostly related to performance and adaptability of the solutions.

In view of the aforementioned aspects, in the next chapter, we introduce a general architectural view of our proposal. We propose a system to perform Internet traffic classification with ML in a Satellite architecture.

Chapter 3

QoS management in Satellite Communications with ML

Contents

3.1 Introduction **31**
3.2 Capella **32**
3.2.1 Operational analysis 33
3.2.2 System Analysis 33
3.3 Reference high-level architecture: QoS management in Satellite Communications with ML **34**
3.3.1 Passive monitoring 36
3.3.2 Feature Extraction 36
3.3.3 Classification system 37
3.3.4 Policy based network architecture 39
3.4 Requirement analysis with Capella **40**
3.4.1 Operational analysis 41
3.4.2 System analysis 44
3.5 Summary **49**

3.1 Introduction

The result of the state of the art analysis remarks the need for implementing an ML solution for traffic classification in Satellite Communications. Current approaches and commercial products are mightily pointing to this kind of solution; however, there are few real implementations. Traditionally, traffic classification is widely implemented by Payload Inspection solutions, however, recent efforts start including or proposing ML learning solutions to their engines [145]. Moreover, according to our knowledge, few works addressed the problem from the Satellite point of view. In this chapter, we outline what will become our first contribution: a proposal for integrating an ML solution in a Satellite architecture. We use as reference model an Architecture proposed by Thales [44], which in turn will serve us as guidance along with this chapter. Therefore, this chapter focuses its attention on developing a framework that can be deployed in Satellite architectures. 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 also propose to identify the requirements to integrate the solution in such architecture.

In order to formally define the requirements of the system, we follow the Model-Based System Engineering methodology proposed by ARCADIA and the open-source methodology tool named Capella. This tool will help us to identify the needs of the system in order to create an implementable model. These requirements are in compliance with the main challenges found in the literature in Chapter 2.

The remainder of this chapter is given as follows. Section 3.2 presents a brief overview of the Capella methodology. Section 3.3 theoretically introduces the proposed architecture. In Section 3.4, we model the proposed architecture by using ARCADIA concepts and diagrams. Finally, Section 3.5 concludes this chapter.

3.2 Capella

Capella is a software engineering tool based on the ARChitecture Analysis and Design Integrated Approach (ARCADIA) methodology to Model-Based System Engineering (MBSE) systems. Both the methodology and the tool have been designed and developed by Thales in the framework of several industrial projects. Briefly speaking, ARCADIA is a method for systems, hardware, and software architectural design. ARCADIA tries to make a clear distinction between the operational context, the needs to be satisfied by the system, the internal components, and functionalities to be provided. Capella follows ARCADIA principles to provide methodological guidance, intuitive model editing, and viewing capabilities for Systems, Software, and Hardware Architects [26].

In Figure 3.1, we show the main phases to achieve the modeling and implementation of a system. In this figure, the Operational analysis and System analysis are closely related, they help to find and define the requirements of the system. Whereas, the Logical and Physical architectures aim at developing the solution.

More specifically, in Figure 3.1, from the left to the right, we find several views integrated into viewpoints that hold the models and architectural designs for each level. Briefly speaking, the first level consists of representing the operational activities related to the problem, as well as the related actors and entities. These activities are transformed into a flow of functions during the System Analysis, it is here where the needs of the system are shaped. Following, in the Logical Architecture, the functions are refined and allocated to specific components. Finally, in the physical architecture, these components are expanded in view of their implementation over software or hardware entities.

The viewpoints use different concepts which can be described by dataflows, scenarios, functional chains, etc. In the next section, we will briefly describe how to build the first two levels of this methodology

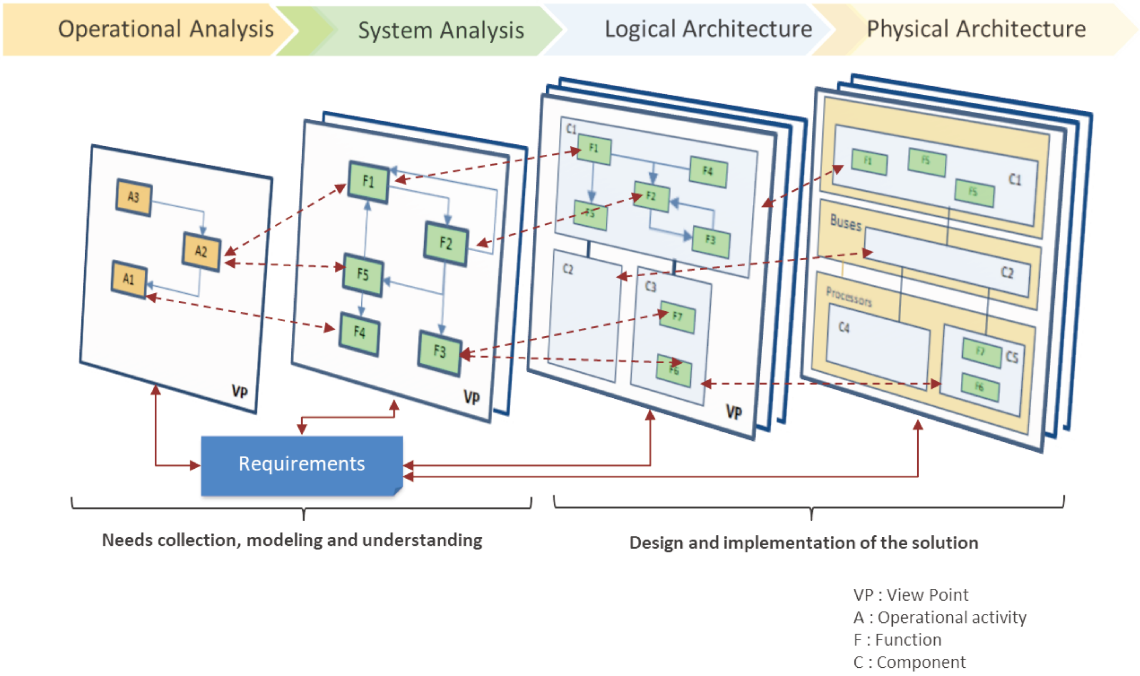


Figure 3.1: Phases for modeling with Capella.

3.2.1 Operational analysis

The operational analysis helps to define what the entities or users need to accomplish. In this viewpoint, it is identified the actors and entities that will interact with the system to be developed. The main activities that the actors/entities should or must perform are described; in addition, the operational activities followed to fulfill the requirements of the problem are also represented. In this phase, several concepts can be used to represent the operational analysis. Some of them are described as follows,

- **Entities and actors**: the actors and entities interacting with the system.
- **Capabilities**: they help to find the interactions between the actors/entities with the system.
- **Actor Activity**: it describes the flow of activities that the actor/entity follows to develop a capability.
- **Entity Scenarios**: They complement the actor/entity activity diagram with a sequence diagram like structure.
- **Operational context**: It captures the allocation of the activities into the actors/entities in only one diagram. This is the final result of this phase.

3.2.2 System Analysis

The activities previously defined are broken down into system functions. In this phase, more details about the needs and functionalities of the system are outlined.

This analysis is focused on what the system needs to accomplish for achieving the actors/entities' goals. Some concepts commonly used in this phase are listed below.

- **Missions and Capabilities**: it is similar to the previous phase, however, the capabilities can be finely detailed and can be specialized to better precise the needs at the system level.
- **System function**: In this diagram, more fine-grained functions develop the activities defined in the operational analysis.
- **System architecture**: The system functions are allocated into the system actors/entities. It is important to mention that these actors/entities are retrieved from the operational analysis. Nonetheless, new component functions can be defined to accomplish the mission of the system.

The ARCADIA methodology will be used to analyse and define the requirements of the ML-based classification system, as well as to design and implement our proposal. In order to do so, we use the Capella tool to develop the two phases previously described; but at first, let us introduce in the following section the theoretical basement of our proposal to later perform the requirement analysis.

3.3 Reference high-level architecture: QoS management in Satellite Communications with ML

The architecture in Figure 3.2 is an operational architecture already studied and proposed by Thales [44]. This architecture takes the network functions of Figure 2.1 and the PBN in Figure 2.2 integrated with the ML-based classification system. In this figure, an abstraction of the elements in a real Satellite network distribution is given. In the figure, we have added three new functional elements to provide Internet traffic classification: *Passive monitoring*, *Feature extraction* and *Classification System*. Internet classification is forwarded to the PBN, following the PBN will take the necessary actions to improve the QoS.

In the figure above, we also show three essential components in the PBN: PDP, Resource allocation PEP, and QoS server. Briefly speaking, traffic classification is forwarded to the PDP that will define what QoS policy should be applied to a flow or set of flows. The QoS policy is then sent to the Policy Enforcement Functions (PEFs) such as the QoS servers and Resource allocation PEP. On the one hand, the QoS server applies the QoS rules on the equipment that handles the traffic (GW, ST). The QoS Server could be composed of a PEP receiving the QoS Policies and to a PEP formatting the QoS policies toward the concerned equipment: ST, GW, routers, proxies. It is important to mention that PEF functions corresponding to QoS Servers can be centralized or distributed in GWs and STs for the Quality of Service enforcement. On the other hand, the resource allocation PEP provides the resource when needed to the User Terminal. To instantiate the resource allocation toward the terminal, the Resource allocation PEP is composed of a PEP receiving the QoS

policy for the flow from the PDP communicating with an internal PDP. Additional functional elements of the PEF functions are omitted due to they do not affect the modeling of our approach. In particular, what will be of paramount importance is correctly handling the traffic monitored and signaling the classification to the PBN.

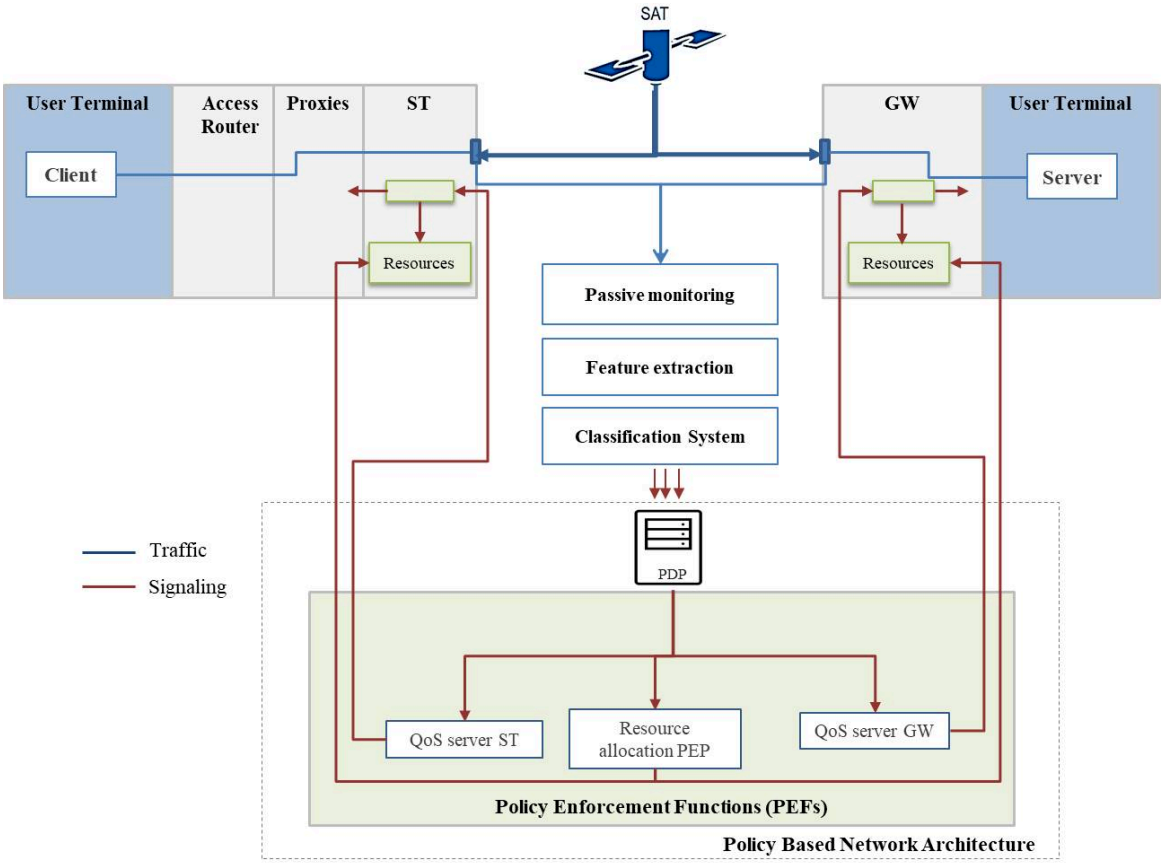


Figure 3.2: Example of a Satellite Network Architecture with traffic classification.

In the proposed reference architecture, the flow of activities of interest are:

1. Intercept Internet traffic in the GW and ST through the *Passive monitoring* points.
2. Perform feature extraction over 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 Resource allocation PEP and the QoS servers will execute those decisions.

In Figure 2.2, the Internet traffic will be captured from both the GW and ST components and decentralized ML-based classification will be performed. However, another option will be to capture and to treat the traffic only in the GW. Moreover, the PEF functions corresponding to QoS Servers can be centralized or distributed in

GWs and STs for the Quality of Service enforcement. These modifications depend on the architecture choices, and they will not affect the functional operations of our ML-based classification solution. These functional operations will be described in the sections above.

3.3.1 Passive monitoring

Internet packets are captured to be organized into flows. This action comprises steps such as packet capture, flow metering, and exportation, detailed in Chapter 2. The passive monitoring component will be in charge of monitoring the packets of a client from the beginning of the connection until the end (for example, when the timeout is reached). Among the most important parameters to gather per-flow in the monitoring process, we have:

- 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

3.3.2 Feature Extraction

Statistical based features are computed for each flow in order 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 3.3. Then, statistical-based features are computed for each flow in order to describe the communications.

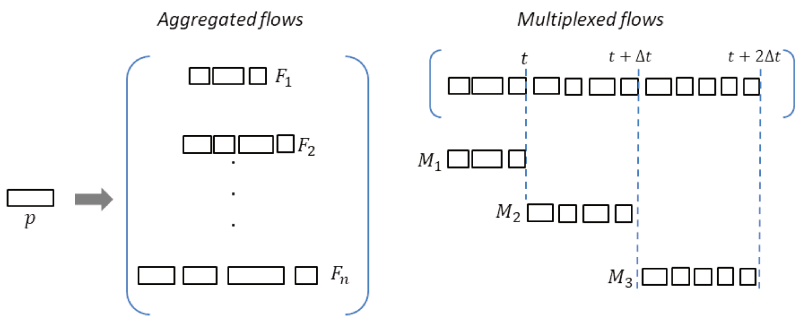


Figure 3.3: Flow reconstruction.

In brief, the properties such as Inter-Arrival Time (IAT) between packets and packets length ($pktlen$) are the most important characteristics considered, with their metrics, such as maximum, minimum, mean and standard deviation, among others.

In addition to the statistical behavior of the IAT and $pktlen$, one can add new characteristics such as the counts of packets arriving within an interval time Δt, as well as the total bytes captured. The feature extraction process will be detailed in Chapter 5.

Conveniently enough, the *Passive monitoring* and *Feature extraction* processes are found interleaved. In consequence, Table 3.1 shows the procedure to achieve these processes. 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. 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 PDP to keep track of the closed sessions.

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 $f \in F | t(p) = t(f)$
   - 2.1. If tf is False (f is not tunneled),
     - 2.1.1. Compute statistical features SF over f
     - 2.1.2. Update $t-1$ parameters of the flow
   - 2.2. Else,
     - 2.2.1. if $t_e(f) < \Delta t$, do the steps from 2.1.1. to 2.1.2.
     - 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
5. Send the IDs $t(f)$ of terminated flows to the PDP component

end for
Output: SF, $t(p)$

Table 3.1: Macro algorithm for passive monitoring and feature extraction

3.3.3 Classification system

Particularly, this system proposes an automatic and logic process to analyze traffic in a hierarchical manner. The classification system is displayed in Figure 3.4. 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). The unitary connections will be treated by a classical ML model called $Cl2$, while the tunneled ones by *Multi-label classifier 1* $MLC1$. 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*. We can notice that the *Model repository* gathers $Cl1$,$Cl2$,$D1$, $D2$ and $MLC1$. In particular, $Cl1$ and $Cl2$ are ensembles of classifiers formed by the set $\{C_1, ..., C_n\}$. The *Online configuration* will be somehow applied over these ensembles. We will give more details about all the components of this *Classification System* in the following sections.

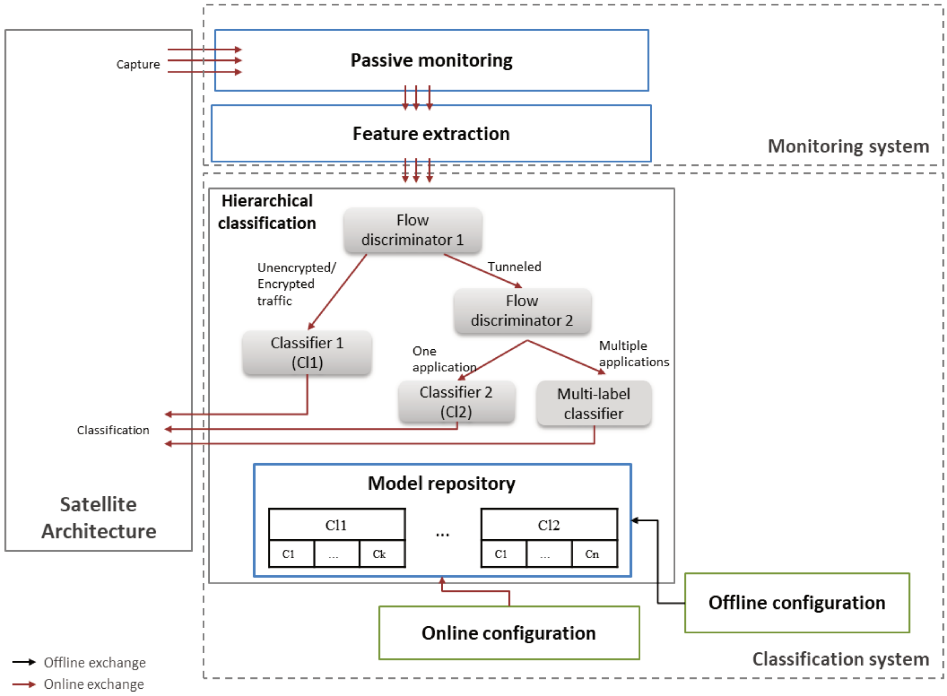


Figure 3.4: Classification framework

Offline configuration component

This component will be in charge of setting all the initial parameters of the classification and online configuration components. Basically, this component will set the primary knowledge of the classification models. In addition, it will define the requirements for monitoring in the Internet point selected, such as the packet length and Inter-Arrival Time (IAT).

Hierarchical classification component

The classification prediction is computed by this component to classify Internet traffic. This system will receive inline Internet flows that will be treated in the following order:

- *Flow discriminators*: classification models that separate different types of distinct Internet communications, such as non-encrypted traffic, encrypted traffic and multiplexed traffic.

- *Classifiers*: set of ML classifiers that identify the type of QoS traffic per flow.
- *Multi-label classifier*: a classification strategy to deal with tunneled connections with more than one application simultaneously.

This component will forward the classification performed and relevant information regarding the communication studied. In addition, it will inform the *Online reconfiguration* about possible class distribution changes. The algorithm proposed to perform this task is given in Table 3.2.

Macro algorithm

Input: ST, $t(f)$
Procedure:

1. Evaluate the flow discriminator $d = D1(ST)$
2. If d is not multiplexed, Evaluate the classifier $c(f) = Cl1(ST)$
3. Else,
   3.1. Set tf True
   3.2. Evaluate the flow discriminator $d = D2(ST)$
   3.3. If $d == Unitary$, Evaluate the classifier $c(f) = Cl2(ST)$
   3.4. Else, Perform the multiplexed treatment $c(f) = MLC1(ST)$
4. Deliver results $\{t(f), c(f)\}$ to the PDP

Output: $\{t(f), c(f)\}$

Table 3.2: Macro algorithm for the hierarchical classification component for a new flow

Online reconfiguration component

This additional step will be in charge of monitoring the behavior of the input samples in order to perform updates or upgrades on the classifiers. 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. The steps of this process are listed in Table 3.3. In brief, the Incremental Learning Model (ILM) will indicate when a class of interest is changing its behavior with the value of b. The simplest way to demonstrate this change is when a batch of samples is filled by this class. This phenomenon will be detected by a semi-supervised algorithm that will be presented in Chapter 6. Nonetheless, this approach can be replaced by a retraining process when new data is collected.

3.3.4 Policy based network architecture

This component comprises three main modules: PDP, Resource allocation PEP and QoS Server. The scope of this work only delves into the entries that this architecture requires.

Macro algorithm
Input: ST, *name* either $Cl1$ or $Cl2$ Procedure: 1. Get ILM given *name* 2. Evaluate the ILM $c, b = ILM(ST)$ 3. If b shows an expansion of the class predicted c, • Get the training dataset (X, y) • Train the base classifier C_j • Add C_j to *name* in the model repository in the hierarchical classification Output: C_j

Table 3.3: Macro algorithm for the *Online reconfiguration* component

The basic interaction between these components begins with a *request* of a policy decision sent to the PDP as denoted in Table 3.4. This request is sent by the *Classification system* and includes flow specifications such as a flow tuple $t(f)$ and classification $c(f)$. The decision taken by the PDP depends on the value of c, named in this work QoS class (reviewed in Chapter 4, Section 4.2). The PDP returns a policy decision to the Resource allocation PEP and QoS server. The Resource allocation PEP then enforces the policy decision by appropriately translating it into commands that will be executed allocating appropriated bandwidth to the related terminals. On the other hand, a QoS server can be deployed for each ST and GW component in order to offer local traffic shaping.

Macro algorithm
Input: $request = t(f), c(f)$ Procedure: 1. If PDP received a *request* • Evaluate the PDP rules over *request* • Get the PDP decision d • Send d to the Resource allocation PEP and QoS Server • Execute PDP's decisions Output:

Table 3.4: Pseudo code for the *PBN* component

In this section, we have informally introduced the needs and functionalities of our ML-based classification system. Hence, the formal modeling of this architecture is proposed in the next section.

3.4 Requirement analysis with Capella

In this section, we specify the logical interaction between all the actors involved in our architecture. We also identify the most important requirements and the most

important functions guiding the design of our solution. Therefore, in order to correctly proceed, several assumptions are stated for modeling the system. These are:

- Several components of Satellite communications are omitted from the model in order to make an abstraction in client-server communications. For instance, the network backbone is transparent to this model due to it is disconnected to the QoS management of the Satellite System.
- In the same manner, only the network functions of interest are taken into account, such as the PDP, Resource allocation PEP and QoS server for improving the QoS.

Given these assumptions, we start modeling the problem and the proposed solution.

3.4.1 Operational analysis

In this section, we try to answer what the users of the system need to accomplish globally. Therefore, we define one by one each of the elements of the operational analysis as follows.

Operational Entities/Actors and Capabilities

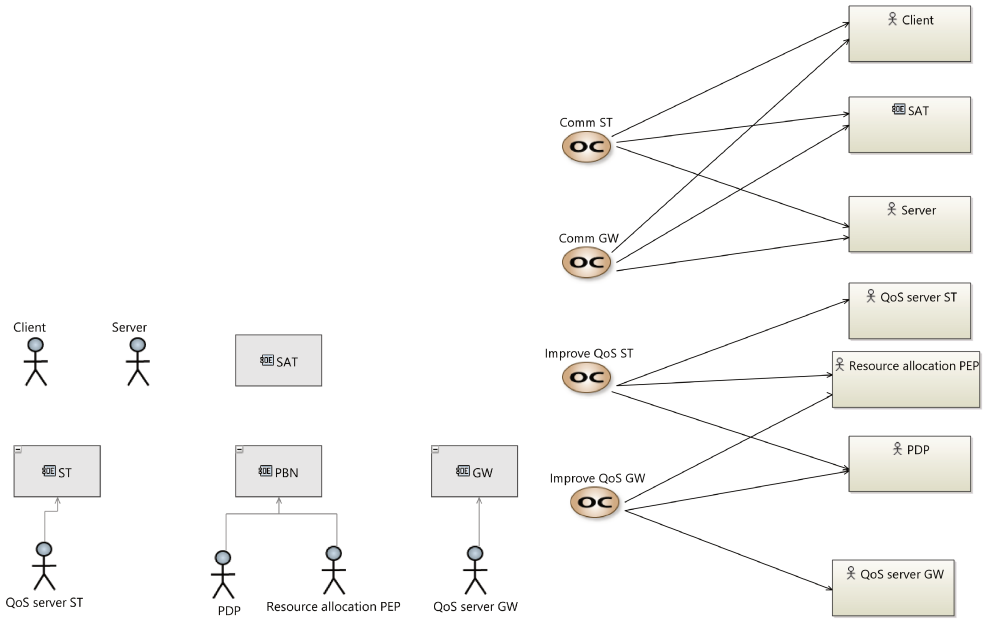
The first step is to define the operational entities and actors. These elements interact along the modeling process to achieve the objectives defined. The entities and actors are shown in Figure 3.5(a). The entities agree with those in the reference models presented in the figures 2.1 and 2.2: Satellite (SAT), Satellite Terminal (ST), Satellite Gateway (SW), and Policy-based Network (PBN). On the other hand, the actors are those involved in a communication session (Client and Server), and network functions (PDP, Resource allocation PEP and QoS servers).

On the other hand, the main capabilities that our system should be able to fulfill are illustrated in Figure 3.5(b). The main capabilities are the communication between client-server and the improvement of QoS. In this sense, communications between clients and servers, either through the GW or the ST, are expected (*Comm GW* and *Comm ST*). In the same manner, QoS improvement is provisioned for communications in the GW and ST. In the figure, the entities/actors involved in each capability are shown.

It is worth mentioning that the ability to offer client-server communications in the GW is equivalent to the ST. In the same manner, QoS management in the ST and GW are equivalent in terms of the operational analysis. Therefore, we present as follows, the model of these two scenarios altogether.

Scenario 1: Communication GW/ST

We will present the general activities to achieve a communication between a client and a server (denoted as a session). Only the handshake or connection establishment is shown. In addition, intermediary network components are omitted to understand and to simplify the model of a Satellite communication. The diagram in Figure 3.6



(a) Entities and Actors (b) Capabilities

Figure 3.5: Operational Entities/Actors and Capabilities.

shows the activity interaction between the client and the server. The main functions are displayed.

Basically, this model abstraction presents the client-server handshake in a Satellite communication. The client traffic is supposedly passing through the ST, SAT, and GW to arrive at the server, while the server inversely follows the same path to finally perform the activity *Sharing/Receiving*. To this extent, different communication patterns can appear (e.g. client-client communications); however, for QoS management, what is important is the traffic captured whether in the ST or the GW.

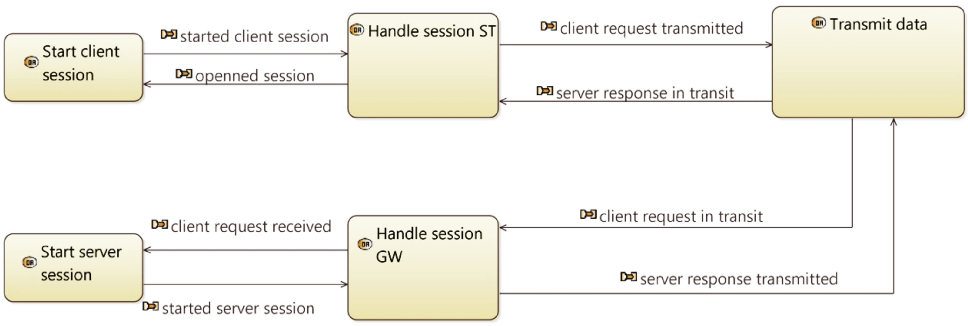


Figure 3.6: Operational activity interaction of a client-server communication in the GW/ST.

Scenario 2: QoS management in the GW/ST

An opened session is monitored to perform a classification step. Depending on the classification results, some decisions will be taken and executed to shape the traffic.

The operational activity interaction of this scenario is shown in Figure 3.7. In brief, the main objective is to monitor client-server traffic from the GW/ST in order to perform traffic classification. The classification result is translated to a QoS class, this QoS class helps to define decisions in the PDP. The PDP transmits the decisions to the Resource allocation PEP and QoS servers that will translate them into actions (for instance, reduce or increase the bandwidth allocation).

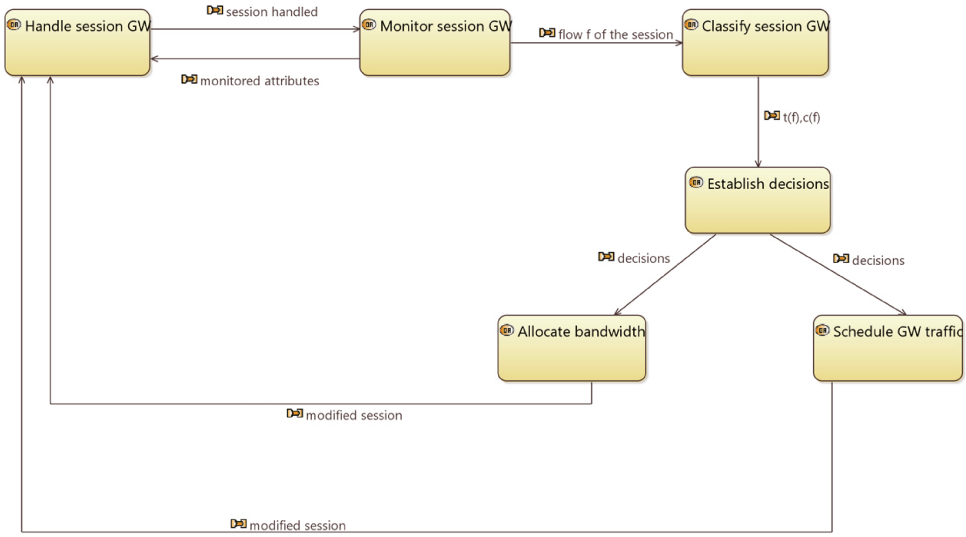


Figure 3.7: Operational activity iteration of the QoS management in the GW/ST.

As it was previously mentioned, the GW and ST capabilities are equivalent. Conversely, additional operational models are unneeded.

Operational context

The operational architecture or product of the operational analysis is given in Figure 3.8. This figure summarizes the activities and interactions between the actors and entities. The GW and ST perform equivalent operational activities. In the center, the client and server actors can eventually open communication sessions passing through the GW and the ST. Meanwhile, the ST and GW will handle the transmission to the SAT. Once a communication session is started, monitoring and classification tasks will be performed in the GW and the ST. The classification result is managed by the PBN entity.

To conclude, the scenario to be developed is the QoS management with ML. The communication between client and server is already well-known and deployed in Satellite Architectures. However, we illustrated it in this architecture as additional information to help to understand the problem. In the following sections, we perform the system analysis of the model, taking into account the capability of interest (*Improve QoS*).

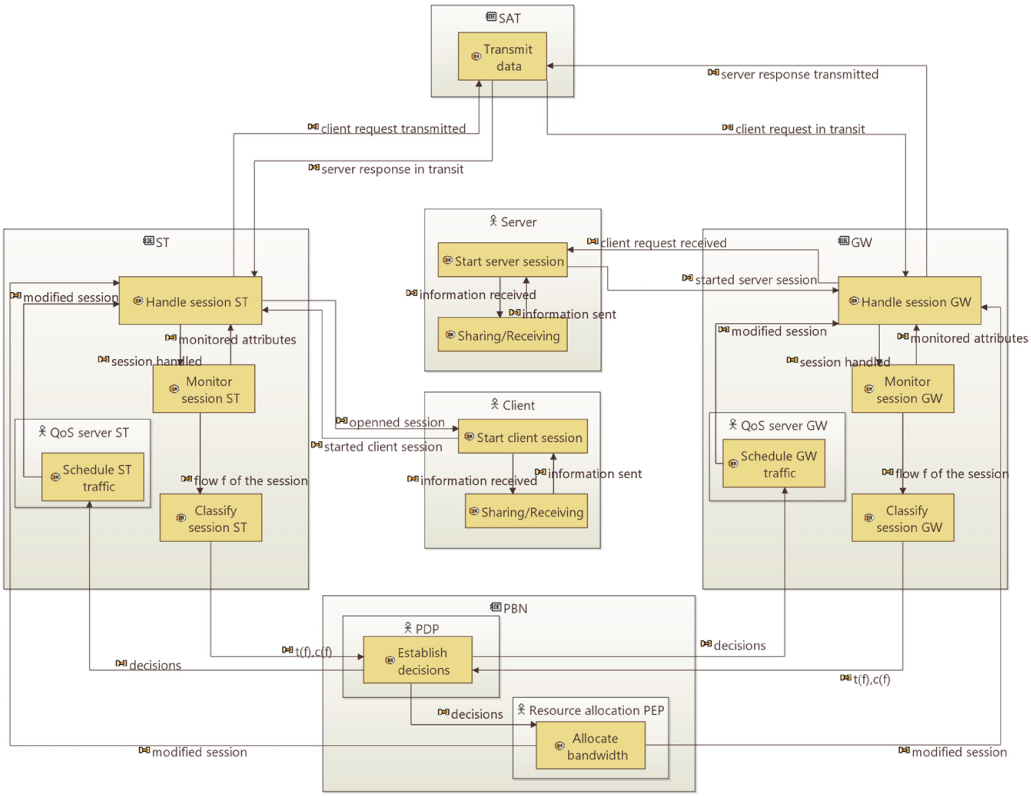


Figure 3.8: Operational context.

3.4.2 System analysis

We now know what the users need. At this stage, with the system analysis, we develop how the system will work to accomplish the users' needs.

Missions and Capabilities

The principal mission of the system is to improve QoS through traffic classification. In order to do so, our system has to be equipped with monitoring, classification and management capabilities. Figure 3.9 shows the missions and capabilities proposed.

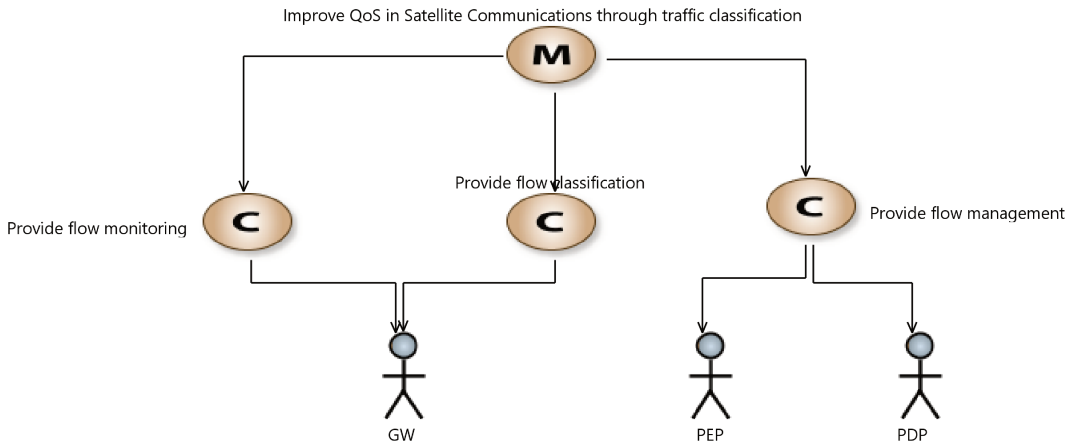


Figure 3.9: Missions and capabilities of the system analysis.

System architecture composite

Firstly, we present a global architecture with composite functions that will allow us to achieve the mission. These functions are depicted in Figure 3.10 and summarized as follows.

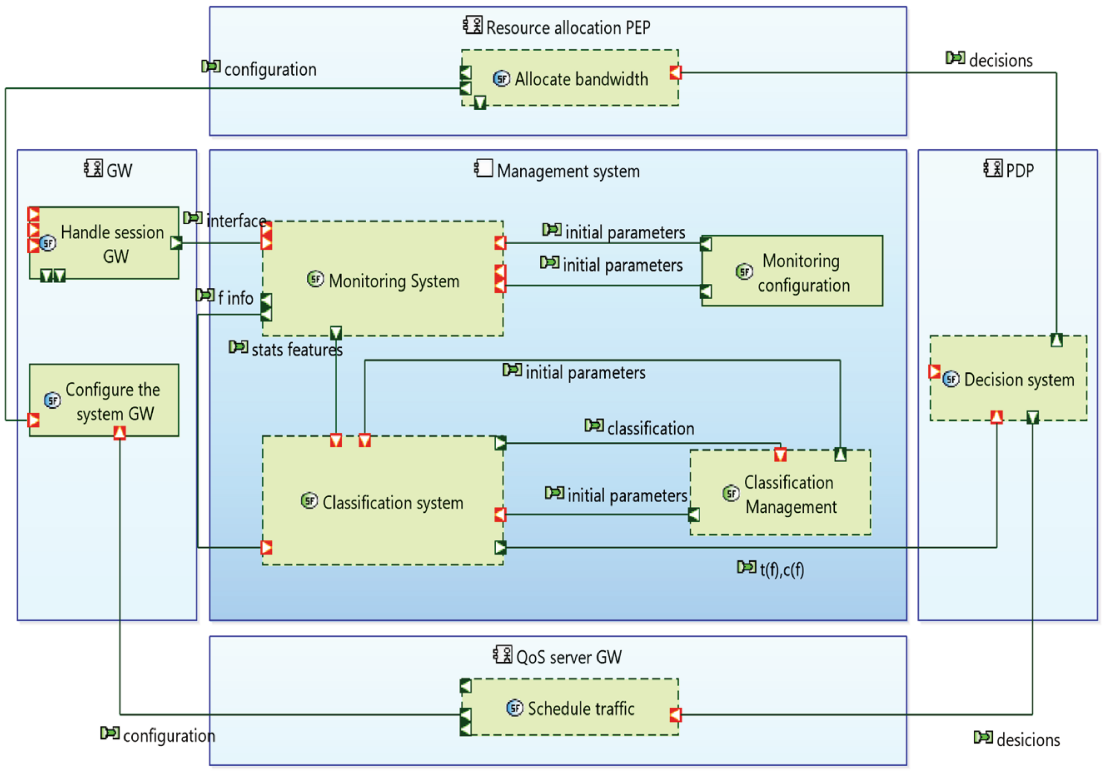


Figure 3.10: System architecture.

- Management system entity: As a central control system, it will coordinate some operations to classify Internet traffic. In this sense, four functions are disposed of:
  - Monitoring system: it will be in charge of Internet traffic monitoring
  - Classification system: It will be in charge of deploying the classification system as in Section 3.3.3
  - Classification management: it handles configurations of the *Classification System*
  - Monitor Configuration: it handles configurations of the *Monitoring System*
- PDP entity: its main function is to process classification results to define suitable decisions. This accomplished by the *Decision system*.
- Resource allocation PEP entity: The decisions provided by the PDP are executed by the *Allocate bandwidth* function
- QoS server GW entity: Similar to the previous case, the PDP will order to take some actions that will be performed by the *Schedule traffic* function.

- GW entity:
  - Handle session: it is a functional port where the Internet traffic can be monitored
  - Configure the system GW: This function makes an abstraction of several components that perform configurations over the GW.

Figure 3.10 is an initial system architecture that contains the main functions to be developed in further system analysis. These composite functions are detailed as follows.

Monitoring and Classification: system functions

Figure 3.11 presents the functional exchanges between the *Monitoring* and *Classification* system. In order to show a simplified diagram, we only show the case when the data flows go into one discriminator and classifier (let us say $D1$ and $Cl1$). In the figure, two main functional chains are remarked: traffic monitoring and traffic classification. Traffic monitoring is to follow these steps:

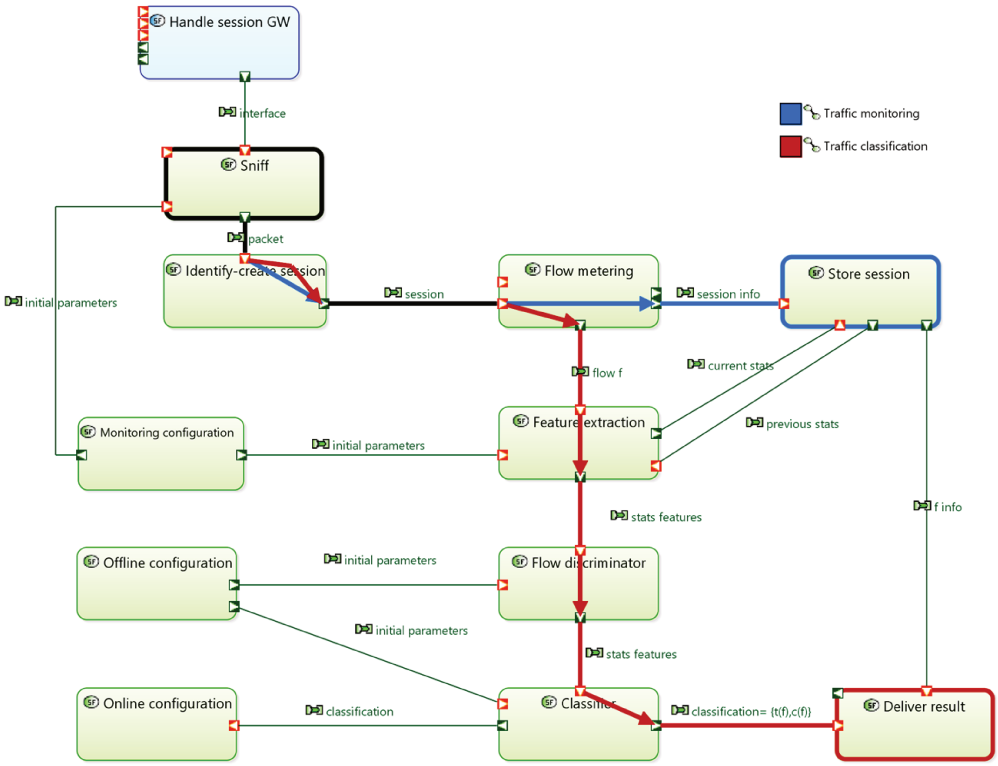


Figure 3.11: Monitoring and Classification system function.

- Sniff a communication session in the GW/ST
- Allocate the packets sniffed in a new or existing session
- Perform a metering process assigning the internet packets to their respective flows

- Some information about the flow/session are stored for further analysis

On the other hand, a traffic classification process for a flow can be viewed as the following sequence of steps:

- Execute the feature extraction and update the flows. Previous flow' states are used for computing the features
- Pass the flows' statistics through a *Flow discriminator* classifier to define which classifier will be used.
- Evaluate the flows' statistics by the *Classifier* and deliver the result to other system functions.
- Perform online or offline configurations when needed. The online and offline configuration of the classifiers can occur in parallel. However, the offline configuration is normally executed to initialize the system for the first time.

QoS management: system function

In this section, we present the functional activities to provide flow management. In Figure 3.12, the result coming from the classification system is taken to perform the following operations:

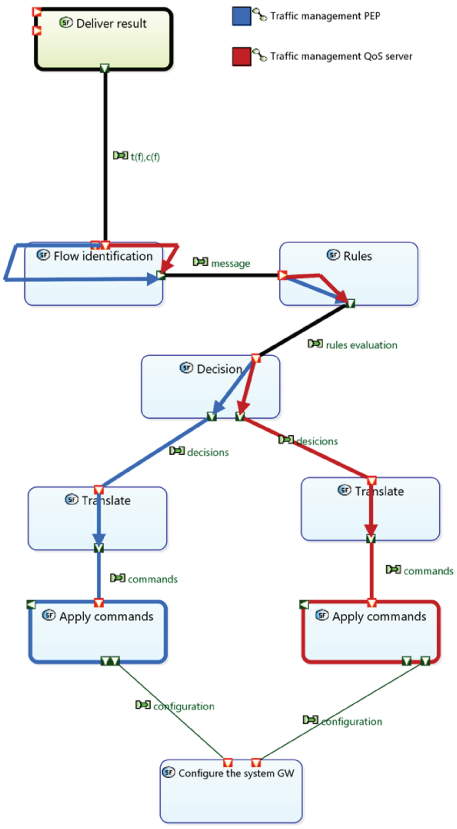


Figure 3.12: QoS management system function.

- Identify this flow and perform a set of rules. The evaluation of the rules gives, as a result, a set of decisions.
- Translate these decisions into commands with the Resource allocation PEP and QoS servers, to later apply them.

The result of these activities is the shaping of the traffic according to the classification results and the PDP previously defined rules.

Final System architecture

We allocate each of the functions in the sections 3.4.2 and 3.4.2 to their respective composite functions. These functions are displayed in the System Architecture, in Figure 3.13, standing up the functional chains defined in the previous sections.

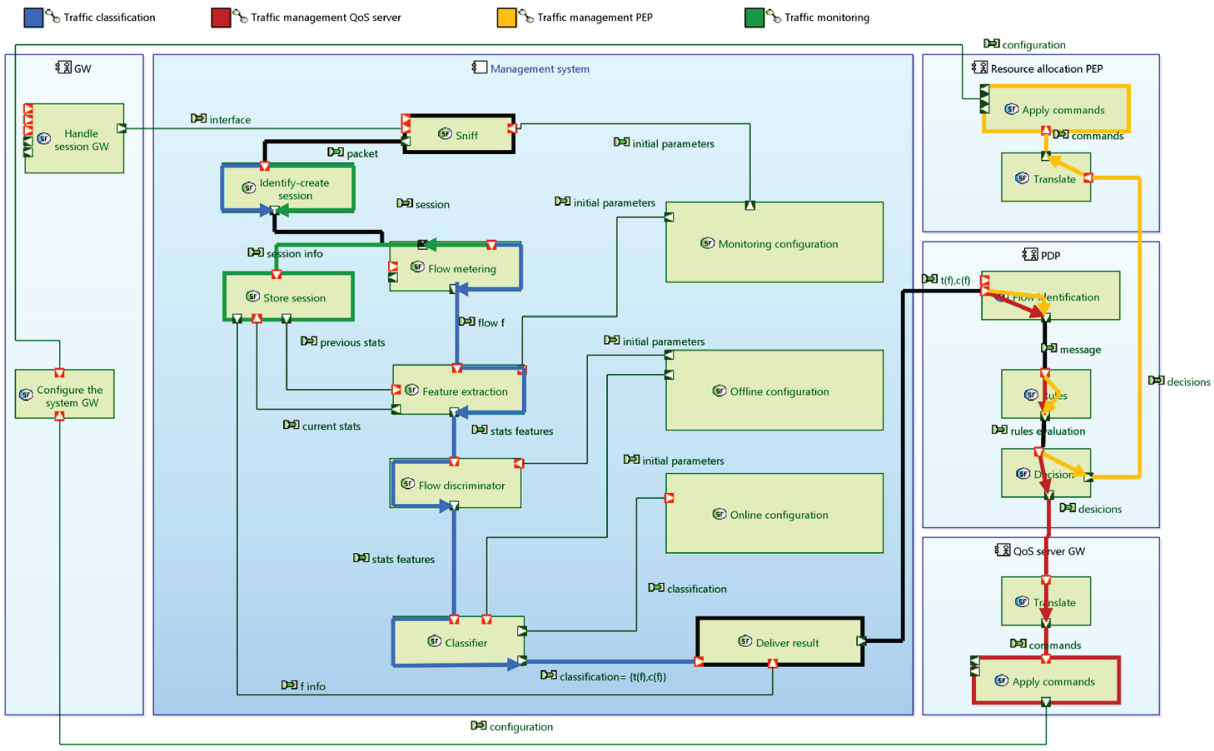


Figure 3.13: System architecture.

3.5 Summary

In this chapter, we follow an MBSE methodology in order to explicitly specify the classification system's requirements and to design the structural and behavioral dimensions of our solution. Conversely, we present a summary of those needs, and where they will be developed along with the following chapters.

- Offline configuration component: in other to initialize the primary knowledge of the ML solution, we need a large historical Internet network data. This data has to represent all the conditions that the ML solution might face in a Satellite Architecture. This requirement will be theoretically and practically developed in Chapter 4.
- Feature extraction: a correct representation of the Internet streams is necessary to perform correct Internet traffic classifications. This aspect will be presented in Chapter 5.
- Flow discriminator: This block emerges from the need of treating protocol communications differently. In this sense, a dedicated classifier will be designed for this aim in Chapter 6.
- Multi-label classifier: the tunneled traffic will be treated by the multi-label classification approach in Chapter 6.
- Classifier and Online configuration component: these elements will allow us to have a self-reconfiguring solution coping with one of the challenges previously detected, the evolution of the network. This need is also taken into account by our work, and it will be detailed in Chapter 6.

To summarize, the modeling of the system needs allows us to focus on the design of certain components. These needs will be sustained by scientific and experimental results that will conceive the implementation of our solution. Such implementation will be based on the two lower layers of the ARCADIA methodology: Logical Architecture and Physical Architecture (both in Chapter 7). In the next Chapter, we fulfill the first requirement that consists of selecting and collecting historical data set to build the ML-based classification solution.

Chapter 4

Characterizing and collecting Internet traffic

Contents

4.1 Introduction **51**
4.2 QoS categories in Internet traffic **53**
4.3 Virtual Private Network **54**
4.4 Public available datasets **55**
4.5 Internet traffic from an emulated network on a cloud platform **57**
4.5.1 User agent 58
4.5.2 Coordination agent 59
4.5.3 Traffic collection 60
4.6 Satellite dataset **61**
4.6.1 User agent 62
4.6.2 Scenarios 62
4.6.3 Data collection 62
4.6.4 Results 63
4.7 Preliminary analysis of the datasets **64**
4.8 Summary **65**

4.1 Introduction

Historical data is a very important source of knowledge for building ML solutions. A rich and complete set of observations regarding a problem can improve the performance and generalization of the ML models. In Internet traffic classification, a data collection process needs to be established to correctly create this historical data. The overall structure of the data collection procedure takes the form of Figure 4.1. In this figure, an abstraction of three levels is given in the following order : i) the network environment that is defined by the conditions in which the traffic is triggered, such as real, generated or emulated, ii) the Internet network itself, and iii) the data collection procedure, which refers to how network packets can be treated. In the figure, we can see in the first level at the left how the Internet traffic is captured either by real

interactions between users and servers, by artificial Internet traffic generated from scripts, or by imitating the user behavior.

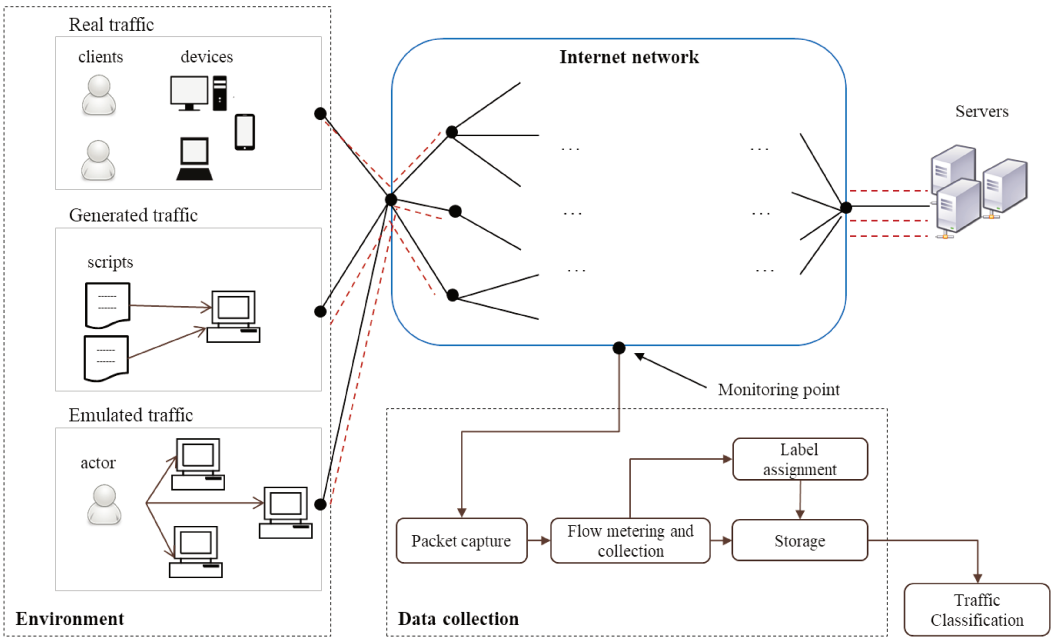


Figure 4.1: Main data collection components for traffic classification

Let us remark that the real Internet traffic available for the community does not fulfill one of the most important requirements to apply ML; that is the ground truth labels of the Internet traces are unknown. For instance, the reader can refer to the MAWI repository [178] that gathers Internet traces at different monitoring points in a backbone network. The deficiency of this and many other similar repositories [170, 24, 39, 32] is that the labeling process is not performed; therefore, a classification process is hard to achieve. On one hand, unsupervised techniques can be viewed as the logical path to take; however, the labeling problem is still a deficiency due to the unsupervised approach that needs to be validated with the ground truth information. On the other hand, DPI software can be used to label the Internet traces; despite that, encrypted traffic will be left aside.

In this section, we present the data used to conduct this investigation. We propose to use two publicly available data sets; moreover, we generated our own data in two different platforms. First of all, Section 4.2 introduces the QoS categories in Internet traffic, which in turn will be largely used for the production of the Internet network data and for the classification solution. On the other hand, we give a brief description of a VPN technology that will be used for generating tunneled communications in Section 4.3. Following, we present the public datasets used to validate our proposal in Section 4.4. In addition, we propose a prototype system on the cloud to generate Internet traffic traces in Section 4.5. In a greater scale, the former experiments were reproduced in a Satellite architecture in Section 4.6 by **Thales** under a collaborative project "R&T CNES: Application du Machine Learning au Satcom" [42].

4.2 QoS categories in Internet traffic

The advent of new network technologies, applications and grown of the Internet service promote to offer better or customize services to certain users; such a paradigm leads to improve the QoS. We already stated that the idea behind improving the QoS is to provide an adequate QoE from the user's point of view [162]. This is achieved by fulfilling some performance requirements for different services and applications [85]. These requirements are mainly correlated to the packet loss rate and delivery delay; said differently, the QoE can be measured mainly by the delay and the information loss perceived by the user. In reference to our discussion, the work in [84] presents a reference model where the QoS categories are defined and organized by their performance requirements (delay and information loss) from the user's point of view. This study set a clear division between the QoS classes that can and cannot tolerate these error conditions. Let us examine one of many QoS categorizations in Figure 4.2. In this figure, two main groups are defined as follows,

- **Voice and video** comprise the applications of VoIP, Interactive video and Video streaming. On one hand, according to [84], VoIP and Interactive video applications are very sensitive to delivery delays, to be specific they can tolerate around 100ms. On the other hand, Video streaming has lower requirements in this aspect with 10s of tolerance. Regarding packet loss, [84] states that VoIP and Interactive video can permit more packet loss than the Video streaming class accounting for 3% for the first one and 1% for the last one.

- **Data** is composed of a diversity of applications that we will briefly introduce. The Chat category states for the interactive messaging, and Client/server transactions for interactions like Web Browsing. These two categories do not present high requirement performance. Bulk data includes FTP transfers, e-mail, database synchronization, among others; which have less priority in terms of QoS requirements as well.

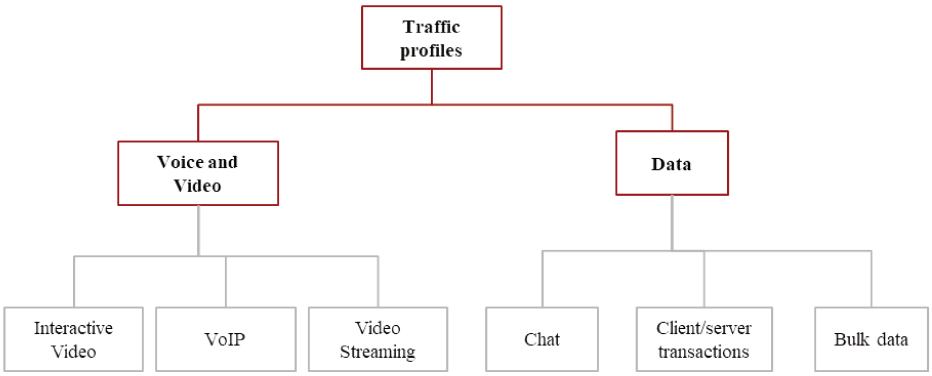


Figure 4.2: QoS classes for Internet traffic classification.

This figure leaves aside some categories that are less important for QoS management, but that still can be found on the Internet (e.g., Peer to Peer (P2P)). Different categorizations exist in the literature, as an example, the DPI solutions define their

own application categorization which varies and can reach until at least fifteen classes [22]. Nonetheless, for the objectives of this investigation, what is important to retain is the most critical QoS classes: Voice and video, voice and video messaging and streaming audio and video (that can also be considered within P2P comunications).

To complement this subject, we study the Internet applications within the categorization presented, that should be detected by our solution along with the most vulnerable/critical and primordial applications. In this sense, it is worth acknowledging the worldwide application usage to know what to expect in the Internet network. So then, [33] forecasts the application of traffic growth from 2015 to 2021. As we could expect, the study found that Internet video will dominate most of the network traffic. This QoS category includes IP Video on Demand (VoD), video-streamed gaming and video conferencing, among others. However, other classes such as file sharing and Web/Data will also account for an important amount of consumption. In summary, the consumption percentage of Internet video varies from 51%-67.4%, Web/data 18%-11% , file sharing 8%-3% and gaming 1%-4%, from 2013 to 2018 respectively. We found that Internet video can be divided into some many others as described by [33]; currently, we can find: i) video delivered through the Internet to a TV screen, ii) video messages or calling delivered on fixed Internet initiated by smartphones, non-smartphones, and tablets, and iii) Peer-to-peer TV (excluding P2P video downloads) and live television streaming over the Internet, among others. On the other hand, in terms of the most used websites, the survey conducted by [4] shows that the top ten are Google, YouTube, Facebook, Baidu, Wikipedia, Qq, Taobao, Tmall, Yahoo and Amazon.

It is worth mentioning that one of the interests of this investigation is to provide a fast classification response, in particular to QoS classes not tolerant of delay. Internet video surfaces as one of the most used and sensitive applications. We will notice along with the development of this project that all the QoS classes are important, making a particular emphasis on the most critical ones (based mainly on critic near real-time constraints).

4.3 Virtual Private Network

The Virtual Private Network (VPN) technology has been largely used nowadays with different applications. One of the most popular applications is among enterprises, which use such technology for secure communications between their employees. On the other hand, many cyber-attacks have been carried by using VPNs; due to, among other reasons, within a VPN connection the data is encrypted, and the actual IP is hidden disabling the user location identification. Basically, a VPN assures a secure connection between a local and a public network. This private network is built by establishing a virtual point-to-point connection through the use of dedicated connections, virtual tunneling protocols, or traffic encryption.

A VPN assures a virtual secure isolated connection over the public Internet through the construction of a tunnel. This tunnel encrypts the source/ destination and message between two endpoints. The secure tunnel is created by using different

encryption protocols. Internet traffic classification over a VPN connection is hard to achieve due to, among other things, the modifications applied over the flows and packets, such as flow multiplexing, packet fragmentation, padding and added control packets. The work in [48] surveys the former techniques and classified them into the following groups:

- Encryption
- Randomization
- Mimicry
- Tunneling

VPNs are normally categorized in the last group, however, there are some technologies that use a mix of these obfuscation techniques, the reader can refer to Obfsproxy and Tor. This study uses OpenVPN as a reference tool to build tunneled connections [133]. OpenVPN is widely used and deployed over the network thanks to its low-cost feature. One advantage of this tool is that only uses tunneling obfuscation technique. OpenVPN normally uses SSL to create the tunnels [163].

Internet classification over VPNs is of great importance due to its common presence in the Internet network. DPI solutions only achieve to detect the VPN connection through the ports and protocols. However, they are unable to identify the type of application(s) within the tunnel. In constrast, ML solutions can find ways to classify traffic based on this technology. In general, the first step is to isolate the VPN connections from the rest of the traffic in order to perform a specialized treatment [6, 77]. We notice that the unitary scenarios (where only one application is within the tunnel) can be treated by the classical classification approaches [119, 23, 174]. In the case of multiplexed traffic (more than one application within the tunnel), most of the papers are focused on detecting only one QoS application of interest [49, 174, 184]. In this sense, particular class behavior (such as codecs patterns in VoIP applications [160]) allows making some assumptions to obtain a fair classification.

In our study, we will try to get tunneled traffic in order to design adequate classification techniques. In the next section, we describe the public available datasets found to design and evaluate our classification solution.

4.4 Public available datasets

Two complete datasets, one with non-encrypted data and another one with encrypted data, were selected for the development of our approach. These datasets are described as follows.

- PAM : the authors in [22] created an emulated environment that allowed them to acquire complete flows from several end-to-end communications. They used 4 hardware machines, 2 with Windows 7 and 2 with Ubuntu, plus 3 virtual machines with Windows 7, Windows XP, and Ubuntu, as data generating stations. A server machine was used for data storage. The Volunteer-Based System

(VBS) was used to collect the information describing the flows, such as start time of the flow, the number of packets contained by the flow, local and remote IP addresses, local and remote ports, transport layer protocol, along with detailed information about each packet.

- VPN-NonVPN: the authors in [49] propose to launch the most common applications in the Internet network, and to reproduce the same experiments encrypting the data under a VPN connection. Therefore, the data counts with four QoS classes (Chat, Streaming, VoIP, Browsing, P2P, File Transfer, and Email) plus the same classes encrypted. The authors captured the complete set of flows for each class, and the sessions were saved in binary files with their respective labels.

Table 4.1 details the number of opened sessions (traffic flows) and number of classes in each dataset. Additionally, these datasets were processed by a DPI tool called nDPI [45], in order to obtain the name of the application, and the category of each flow. In Table 4.2, the application names are listed for the classes of interest such as Browsing, VoIP, and Streaming, among others. It is important to mention that nDPI was not applied for the encrypted data in the last dataset, and their labeling was done using the file identifier. For both datasets, some flow sessions were discarded; for instance, classes with samples lower than 20 flows and unknown unlabeled flows.

Name	sessions	classes	Size
Pam	173429	17	22.7 GB
VPN-NonVPN	311686	22	28 GB

Table 4.1: Description of the datasets

These public datasets will allow us to test and validate our classification system, however, they count with some drawbacks listed below:

- The PAM dataset is one of the most complete datasets found. Unfortunately, it was not possible to find the ground truth values of the files. In addition to this, the data is already obsolete in some cases, and encrypted traffic cannot be studied.

- On the other hand, the VPN-nonVPN is more recent with appropriate labeling for the files. However, for VPN connections, only unitary applications are launched within the tunnel. Thus, it cannot be exploited the VPN goodness of multiplexing several applications in one session.

In light of these disadvantages, in the following section, we propose an emulated environment on the cloud that will allow us to create our own Internet traffic data under the desired conditions.

	PAM		VPN-NonVPN	
Class	Flows	Application	Flows	Application
Streaming (S)	5757	Flash, NetFlix, QuickTime, RTMP, Twitch, Vevo, YouTube	474	Flash, NetFlix, QuickTime, RTMP, Spotify, YouTube
File transfer (FT)	1125	-	163	-
Browsing (B)	88806	Amazon, Apple, AppleiTunes, CNN, Cloudflare, GMail, Google, GoogleMaps, HTTP, HTTP Proxy, Instagram, LastFM, MSN, QUIC, SOCKS, SSL, SSL No Cert, Tuenti, Wikipedia, Yahoo, eBay	11764	Amazon, Cloudflare, GMail, Google, HTTP, HTTP Proxy, MSN, QUIC, SSL, SSL No Cert, Unencryped Jabber, Yahoo
P2P	49184	BitTorrent, eDonkey	1833	BitTorrent
Chat	48		59	-
VoIP	4787	CiscoSkinny, H323, IAX, RTCP, RTP, SIP, Skype	831	RTCP, RTP, SIP, Skype, VHUA
Network protocols	11170	-	254197	-
RPC	2105	-	38	-
Mail	-	-	150	-
VPN	76	-	-	-
VPN-Chat	-	-	4038	-
VPN-FT	-	-	918	-
VPN-P2P	-	-	487	Bittorrent
VPN-B	-	-	2501	-
VPN-VoIP	-	-	12010	Facebook, Hangouts, Skype, VoIPbuster
VPN-S	-	-	490	Spotify, Vimeo, YouTube
VPN-Mail	-	-	300	-

Table 4.2: Flow and class distribution of the public datasets PAM and VPN-nonVPN.

4.5 Internet traffic from an emulated network on a cloud platform

In our study, we have designed and developed a small cloud-based platform on Proxmox [144]. Proxmox is an open-source server virtualization management solution, that allows managing clusters, virtual machines, containers, storage and networks with an integrated, easy-to-use web interface. This platform will allow us to emulate normal user behavior on Internet [100, 38]. In order to do so, Figure 4.3 shows the implemented platform. In this figure, we notice that a Proxmox node cluster holds a set of VMs and routers, each dedicated to a specific function.

More specifically, in the figure, the subset 1 is set to hold several VMs, these will emulate the user behavior when using different applications. In this sense, we create a VM for each QoS class (such as VoIP, Streaming, Mail, etc). Within these VMs, an *User agent* will emulate the behavior of real Internet users. Whereas, in an upper level, a *Coordinator agent* will be in charge of controlling how and when the *User agent* will act. The routers will intercept all the traffic from/to the VMs. In this last

component, data collection procedures are deployed to store historical data for the ML solution.

On the other hand, subnet 2 serves as an environment to get encrypted data. The traffic generated in subnet 1 will pass through a VPN configuration to later access to the Internet. This element is disposed of emulating tunneled connections in the Internet backbone. The VPN technology selected was OpenVPN [133].

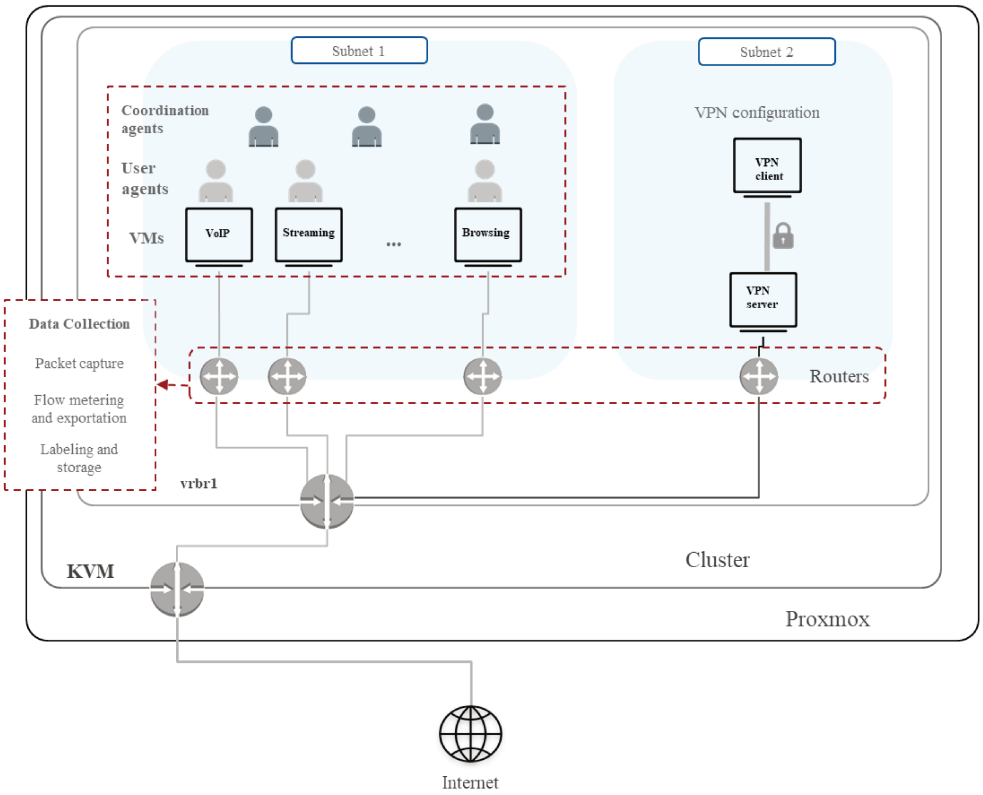


Figure 4.3: Traffic emulation platform proposed.

In our platform, we have created a node cluster based on several physical servers with the following configuration: 8 x Intel(R) Xeon(R) CPU D-1521 2.40GHz (1 Socket), Kernel Version Linux 4.13.13-5-pve, 4x32 Go RAM, 2x2000 Go of hard disk, 1.5Gbps of network bandwidth. Over this cluster, we implemented the platform in Figure 4.3. This emulated network will be a specialized platform for monitoring and classifying inward/backward Internet traffic passing through the routers. We briefly report the most important elements of this implementation as follows.

4.5.1 User agent

The main objective of this agent is to emulate the behavior of the applications and the users, that normally intervene in the Internet network. In each VM, different scenarios will be proposed to emulate the user behavior without the intervention of the human. The main idea is to automatically launch applications in the same or similar way as a real user. These scenarios will cover from non-encrypted to encrypted data. In the non-encrypted case, applications of type Streaming, web browsing, and VoIP will be

reproduced. In the encrypted case, a VPN communication will be established, and the previous applications will pass through it.

In order to implement our agent, we studied several tools that aim at testing web applications. The test is to reproduce the nominal behavior of the clients and repeat it in order to find failures. In our particular case, we will use one of these tools to emulate user behavior instead of using the same principle exposed before. We selected the well-known tool Selenium for this aim [154]. This tool is very flexible and allows easily reproducing user's actions. In Figure 4.4, we show two ways of automating web applications with selenium through a browser: (1) Use Selenium IDE and record a sequence of actions that will be repeated to emulate user's behavior, and (2) inspect the HTML design of a web application and use the Selenium drive to reproduce user actions. In our particular case, we implemented the second option.

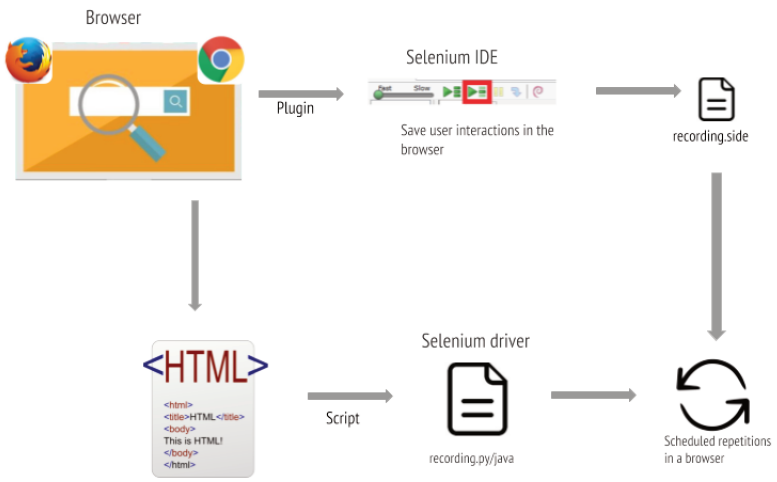


Figure 4.4: Application and user emulation with Selenium.

The user agent is represented by a set of scripts, each of them defining the behavior of an application (e.g. YouTube, Facebook, Netflix, etc.).

4.5.2 Coordination agent

In order to storage a significant number of scenarios, the way in which each application is launched will be coordinated to obtain different variations of themselves. In addition, a scheduling will be proposed to perform the task for a period of time. These tasks will be performed by a *Coordination* agent.

To illustrate the interactions between the *Coordination Agent* with the rest of the actors/entities in the cloud platform, we present in Figure 4.5 an activity diagram example with only one VM. In this figure, we notice that the *Coordination Agent* will start the process, and will set the parameters needed in the *User Agent* and the *GW*. Selenium (denoted as *User Agent* in the image) will be in charge of starting the user's emulation in the VM. The emulation of the user is translated to executing the *scenario i*, which in turn triggers a set of actions. Whereas these actions are triggered, the *User agent* gets into a state where it is emulating the user's behavior, and the browser is executing such actions. On the other hand, the mission of the *GW*

is to start sniffing the traffic coming from the VM and to perform the *Data Collection* process. The *Coordination Agent* will force the end of the process according to the settings established, these settings are normally related to the time and the user agents.

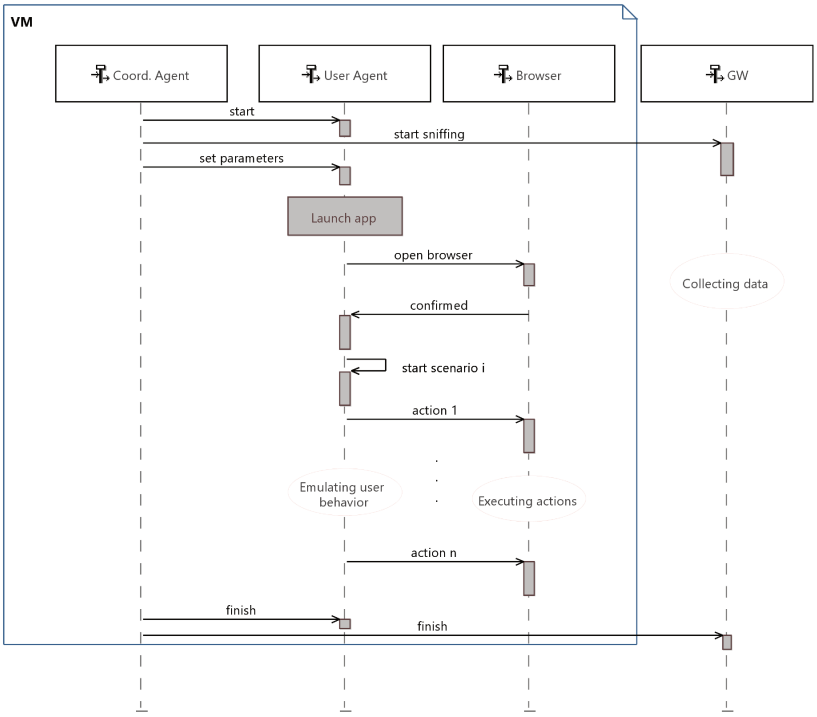


Figure 4.5: The protocol to perform the data collection in the traffic emulation platform proposed.

4.5.3 Traffic collection

Once a network traffic environment is adopted, the following issue to address is how to export the packets, construct IP flows, and assign the application name (if needed) of such flows, in order to finally store or use this data. The work in [78] presents a comprehensive procedure for flow extraction using NetFlow and IPFIX. In brief, the steps needed to perform these steps are:

- Packet capture: The packet capturing step refers to the procedure of extracting the binary data from monitoring points, in this step each packet is considered as a single independent entity. The key aspects to be considered at this stage are the size of the sample and the sampling time.

- Flow metering: the flow metering process aims at aggregating the packets into flows. 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.

- Labeling process: This is an additional component needed for the ML approach to building classifiers. It aims at defining the application name per flow, or any

other identifier needed to enrich the knowledge base, in particular for supervised ML techniques

- Storage: it saves the flows exported. Each of the flows is named with the name of the application launched by the VMs. In this sense, the labeled process will be performed.

The cloud-based Internet traffic emulation platform has served as proof of concept that was later implemented on a greater scale by Thales in a collaborative project in which we actively participate. In this project, we provided the user agent implementations and guidance for the data collection process and the labeling of the Internet data. In the next section, we will give more details about this project and the results yielded.

4.6 Satellite dataset

A data generation process, equivalent to the former one, was developed in a Satellite emulated architecture. This work form part of the Thales' project called "R&T CNES: Application du Machine Learning au Satcom" [43]. The architecture selected to perform such a task is given in Figure 4.6. This figure corresponds to the reference model of a multi-gateway satellite network presented in Section 2. However, a VPN configuration is disposed between the ST and the GW, with the objective to emulate tunneled communications in a Satellite architecture.

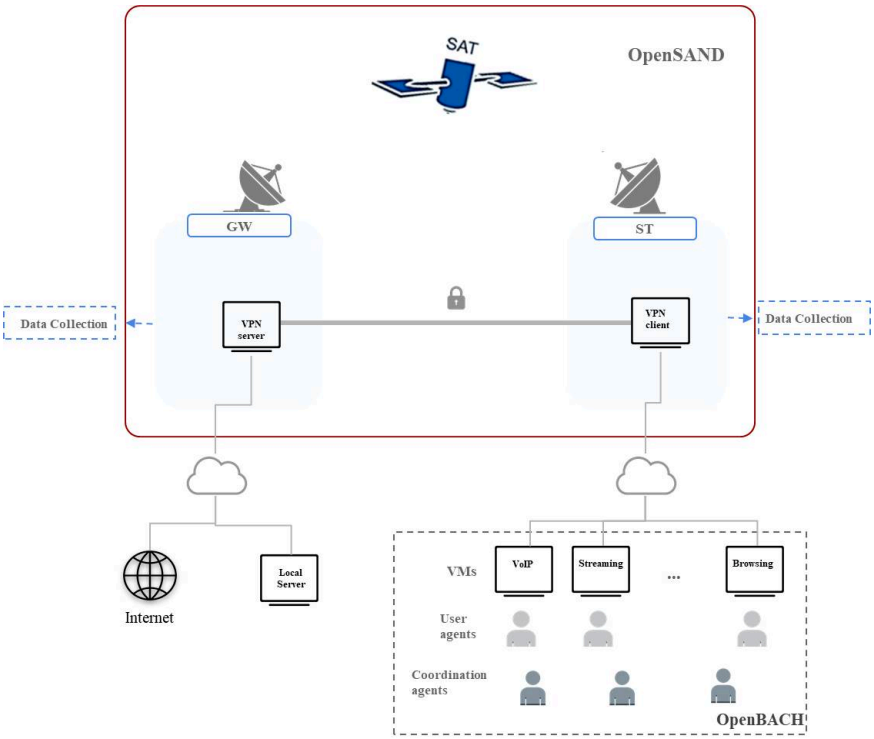


Figure 4.6: Traffic emulation platform proposed in a Satellite Architecture [43].

The architecture in Figure 4.6 is an emulated architecture built with OpenSAND [132]. OpenSAND is a software used to emulate satellite networks. This is composed of elements the network functions and the elements to enforce QoS such as PEP, PDP, and QoS Server. In addition to this, OpenBACH was used to implement the user and coordination agents. OpenBACH is a tool developed to configure, supervise and control networks under test (e.g. terrestrial networks, satellite networks, WAN, LAN, etc.)[131]. We described as follows the variations that this work carried.

4.6.1 User agent

The User agents are equivalent to those introduced in Section 4.5.1. Moreover, more applications were included besides some other variations/extensions that are listed in Table 4.3. Regarding the Coordination Agents, they perform tasks such as those in Section 4.5.2 but more generic and implemented on OpenBACH.

QoS class	Application	Parameter	Value
VoIP- Voice	Skype Facebook Twinkle	Audio track Piste audio codecs	3 pistes randomly launched 3 tracks randomly launched G.711, G.726 and GSM randomly launched
VoIP- Video	Skype Facebook	Video track Video track	3 pistes 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 4.3: Description of the applications launched for the SAT data.

4.6.2 Scenarios

The work also envisaged performing two main scenarios described below.

- Unitary scenarios: only one application at a time is launched.
- Multiplexed scenarios: several applications are launched at the same time. The packet flows are multiplexed by the tunnel.

Additionally, some network configurations were imposed on OpenSand and OpenVPN.

4.6.3 Data collection

The data collection process is equivalent to the previously presented. In this particular case, the difference is where the monitoring process is performed. 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 application launched. However, for the VPN tunnel, a 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.

4.6.4 Results

Figure 4.7 shows the structure of the resulting SAT data. The applications were launched without a VPN (denoted in the figure as None) and with a VPN. In the VPN branch, four protocols were used: *tcp* and *udp* use TCP and UDP as the transport protocol, the packets are only tunneled with OpenVPN. This configuration is commonly used to accelerate the response time. Whereas, *tcp_sec* and *udp_sec* use TCP and UDP with the packets encrypted and tunneled. 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 this particular work, we used only the data captured in the GW with the applications in Table 4.4. 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 4.4, 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 5min up to 15min.

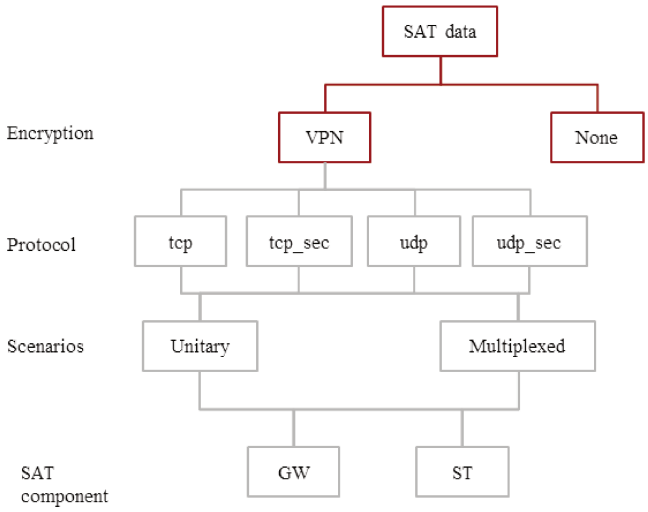


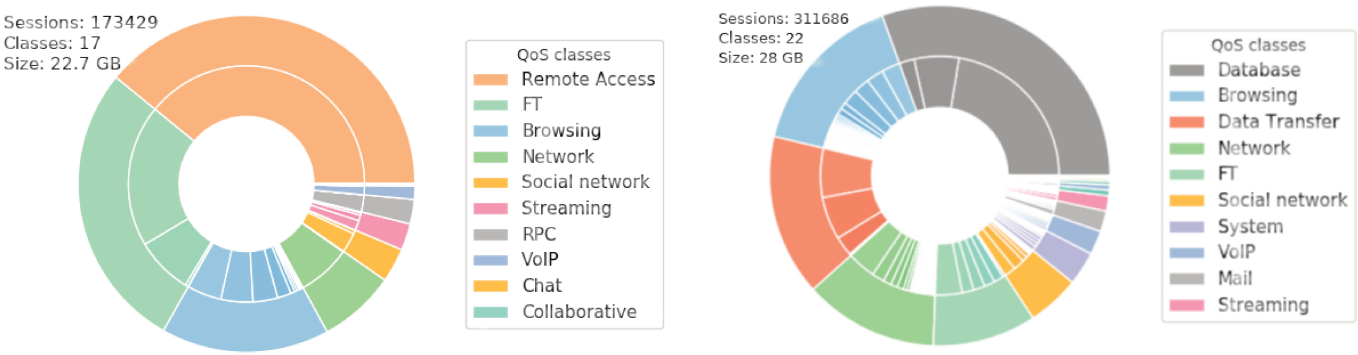
Figure 4.7: Illustration of the experiment launched in the emulated SAT plaform.

		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

Table 4.4: Class, packet and flow distribution of the SAT data in the GW.

4.7 Preliminary analysis of the datasets

What follows is an illustrative analysis of the data used by this research work. The main aim is to know the distribution of Internet QoS classes and applications within the binary files. We took the public datasets PAM and VPN-nonVPN and study the class distribution of their non-encrypted samples. We can notice in Table 4.2 the number of classes and the number of samples for the PAM and VPN-nonVPN. It is noticeable that some classes are more dominant in numbers than the other ones. In order to verify this, we display in Figure 4.8 a pie chart showing the class' distribution. The outer pie represents the QoS classes, while the inner pie the applications available for each QoS class. As it is also apparent in the figure, the amount of applications does not represent realistic numbers; the Internet traffic accounts for a higher rate of applications in each QoS class. Nonetheless, it suffices with evaluating the most used and important ones for a great audience on the Internet. In the same manner, for our SAT dataset, we plot the distribution of the classes in Figure 4.9.



(a) Pam (b) VPN-nonVPN

Figure 4.8: Class and application distribution of the PAM and VPN-nonVPN dataset.

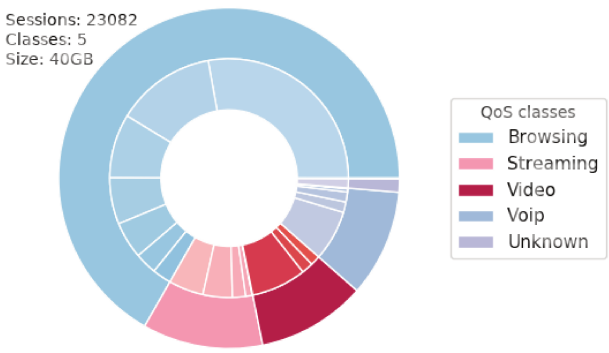


Figure 4.9: Class and application distribution of our SAT dataset.

As regards the PAM dataset, we would like to make some interesting remarks for our study. Unlike the VPN-nonVPN and SAT datasets, this dataset mostly contains non-encrypted traffic and its labeling procedure was performed by a DPI tool. In

Figure 4.8, we can notice that the majority classes is *Remote Access* . This rough coincidence can be associated to the fact that most of the Internet applications were launched in an emulated environment and different control protocols were used to coordinate the data collection.

This study unveils one characteristic of the data called in the ML field as class-imbalance: one or more classes have more samples than another. This case is very harmful to most of the ML classifiers due to it will bias them to learn more from the majority classes than the minority ones. It can also cause over-specification (overfitting) of the minority class, which means that the classifier models too well the training data to the extent that it negatively impacts its performance on new data. This behavior is commonly found in traffic classification as it is exposed in [99, 105]. There are a variety of ML algorithms that deal with the unbalance problem [15]. Basically, the majority of these methods use strategies to balance the data per class. For instance, under-sampling and over-sampling present ways to generate new samples in the classes which are under-represented and over-represented, respectively. There also exist combinations of the mentioned approaches and more advanced algorithms [72]. However, we have to be aware that balancing the data can cause a loss of information. Particularly, in Chapter 6, we will see how this property of the Internet traffic classification will affect our Classification System, and how we intend to deal with it.

4.8 Summary

In this chapter, we analyzed the Internet data that will be repeatedly studied along with this work. We describe the QoS categories that will allow us to prioritize some classes over other ones. The VPN technology was also reviewed in order to show how we will use it and how current works treat it. Two public datasets will allow us to validate our approach along with a data captured based on the work done on a cloud platform.

This chapter shows the second project contribution with the creation of an emulated network to capture and analyze Internet traffic. This initial prototype set on a Proxmox-based cloud platform serves as a guidance to reproduce it on a bigger scale; as it was done in our collaborative project [43], obtaining as a result the SAT data. Even though not all the applications found on the Internet were launched, the most used worldwide were considered. A controlled environment was set to launch those applications within a VPN, the advantages of this setting are that a correctly labeling of the multiplexed data was performed. Later, by studying the data, we conclude that a class-imbalance problem was detected. This characteristic is a key factor to be considered by our ML solution.

Knowing the available data, we can proceed to design our solution. The next chapter will be in charge of defining the feature extraction process for the Internet data stream.

Chapter 5

Feature extraction from communication streams

Contents

5.1 Introduction **67**
5.2 Statistical features for data streams **68**
5.3 Feature extraction for aggregate flows **69**
5.3.1 Statistical based features 70
5.3.2 Statistical feature generation with Linear Regression 71
5.4 Feature extraction for tunneled flows **79**
5.4.1 Slicing tunneled connections 79
5.4.2 Categorizing tunneled packets 81
5.5 Result **82**
5.6 Summary **83**

5.1 Introduction

Once a significant amount of raw Internet data is available, the next step is to find a way to represent or describe this data. In ML, this process is called Feature Extraction. This phase aims at finding attributes that will help to differentiate traffic behaviors. In our work [135], we studied all the possible approaches getting into the conclusion that the statistical-based features are the most reliable and adequate.

For decades, statistical-based features are used in different fields, such as in industrial processes for process and quality control, in medicine for analyzing the evolution of patient treatments or patient system regulatory processes, in business for tracking corporate metrics, stock market analysis and economic forecasting, in image processing and Internet traffic analysis. Particularly, statistical features are highly used for the description of time series data. For instance, Internet traffic presents different characteristics that make statistical features able to proportionate valuable information about the communications: i) ordered events are assessed to start and to close communication sessions, ii) Internet applications use different protocols to perform their operations, iii) such protocols change the way in which the information is exchanged in terms of time and packets length.

Network traffic classification with statistical features is preferred by practitioners in this field due to their simplicity and characterization capabilities [135]. Additionally, this approach does not intrude into the packet content, enabling its use for both non-encrypted and encrypted traffic while respecting privacy. Several approaches have already validated their performance by measuring the classification capabilities of several Machine Learning (ML) models [139, 161]. The authors in [139] tested ten classifiers to get into the conclusion that these features allow the classifier to achieve high accuracy. However, when the flow sequence present abnormalities, the statistical-based features can be affected causing a decrease of the classification performance. For instance, one common abnormality is the packet retransmission in TCP connections. In this case, a partial solution is to treat the retransmission before performing the feature extraction as reported by [79]. The authors proposed a system to discard retransmitted packets, detecting and using only the original packets. To treat abnormalities, feature reduction and selection are deployed to discard noisy statistical-based features. For instance, the works in [188, 56, 55] study the most relevant statistical features for Internet traffic classification. In [56], several Feature Selection techniques are used to obtain the most important features, while a newly proposed method selects the smallest set.

Statistical-based features have been chosen by this thesis project. Particularly, we adapt some statistical-based computations in order to achieve accurate Internet traffic classification for encrypted and unencrypted packets. To do so, let us introduce the statistical feature computation in Section 5.2. Following, in Section 5.3, we formally define the feature extraction process for non-tunneled Internet flows, self-called aggregated flows. Whereas, in Section 5.4, we describe the procedure to correctly extract statistical features over tunneled/multiplexed connections. Section 5.5 gathers the results, and Section 3.5 concludes this chapter.

5.2 Statistical features for data streams

Statistical based features extracted from data streams are in common use in ML. For instance, the classical metrics are the mean and standard deviation of data streams: moreover, in the time domain, metrics such as Root Mean Square (RMS), crest factor, mean, standard deviation, variance and skewness are also widely deployed.

Particularly, the mean of a data stream provides insights of the behavior of the process. This measure in conjunction with the standard deviation are often used in a diversity of problems. For data streams $x = [x_1, x_2, ..., x_n]$, the classical computation of the mean (see Eq. 5.1) is not feasible due to the huge volume of data streams that unables to have a memory of the past observations. In consequence, several alternative approaches have emerged, where the most simple and effective is the moving average described in Eq. 5.2. This mean approximation uses only the current observation and the prior mean value. In data stream processing scenarios, this metric is suitable for feature extraction. This metric has some properties such as i) reduce the online fluctuations on the data to obtain a reliable measure in short and long term, ii) try to smooth out shifts in short sampling window scenarios; on the contrary, in large

sampling windows, it will stress tendencies, and iii) mathematically speaking, the moving average is interpreted as a convolution function, while in signal processing as a low-pass filter. To conclude, its main property is to average online streams for sampling processes in different applications [20].

$$\mu_n = \frac{1}{n}\sum_{i=1}^{n} x_i \tag{5.1}$$

$$\mu_n = \mu_{n-1} + \frac{1}{n}(x_n - \mu_{n-1}) \tag{5.2}$$

Several modifications and improvements to the moving average are found in the literature. For instance, the exponentially weighted mean is a variation of the moving average, where the error is modeled by an exponential function. Eq. 5.3 depicts this modification with α is the parameter of an exponential distribution. The effect of this parameter over the mean computation is to weight the error with an exponential decrease over time [114]. In brief, the weight is mainly introduced to smooth the average. In the same manner, other error approximations can modify the computation of the moving average: for instance, the kernel average, the nearest neighbor and the local linear regression smoother, among others, also try capturing the most appropriate value of the mean [73].

$$\mu_n = \mu_{n-1} + \alpha(x_n - \mu_{n-1}) \tag{5.3}$$

The mean computation in data stream environments can be also used to obtain some other statistical features, where the most closely related is the standard deviation. Such a feature can be obtained through the expectation function which is a generalized version of the mean. For instance, an online approximation of the standard deviation is presented in Eq. 5.5 [94].

$$S_n = S_{n-1} + (x_n - \mu_{n-1})(x_n - \mu_n) \tag{5.4}$$

$$\sigma_n = \sqrt{S_n/n} \tag{5.5}$$

As a final note, this investigation uses the moving average and its derivations (such as the standard deviation) to describe Internet flows. In addition to that, we also propose new features as will be described in the following sections.

5.3 Feature extraction for aggregate flows

In traffic, classification is commonly used the term flow to describe communications between peers. An aggregated flow, according to [34], 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,*

$$F_i = \{H_i, P_i\} \tag{5.6}$$

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,*

$$H_i = (IP_{src}, IP_{dest}, port_{src}, port_{dest}, proto) \tag{5.7}$$

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. One of the main properties of the traffic flows is that they are ordered sequences of packets between the source and destination; this order is defined by communication protocols. For modeling purposes, we use a modification of the previous definition called Bag of Flows (BoFs) [189]. A BoF can be defined as a group of flows that shares the same destination and source IP, and that presents variations in the opened ports of a connection as follows.

$$H_i = (IP_{src}, IP_{dest}, proto) \tag{5.8}$$

This behavior is commonly seen in communication sessions. Several works already tested the advantages of using BoFs to improve the performance using ML classification and clustering approaches [175, 47]. Therefore, once the flow is constituted for Internet communications, one can start computing appropriate statistics as below.

5.3.1 Statistical based features

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) that are unique for certain types of applications, and enable different source applications to be distinguished from each other. Under this assumption, the work in [121] proposes 249 statistical features, which can be extracted from flow network traffic. Traffic characterization based on statistical features has been largely reported [41]. We illustrate the process in Figure 5.1, where a packet is "aggregated" to its corresponding flow; for each flow, a set of properties is extracted. 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.

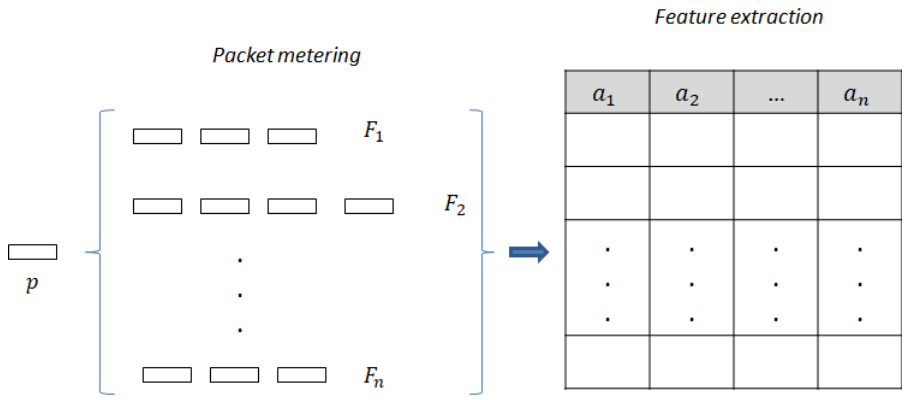


Figure 5.1: Feature extraction overview

Table 5.1 reports the features selected for this study. These features are computed for the bidirectional flows, where F is the bidirectional flow, F_{src} is the forward traffic generated by the source and F_{dst} is the backward traffic generated by the destination in response. In the table, there are also depicted the formulas used to compute such metrics.

The features in Table 5.1 are extracted from the flow level point of view, meaning that no intrusion within the packet datagrams is performed. We can extend to 249 statistic features as proposed by [121]; however, it is noticeable that the extended features are derived from Table 5.1. They are mainly the same statistics coming from packet length in the Data, Data Control/wire, Ethernet, IP, and among others. Additionally, the packets with the flags of Ack, Sack, dSack, Rexmt, FIN/SYN are also considered in [121]; nonetheless, we avoid inspecting these flags to not intrude into the packet content. It has been found in the literature that with only the general statistical features shown in Table 5.1, an accurate flow classification can be achieved [135].

It is worth mentioning that the moving average is directly deployed to envisage an online environment. In order to reinforce the online statistical-based features, we proposed new features that will try finding better approximations to the real mean. This work was published by the authors in [134], and a summary is given as follows.

5.3.2 Statistical feature generation with Linear Regression

Acknowledging that the Internet traffic representation will be centered on the statistical features, we decided to take a deep look at them and find a way to improve their calculations. If we imagine that the Internet communication properties are time-series streams, why not to model their curve shape. For instance, we can propose to model the packet length overtime in an opened session. This brings us to propose a new way to compute the mean of a property A. Similar to the moving average, we propose to compute $\mu = m(A)$ with m a new mathematical estimated relationship. This m function has some parameters that will be set thanks to a Linear Regression (LR) model trained with raw Internet data.

Property	Feature	Formula
Packet length in F, F_{src} and F_{dst}	Minimum	$min(len(p)) \quad \forall p \in F$
	Maximum	$max(len(p)) \quad \forall p \in F$
	Mean	$\frac{1}{m}\sum_{i=0}^{\lvert F\rvert} len(p_i)$
	Standard deviation	$\sigma = \sqrt{\sum_{i=0}^{\lvert F\rvert}(len(p_i) - \mu)}$
Inter-arrival time (iat) F, F_{src} and F_{dst}	Minimum	$min(iat(p_{i-1}, p_i)) \quad \forall p \in F$
	Maximum	$max(iat(p_{i-1}, p_i)) \quad \forall p \in F$
	Mean	$\frac{1}{m}\sum_{i=0}^{\lvert F\rvert} iat(p_{i-1}, p_i)$
	Standard deviation	$\sigma = \sqrt{\sum_{i=1}^{\lvert F\rvert}(iat(p_{i-1}, p_i) - \mu)}$

Table 5.1: Features extracted from the bidirectional flows

A simple LR model is useful for finding relationships between two continuous variables by fitting a linear equation to observed data; with one independent variable, and other dependent variables [61]. For example, the previous mean μ_{t-1} and current property A_t values can help us to estimate the current mean μ_t. Therefore, the main aim is to estimate feature values by modeling the statistical distribution of a set of observations. In this particular case, the modeling will be performed over the moving average. This modeling will be achieved separately for each class.

Figure 5.2 shows the intuition of our approach. The general idea is described below:

- The raw data is labeled for its further separation by class. Each color in Figure 5.2 represents a class.
- A feature modeling process will return as a result of a mathematical relation that will model the feature (moving average) for each class

- The feature models M are created by using an ML algorithm, Linear Regression (LR). The training of such models is performed by using the samples for the class.
- For new incoming observations, all the features models by class are assessed, and the result is given by the one that shows the highest similarity with the past estimation.

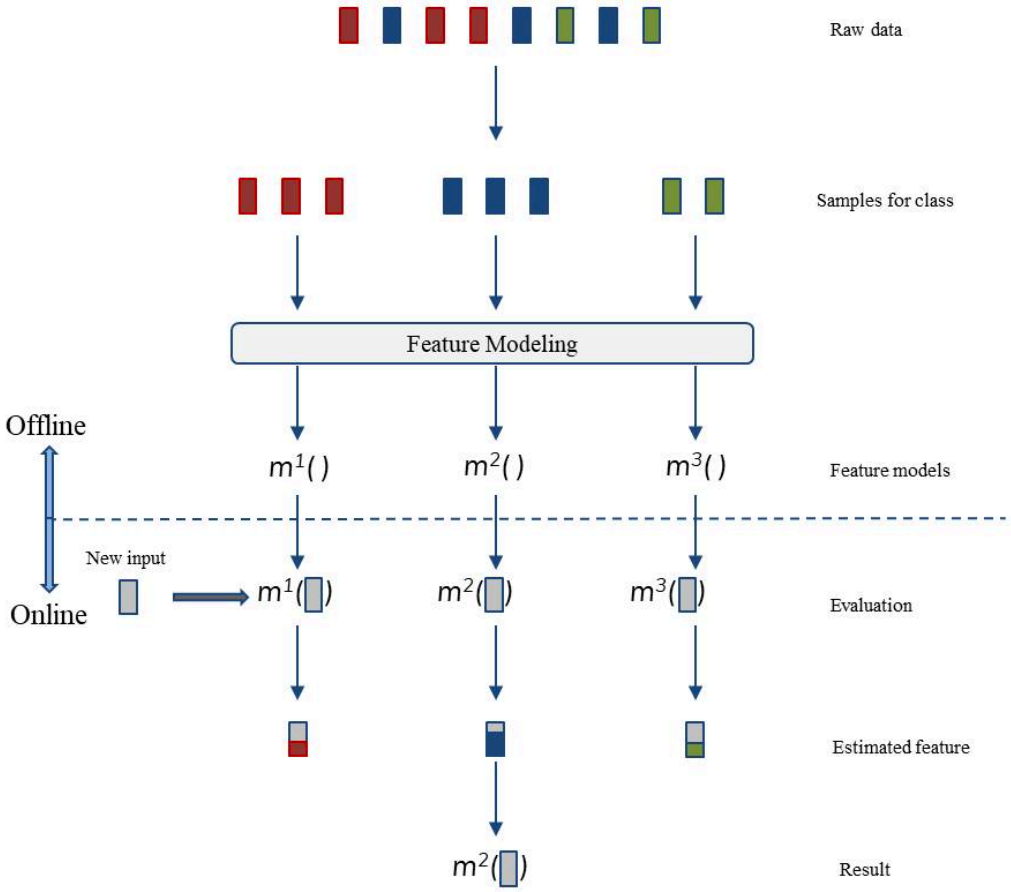


Figure 5.2: Graphical description of the proposal.

To correctly apply this proposal, the following assumptions are set,

Assumption 1. We can differentiate a raw input, which will be interpreted as an ordered sequence of samples in a finite period of time, from another raw input.

Assumption 2. The statistical behavior of a variable is different from class to class, enabling to differentiate them from one another.

Assumption 3. Given the previous statements, it is possible to model statistical feature behaviors for each class separately.

Let $X = [X_1, X_2, ..., X_n]$ be the raw data from a historical dataset with class labels $L = [l_1, l_2, ..., l_n]$, $C = [c_1, c_2, ..., c_p]$ distinct classes identified in L, F the current statistical based features, and A the selected features to be modeled. In Table 5.2, it is depicted the general procedure of our approach. This algorithm starts selecting a subset of raw samples $\{X_s, L_s\}$ from the original raw data $\{X, L\}$. For this particular case, it is preferable to use the training set. The next step is to select the features A

to be estimated; this proposal chose the moving average. Following, the models M are created and a FE process is executed.

In the following sections, the main components of the proposal are detailed.

Macro algorithm

Input: X and F.
Procedure:

% Proposal

1. Select a subset X_s from the original raw data X
2. Select the feature set A to be estimated/modeled from F,

   % Create the feature models

   2.1. Create the models M with X_s using the algorithm in Table 5.3

   % Feature extraction

3. Perform the FE process with X and M using the algorithm in Table 5.4

Output: M

Table 5.2: Pseudo code of the proposal

Create the feature models

The goal is to obtain a model that can estimate the value of a feature. In signal processing, we can found several methods to find an approximate function of a signal where polynomial representations are widely used. In this field, several ML methods have been also studied to perform this task, such as neural networks. In this sense, the proposal here exposed uses the simple Linear Regression model for estimating the behavior of a feature.

A LR model is created for each class $c \in C$ and for each raw sample in $X^c \in X_s$. This means that $|C| \times |X^s|$ LR models will be created; for instance, in the example in Figure 5.2, we will obtain 3×8 models. This model for class c and sample j is described as follows.

Definition 3 *The LR of a raw sample $X_j^c \in X^c$ with $X_j^c = [x_{ji}, ..., x_{jh}]$ is defined as,*

$$m_j^c = g(\Theta_j^c) \tag{5.9}$$

where Θ_j^c is a vector holding the parameters of the LR model, and g the function that specifies the relationship among them. In this particular case, the type of LR model fixed will depend on the number of historical and current raw features that will be used to make the prediction. In the present work, we propose to use the model in Eq. 5.10 for the mean and the model in Eq. 5.11 for the standard deviation.

$$\mu_i^c = m^c(x_i) = \theta_0^c + \theta_1^c \times \mu_{i-1} + \theta_2^c \times x_i \tag{5.10}$$

$$\sigma_i^c = m^c(x_i) = \theta_0^c + \theta_1^c \times \sigma_{i-1} + \theta_2^c \times x_i + \theta_4^c \times \mu_{i-1}^c + \theta_5^c \times \mu_i^c \tag{5.11}$$

where $\mu, \sigma \in A$ are the attributes to be estimated and, for simplicity, the sample $x_{ji} \in X_j^c$ will be denoted as x_i . Therefore, μ_i^c and σ_i^c are the current predicted values for the sample i by the LR models in the equations 5.10 and 5.11.

This LR function is inspired by the moving average and standard deviation parameters described in Section 5.2. We can also compute the training error for its further use, by using Eq. 5.12.

Definition 4 *The training error of m^c will be measured through the mean squared error as follows,*

$$e_i^c = \frac{1}{h}\sum_{i=1}^{h}(f(x_i^c) - m^c(x_i))^2 \tag{5.12}$$

where $f(x)$ is the feature computation using the classical statistical expression, interpreted as the ground truth; and h the amount of the raw values in a time series sample.

Once all the models are constructed ($|C| \times |X^s|$ in total), the next step is to define only one model by class. Such generalization can be achieved by using **Assumption 3**, where we can presume that the model parameters of each raw sample should be similar for the same class. In this sense, we propose a weighted mean computation of the parameters Θ^c by using the following definition.

Definition 5 *The final parameters for the class c with k samples are given by Eq. 6.2, where e_j^c in Eq. 5.12 will weight the impact of each parameter. This weight is scaled from 0 to 1, meaning that values of $\bar{e_j^c}$ close to 0 had a big error when building the LR model; the contrary case will occur if $\bar{e_j^c}$ is close to 1.*

$$\Theta^c = \frac{1}{k}\sum_{i=1}^{k}\bar{e_j^c} \times \Theta_j^c \tag{5.13}$$

Table 5.3 shows the algorithm to create the feature models for each class. The selected raw set X_s is divided into subsets by means of the unique class labels in C; i.e., the rows in X_s that belongs to the class c are in $X^c = [X_1, ..., X_k]$, with k the number of samples labeled with c. This selection is performed in order to improve the inner-class differentiation based on **Assumption 2**. Consequently, an LR model will be built for each class in the set X_s. The LR models are trained by means of the raw time-series samples and their ground truth feature value (for this case, the mean of the series). Following, to obtain the parameters that generalize the model, the normalized error is used to weight the mean computation of Θ. At the end of this process, we will end with a model for each class in the raw data set; defined as $M = [m^1, ..., m^c]$. These models will feed the next process, in order to add one or more attributes a to the feature extraction process.

Macro algorithm

Input: X_s and L_s.
Procedure:

% Feature modeling

1. Separate the raw dataset X_s into subsets X^c, using L_s
2. For X^j in $\mid X^c \mid$,
   - 2.1. For X_i^j in X^j,
     - 2.1.1. Train a regression model m_i with X_i^j and $f(X_i^j)$
     - 2.1.2. Compute the error e_i by using Definition 4
     - 2.1.3. Save the pair e_i and Θ_i in the sets E and Θ, respectively
   - 2.2. end for
   - 2.3. Normalize the errors in E
   - 2.4. Compute the parameters Θ^c through Definition 5, and obtain the final models M with the parameters.
   - 2.5. end for

Output: M

Table 5.3: Pseudo code to build the feature models for each class.

Feature extraction

The result of the previous phase gives a set of feature models for each class. When the feature extraction process is deployed and new incoming features need to be assessed, only one feature estimation has to be given as a result of the ML classifier. The proposal performs such a task using the algorithm in Table 5.4.

The FE process starts with the computation of the original statistical features. Then, our proposal will compute the feature estimated with all the models in M. The selection of the best feature value will be the one that originates the lowest distance between the prior prediction and the current value as it is depicted in Table 5.4. The definitions and formulas that support the algorithm in Table 5.4 are described as follows.

Definition 6 *The error of m_j will be measured through the Euclidean distance between the previous value computation and the current sample value,*

$$d_j^c = [(a_j^c + w) - x_j]^2 \tag{5.14}$$

This error can be affected by shifts in the sample behavior. In this sense, a momentum w is added at this stage for balancing the final decision. The momentum is given by,

$$w = (a_{j-1} - x_j)/t \tag{5.15}$$

where t is the number of samples currently evaluated. Our approach considers a shifting backward with w for improving the current prediction.

Once each of the estimated metric values is computed (evaluating each LR model), the selection of the model will use the following definition.

Definition 7 *The best approximation to the attribute of interest a, is given by the LR model that obtains the lowest distance value, as follows.*

$$a_j = \{a_j^i \in P \mid argmin(d_j^i)\} \tag{5.16}$$

Macro algorithm

Input: X, M, F.
Procedure:

% Feature extraction

1. For X_d in $\mid X \mid$,
   1.1. For x_j in $\mid X_d \mid$,
      1.1.1. Compute the original statistical features S by using the functions in F
      1.1.2. For m^i in M
         1.1.2.1. Compute $a_j^i = m^i([a_{j-1}, x_j])$ by using Equation **??**
         1.1.2.2. Compute $d_i(a_j^i + w, x_j)$ by using Definition 6
         1.1.2.3. Add to the set $P(i) = (a_j^i, d_j^i)$
      1.1.3. end for
      1.1.4. Select the feature value estimation a_j by using Definition 7
      1.1.5. $a_{j-1} = a_j$
      1.1.6. Append to the data set $data(j) = [s_1, s_n, ..., a_j]$

Output: *data*

Table 5.4: Pseudo code to extract the features from the raw data

In order to test the boundaries of our approach, several tests were carried comparing our approach with different moving average approaches. We present as follows a brief analysis of those tests.

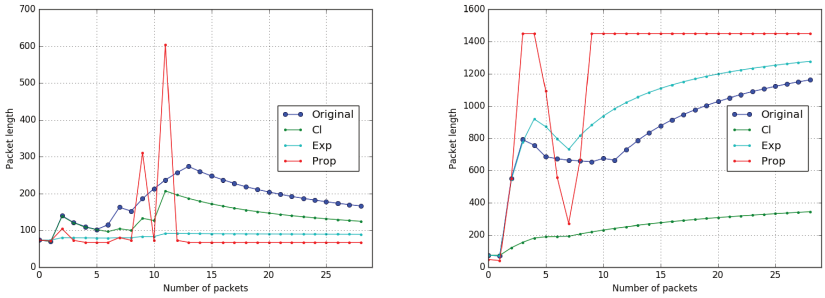
Result analysis

In this section, we will try to interpret the feature estimation given by our proposal. In this sense, we take two flows where one flow belongs to the class *Streaming* in Figure 5.3, and the other to the class *VoIP* in Figure 5.4. We compute the original mean (Original), the simple moving average (Cl), the exponentially weighted moving average (Exp), and finally our proposal (Prop).

In our experiments, the best method to estimate the mean of the packet length is given by the exponentially weighted moving average. However, our approach was outstanding when the sampling window is small. We can try to validate this result based on the design of the proposal. We know that the proposal estimates the mean of the sequence by selecting the most suitable LR model output, in the next iteration the same operation is performed with the same or a different LR model. Therefore, there is not memory about which LR model output was selected as the most suitable in the previous iteration. In this sense, our approach aims at creating a new feature value that tends to their class mean behavior, and not to the raw sample mean. It is for this reason, that in the figures 5.3 and 5.4, the proposal does not follow the raw

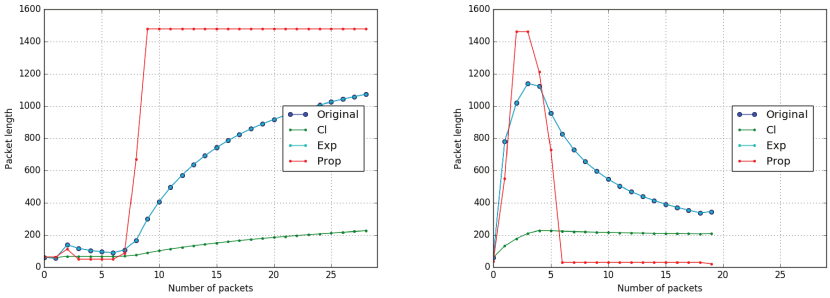
mean value, and starts with trends that can change sharply according to LR model outputs. The latter analysis allows us to infer that our approach does not predict the time-series values, instead, it is oriented to predict the most likely next value regarding each class.

From the traffic classification point of view, in a standard streaming connection, the communication protocol starts with the client making a request and if the server agrees, this last one will send most of the session workload at its maximum capacity (packet lengths with 1500 bytes). Figure 5.3(a) and Figure 5.3(b) are reflecting this behavior. A different case occurs with VoIP session, where both parties can interact in the same way interchanging their roles, as we can notice in the figures 5.4(a) and 5.4(b).



(a) (b)

Figure 5.3: (a) Statistical feature values computed for a packet sequence in the client with the Streaming class. (b) Statistical feature values computed for a packet sequence in the server with the Streaming class.



(a) (b)

Figure 5.4: (a) Statistical feature values computed for a packet sequence in the client with the VoIP class. (b) Statistical feature values computed for a packet sequence in the client with the VoIP class.

The feature models will aim at supporting classical statistical-based computation, and to help the inner-class discrimination. The results showed promising results, promoting the use of these features in our study. We encourage the interested reader to refer to [134], where more experimental details and results are found. In the next section, we describe the statistical-based features extracted from tunneled connections.

The last remains a challenge in this field due to the difficulty of describing multiple sessions encrypted and multiplexed in a tunnel.

5.4 Feature extraction for tunneled flows

Let us move forward to another type of representation, that is 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 5.5.

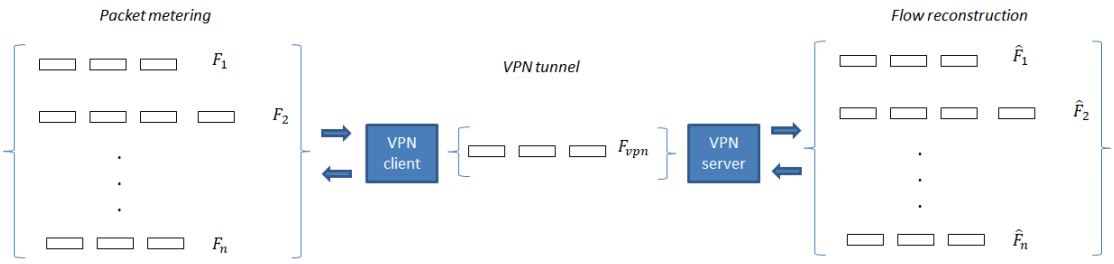


Figure 5.5: Packet and flow tranformation in a VPN communication

Definition 8 *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,*

$$TF = \{TH, F\} \tag{5.17}$$

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 to describe what type of applications are within the tunneled flow of TF.

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 complex 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 constantly cut this TF in chunks of subflows that will be partly "independent" of the global behavior of the tunnel. These partitions are defined as follows.

5.4.1 Slicing tunneled connections

In this section, we define the sliced flows taken from tunneled connections to perform classification tasks. In order to provide a rationale for our slicing process, we state the following considerations:

- In order to find isolated behaviors, we consider to break the tunneled flow into segments of sub-flows and to perform a feature extraction process.
- There is no information about the starting or ending flows within a tunneled flow.
- The main objective will be to detect one or more classes within this sub-flow.
- The flow behavior of past observations helps to define the classes of the current sub-flow.

Given these statements, 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 5.6 illustrates the creation of such sub-flows, which are defined by means of a time window. In the figure, we can notice that Δt_1 defines SF_1 with 8 units of time, while Δt_2 gathers the statistical information about BF with 24 units of time. In further experiments, we will define the most 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.

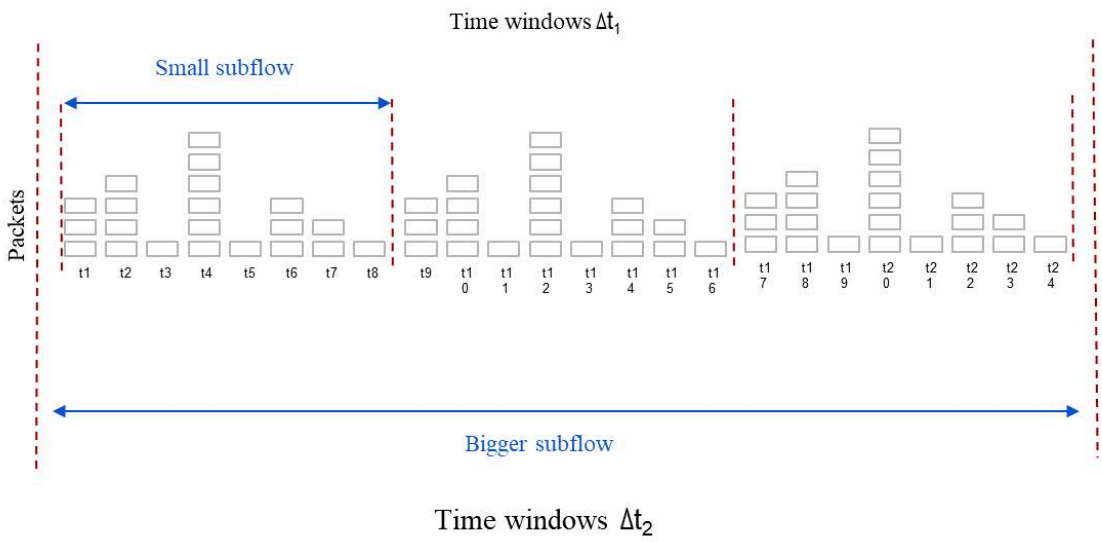


Figure 5.6: Flow construction for a VPN communication

Once the flow construction is finished, we used the same statistical based features described in Table 5.1 for the SF and BF in F,F_{src} and F_{dst}. However, more information is required to characterize applications within a tunnel. One can consider to study the distribution of the packet's length regarding each application. Therefore, we propose to extend the statistical features by computing the metrics over each packet category, as it will be detailed as follow.

5.4.2 Categorizing tunneled packets

In order to find the packet length distribution, we can think about a clustering approach that groups similar packet lengths. However, in order to improve the inner-class separation, we build instead of a Decision Tree (DT) over the packet lengths. But how a DT helps us with this task. If we overview the construction of a DT, we found that this technique can create highly interpretable models for classification. The principle is based on partitioning the feature space into disjoint regions, which in turn are associated with a class. This partition is represented by a tree, where the nodes are the features (attributes) and the leaves are classes [179]. Each level of the tree is built using the attributes that distinguish between one class and another.

In our particular case, we count with only one attribute the packet length (*pktlen*). The *pktlen* is represented in the tree as nodes and the possible values of it are the branches. A metric is used to determinate the information that is contributed to the attribute with regard to the classes, this metric is used for the tree construction. The entropy or Gini degree is used to decide how to build the tree. Intrinsically, these metrics indicate the information degree that is provided by *pktlen* to each class. Considering this, if we build a DT with only the packet length and its corresponding class, we divided this property into a range of values. This naive approach allows us to find an adequate packet categorization. For instance, using the SAT data with *udp* and *udp_sec* as transport protocols, we get as a result the arrangement below (the details of this construction can be found in Appendix A).

Category (*cat*)	Range
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	$pktlen > 1500$

Given this result, we empirically study the packet distribution with violin plots. We notice that for each application launched the packet's lengths are normally within the interval value defined by the DT as seen in Figure 5.7.

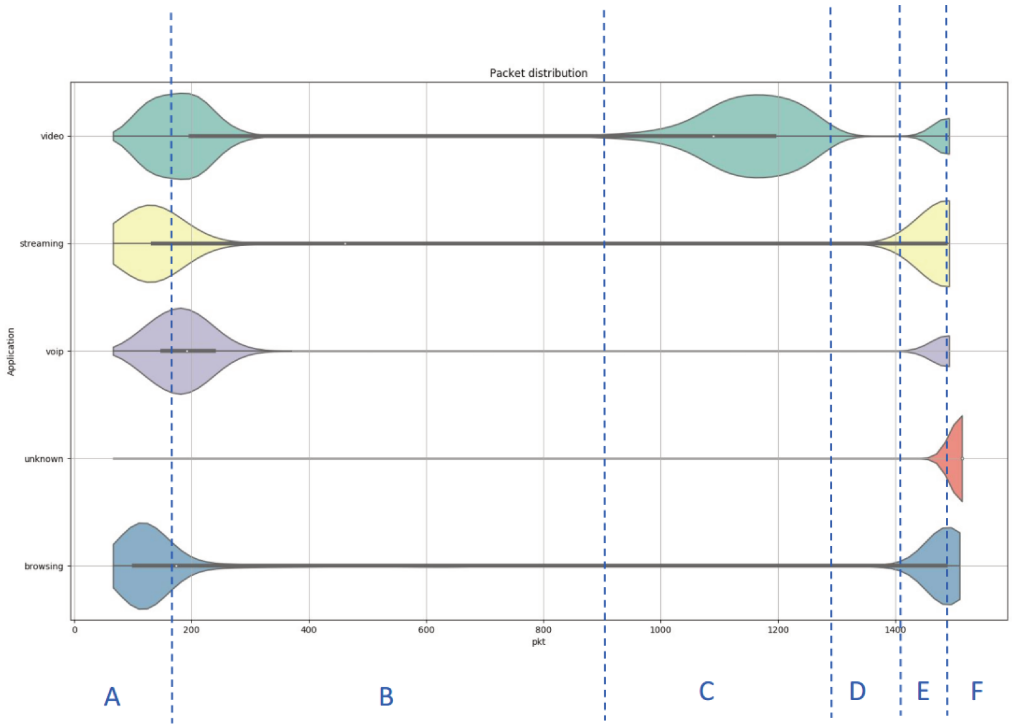


Figure 5.7: Packet distribution of the tunneled connections using udp and udp_sec.

5.5 Result

Now let us put all the features together as a result of this section. To start, let us remark the features that will be used for each of our proposed classifiers. In order to do so, Table 5.5 shows the flows needed for each classifier in order to obtain significant information.

		Aggregate flows	Tunneled flow	Tunneled's sub-flow
Classifier	Description	F,F_{src} and F_{dst}	BF,BF_{src} and BF_{dst}	SF,SF_{src} and SF_{dst}
Flow discriminator 1 (D1)	Traffic discriminator			
Classifier 1 (Cl1)	Classifier of unencrypted traffic			
Flow discriminator 1 (D2)	Traffic discriminator			
Classifier 2 (Cl2)	Classifier of unitary tunneled traffic			
Multi-label classifier 1 (MLC1)	Multilabel classifier of tunneled traffic			

Table 5.5: Flows used for each classifier

In Table 5.6, the features computed for aggregated flows are listed along with their description and the flow direction used. Whereas Table 5.7 shows the features for the tunneled connections. We can notice that the difference is that in aggregated flows the LR estimated features are computed, while in the tunneled connections we add the statistical features for the categories previously defined. Moreover, the flow reconstruction is performed differently for each case.

Feature	Metric	Additional Information	Total of features	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
lr_pktlen_[m]	[m] of the packet lengths	"lr" refers to the LR proposal	2	4
bytes_[Δt]	bytes per [Δt]	"Δt" is the time windows	3	3
pkt_[Δt]	packets counts per [Δt]	-	3	3
Total				42

Table 5.6: Result of the feature extraction process for aggregated flows

Feature	Metric	Additional Information	Total of features	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] *cat* defined in Section 5.4.2

Table 5.7: Result of the feature extraction process for tunneled flows

In addition, we would like to remark some functional and nonfunctional aspects that these features must fulfill.

- Speed: we have to assure that the feature extraction is fast regarding the number of packets per second received per flow. We will see in Chapter 7 that the computation of the statistical features selected takes a small time almost negligible compared with other tasks for traffic classification.

- Memory: two things can be considered here. On one hand, the memory used to gather the current statistical computation in t; and, on the other hand, the past observations needed to compute the statistical-based features (remembering that metrics such as the mean, std, min and max need a set of previous values). In this particular case, only the $t-1$ observations are gathered for the computation of the metrics. In Chapter 7, we make an estimation of the average memory needed by the features selected.

- Smoothing property: this particular constraint refers to the robustness of the metric at facing noisy packets. In this sense, what is desirable is that the metric smooths short-term fluctuations and highlights longer-term trends. We make sure that this property is maintained by using the moving average and the new proposed features based on LR.

5.6 Summary

In this chapter, we describe the feature extraction processes deployed over different Internet communication streams. We notice that the two types of flows treated by this investigation are either aggregated or tunneled flows.

On one hand, for aggregated flows, we investigate a new novel way to extract statistical-based features that can improve the discrimination capabilities of the classical approaches. For instance, we outline the classical selected features and their computation. In addition to those features, LR based features are introduced to address the statistical-based feature computation taking the distinctive behavior of each class. Therefore, an LR model is created for each class; this LR model belongs to a set that models the statistical behavior of a feature such as the mean. In Internet traffic classification, correctly detecting some types of applications at the beginning of a communication is very important because it enables taking fast actions to guarantee QoS in Satellite communication. Conversely, this effort helps to assure that the statistical features computed are correctly measuring the communication behavior for aggregated flows.

On the other hand, for multiplexed flows, we defined a procedure to describe the tunneled connections. This is translated to sub-flows that will describe the status of the tunneled connection in a short time window, aided by a bigger flow that represents the more generic behavior of the tunneled connection. In this sense, we aim to find tendencies over the tunneled connections using statistical features over this mixture of flows.

To conclude, we stated the flow features that will be used by each classifier proposed by our initial architecture. Giving this result, in the next section, we can continue shaping our solution by presenting more formal aspects of the Classification System.

Chapter 6

A Heterogeneous Classification System for Internet traffic

Contents

6.1 Introduction **85**
6.2 Classification System overview **87**
6.3 Flow discriminator **88**
6.3.1 Experimental results 89
6.4 Multi-label classifier **91**
6.4.1 Proposal 92
6.4.2 Multi-label classifier: Experimental results 94
6.4.3 Packet classifier: Experimental results 98
6.5 Classifiers **100**
6.5.1 Base classifiers 101
6.5.2 Meta-classifier 102
6.5.3 Reconfiguration process 102
6.5.4 Experimental results 104
6.6 Incremental learning model **104**
6.6.1 One-class classification 106
6.6.2 Proposal 107
6.6.3 Model build 108
6.6.4 Test 109
6.6.5 Online evaluation 111
6.6.6 Experimental results 112
6.7 Classification System perspective **118**
6.8 Summary **119**

6.1 Introduction

The interest in implementing ML solutions for Internet traffic classification in recent years has promoted the development of a significant amount of works in this area. We can find in the literature approaches that use classic ML algorithms to detect

QoS classes [135], as well as more specialized methods such as Online Sequential Extreme Learning Machine [158], Convolutional Neural Networks [113], Wavelet Kernel Extreme Machine Learning (WK-EML) and Genetic Algorithm (GA) [54], among others. Few works have remarked the need for dealing with the Internet protocols differently [49]. That is to say, that protocols such as IPsec, HTTP2 and SSL/TLS prevent the ML solutions from working correctly due to their properties to change the initial datagrams by adding new headers or dividing and joining the packet content. In this sense, we propose in our architecture (Chapter 3, Section 3.3) a hierarchical approach that will be dividing the traffic according to its encryption characteristics. An ML classifier will evaluate each type of traffic, and different strategies will allow us to obtain a good classification performance.

Moreover, evolving solutions are required for Internet traffic classification. New applications and concept distributions are seen in a short space of time making static Machine Learning (ML) solutions obsolete. Evolving classification solutions are also a challenge in the ML field by itself. In this subject, the more close-related subfields are concept drift, incremental learning, active learning, and online classifiers, among others [105]. In a classical ML classification problem, a classifier is trained with historical data; when new class behaviors are seen, this classifier is unable to detect them. One of the most common approaches used by practitioners in this field is to propose scheduled retrainings. However, in most of the cases, it is hard to capture new historical data for such aim, mainly due to the implied cost. Therefore, we will also present a proposal that will combine several ML approaches to detect concept changes in a semi-supervised manner. The idea is to somehow add competent classifiers in selected areas of the data.

The contributions of this chapter would be two-fold,

- For the hierarchical classification, we define the algorithms that will build the *Flow Discriminators*, *Multi-label classifier* and the *Classifiers*. We took into account the problem of class imbalance detected in the data description; so then, we focus our attention on the ML algorithms adequate for this application, such as the ensemble approaches. In addition to this, we also take into account that lightweight models should be constructed to perform fast Internet traffic classification. Lastly, the performance of the ML models is also taken into account. All the previous conditions biased us to select tree-based ML algorithms. We will demonstrate that this approach fulfills all of our requirements for the hierarchical classification.

- Lastly, one important component of the Classification System is also defined in this chapter, i.e., the Online Configuration. This component is presented as an ML approach based on K Nearest Neighbors (KNN) [51]. We will describe some modifications performed over KNN by using the One-class classification approach [97] for dealing with the class imbalance problem. The result of this system will be denoted as Incremental Learning Model (ILM).

The remainder of this chapter is organized as follows. Section 6.2 presents a short overview of the proposed *Classification System*. This section shows that we will

have mainly *Flow discriminators*, *Multi-label classifiers*, *Classifiers* and *Incremental Learning Models*. In this order, in Section 6.3, we show the flow discrimination capabilities of some ML models. Following, we define and evaluate the multi-label classification approach for tunneled connections in Section 6.4. The classifiers are proposed and validated in Section 6.5. Section 6.6 shows the theoretical basement and tests that allow analyzing and validating our ILM. Moreover, to complement the classification system proposed, in Section 6.7, we highlight some perspectives. Finally, we conclude this chapter in Section 6.8.

6.2 Classification System overview

The system proposed is depicted in Figure 6.1. In summary, after inline traffic is processed, flow features are evaluated by several classifiers disposed of hierarchically. That is to say, that the flow of information is bifurcated depending on its properties. Let us recall each of the levels of this system.

- In the first level, flow discriminators are disposed. The *Flow discriminator 1* separates tunneled from not tunneled flows. Whereas, the *Flow discriminator 2* detects if a tunneled flow has unitary or multiple sessions.
- In the second level, a standard classification is performed. The *Classifier 1 (Cl1)* marks the not tunneled/aggregated flows, and the *Classifier 2 (Cl2)* marks the unitary tunneled connections. A multi-label classifier is also placed at this level to treat the tunneled flows with multiple sessions
- Finally, in the third level, class evolution might be detected by an *Incremental Learning Model (ILM)* that will induce reconfigurations over the $Classifiers$.

Regarding the evolution of the Internet, in one of our works [136], we showed the need for counting with autonomic systems that can somehow be self-reconfiguring given the knowledge acquired from online traffic. The main objective behind it 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. This idea is retaken and considered by our classification system, as shown in Figure 6.1. As it was initially proposed, our system will have active traffic classifiers, allocated in the *Model Repository* component, dealing with the traffic classification task. The *Model Repository* represents the brain of our system, which is classically implemented as a static module. In our particular case, this module is dynamically updated by creating an ensemble of models that represent the traffic classifiers (e.g., *Classifier 1*, *Classifier 2* and *Multi-label classifier*), and by progressively adding base classifiers to the ensembles of this repository when evolution is seen.

Let us understand the problem with a set of question-answer statements.

What are we going to modify? An ensemble of classifiers that must have a structure that can be easily adjustable as it will be presented in Section 6.5.

Where are we going to modify? Within this ensemble, we can add or delete base classifiers denoted as Reactive Classifiers (RCs), whereas the Static Classifiers (SCs) are built with a reliable historical dataset.

How are we going to modify it? An incremental learning model will induce the parameters to alter the corresponding repository.

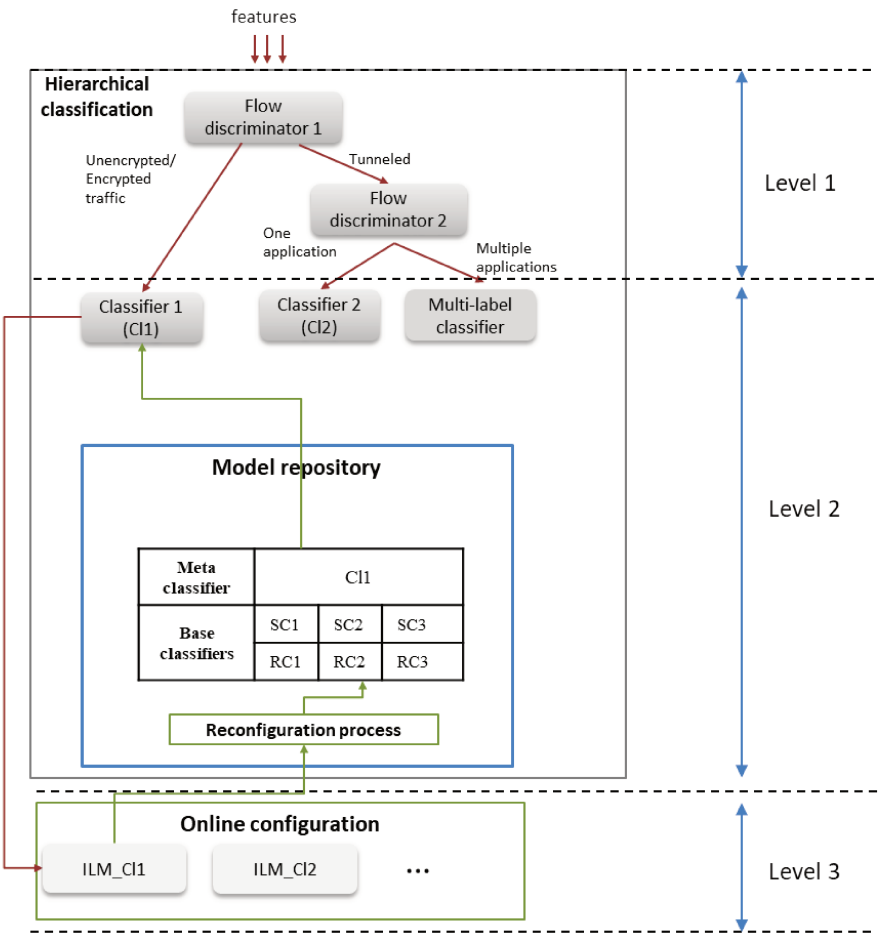


Figure 6.1: Hierarchical classification system.

This work only covers the evolution of the multiclass classifiers Cl_1 and Cl_2. Although, that the *Multi-label classifier* will eventually need a retraining or incremental learning process, we start validating our approach on the multi-class models. Afterward, we could propose a possible equivalent approach for the multi-label case.

Following this set of ideas, we present as follows the elements of the classification system. We also demonstrate that the components of the hierarchical system and the incremental model are reliable and comparable with classical approaches. To do so, comparisons between different methods are set, and analysis of the results is given.

6.3 Flow discriminator

This module is in charge of separating different types of traffic as it is illustrated in Figure 6.14. In this sense, we evaluate different classical ML classifiers to perform this task. For instance, multiplexed connections are detectable because the packets

counters per second are considerably higher than an aggregated connection. The same logic could be applied over the tunneled flows with one or more sessions. Based on this, we would like to test the capabilities of some ML algorithms to separate those Internet flows.

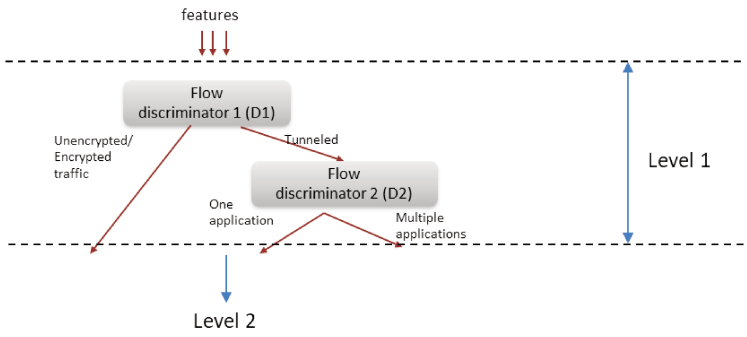


Figure 6.2: Flow discriminators of the hierarchical system.

In principle, based on our experience with this application, we found out that the most accurate ML approach is the tree-based one (such as Decision Tree introduced in Chapter 5, Section 5.4.2). Particularity, Random Forest (RF) also highlights thanks to its capabilities for dealing with class imbalance. Also, this method is suitable for multi-label tasks. The ideal would be to have a set of models that can be interpretable, easy to implement, and modify.

Tree-based approaches will serve as important knowledge for understanding the flow discriminator results, and also for subsequent sections. At this stage, we would like to provide a brief introduction to this approach. RF 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 [21]. 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 [73]. RF is well-known for reducing the variance by decreasing the correlation between the trees. This is achieved with the random selection of the input variables.

In addition to RF, we test other ML algorithms such as the K-nearest neighbors (KNN), Decision tree (DT), Extra Trees, Voting, and AdaBoost to build the flow discriminator. We introduce right below the experimental results of this phase.

6.3.1 Experimental results

In Chapter 5, we defined the suitable features to perform this task. Therefore, we need to select adequate classifiers *Flow discriminator 1 (D1)* and *Flow discriminator 2 (D2)* that will perform the discrimination tasks. For this experiment, two data sets where selected: VPN-nonVPN and SAT data. In addition to this, we evaluate the average most important variables to perform the classification. This study is conducted training several RFs and registering the entropy value of each feature. This feature selection process is applied to build more lightweight classifiers and to offer fast discrimination outputs.

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 $\Delta t = 10ms$ 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 to measure their accuracy, as displayed in Table 6.1. In addition to this, the most important features are retained, and the classifiers are retrained with them. The result lets us conclude that with only the ten most important features (*pkt_0_min* , *c_pkt_0_min* , *pkt_min* , *and c_pkt_4_mean*, among others), we can perform an accurate flow discrimination task. From the table, we can select a DT as $D1$.

	SAT-data					
	VPN-NonVPN		GW		ST	
Classifier	All features	10 features	All features	10 features	All features	10 features
Desicion Tree (DT)	0.9981	0.9939	0.9989	0.9985	0.9989	0.9985
Random Forest (RF)	0.9987	0.9947	0.9989	0.9989	0.9989	0.9989
KNN	0.9853	0.9916	0.9946	0.9978	0.9961	0.9978
Extra Tree	0.9986	0.9951	0.9993	0.9989	0.9995	0.9987
Voting	0.9983	0.9947	0.9989	0.9987	0.9991	0.9987
AdaBoost	0.9984	0.9930	0.9993	0.9982	0.9995	0.9982

Table 6.1: Accuracy results after testing the $D1$ classifiers

To complement this result, we took the SAT data which counts with 135306 multiplexed window flows and with 3102 nonmultiplexed, and analyze the test results of DT acting as $D1$ with its confusion matrix in Table 6.2. We can see from the table that the class identification is appropriately done, with zero misclassifications.

		Actual	
		non-multiplexed	multiplexed
Predicted	non-multiplexed	3102	0
	multiplexed	0	135306

Table 6.2: Confusion matrix after evaluating $D1$

Finally, for $D2$, we want to identify the unitary connections from the multiple ones. We took all the multiplexed connections and divided them into windows of $10ms$, labeling each window with is respective class (Unitary or Multiple). The results are similar to those for $D1$ with an accuracy equal to 0.9661, allowing us to conclude that a simple DT is also suitable for this task. The confusion matrix of this evaluation is given in Table 6.3. We can notice that some of the multiple tunneled connections can be classified as unitary (4342 to be specific), while fewer flows are misclassified as Unitary (223 flows). This result is expected due to during the launching of the multiple applications within the tunnel, some of them appear as unitary for short time periods.

		Actual	
		Unitary	Multiple
Predicted	Unitary	4337	4342
	Multiple	223	126414

Table 6.3: Confusion matrix after evaluating $D2$

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 ML technique for each type. For instance, the tunneled connections with multiple applications can be treated by a multi-label classifier as it is detailed as follows.

6.4 Multi-label classifier

This block aims at detecting the classes within a tunnel with multiple sessions as it is placed in Figure 6.3. The multi-label classification task seems to be a suitable path, given the characteristics of the problem. The work in [155] already demonstrated that multi-label classification could be achieved by a Neural Network-based approach to detect Streaming connections in tunnels. Other methods present different strategies based on classical multiclass classifiers to find a targeted class in a tunnel. For instance, the work in [184] intents to detect VoIP traffic within IPsec tunnels. Results show that the approach is accurate; however, few tests were performed mixing Internet traffic which might decrease the accuracy. Similar methods are found in [5] to identify three main classes (Bulk transfer, Interactive and Streaming), and in [106] for Streaming communications. In this last work, background traffic was added for Streaming communications; the classification results decreased considerably given the noise of these background packets. Those studies are thus giving hints about the possible incapability of the ML model to maintain their classification accuracy for tunneled sessions with mixed traffic. Nonetheless, the multilabel classifiers might be able to take into account this mixed traffic as perceived by [155].

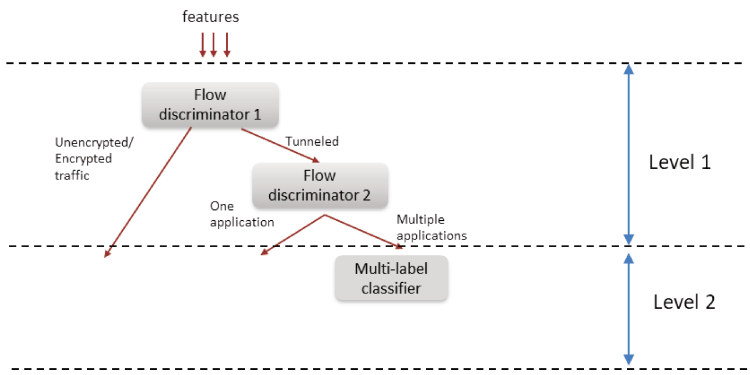


Figure 6.3: Multi-label classifier of the hierarchical system.

But what is the paradigm of multilabel classification? If we take a standard classification approach, we know that the problem is to assign a sample to a single class. In contrast, the multi-label classification problem states that a sample belongs to more than one class at the same time [147]. 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, and an image can be classified into different classes and can have embedded different sub-images at the same time.

Before continuing, let us define the classical multiclass classification problem with 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

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 [168]. To overcome those issues, several ML algorithms include the multi-label learning task such as multi-label decision trees [95], multi-label kernel methods [115], multi-label neural networks, and multi-label KNN [191]. 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 [2]. We present our multilabel classification proposal as follows.

6.4.1 Proposal

Once the flow discriminator $D2$ detects a tunneled connection with multiple applications, a multi-label classifier will be in charge of defining what type of applications are within the tunnel. Somehow this approach allows us to determine the QoS classes in the multiplexed session. For tunneled applications, we defined a different way to build the Internet flows: a small sub-flow SF and a bigger sub-flow BF that represents the local and global behavior of the tunnel, respectively.

In Figure 6.4, we propose two approaches to deal with multiple connections:

- **Flow classification**: a multi-label classifier defines the classes of the flow SF. In this case, taking RF as a multi-label classifier, we expect to have a set of positive labels with a certain probability. In Figure 6.4, we define the positive class those with a probability higher than 0.5, for example, such as the VoIP

and Streaming classes. The probability threshold is set in most of the cases at 0.5; however, some ML algorithms define this value depending on the training performance results.

- **Packet classification**: an incoming packet from the flow SF is classified into only one class by using the packet's length and statistical-based features of SF and BF. The packet classification task will be treated as a multi-class classification problem.

In the **Flow classification** task, the multi-label output is transmitted to the QoS architecture, and the complete tunnel might be prioritized when a class of interest is detected. On the other hand, if a packet classification output is provisioned, packet prioritization can be achieved. More about this will be discussed in Chapter 7. The packet classification case has not been yet included in our architectural framework. However, it is a desirable solution and needs more exhaustive work to its validation. Nonetheless, in this section, we give some hints to solve the **Packet classification** task.

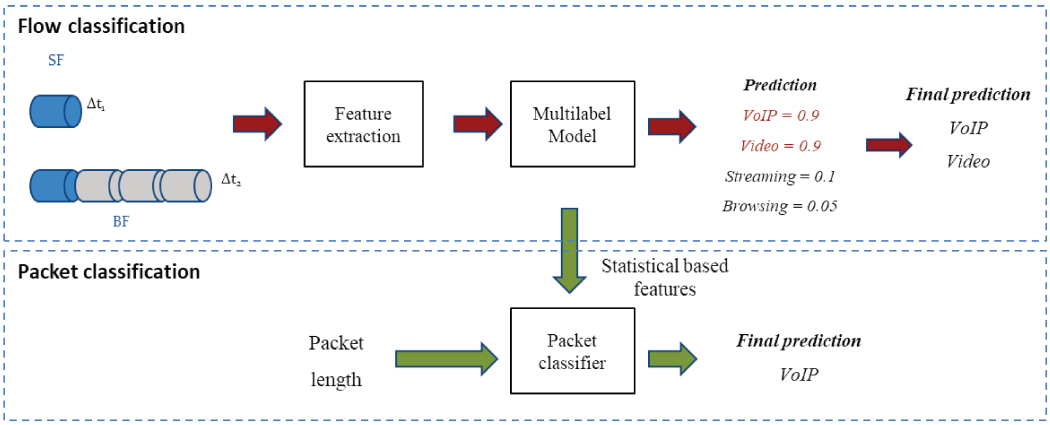


Figure 6.4: Classification process for multiplexed flows.

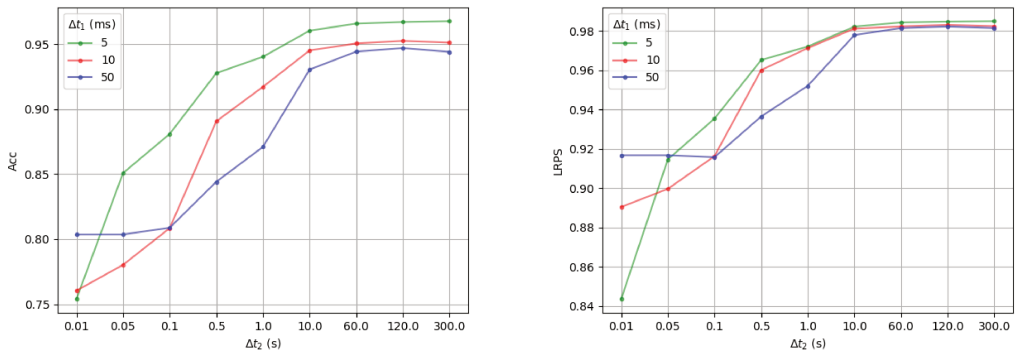
In the following sections, we present the experimental results of both approaches.

6.4.2 Multi-label classifier: Experimental results

For this experiment, only the SAT data contains multiple applications multiplexed within a tunnel. Therefore, we take the data under udp as the transport protocol. First of all, we analyze the effect caused by the feature extraction process proposed in Chapter 5 to an RF multi-label classifier. Metrics such as accuracy (Acc) and Label ranking average precision score (LRPS) 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 make a brief analysis of the statistical-based features in tunneled connections.

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 [139]. 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

Figure 6.5: Accuracy and LRPS of a RF with $\Delta t_1 = \{5ms, 10ms, 50ms\}$, and Δt_2 varied from 0.01s to 300s.

Regarding the most suitable value of Δt_2, it is noticeable from the figures that values upper than $10s$ help to increase the accuracy and the LRPS. While for Δt_1, we can conclude that while smaller 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. We will retake this discussion in the implementation section (Chapter 7).

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 6.4, we validate the use of RF as the multi-label classifier by comparing it with close related approaches.

Algorithm	Acc	LRPS
RF	0.9537	0.9805
DT	0.9111	0.9658
Extra trees	0.9089	0.9544
MLP	0.4252	0.6656
One vs All (Logistic regression)	0.5398	0.7878

Table 6.4: Accuracy result after evaluating the multi-label classifier

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; while the confusion matrix of this test is presented in Figure 6.6.

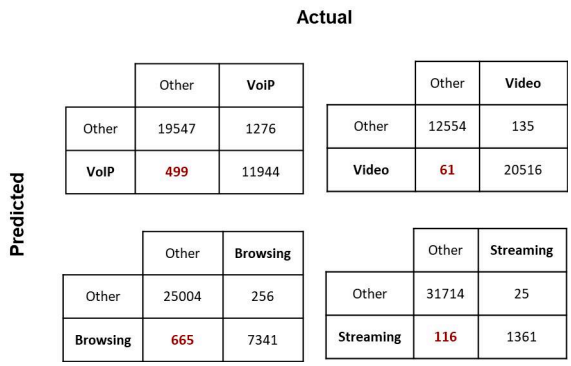


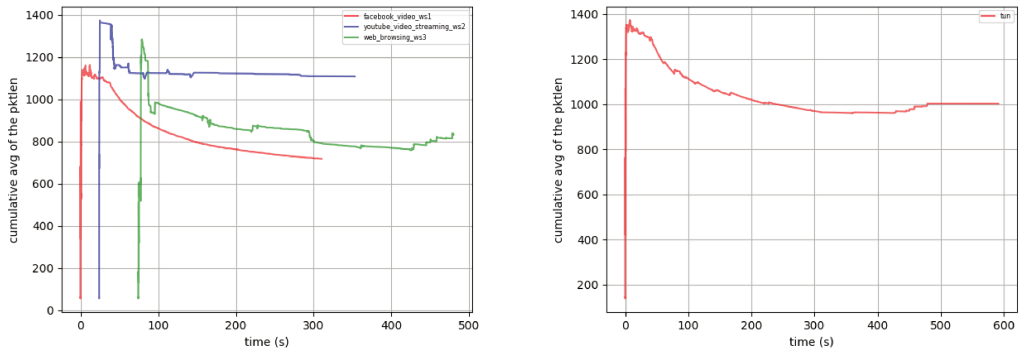
Figure 6.6: Confusion matrix for Multiplexed tcp sessions with $\Delta t_1 = 10ms$ and $\Delta t_2 = 60s$

Analysis

The following shows a graphical analysis of the statistical features' discrimination capabilities over tunneled connections. In this analysis, we took the packet length raw values before and after the tunnel, and compute their cumulative mean value. Three cases were analyzed as follows.

- Video-Streaming-Browsing: destination direction

In Figure 6.7(a), the packet length behavior of the three sessions is depicted. We can notice that the beginning of these connections is very similar, and afterward, we can notice that the time series varies from 1000 to 1400. Figure 6.7(b) shows the equivalent tunneled connection. We notice that the behavior of the beginning of the connection for the three applications is lost; however, an indistinguishable new time series behavior emerges. This behavior was also found in some of the statistical-based features; as an example, for the *pktlen* categories.



(a) Before the tunnel (b) After the tunnel

Figure 6.7: Cumulative mean value of the packet lenght of three applications Video-Streaming-Browsing, before and after the tunnel for the destination direction flow.

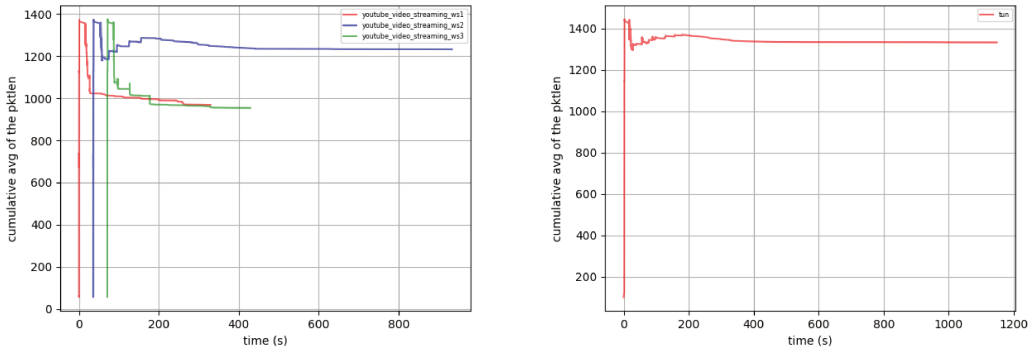
- Streaming-Streaming-Streaming: destination direction

  When we analyze three parallel applications from the same category, we saw that the resulting tunneled stream behaves as if only one application is in the tunnel. For instance, in Figure 6.8(a), we count with three YouTube flows with cumulative means varying from 1000 to 1400. Figure 6.8(b) shows the equivalent tunneled connection.

- Streamingx3-Browsingx3-Voicex2: both directions

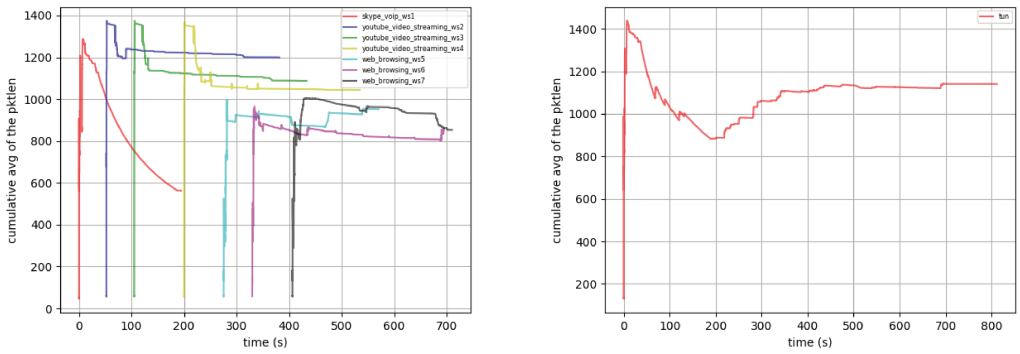
  Let us now increase the number of parallel connections as in Figure 6.9(a). The tunneled mean is smoothing its behavior according to the type of parallel connections perceived as it is shown in Figure 6.9(b). A similar case occurs for the source flow streams displayed in Figure 6.10.

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 classifier is the most adequate for this problem. Finally, analyzing the tunneled and not tunneled data statistical behavior, we understood the behaviors of the flows, which in turn help to somehow validate our approach.



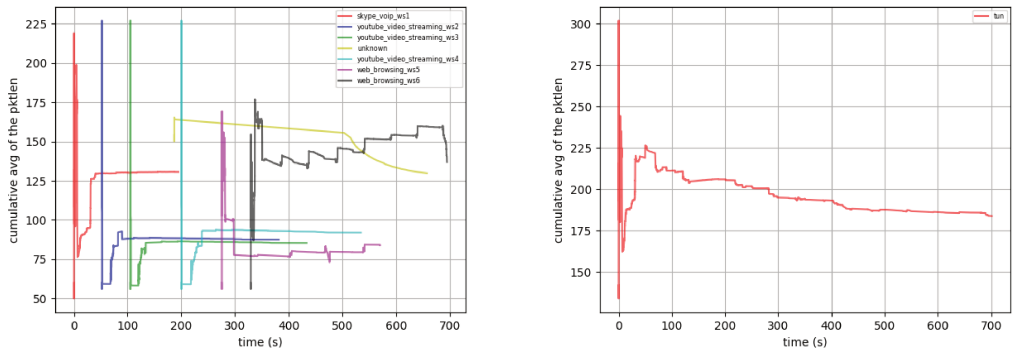
(a) Before the tunnel (b) After the tunnel

Figure 6.8: Cumulative mean value of the packet lenght for three applications Streaming-Streaming-Streaming, before and after the tunnel for the destination direction flow.



(a) Before the tunnel (b) After the tunnel

Figure 6.9: Cumulative mean value of the packet lenght for 8 applications, before and after the tunnel for the destination direction flow.



(a) Before the tunnel (b) After the tunnel

Figure 6.10: Cumulative mean value of the packet lenght for 8 applications, before and after the tunnel for the source direction flow.

6.4.3 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.

To built 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 flow SF.
- The statistical features of the multi-label classifier with $\Delta t_1 = 10ms$ and $\Delta t_2 = 60s$, and the current packet length and its class form our historical dataset.
- With the filtered data, we build a packet classifier represented by a Random Forest.

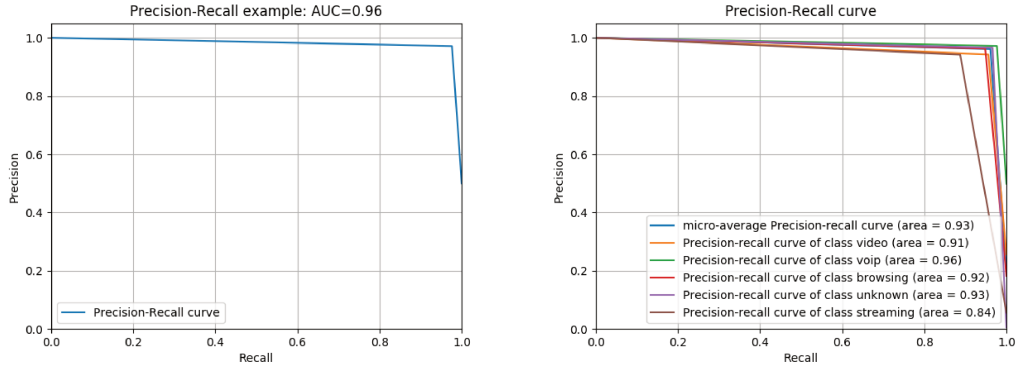
The dataset characteristics are depicted in Table 6.5. In this table, we can notice that the total number of samples is equivalent to the total number of packets in a tunneled communication. Whereas, the filtered samples represent the packets that fall into a window Δt_1 with more than one class within the tunnel. The accuracy (acc) in Table 6.5 lets us conclude that a good packet classifier can be built with the flows and packet length information.

Dataset	# samples	# filtered samples (% total)	acc
udp	3527278	196778 (5.58%)	0.9610
udp_sec	3425113	216210 (6.31%)	0.9655
tcp_sec	3145291	124512 (3.96%)	0.9586

Table 6.5: Accuracy result after evaluating the multi-label classifier

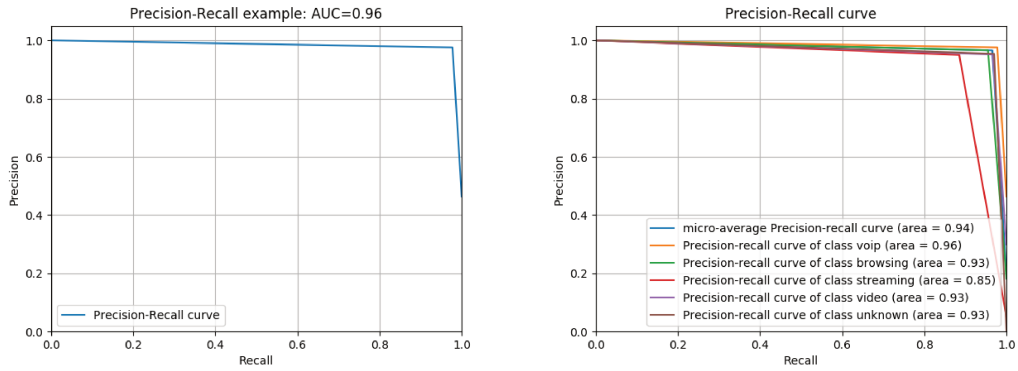
In addition, we show the performance per class expressed by a Precision-Recall curve in the figures 6.11, 6.12 and 6.13 for *udp*, *udp_sec*, *tcp_sec*, respectively. On the one hand, for all the cases, we compute the Area Under The Curve (AUC) which is a performance measurement for classification models at various thresholds settings. This metric tells how much model is capable of distinguishing between classes. AUCs close to one indicates good model prediction capabilities. 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 globally. However, we can notice that a small decrease is presented for the class “streaming” which in turn in a minority class.

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 6.5). This characteristic can be attributed to the construction of the data; 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.



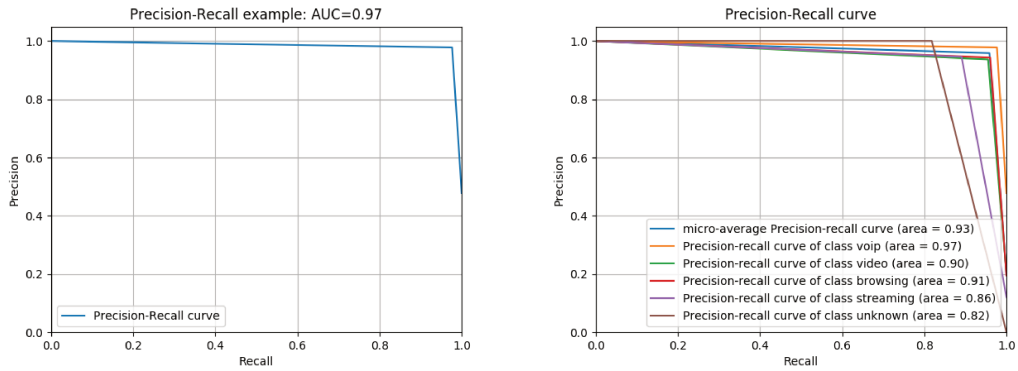
(a) Global Precision-Recall (b) Precision-Recall for class

Figure 6.11: Precision-Recall curves for the *udp* dataset.



(a) Global Precision-Recall (b) Precision-Recall for class

Figure 6.12: Precision-Recall curves for the *udp_sec* dataset.



(a) Global Precision-Recall (b) Precision-Recall for class

Figure 6.13: Precision-Recall curves for the *tcp_sec* dataset.

We culminate this section with two approaches to deal with multiplexed connections. Up until now, we validated the multi-label classification for our architectural

framework. In the next section, we present another essential component of this architecture the *Classifiers*.

6.5 Classifiers

The classifiers belong to the second level of our hierarchical classification as in Figure 6.15. In this proposal, we notice that an *ILM* might continuously update a repository of models. In this section, we describe and build all the elements of these classifiers.

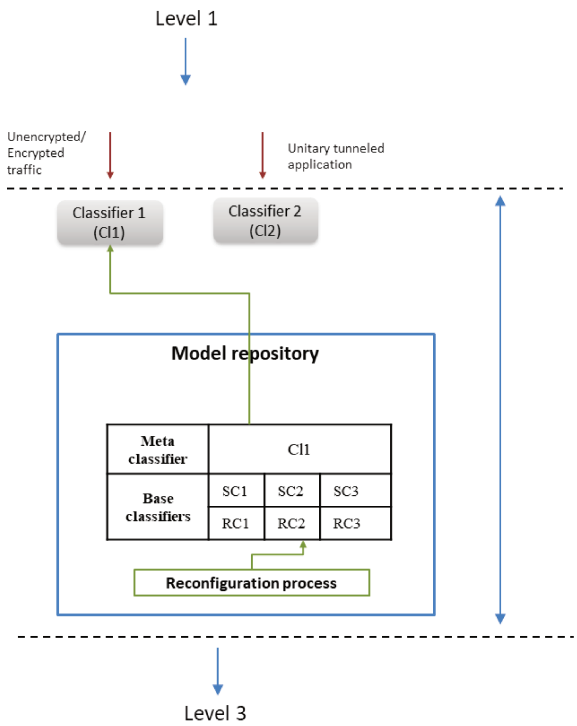


Figure 6.14: Classifiers of the hierarchical system.

There already exist ML-based classifiers that can perform the task needed by the *Classifiers*. However, they are static and require constant retraining processes for their updates. Modern applications with massive volumes of data are in continuous search of automatic, self-learning, and auto-reinforced approaches that can cope with the evolution of data. Human experts can annotate a negligible fraction of data, and Internet traffic is out of this fraction. The most reliable solution for highly evolving environments is scheduling retrainings; however, labeled data dependency is a deficiency. In Internet traffic, this latter approach is the most provisioned [91, 47]. Few are indeed the efforts to create online reconfigurable classifiers that still depend on labeled data, such as online Random Forest [153, 66] or concept drift based ones. In particular, concept drift aims at detecting data distribution change over time. For instance, some of these methods monitor the classifier performance in an online manner; if the performance considerably decreases a drift is detected, and the classifier must be updated. These methods are usually supervised based approaches[11, 63, 67].

An interesting approach, presented by [152], creates an ensemble of classifiers that is adaptive over time. New base classifiers are added with selected labeled data to improve the performance of online classifiers. Following this idea, let us say that we achieve to fill a batch of samples that represents a new distribution; then, the classifiers should be aware of this distribution change. In this sense, we would like to propose an ensemble of classifiers that can be easily modified. The general idea is to add base classifiers to our repository that represents the new distribution. Therefore, this ensemble of classifiers has some parameters that will be presented in Section 6.5.1. How the classification decision is performed is presented Section 6.5.2. Moreover, Section 6.5.3 details the procedures for penalizing, pruning, and adding new classifiers to the *Model Repository*.

6.5.1 Base classifiers

The ensemble of k base classifiers will be defined as the tuple below,

$$C_i = (name, m_i, t, W_i, F_i, Q_i, s) \tag{6.1}$$

where,

- *name* is the identifier of the classifier
- m_i is the model trained
- t is the type of classifier either static or reactive.
- F_i represents the f-score metric for the classes of the problem as follows $F_i = [f_i^1, f_i^2, ..., f_i^c]$ with c the number of the classes.
- $W_i = [w_i^1, w_i^2, ..., w_i^c]$ stands for a classification weight that will be used to penalize the classifier by class.
- Finally, $Q_i = [q_i^1, q_i^2, ..., q_i^c]$ is a vector that stores the cumulative value of the classifier's quality on predicting an input sample for the class j.

The F-score is computed when the classifiers are trained. These values will be used to compute the quality of the classification given by the classifier for a specific class. This will be detailed in Section 6.5.2. While the qualities in Q are updated any time that a new input is evaluated.

We will denominate W the vector $W = [w_1, ..., w_k]$, likewise $Q = [q_1, ..., q_k]$ and M the set of models $M = [m_1, ..., m_k]$. The prediction given by the k classifiers will be denoted as $P = [p_1, p_2, ..., p_k]$. Note that initially all the classifiers have the same weight $w = 1$, the variation of this value will be given by the reconfiguration process.

In addition, we set a differentiation between the base classifiers with the attribute t, as follow:

- Static classifiers (SC): they are mostly trained with the historical data

- Reactive classifiers (RC): they are trained with online data being specialized in one class.

This division is made to avoid losing generalization. For instance, if in a time interval, the amount of classifiers favors only one class; therefore, the ensemble might be biased to the just added reactive classifiers. We will notice in the following section that the reactive classifiers will be treated differently than the static classifiers to maintain the performance and the integrity of the ensemble.

6.5.2 Meta-classifier

The meta-classifier consists of a combination method used for making a consensus of the ensemble prediction. In the literature can be found several suitable combination methods for this aim such as hard and soft voting [169]. In this particular approach, any of the existing combination methods can be used to define the final output. The meta-classifier has as objective to compute the quality of the prediction. This variable is heuristically set, and it will be calculated as follows,

$$Q' = P' \frac{(W' + F')}{2} \tag{6.2}$$

P' is the vector of probabilities for the class predicted y' by the models. As a result, we will obtain a vector of qualities for each classifier.

6.5.3 Reconfiguration process

Once the incremental learning system has evaluated the sample x_i, two things can happen, either the sample represents a change in a class distribution, or not. If a change is detected, a new classifier will be added to the ensemble, and an old one might be penalized or discarded.

1. Penalizing a reactive classifier

   We envisage penalizing the learner that provided the lowest classification quality for the class y'. This action is tied to the event of a class distribution change. Therefore, the classifier that got the lower classification quality is given by,

   $$\{C_i | \arg\min_{q'_i \in Q}(Q)\} \tag{6.3}$$

   The classifier i that got the lower q' is penalized as follows.

   $$w'_t = w'_{t-1} \times \left(1 - \frac{1}{k+c}\right) \tag{6.4}$$

   In this case, w_{t-1} is the previous weight, k the number of classes and c the number of classifiers. The justification is that one of the classifiers is not answering correctly for the class y', that is why its classification capability decreases.

2. Adding a reactive classifier

   A classifier is added when the incremental learning system has detected a change in one or more class distributions by sending the new training set $\{\hat{X}, \hat{y}\}$. The new learner is marked as a reactive classifier, due to it is trained with unreliable data.

   To know which type of algorithm and model will be used to train a new classifier, we detect the most "accurate" classifier as follows,

$$\{C_i | \arg\max_{q_i \in Q}(Q)\} \tag{6.5}$$

   Therefore, we train a new base learner with the same structure (model and algorithm) than C_i. When adding a new classifier, all the parameters of the ensemble are updated, such as W, F, Q and M. In case that, it is the first time that a new RC is added for a class, we attach as many RC as the amount of SC.

3. Pruning the ensemble

   The ensemble is pruned in the sense that reactive classifiers will be deleted over time. In this particular case, we consider that the amount of RC for a class won't be higher than the amount of SC. Therefore, a new added RC replaces the weakest RC in the ensemble.

To conclude this section, we present the sequence of steps of this module in Table 6.6.

Macro algorithm

Input: Samples $\{\hat{X}, \hat{y}\}$
Procedure:

1. Obtain the weakest learner C_i by using Eq. 6.3
2. Penalize C_i by using Eq. 6.4
3. if a distribution change was detected,
   - Detect the most capable model C_j with Eq. 6.5
   - Train a new RC with the same structure (model and algorithm) as C_j and with $\{\hat{X}, \hat{y}\}$
4. if $|RC^c| > |SC|$, remove the weakest classifier in RC^c
   C_i from the ensemble
5. Update all the parameters of the ensemble, such as W, F, Q and M

Output: Ensemble updated

Table 6.6: Pseudo code for penalizing and adding a base classifier

6.5.4 Experimental results

This test allows us to validate the construction of our classification system. To do so, the ensemble will be trained with two variations of the static base classifiers, and it will be compared with classical approaches. We selected the Random Forest (RF), Decision tree (DT), Extra Trees, Voting, and AdaBoost Classifiers. The classifiers' settings were modified to obtain their overall best performance. Moreover, for the dynamic ensemble, we will consider two variations: i) CS_1: 3 DTs base classifiers and CS_2: 3 RFs base classifiers. We can notice from Table 6.7 that our approach provides a good accuracy value, comparable with well-known classical methods such as Random Forests and Extra Tree classifiers. However, this result is expected due to the construction of our ensembles with well-known classifiers. These ensembles have some settings that will allow us to perform modifications in an online manner.

	Cl1			**Cl2**
Algorithm	PAM	VPN-NonVPN	SAT-gw	SAT-gw: Unitary
KNN	0.8783	0.9785	0.8634	0.9209
RF	**0.9778**	0.9876	**0.9186**	**0.9401**
DT	0.9356	0.8698	0.9066	0.9321
Extra Tree	0.9466	**0.9879**	0.9174	0.9304
Voting	0.9335	0.9850	0.9135	0.9358
AdaBoost	0.9466	0.9832	0.6242	0.8333
CS_1	0.9450	0.9498	0.9094	0.9308
CS_2	*0.9677*	*0.9876*	0.9180	0.9389

Table 6.7: Result after training the classifiers with all the flow sequence

Additionally, we demonstrated that $Cl1$ and $Cl2$ could be built with adequate accuracy. For instance, for the SAT data, we got an accuracy between 0.90-0.95 %; however, if we balance the data before creating the ML models, we reach values close to 1. Therefore, for building the ML models, we recommend proper balancing before the training process. Given these results, in the next section, the last component of the classification system: the Incremental Learning approach.

6.6 Incremental learning model

The last level of our hierarchical classification system is composed of ILMs for each classifier $Cl1$ and $Cl2$. This component is placed at the end of the chain to provoke modifications over the *Model repository* as it is depicted in Figure 6.15. The motivations and basements of this approach are presented as follows.

From the previous section, we notice that most of the self-learning approaches use labeled data. Fortunately, there are some other promising options, such as semi-supervised learning. Unlabeled data taken for real applications can be reused as labeled samples for the model discrimination improvement. In novelty detection, one of the most popular and simplest approaches is based on clustering methods. Distance or density information can provide insights about the data distributions. Several works try to use clustering constructions to detect samples that might be changing the class distributions. These samples can be continually stored to later induce retraining or modifications in the classifier's parameters [57, 74]. The assumption is that the

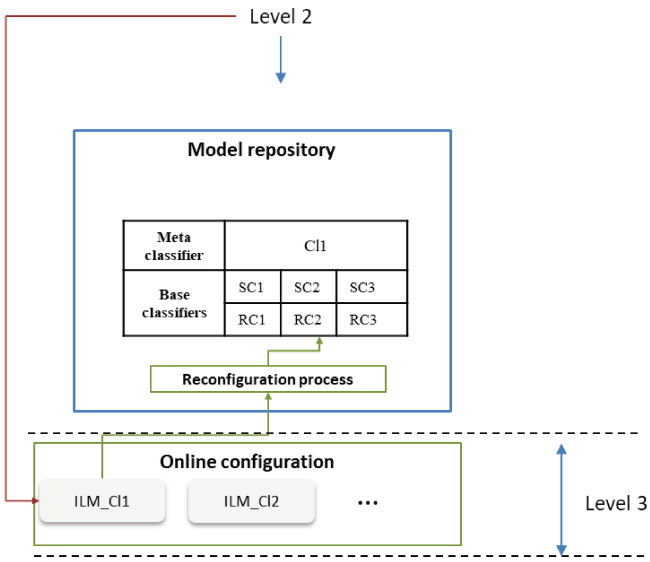


Figure 6.15: Incremental learning model (ILM) of the hierarchical system.

commonly known concept distributions stay in their respective clusters, while new concepts or unusual ones will form new patterns around the existing groups. One of the main drawbacks of this approach is that the groups have to correctly represent the data, which is a difficult and challenging task in this area. Well-separated classes might be well represented by clustering constructions, which implies that the features of the data describe and differentiate each of the classes. However, in the presence of high dimensional spaces, we cannot make the same conclusions. Some of the questions that arise are: i) What about not linearly separated data? For such a case, many of the clustering approaches fail dramatically; the clusters can be overlapped in different areas, which is not desirable for separating the data. ii) Will the sample belong to a shared region assigned to the correct cluster? For the former problem, the best option is to use a classifier that better performs data separation.

Presuming that we can find a system that somehow updates online classifiers with partial new knowledge, we could achieve better performance. In particular, these new classifiers should be representing the evolution of a class. This evolution can be detected and targeted by experts in the field, applying active learning. However, most of the time, it is hard for the experts to conclude about the behavior of the samples. Another approach can be to have a repository of a training dataset that is updated continuously. Nonetheless, in some cases, it is now hard to capture real-world data measurements making this option non-desirable.

Evaluating all the arguments above, we will present a proposal so-called Incremental Learning Model (ILM) that will combine several ML approaches to detect concept changes in a semi-supervised manner, and to add competent classifiers in selected areas of the data. In this phase, we present the incremental learning method designed to evaluate the samples given by the classification system and to detect a new class distribution. Experts target this class that contains a low amount of samples typically. The main idea is to take samples from the online process to complete

the data available in this targeted class (minority class). When sufficient examples belonging to the targeted class are seen, new base classifiers are added to the ensemble in the classification system. The main properties of this model are listed below.

- The method is based on One-class classifiers that will detect new class distributions in a semi-supervised manner. The One-class classifiers are modeled with KNN and reinforced with K-means. The new class distribution is detected based on the similarity given by the Euclidean distance.
- A resampling procedure is designed for the one-class classifiers to assure a good data representation in the presence of class-imbalance.

To start, we will introduce the One-class classification approach, and how KNN is adapted to it as follows.

6.6.1 One-class classification

One-class classification (OCC) is an ML approach that handles the classification problem differently to the classical methods. Classical pattern classification disposes of a training set to learn a model that characterizes the problem; typically, the training set gathers a variety of concepts (classes). In OCC, the principle is: one unitary concept is available for the training and a boundary like function models this concept. After the boundary function is built, samples that fall outside are considered as outliers. This approach is commonly denoted as learning in the absence of counter-samples.

Formally speaking, an one-class classifier model for an unitary concept is described as a model $M_c : X \rightarrow \{c, c_o\}$ with c the concept or class. Then, the model for each class available is,

$$M_c(x) = \begin{cases} c, & \text{if } F_c(x) \geq \theta \\ c_o, & \text{otherwise} \end{cases} \tag{6.6}$$

where F_c is the boundary function, c_o the outlier class, and θ the classification threshold.

For the one-class multiclassification problem, a model is built for each class. A new sample x_i is evaluated by all the models. Considering that an input can fit into one or more one-class classifiers, a decision or combination function will give the final classification of the sample [97].

We can find four main trends in this area, differentiated mainly on how the boundary and the decision or combination function are defined [98]. For instance, density-based methods find estimated distributions for each class. Incoming samples are contrasted with these distributions, and the decision function is based on their similarity. Boundary based methods enclose the data into boundaries such as One-class Support Vector Machines, Support Vector Data Description, and Minimum Spanning Tree Classifier. Finally, reconstruction-based methods are based on clustering and data modeling to find the data structure of the classes. In this approach, outliers will be significantly dissimilar to the data representation. The reconstruction based approaches are mainly based on k-means, self-organizing maps, and auto-encoder neural

networks, among others. This latter approach is the interest of our approach due to KNN is used as a reconstruction method for our incremental learning system. We present as follows the principle of KNN, and its applicability in OCC.

KNN and one-class classification

KNN is a distance-based approach based on the distance between samples in a dataset; in general, a small distance means high similarity. The distance value between two samples can be measured by different metrics, such as the Euclidean, Cosine, Pearson, Mahalanobis, and City-block distance, among others. The samples are arranged according to their similarity; after that, samples with high similarities are grouped or classified using an unsupervised or supervised learning process [51].

The algorithm KNN is one of the simplest and most effective learning algorithms based on distance. KNN locates the K nearest neighbors of a sample x_i, computing the distance between x_i and the rest of the samples in the training dataset. Commonly, a sample x_i will belong to the class where the distance to its K nearest neighbors is the minimum [104]. In the case of one-class classification, the samples can be separated by class, and a combination method is defined to perform the final decision. Such a combination method can be:

$$c_i = argmin_{c_k}\{d^c_{nn}\} \quad c = 1, \cdots, |C| \tag{6.7}$$

where d^c_{nn} is the distance between the sample x_i, and the samples $\overline{X}$ in each one-classifier as follows.

$$d^c_{nn} = avg\{min_k\{d(x_i, x_j)\}\} \quad j = 1, ..., |\overline{X}_c| \tag{6.8}$$

6.6.2 Proposal

In this section, we present the algorithm and model to perform an incremental learning kind task using a semi-supervised approach. In general, our approach is composed of three main steps: model build, test, and online evaluation. In brief, each of the steps will be in charge of:

- Model build: it will be in charge of setting the organization of the data. Our approach will be based on KNN to create the classes, and K-means to form clusters in the classes called microclusters. Therefore, the training of the model is to place the samples in an n-dimensional space labeled with their respective classes.
- Test: in this step, we will evaluate the test set, and we will detect overlapping zones to reorganize the data model.
- Online evaluation: finally, we will evaluate samples in an online manner. Batches of samples can be built for each class in the data model. These batches of samples can be used furthermore for the classification system.

We show a graphical workflow of our approach in Fig. 6.16. In the following sections, we formally define each of the steps.

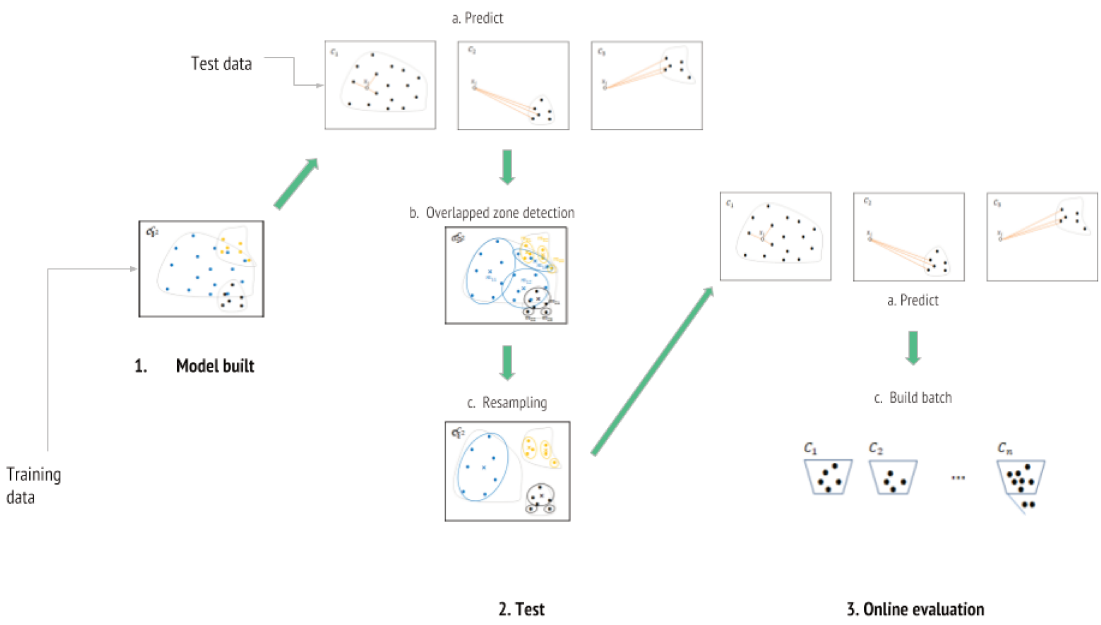


Figure 6.16: Graphical description of the proposal.

6.6.3 Model build

For this case, we represent a class by the tuple in 6.9,

$$C_r = (\overline{X}_r, MC_r, B) \tag{6.9}$$

where $\overline{X}_r$ is the set of samples $\{x_i | M(x_i) = C_r\}$, and MC_r is a vector that denotes to which micro-cluster each sample belongs. Finally, B stores samples evaluated in an online manner.

The standard procedure to build ML models is to perform a training process that will allow defining the parameters or structure of the model. In our approach, any training is required; instead, this process is translated into an organization of the samples based on their similarity.

This organization is performed by dividing the samples into c disjointed groups. These groups can be classes or clusters. For our particular application, we will part from the assumption that the historical data is labeled; therefore, only classes can be formed in the construction step. At first, we use the OCC approach, where: a) we separate the data by class, and b) for each class, we divide the data by clusters, which will be called microclusters (MC). We can use any clustering approach, whether k-means, spectral clustering, etc. Figure 6.17 illustrates a data distribution of three classes and their microclusters. In Table 6.8, we present the algorithm of this step.

When a new sample x_i is evaluated, the average distance between the sample and all the k nearest neighbors in each class X^c_{nn} is separately computed by using the

Macro algorithm

Input: Samples X, classes C
Procedure:

1. For c in C,
   - Identify $X_c \in X$.
   - Create m micro clusters using K-means and X_c.
   - Create the tuple in 6.9
2. End.

Output:M

Table 6.8: Pseudo code of the model build

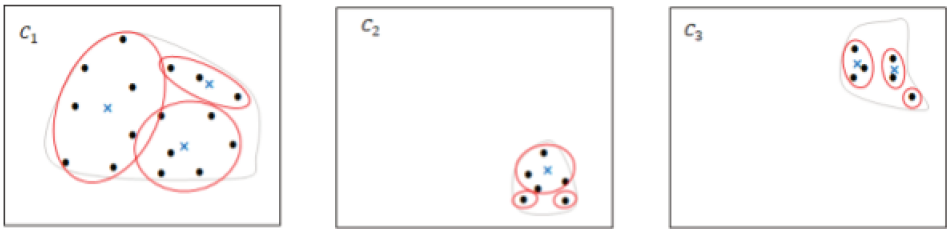


Figure 6.17: Graphical description of the model build step.

expression in Eq. 6.8. The sample is assigned to the class that gets the minimum distance as in Eq. 6.7.

We can think that this approach is an intrinsic KNN classification; nevertheless, the separation of the data by class and the fusion metric differentiates these approaches. We might expect similar performance between them. It is worth mentioning that improving the classification performance is not our aim. Our primary interest is to detect the areas of misclassifications or evolution for creating more capable classifiers. Classifiers based on non-distance approaches will be more suitable for this task, such as the classifiers proposed in Section 6.5.

6.6.4 Test

The test task will be a crucial aspect of our approach. This process will allow us to refit the model, in the sense that a new reconfiguration of the data might emerge from this step. We are going to base this procedure on the following assumption,

Assumption: The test dataset helps to determine overlapping areas between the one-class classifiers when they are misclassified in the test process.

This assumption allows us to detect overlapped areas between the one-class classifiers, and what is worth the microclusters implied in these zones, as it is illustrated in Fig. 6.18.

Given the overlapped areas, we define a set of micro clusters that are sharing the same zone as it is shown in Fig. 6.19. In this toy example, we detect two sets of microclusters that might be in overlapping: $S_1 = \{m_{12}, m_{21}\}$ and $S_2 = \{m_{13}, m_{31}, m_{33}\}$. These tuples of microclusters might indicate either class-imbalance or data-imbalance. In our particular case, we will assume that we are working with class imbalance, given

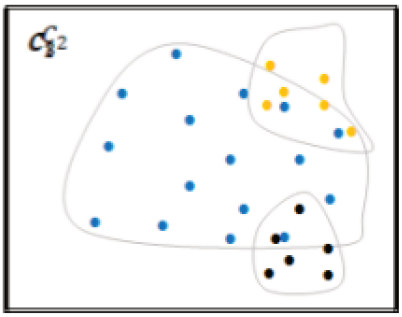


Figure 6.18: Graphical description of the overlap detected between three one-class classifiers.

the characteristics of the Internet traffic classification. Therefore, we can formulate the following assumption to correct this behavior.

Assumption: Micro clusters in a shared area or "overlapped" area that belongs to the majority classes will be deleted to perform resampling of the data and to improve the model's data representation.

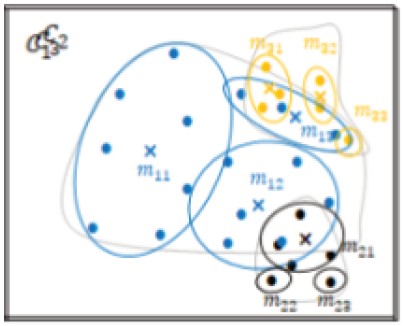


Figure 6.19: Graphical description of the micro clusters in overlap detected between three one-class classifiers.

This last assumption will allow us to perform a resampling, based on the misclassifications found with the test set. The result of this operation over our toy example is shown in Fig. 6.20.

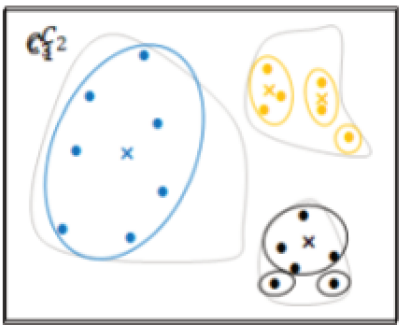


Figure 6.20: Graphical ilustration after resampling the data.

As an additional point, we consider the shared areas denoted by the sets S as valuable information for the classification system. Therefore, the data therein can be shared with the classification system to add expert classifiers in these zones. To

conclude this section, we present the algorithm in Table 6.9. In some cases, the resampling process is not executed because any majority class is involved in the misclassifications; these sets will still be considered for the online evaluation.

Macro algorithm

Input: Training samples X, real outputs Y and the name of the majority classes N
Procedure:

1. Evaluate X in M, and get the classes Y'
2. Detect the missclassifications with Y and Y'
3. Obtain the set of shared areas S
4. For s in S:
   - For c_m in N:
     - If c_m is in s, delete the micro clusters involved.
     - Reorganize the data in c_m, creating new micro-clusters.
   - end for
5. end for

Output:M

Table 6.9: Pseudo code of the test

6.6.5 Online evaluation

As it was previously mentioned, this approach does not have as an objective to correctly classify input samples. Instead, it tries to find similarities between classes to detect new behaviors. These solution aims to reinforce or support a more accurate classifier.

Generally speaking, in an online manner, the samples are placed to their more similar one-class classifiers. However, for those samples assigned to micro-clusters in S, they will be denoted as inconclusive by our model. In an online manner, we envisage to stock samples for each class in a batch. The user defines the size of the batch, and it will help to gather new behaviors of a class. When a batch is filled, the data in the *ILM* and the batch are sent to the *Model repository* to activate a reconfiguration process. The algorithm of this process is presented in Table 6.10.

The evaluation of the new sample x_i to our model will sightly change in the sense that the samples in the batches B will be considered to obtain the nearest neighbors, following the following assumption.

Assumption: Given a set of new samples in a period, that is being assigned to the same class, we can assume that a new or similar behavior is emerging for this class.

The average distance presented in Eq. 6.10 is changed to,

$$d_{nn} = d^{c}_{nn} + d^{b_c}_{nn} \tag{6.10}$$

Macro algorithm

Input: input sample x_i
Procedure:

1. Evaluate x_i in ILM, and get the class y'
2. If x_i is S,
   - Target x_i as inconclusive
   - End
3. Else,
   - Save x_i to its corresponding batch b_c.
   - If b_c is filled, empty b_c and send the new dataset $\{\hat{X}, \hat{y}\}$ to the classification system.
4. End.

Output:b_c

Table 6.10: Pseudo code of the online evaluation

where b_c is the batch of the class c and $d_{nn}^{b_c}$ is the minimum average distance of the k samples in the batch.

This new measure will allow us to partially consider the new "behaviors" included in the one-class classifiers over some time.

We present as follows the datasets selected to test and validate our framework.

6.6.6 Experimental results

The ILM has some essential parameters defined for the training process. In Appendix B, we listed their values and how they were selected for this application. In Table 6.11, we present the accuracy results after training and testing the ILM against its more similar adversary KNN. We can notice that the results are very similar to KNN's performance. As was previously mentioned, in this case, we are not trying to improve the accuracy performance, instead, we intend to represent the data correctly.

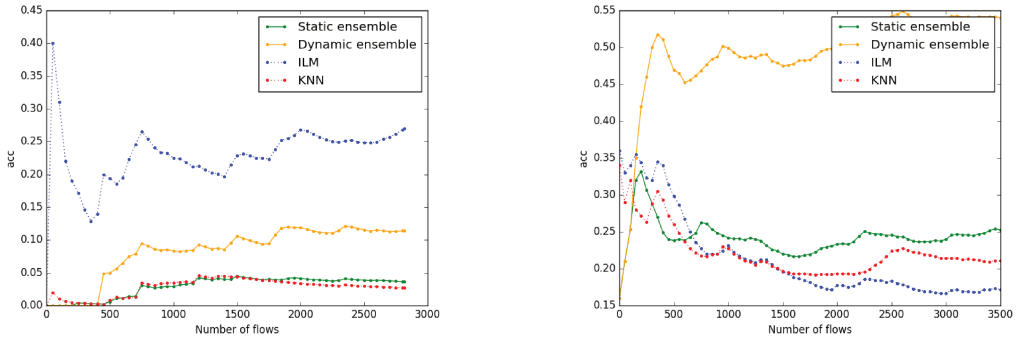
Classifier	PAM	VPN-NonVPN	SAT-gw
KNN	0.8783	0.9785	0.8634
IML	0.8643	0.9276	0.8653

Table 6.11: Accuracy result after evaluating the ILM and KNN models

On the other hand, to demonstrate that this approach can detect new class distributions, we present as follows several tests that will emulate new incoming classes and how the ML models behave. Most of the results of the next section will be displayed graphically with one image that will measure the accuracy every time that a new sample (of the new class) is evaluated. The setting of this test is detailed in Appendix B.

PAM results

The PAM dataset, built on an emulated environment, allowed the recollection of more than ten types of classes as it was described in Chapter 4, Section 4.4. In this case, we illustrate the classification results after evaluating a new incoming behavior from the class Streaming. Figure 6.21(a) summarizes the online conduct of the application YouTube. From the figure, we can notice that the original ensemble fails dramatically. On the other hand, the ILM started progressively correcting the classification system by adding new learners, who are experts in the class Streaming. A remarkable improvement of the accuracy is perceived after these modifications. The same experiment was reproduced for the Skype application, which belongs to the VoIP class. In Figure 6.21(b), we can notice that the results are very similar to the previous case.



(a) YouTube (b) Skype

Figure 6.21: Accuracy for two applications of the class Streaming and VoIP, with $B = 50$.

Additionally, we present the results after evaluating our approach with some applications of the majority classes, such as Browsing and P2P. In Figure 6.22, we can notice that both the static and dynamic ensemble give satisfactory results. We can also see that the ILM does not perform well for the majority class, which is expected due to the method was designed to favor the minority class. The resampling process deprives the ILM to get good accuracy with the majority class; the contrary case is found with KNN. This particular case does not cause a problem for our classification system due to the static classifiers are working adequately for the majority classes.

The results in Figure 6.23 demonstrates that our approach could also improve the accuracy for identifying new applications of the majority class, such as the case of eDonkey belonging to the P2P class.

VPN-nonVPN results

Similarly, we evaluate some non-tunneled and tunneled applications as new unknown behavior. However, this dataset presents some constraints because the authors collected a few samples of different applications. For this case study, we treated the tunneled and non-tunneled data differently, in the sense that a dynamic ensemble and an ILM is built for each case separately. We already demonstrated that that

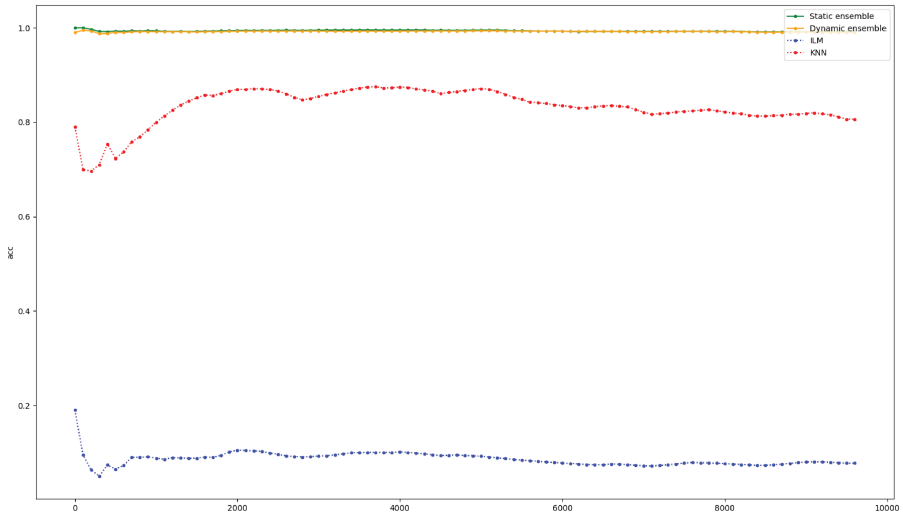


Figure 6.22: Accuracy for the application Wikipedia, $B = 50$

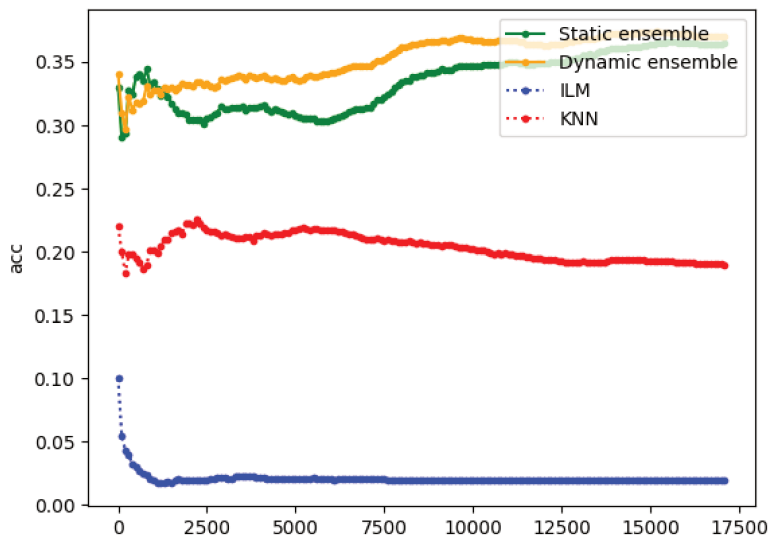
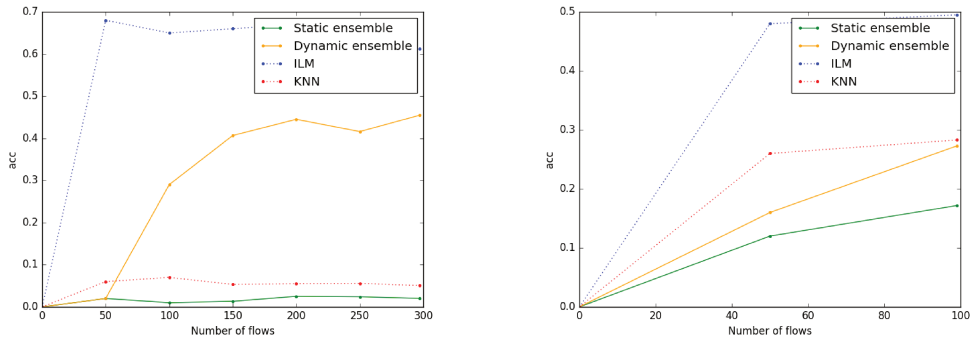


Figure 6.23: Accuracy for the application eDonkey, $B = 50$

tunneled and non-tunneled data could be easily separated using their statistical features.

For the non-encrypted case, we evaluate YouTube and NetFlix applications as a new behavior. We can notice from Figure 6.24 that similar results than the previous cases are perceived. In the case of the majority class, for the non-tunneled case, we evaluate several samples belonging to the Browsing class; these samples belong to unseen new applications. We can notice in Figure 6.25 that the classifiers behave correctly. Although the ILM was not able to place their correct class, the addition of

reactive classifiers does not affect the performance of the dynamic classifier.



(a) YouTube (b) NetFlix

Figure 6.24: Accuracy for the applications with $B = 50$

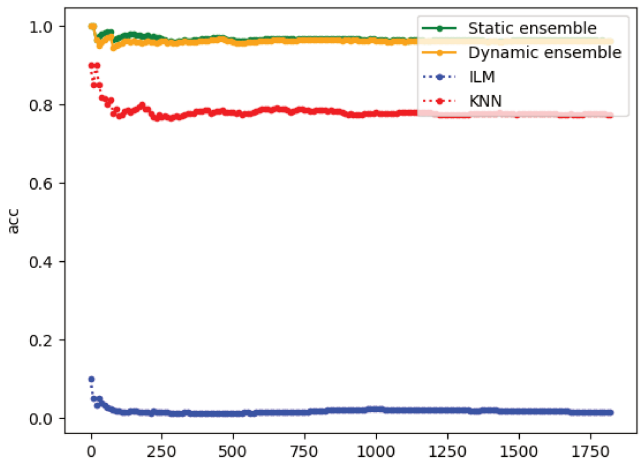
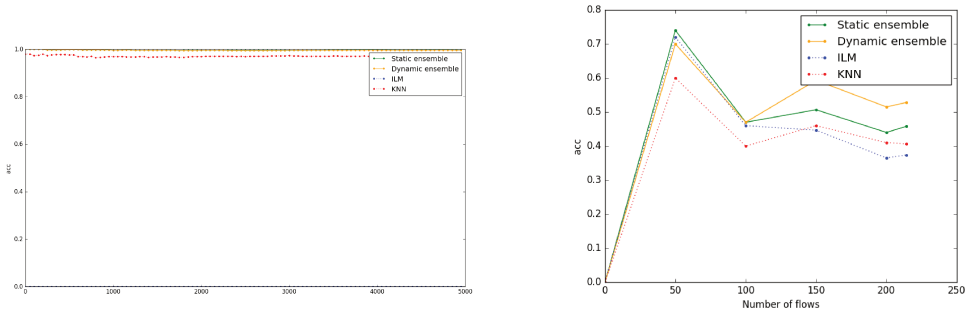


Figure 6.25: Results for the applications Amazon, Yahoo and others, $B = 50$

In the same manner, we evaluate tunneled Hangouts and YouTube applications as new unknown behavior. In Figure 6.26, we can see the results of these tests. In this particular case, the class VoIP belongs to the majority class, while the Streaming application to the minority class. To conclude, we found similar result patterns with the non-tunneled and tunneled data tests.

SAT results

We repeated the same experiments as in the previous cases, and the only difference remains in the treatment of the data. We merged the classes Video and Voice coming from the same application, let us call these QoS classes Interactive Skype and Interactive Facebook. The logic behind it is that the beginning of the connection (handshake for Video and Voice) is equivalent to the application of Skype or Facebook.



(a) Hangouts (b) YouTube

Figure 6.26: Accuracy for the applications with $B = 50$

Interactive Facebook and Skype are considered as new incoming behavior for the Static and Dynamic ensemble. In figures 6.27 and 6.28, we get remarkable good results by using our proposal. It is noticeable in Figure 6.21 that the accuracy can be easily elevated if enough input samples are seen form the same class.

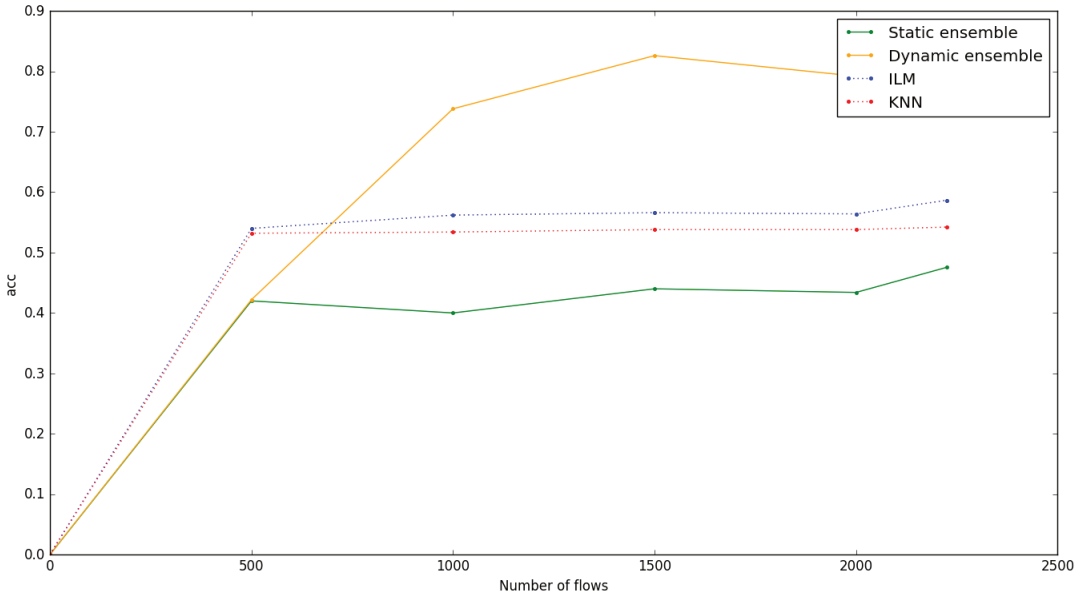


Figure 6.27: Results for the application Skype, $B = 500$

Table 6.12 shows the final accuracy of the previous tests. Additionally, we set experiments with validation tests to show that the generalization property is not lost. We can notice that static classifiers give a fair accuracy, while our approach is slightly higher. The samples of the new classes are randomly placed in the validation test set, and this affects their learning. However, we estimated that after some time, a new application could be detected by our approach, and this will depend on its usage on the Internet.

We would also take into account the modifications suffered by our approach in Table 6.13 with only the unknown application, and Table 6.14 with the validation test set. We can list the following observations:

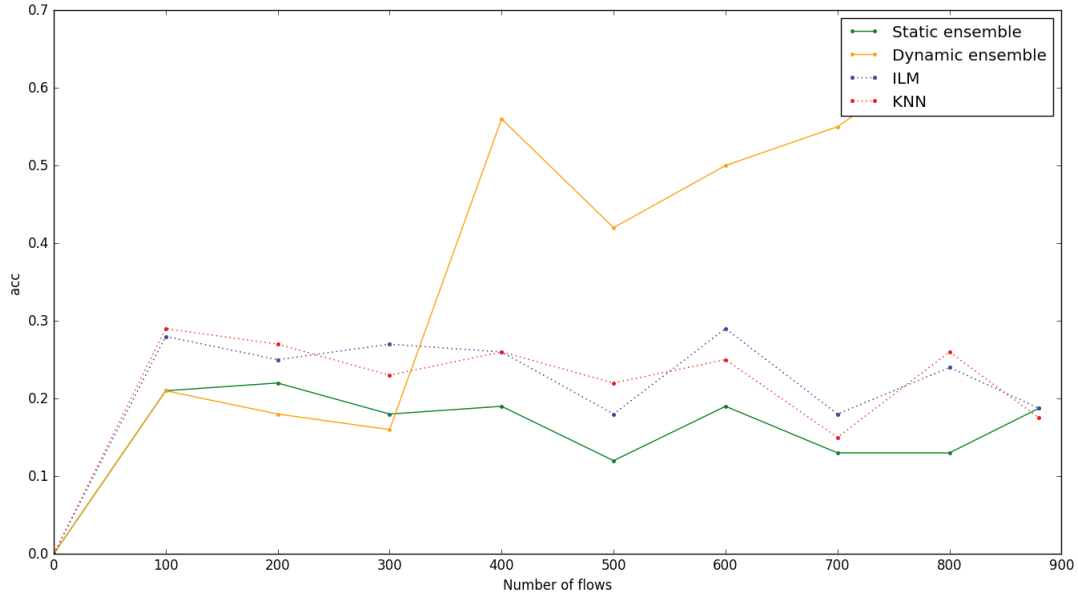


Figure 6.28: Results for the application Facebook, $B = 100$

	One unknown application		Validation set	
	RF	DE	RF	DE
Interactive Skype	0.4287	0.7083	0.4194	0.5168
Interactive Facebook	0.1829	0.4602	0.4983	0.5480

Table 6.12: Final accuracy after evaluating only one unknown application and a validation set.

- In both tables 6.13 and 6.14, few samples fell into inconclusive zones, and in consequence were filtered by ILM.
- In Table 6.13, we can notice that not more that 6 times an update was produced over the ensemble. Whereas in the validation case, in Table 6.14, for the Facebook application, 15 updates were executed.

	MA	S	added RCs	samples of the class	Total of samples	Filtered by IML
Interactive Skype	1	12	1	1840	1840	79
Interactive Facebook	1	12	6	880	880	37

Table 6.13: Online modification with one unknown application.

	MA	S	added RCs	samples of the class	Total of samples	Filtered by IML
Interactive Skype	1	32	1	2225	3049	0
Interactive Facebook	1	27	15	880	1812	142

Table 6.14: Online modification with the validation set.

For the SAT data, more promising results were found thanks to the good construction of this data. We believed that more challenges could be added to this approach when treating with more interactive applications at the same time.

To conclude, we present a more fine detail study in Appendix C about how the selection of base classifiers affects the performance of the new class identification. We

found out that with only Decision Trees as base classifiers, we can eventually learn one class at a time. We also made a study of the effect that the batch size might bring to this task. We notice that this will depend on the number of samples available for testing our method. However, we recommend a batch size considerably large in a real-world environment.

Up until this point, we already defined all the elements that conform to our Heterogeneous Classification System for Internet traffic. What we present in the next section is the perspective of this approach.

6.7 Classification System perspective

The former theoretical and experimental analysis provide a reconfigurable system that can somehow learn one minority class. 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 6.29. New historical behaviors captured from the Internet network might lead to design new *Flow discriminators* either by using ML techniques or classical approaches (such as port and protocol identification). Following, a dedicated classifier might correctly analyze each type of communication. Moreover, the class evolution is prevalent for any communication technology. Therefore, it will be necessary to set evolving models such as ILM to update the classifiers designed.

In Internet traffic classification, the detection of some classes is a crucial factor to mainly improve the QoS. In this case study, we found that the datasets available for the research community are limited; in addition to that, the publicly available data counts with class-imbalance. In this context, minority classes might not be well represented by ML models. The present investigation tries to find an approach that can learn one class at a time, in particular, that can learn the minority class. We remark the need for building dynamic models that can be shaped according to unlabeled online knowledge. Conversely, two classification systems work in parallel to cope with our objectives.

The results after synchronizing these two elements were promising, inspiring new future paths in this field. For instance, a concept drift detector can be added between the classification system and the ILM, to filter the samples that are considered as new knowledge. On the other hand, dynamic feature selection can be deployed, where a set of features can better describe a minority class than another one. To build the DTs, we can directly consider the trees produced by RF, and use them as base classifiers. Finally, it is important to make more efforts to adapt this solution to a multi-label classifier and other possible classifiers as it is shown in Figure 6.29.

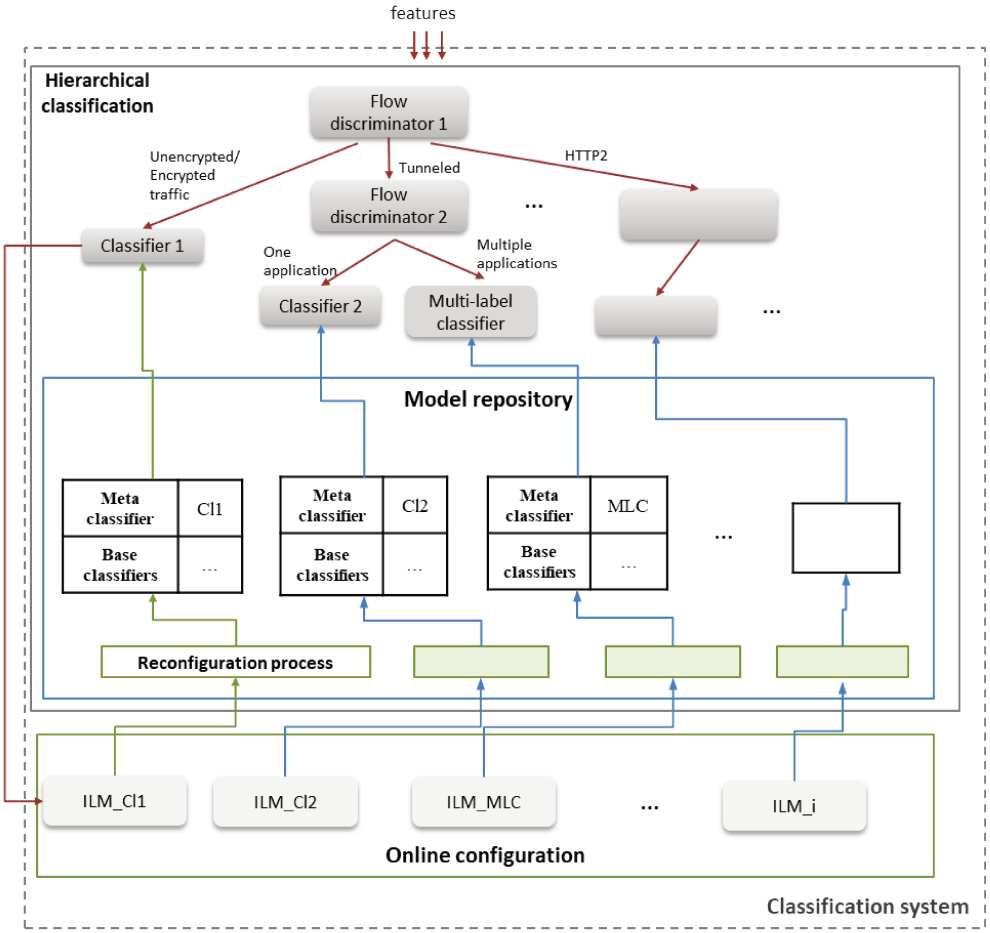


Figure 6.29: Graphical perspective of the Classification System

6.8 Summary

In this chapter, classification problems were addressed. To start, we build ML models that will serve as flow discriminators and classifiers in our Classification System. Following, we define the multi-label classifier that will be in charge of treating tunneled connections. We analyze the results of a multi-label classification and its reliability. Also, we present an approach to perform packet classification over tunneled flows with multiple tunneled sessions. Finally, we present a method that can learn new class behaviors in Internet traffic classification. This approach aims to learn one class at a time, and in particular the minority classes. We can outline the following discussions about this last approach.

- The classifier proposed is a classic ensemble of classifiers that allows the replacement and the addition of base classifiers. The new base classifiers are trained with historical data, and new unlabeled samples are considered as new knowledge managed by the ILM.
- The ILM is based on one-class classifiers, which in turn are modeled by KNN. This representation allows us to find new behaviors similar to the known classes.

In this sense, this model must deal with the class-imbalance through the resampling process.

- We demonstrated that the resampling process and modification of the ensemble do not affect the generalization properties of the classification system.
- In Section 6.6.6, we notice that our approach is not 100 % accurate for the new class behaviors. However, it shows better performance than the static classifiers and the KNN by itself. Conversely, it is reasonable to think that with more input samples, the solution can notably improve the accuracy.
- Although the amount of new behaviors per class is limited, we selected the most critical applications to be detected by the practitioners in this field. The results were satisfactory for each of the experiments.
- A lightweight implementation was proposed taking into account the requirements of the case study. We notice that if we replace the RFs by DTs as based classifiers, we also obtain a good performance. Besides, this gives us more flexibility to implement the solution.
- Finally, a possible extension of our *ILM* for multi-label tasks is feasible due to KNN and OCC that can be adapted to work as multi-label classifiers.

In addition to this, we described the perspectives of the Classification System designed. More *Flow Discriminators*, *Classifiers* and *ILMs* might be needed to cover other communication technologies. In this sense, more historical data must be captured to extend our approach. In the next chapter, we will cover more technical aspects to implement the ML solution into a network appliance.

Chapter 7

Towards the implementation of the Classification System

Contents

7.1 Introduction **121**
7.2 ARCADIA and Capella for the solution implementation . **122**
7.2.1 Logical architecture 122
7.2.2 Physical architecture 123
7.3 Implementation proposal **123**
7.3.1 Logical architecture 123
7.3.2 Physical architecture 127
7.4 Experimental results **129**
7.4.1 Experimental setup 130
7.4.2 nDPI vs hierarchical classification results 131
7.4.3 Load/Stress performance 132
7.4.4 Online Display 134
7.5 Notes about the QoS management **135**
7.6 Summary **136**

7.1 Introduction

After studying the theoretical aspects of this project, we come back to more technical matters that drive the implementation of our proposal. We acknowledge that few classification systems based on ML have been genuinely implemented in real network appliances [146, 88, 166, 140, 68]. Most of the works are deployed over home networks, cellular networks, routers or PoPs of large ISP. In Satellite Communications, few works address this task mainly due to the limited access to Satellite components and data.

In Chapter 3, we detected and established the requirements of our solution. The primary need was to provide for a classification system that can feed a QoS management architecture. What we would like to asset in this chapter is the potential functional and physical interactions required to deploy our proposal. We use the last

phases of the ARCADIA methodology to formalize the implementation of the proposal. The result of this analysis is a lightweight prototype that can be coupled to any network appliance where passive monitoring is permitted.

In addition to this, we need to submit our system to different scenarios to evaluate its performance. It would be ideal to have a real Satellite testbed system to execute performance tests. However, this kind of environment will be assessed in our collaborative project. In our case, we set the guidelines to test the classification system in a testbed environment placed on the Proxmox Virtual Environment as in Chapter 4. We will establish a simple communication between peers acting as source (Internet client) and destination (Internet server). A router will be placed between source and destination to intercept and to classify the Internet traffic. We will propose several scenarios to measure the overhead provoked by the classification system to the router. Finally, we conclude with some important points to be considered for QoS management by using the Classification System proposed.

This chapter is organized as follows. Section 7.2 defines the last two phases of the ARCADIA modeling process. Section 7.3 shows the implementation proposal with its logical and physical architecture. Section 7.4 presents an experimental test set on the Proxmox Virtual Environment. Several tests were carried on to measure the performance of the proposal. Even though the system was not tested over a Satellite Architecture, we state the guidelines to asset this system in the GW or ST component. Finally, in Section 7.5, we establish some considerations about how to interpret the classification outputs to perform QoS management. Section 7.6 summarizes this chapter.

7.2 ARCADIA and Capella for the solution implementation

In Chapter 3, the ARCADIA methodology and the Capella software modeling tool were introduced for four main phases. The two first phases develop the system requirements while the left ones the design and implementation. We notice that in the second phase, the resulting architecture represents a prototype composed of system functions. These functions are composed of internal activities that allow them to achieve their goals. Additional concepts and viewpoints must be introduced to define the physical elements that will implement such a solution. Therefore, we briefly describe the last two levels of modeling with Capella.

7.2.1 Logical architecture

In this phase, we enter a more detailed description of the system design for its later implementation. Among the concepts here used, we find,

- **Components**: new components that will allow extending the system beforehand designed.
- **Functional exchanges** between components.

- **Function allocation** into components: this is the final result of this phase, and it allocates all the components, systems and actors in only one diagram.

7.2.2 Physical architecture

The main objective is to design a physical reference architecture, reusing all the diagrams of the logical architecture.

- **Behavioral components** refine the operational components and implement their behaviors.
- **Implementation components** supply the resources needed by the behavioral components.
- **Physical links** establish connections between the implementation components.
- **Physical architecture** is the final result of this phase, where all the components are allocated in the same diagram.

7.3 Implementation proposal

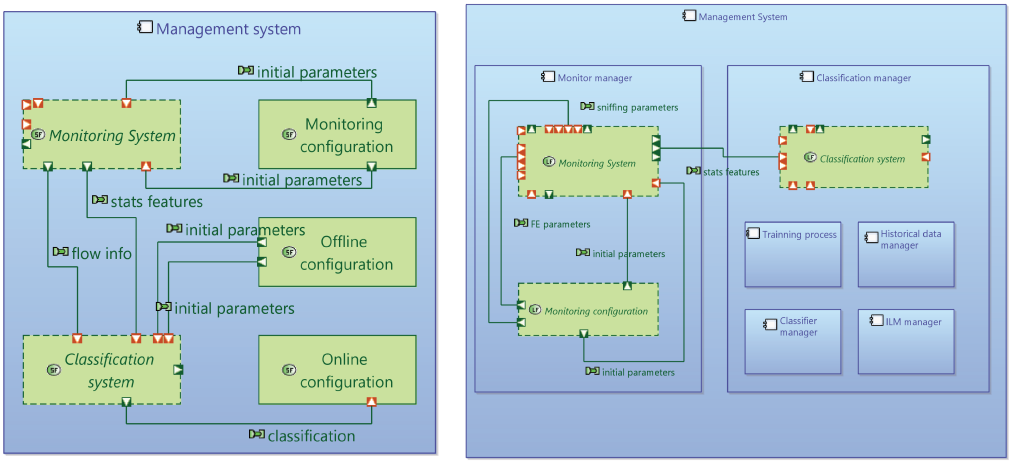
Once the requirements are fully defined, we can introduce the proposed implementation. To build the two last phases of the Capella model, we also make some assumptions regarding the resources available in a real Satellite Architecture.

7.3.1 Logical architecture

In this section, a refinement of the system proposed is carried out. In Figure 7.1, we make a comparison between the system and logical analysis. The subsystems proposed will define how the components of the *QoS management system* will work. Only two points will suffer major modifications,

- The *Offline configuration* will be developed by the *Training process* and *Historical data manager* components
- The *Online configuration* will be unfolded between the *Classifier manager* and the *ILM manager* components.

In the following sections, we will only detail the operational functions of the offline and online configuration processes. The reader can notice that the other operational functions (such as *Monitoring system* and *Classification system*) were already detailed in the System Analysis in Chapter 4. These functions suffer small technical modifications that are not worth to specify at this point.



(a) Phase 2: System Analysis

(b) Phase 3: Logical Analysis

Figure 7.1: Transition from the system to the logical analysis.

Offline configuration: logical function

The sequence of steps, for achieving an offline configuration in Figure 7.2, agrees with the steps described in Chapter 3, Section 3.3.3. Feature extraction for the Flow discriminator (either *FD1* or *FD2*) model and the Classifier (either *Cl1* or *Cl2*) are performed. Data processing is executed when needed. The data is separated into a training and test set for building the ML models. Once the models are validated, they are sent to the external components such as the *Flow discriminator* and the *Classifier*. These functions are placed into the operational architecture to see the connection with the other actors, as shown in Figure 7.3.

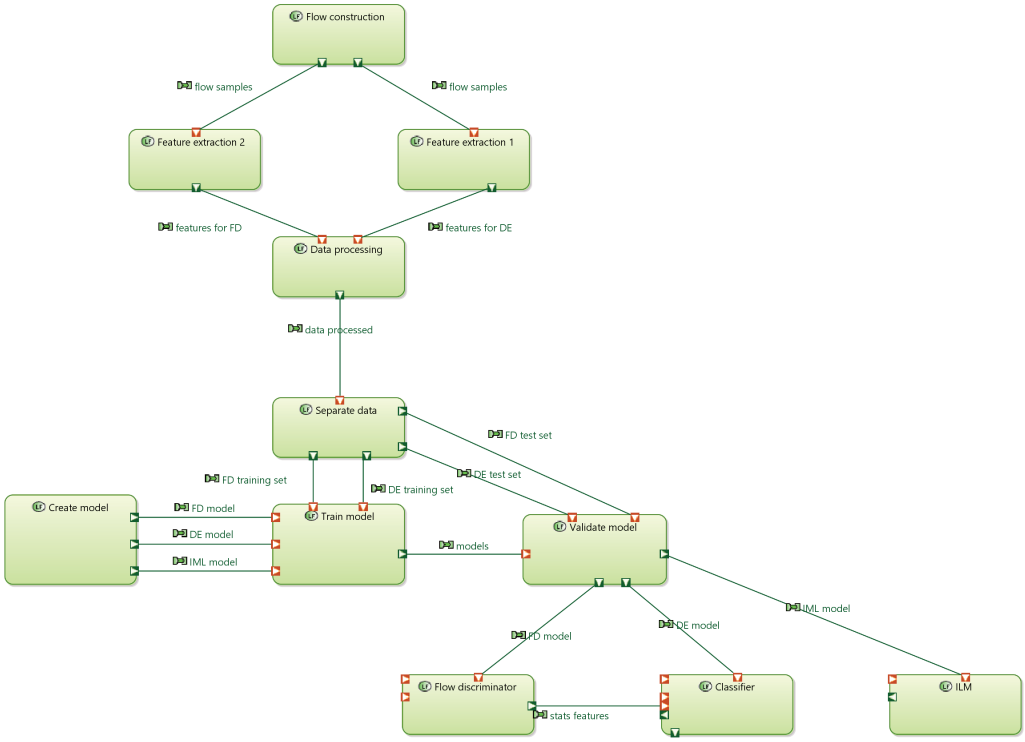


Figure 7.2: Logical function of the Offline configuration system.

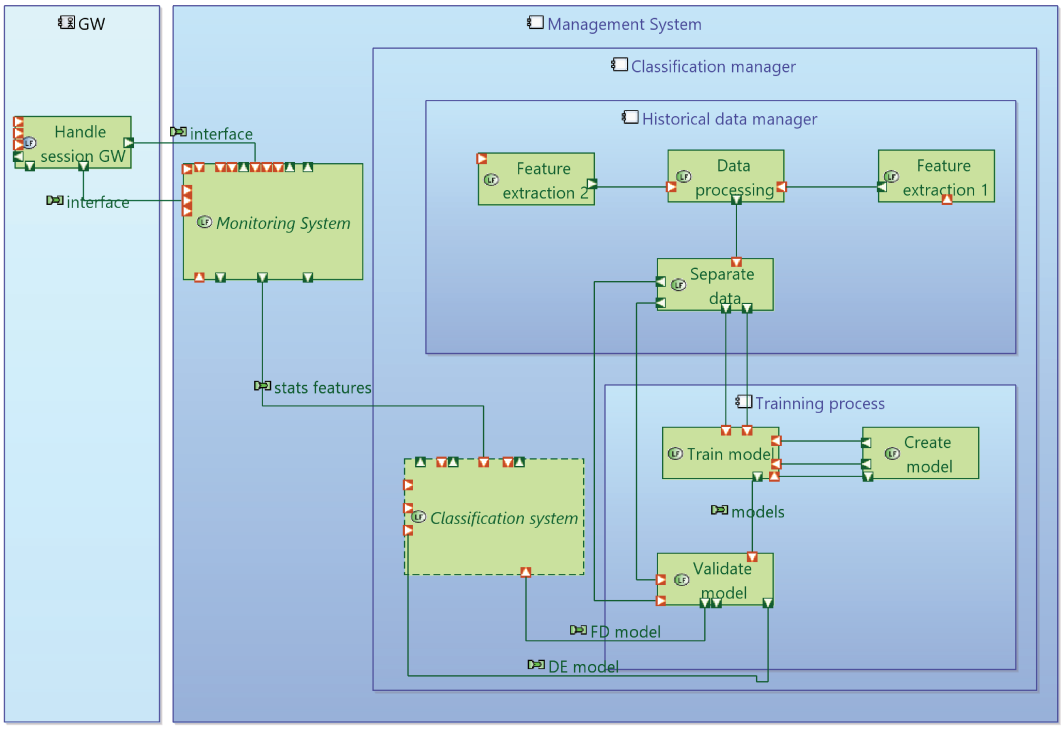


Figure 7.3: The Offline configuration system integrated in the Logical architecture.

Online configuration: logical function

In an online manner, we would like to retain flow behaviors to detect new distributions of a targeted class. In this sense, in Figure 7.4 ,we present the logical function of this

task. Statistical based features are delivered to the ILM, which in turn will evaluate these values and assign them into a batch. When the batch of the targeted class is filled, an RC classifier will be trained and added to its corresponding the *Classifier*.

These operations are displayed in the Logical architecture as in Figure 7.5.

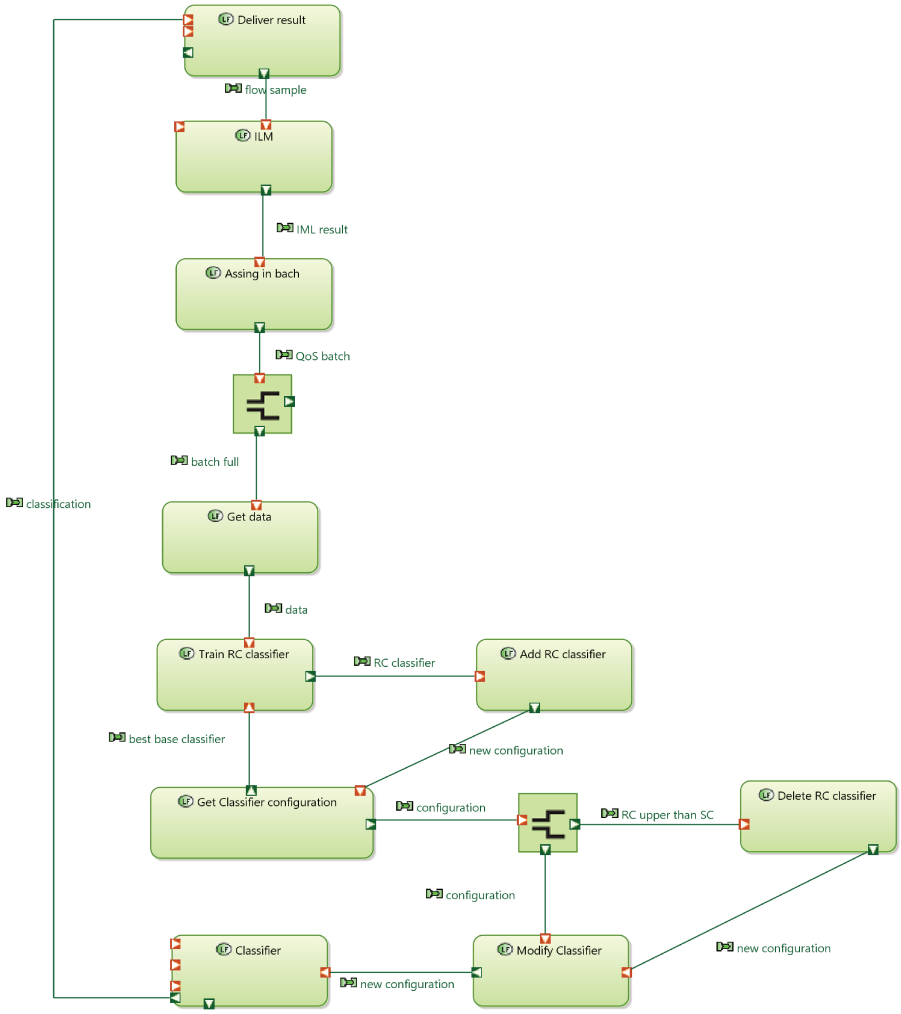


Figure 7.4: Logical function of the Online configuration.

The final result of this phase is a logical way to approach the physical implementation of our prototype. We notice that all the functions proposed were already theoretically developed. In the next section, we show possible implementations of our prototype, and also the tools used and needed for the development of this system.

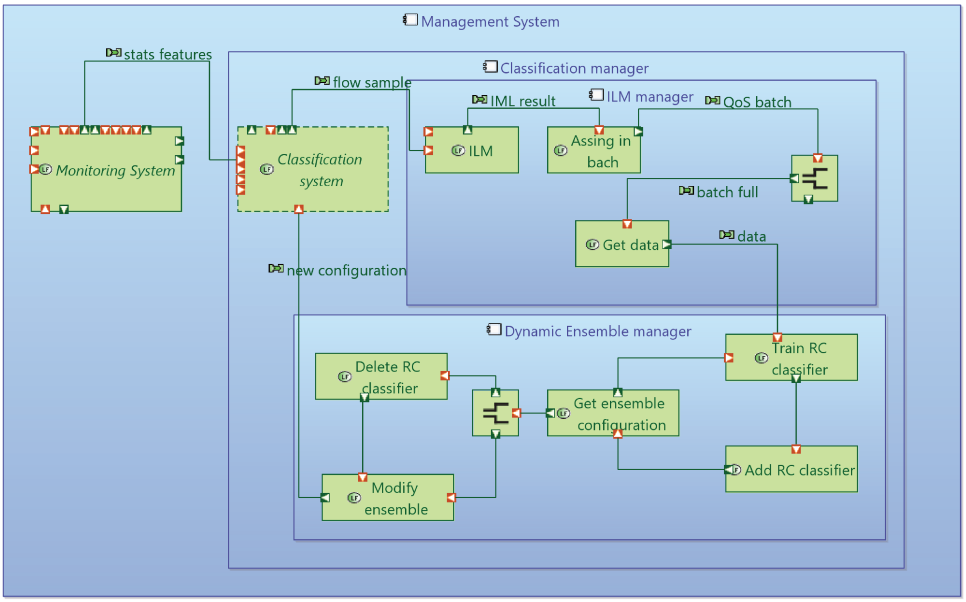


Figure 7.5: The Online configuration system integrated in the logical architecture.

7.3.2 Physical architecture

To culminate, the implementation proposal is presented in Figure 7.6. We define two new 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. In addition to this, we will need 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 API*, we use existing solutions such as Libpcap [108] for performing the sniffing. Then, we add the *Flow Metering* and *Feature Extraction* behaviors.

As an 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. For instance, we present a configuration suitable with the ST actor in Figure 7.7. In this figure, we remarked that the *Management server* is centralized; however, this configuration can be easily replaced by disposing of a *Management server* component for each the GW and ST.

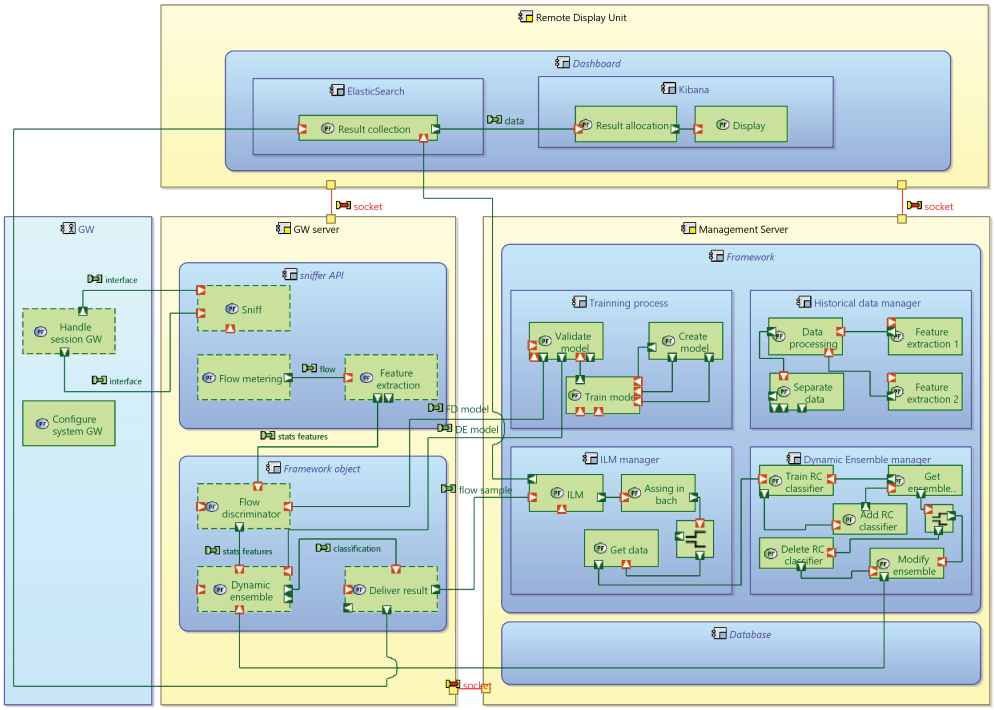


Figure 7.6: Physical architecture of the ML solution in the GW.

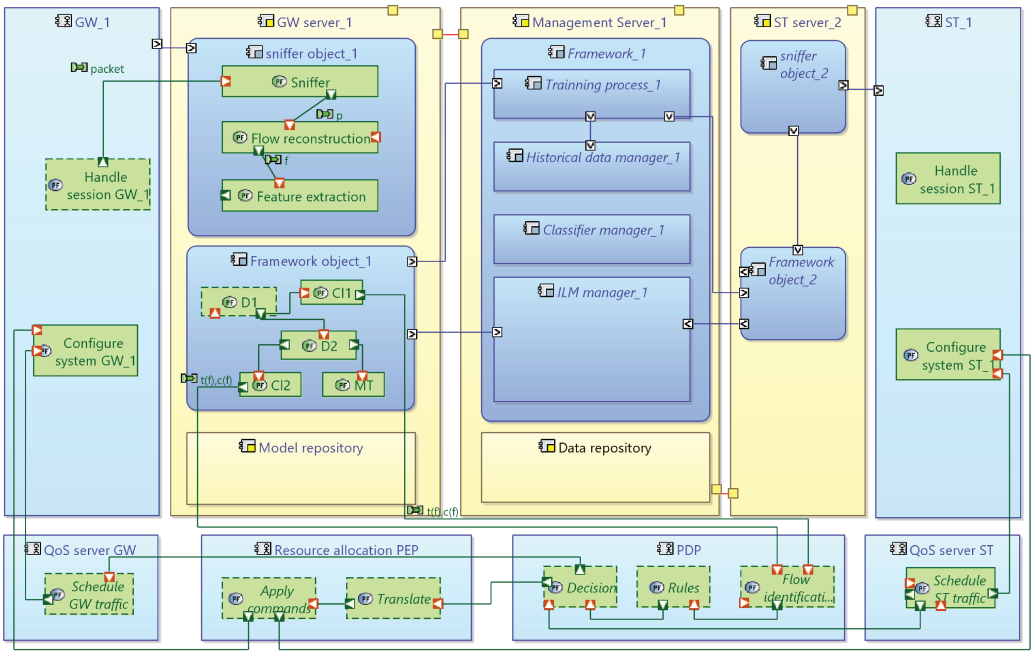


Figure 7.7: Physical architecture of the ML solution in the GW and ST.

Implementation details

The implementation of the solution is partly conceived in C aided by some libraries included in nDPI and in Python. The most important libraries developed are listed below:

- **Framework**: this library coordinates the operations between the classifier and the incremental learning model. Here the training process can be performed for the defined models (Flow discriminators, classifiers and ILMs). In addition, this library coordinates the connection between the online classifier and the offline reconfiguration. This library is developed in Python aided by scikit-learn library [138], and it contains other classes such as
  - **Ensemble**: it gathers the classifiers. Functions such as predict, add ensemble, delete ensemble, save in a file, load from a file, among other useful functions are developed. This library was developed in C, the details about such implementation are detailed in Appendix D.
  - **ILM**: similar to the previous case, it gathers the incremental learning model. Functions such as predict, assign into a batch, save in a file, load from a file, among others useful functions are developed. This library is developed in Python aided by scikit-learn library.
  - **data-utils**: it comprises supporting functions to treat historical and online data. This program was developed in Python.
  - **Dashboard-utils**: It helps to correctly send the data to the *Remote Display*, managing Elasticsearch [53] principally.
- Sniffer API: It is in charge of performing passive monitoring, including the feature extraction process. Within this C API, we find,
  - **Libpcap**: This library manages the sniffing process given an interface, IP address, etc.
  - **flowManager**: It takes the binary packets and performs a metering process. This allows us to differentiate one flow from another. In this particular case, we use a hast table to find existing flows or register a new flow tuple. This library was developed using the example implementation called "ndpiReader" from nDPI which offers an efficient flow metering process.
  - **Features**: it complements the previous library by computing in an online manner the statistical-based features. It also has some complementary functions for treating the different types of encryption. It also handles the features needed for each classifier.

To sum up, we present an ML system that can be integrated into Internet traffic architectures, being the Satellite Architecture of our primary interest. The proposal presented can be comparable with existing DPI solutions, which offer a portable software solution for Internet traffic inspection. In the next section, we perform several experiments to validate our proposal.

7.4 Experimental results

In this section, we present an implementation proposal in a cloud-based emulated network. The main objective is to measure the prototype performance and to give

guidelines to test our approach over a Satellite network component either in the GW or the ST.

7.4.1 Experimental setup

Using the Proxmox Virtual Environment, we create a testbed on the cloud (OVH cloud provider). We set the components in Figure 7.8 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. Let us say that this router is equivalent to the GW server proposed in Figure 7.6. Connected to this router, we placed a management server that will hold offline processed, 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.

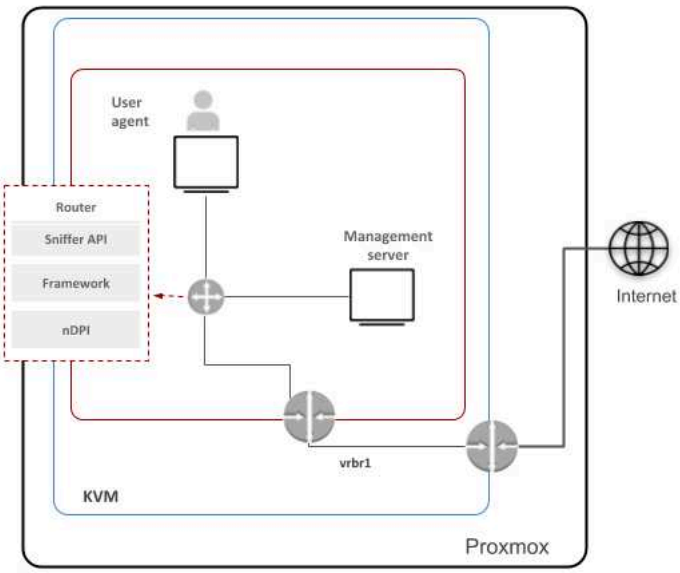


Figure 7.8: Implementation proposal on Proxmox.

If we go deep into the router appliance, we can illustrate in Figure 7.9 the internal process to perform the Internet traffic classification task by our prototype. Briefly speaking, Libpcap captures a copy of the Internet packets which in turn are organized into flows by using a hast table. Following the feature extraction process is executed, and the ML solution can be evaluated. In parallel, we set up nDPI to compare both approaches.

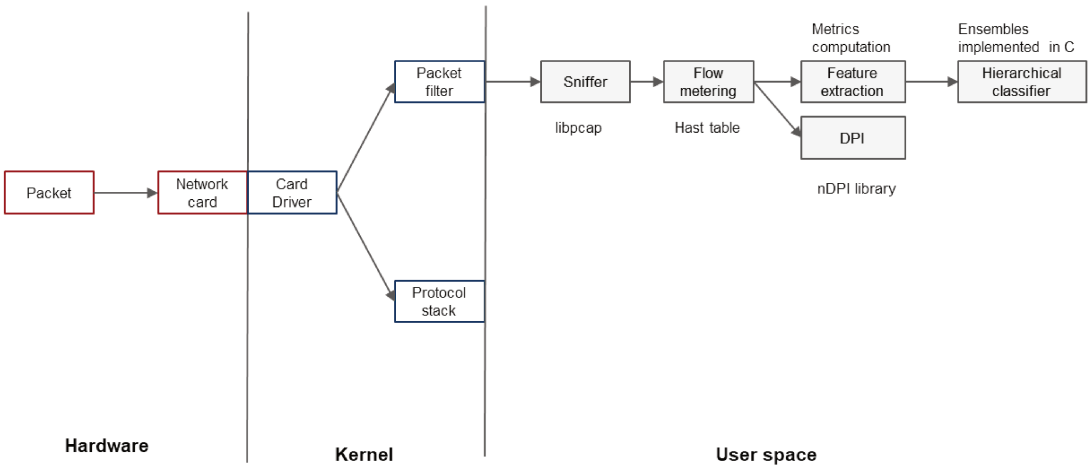


Figure 7.9: Implementation proposal on the router.

7.4.2 nDPI vs hierarchical classification results

In this section, we measure the classification capabilities of our implemented prototype against its more close competitor. The data used to train our proposal is specified in Table 7.1. All the data is used to build $D1$ while the rest of the data is adapted according to their objectives. In order to build $Cl2$ and $D1$, we set $\Delta t_1 = 10ms$ and $\Delta t_2 = 100ms$.

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

Table 7.1: Data settings for building the classifiers.

Following, the complete framework was implemented in C (this implementation is described in Appendix D). In Table 7.2, 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. Regarding the response time of the classifiers, in Table 7.3, 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 7.2: Accuracy evaluating the test data

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

Table 7.3: Average packets per second processed pkt/s

On the other hand, we would like to show in Figure 7.10 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.

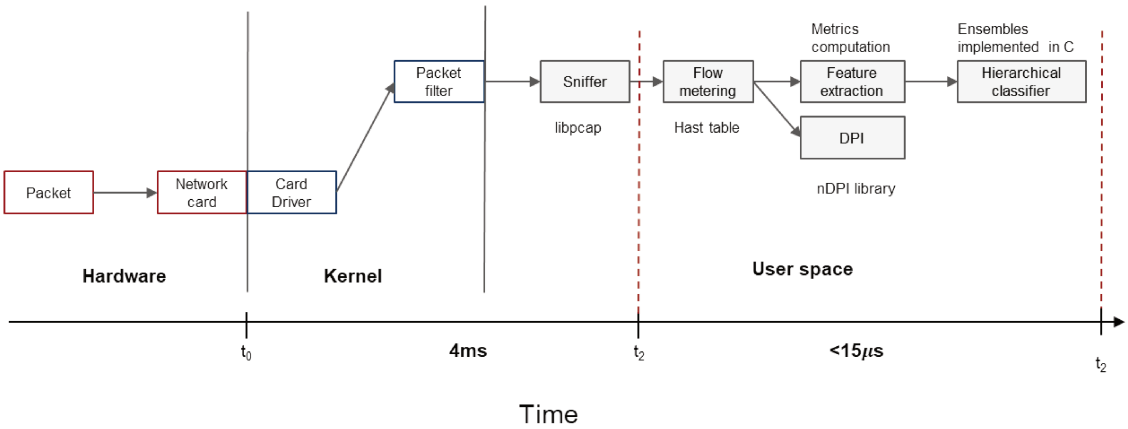


Figure 7.10: Time analysis of the classification process.

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 [40]. In our particular case, DPDK can be programmed as a kernel-bypass toolkit as it was already proposed by nDPI in its architecture [165].

7.4.3 Load/Stress performance

The main goal of this section is to know the load provoked by our approach under different scenarios to the router. In this sense, our router is put through different conditions by varying the data rate and the number of flows. To do so, we used *iPref* [83], which is a tool dedicated to measuring bandwidth on IP networks. This tool allows us to define the desirable flow packets at a specific data rate, transmitted from one VM to another one. In Table 7.4, we show the configuration selected: the data rate (Mbps) and the number of flows sent from the client to the router, and the results of such tests: the RAM and CPU consumption, and the input bytes perceived by the router.

It is important to mention that the tests are limited to the cloud-servers' Internet bandwidth which is around 3.6 Gbps. In a more powerful architecture, these tests

Datarate (Mbps)/# flows	% RAM memory	% CPU consumption (avg)	% of input bytes (Mbps)
1 Mbps/100	4%	2.81 %	101.28
1 Mbps/1000	4%	5.97 %	524.64
15 Mbps/10	4%	2.86 %	78.96
15 Mbps/100	4%	8.67 %	1019.84
80 Mbps/10	4%	10 %	620
500 Mbps/10	4%	17 %	2598.8

Table 7.4: Load/stress performance

can be scaled to perform more fine studies. Nonetheless, the results in Table 7.4 let us remark the following aspects:

- The performance of the solution is mainly correlated to the specifications of the machine. In addition to this, our tests were limited by the cloud-server network, we achieved to send around only 3.6Gbps, and in some cases, we incurred on information loss.
- The load fells under the sniffer and flow metering processes. For instance, the sniffer has to deal with a big load of information at the same time with threaded processes. In the same manner, the flow metering also requires threaded functions for fast flow search and flow classification.
- The average CPU consumption rises steadily as the datarate increases. This is typical behavior caused by the number of preprocessing tasks in the kernel to treat the raw packets. Also, libpcab uses only one core that limits the packet capture speed. This problem has been widely studied; for instance, the work in [8] shows that libpcap causes an exponential increase of the CPU consumption as the bit rate increases.
- Our system only takes the first 15 packets for aggregated flows, which might alleviate the memory and CPU consumption. Once a flow is classified, it is marked as classified, and it is never processed until a time-out is reached.
- 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 7.5. 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 7.5: Memory consumed by our proposal and by nDPI

Future studies should be carried out to enhance the available resource usage. For instance, CPU usage might be decreased by bypassing the default packet handling

mechanism of the kernel with tools such as PF_RING [1] and DPDK [40], among others. 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, our ongoing-project will instance these tests to perform a more extensive assessment of the system. In the next section, we propose an alternative way to interpret the Classification System results through a dashboard.

7.4.4 Online Display

In order to complement the results of the classification system, we can forward the status of the network to a data visualization plugin. This dashboard was set in the *Management server* for simplicity. Elasticsearch [53] was used to the connection and online update with the dashboard. For instance, in Figure 7.11 proposes a dashboard that counts with five data representations,

- DE figure: in the left hand, we can find the ensemble classifications over time. The possible class values are set according to small data captured from the virtual environment.
- ILM figure: in this figure, we can keep recollection about the batch of data that it is being captured for the ILM. Although, in this case, only one class of interest should appear; we displayed all the results that ILM provides.
- DE pie: it summarizes in a pie chart the classification results.
- count_updates : it is the number of flows processed.
- table_bytes : it is a summary of the bytes processed by the classification system.

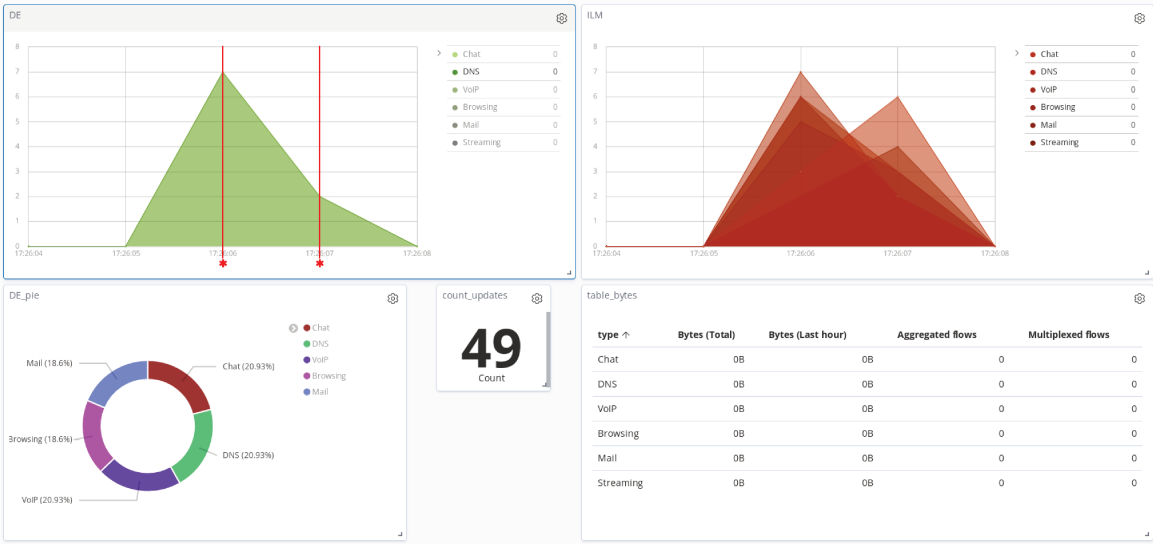


Figure 7.11: Dashboard of the proposal.

[1] https://www.ntop.org/products/packet-capture/pf_ring/

Although this solution is not operational, it is a good possibility to monitor the classification system results. The dashboard is flexible and easy to modify.

7.5 Notes about the QoS management

In this section, we set a discussion about the use of our solution in a Satellite Architecture for QoS management. As we previously mentioned, it suffices to place the classification system over a network appliance that permits traffic monitoring. For instance, in the GW component, the classification output is signaled to the PDP 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 also be benefited from this action. To avoid this, a classification per packet task is desired. The result obtained for the packet classification (in Chapter 6 Section 6.4.3) are promising and might solve the problem for some protocols that allow per packet prioritization.

In tunneled connections, the prioritization per packet means a reordering of the packets that is not always feasible. The IPSec VPN and OpenVPN allow packet reordering with some constraints. For instance, OpenVPN with udp as transport protocol allows reordering of the packets within a certain fixed sequence number window; contrary case occurs when tcp is used as a transport protocol. Any reordering process for OpenVPN tunnels with tcp will be considered as cyber-attacks, and the packets will be dropped as specified in the reference manual [2]. In Figure 7.12, we illustrate the approach proposed for each protocol with OpenVPN.

In addition to this, we need to be sure that the QoS requirements are satisfied on time. For instance, according to [84], VoIP and Interactive video applications are susceptible to delivery delays. To be specific, they can tolerate around 100ms of delay; whereas, another important class such as Video streaming no more than 10s. We notice that the classification task can be achieved in $15\mu s$, giving sufficient time to treat those sensitive classes.

[2] https://openvpn.net/community-resources/reference-manual-for-openvpn-2-4/

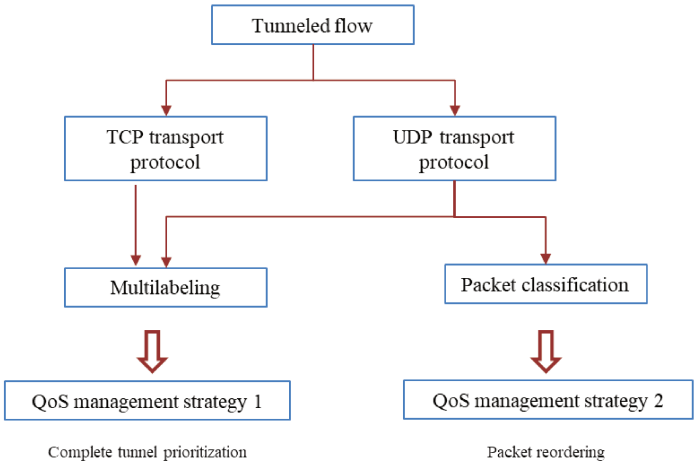


Figure 7.12: QoS management for OpenVPN tunneled connections.

7.6 Summary

This chapter presented the final phase of our project, which is the implementation and evaluation of the proposal. We culminate in the two last stages of ARCADIA to formalize the design and implementation of the proposal. We define all the elements designed to give life to the physical architecture's components. Afterward, our approach was tested over an emulated platform in which we set several conditions that aided to evaluate its benefits and limitations. In this sense, we set some recommendations to test this approach over an emulated or real Satellite architecture that will be covered in our collaborative project.

The experiments let us conclude that the proposal can be comparable in terms of accuracy and overhead with an existing DPI solution. We tested our approach in the GW component, with data captured from an emulated Satellite platform. This approach outperformed in accuracy and processing time to a well-known DPI solution. We displayed the needs of tuning the framework proposed, given different Satellite network conditions. In addition to this, we gave some insights into the effect of the traffic classification over the QoS management architecture. This chapter closes the development of this research work; therefore, the next chapter concludes and shows the future perspectives of our scientific proposal.

Chapter 8

Conclusions and perspectives

Contents

8.1 Introduction 137
8.2 Conclusions 137
8.3 Perspectives 140

8.1 Introduction

In this chapter, we summarized the results of our research project by presenting a set of contributions in Section 8.2. In this sense, we offer the conclusion of our work, along with the lessons learned in this Ph.D. thesis. Finally, in Section 8.3, we present the perspectives and highlight new future research directions.

8.2 Conclusions

ML has become a valuable tool for knowledge extraction in different domains of science. Although, the main procedures, to achieve knowledge extraction through ML, are easy to comprehend; they might change and increase in complexity depending on the application studied. In our particular case, we intended to design Internet traffic classification techniques for QoS management in Satellite Communications. This thesis project tackled the problem from different methodological points of view. The main principle was to follow the steps to achieve ML: i) problem description (architecture, actors, activities, etc.), ii) data collection, iii) feature extraction, iv) classification system design and construction, and v) implementation. This way of approaching our research work allowed us to provide contributions by solving inherent challenges in each step, as it will be detailed as follows.

Problem description: ML solution in a Satellite Architecture

The structure of Satellite Communications presents a high complexity that yields us to study an abstraction of its components and interactions. To do so, in Chapter 3, we theoretically propose a Satellite communication architecture; as well as, all the necessary elements needed to perform QoS management and to integrate an ML-based classification system. Following, this theoretical approach allows us to deliver

more formal modeling by using a model-based software engineering methodology and tool. The results are translated into the needs, the internal components, and the functionalities of the system.

We found out that an ML-based classification solution should cover some components to offer an accurate and possible operational solution. Among these functionalities, we have:

- Offline configuration component: On one hand, building a classification system from scratch can be costly for the final users of this solution. Therefore, it is necessary to provide accessible modules that perform this task automatically. On the other hand, to build an ML-based solution, historical data is required, and special attention needs to be taken into account regarding this subject.
- Hierarchical classification system: the principle of divide and conquer is valid for this item. We propose to decompose the problem of traffic classification into small classification problems that might improve the accuracy and comprehension of the system.
- Online configuration component: the evolution of the Internet network is present; therefore, ML-based solutions should foresee this matter. We propose a component that can somehow offer a self-reconfiguring structure.

The integration of an ML-based classification system in a Satellite architecture allowed us to have a global view of the problem affronted. Besides, this study permitted to establish some boundaries for our solution. In our literature review, few works arrive to deploy this solution correctly, and some of the main challenges were no covered. Hence, our proposal becomes our first contribution in this field. Our second contribution addresses data production for building our solution.

Data collection: Toward automatic Internet traffic generation and collection

The central knowledge base of intelligent systems constitutes significant historical observations of a problem. No matter the domain in which ML is applied, this is a crucial need that contributes to the success of this approach, and Internet traffic classification does not escape from it. We saw that few publicly available data is available, forcing us to propose a cloud-based platform for the Internet traffic data generation. This platform served as proof of concept to later be reproduced in an emulated Satellite architecture. The resulting dataset has become a valuable source of information not only for our research project but also for future works in this area.

Feature extraction: How to describe Internet communication streams

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 propose an enhancement of these statistical properties by introducing new statistical descriptors based on LR models. In addition to this, 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, which becomes our next contribution.

Classification System design and construction

The core of our classification system is a classification per level kind of approach. In this sense, we tried to divide different Internet traffic behaviors by using *Discriminator* classifiers. Following, more specialize classifiers (such as *Classifiers* and *Multi-label classifiers*) will be in charge of predicting the application name of an Internet stream.

Moreover, in the requirement analysis of our system, we remarked the need to have a component that can detect the evolution or distribution change of a QoS class. To do so, we propose an Incremental Learning Model (ILM) that can modify the *Classifier's* structure when evolution needs are detected. Such *Classifier* is represented by a classic ensemble that allows the replacement and the addition of base classifiers. These new base classifiers are trained with historical data, and new unlabeled samples are considered as new knowledge managed by the ILM. Also, we propose a lightweight equivalent implementation that uses Decision Trees as a base classifier. More formal aspects of our implementation approach are given in our next contribution.

Implementation approach

At this point, we culminate in our research project by formalizing the implementation architecture with ARCADIA and Capella. In this phase, all the proposed functionalities are placed in physical and software components. However, before continuing, we consider the fact that in Internet traffic classification, the optimal use of the resources is a crucial and essential factor to deploy any software solution. In our case, we require to integrate a new module that might need physical equipment and consume memory and CPU to work correctly. Following this line of ideas, our implementation fulfills this requirement by deploying online functionalities in a low-level programming language. Whereas, more sophisticated features that do not require fast online responses are coded in a high-level programming language.

To validate our implementation, we submit 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 in our collaborative project. In addition, we gave some insights into how the QoS management architecture should treat the traffic classification.

8.3 Perspectives

The contributions achieved during this investigation introduced novel classification techniques for the field of Internet classification. Up until this date, the company Thales Alenia Space does not count with an ML-based Internet classification system. Therefore, this study opens the doors to a variety of new research directions that can complement and refine our approach. Among the most important points to address, we listed:

i Enriching Internet data:

The cloud-based platform gave us the facility to emulate Internet communication. Nonetheless, only a small fraction of the most used worldwide applications was launched and studied. In this sense, more efforts to produce more labeled Internet communications are needed covering the majority of applications found in Satellite Communications. For instance, the system can be extended to generate data with other protocols not studied in this investigation, such as HTTP2 and IPsec.

ii Covering more types of communication streams:

As a consequence of the expansion of the historical dataset, different featuring engineering approaches might be designed to deal with the features of the new protocols.

iii Expanding our Classification system:

We already outlined the perspective of our Classification System in Chapter 6. As a summary, we found that we could expand the hierarchical levels of our system to include more applications and protocols behaviors. Indeed, we could also propose and validate ILM for each new *Classifiers* designed. In this context, it is important to say the ILM of the multi-label classifiers remains as a future work that can be quickly undertaken.

iv Evaluating and implementing our solution in different Internet network configurations:

ML solutions are most of the time biased to only work correctly in the same or similar environment where the data collection process was performed. In particular, Internet traffic classification is very sensitive to the network configuration due to they change the statistical behavior of some properties such as the IAT. This variation can cause misclassifications of an ML-based classifier trained with data with other conditions. Then, a transformation over the features needs to be performed to transition from one network condition to another. As a perspective, we propose to evaluate the differences between different Satellite networks and to find a mathematical relationship that can harmonize this difference. The main objective is to have a classification solution that can be used in different network conditions.

v Evaluating and implementing our solution in QoS management architectures:

Part of this final perspective will be accomplished in our collaborative project. In this matter, it is necessary to apply all the guidelines proposed in the implementation chapter. Although QoS management was indeed out of the scope of this investigation, it stands as the principal motivation of our work. Therefore, it will be adequate to evaluate the impact of our solution over QoS management architectures.

Appendix A

Packet categorization

This appendix presents the decision tree used to define the categories of the Internet packet. Figure A.1 shows the resulting decision tree.

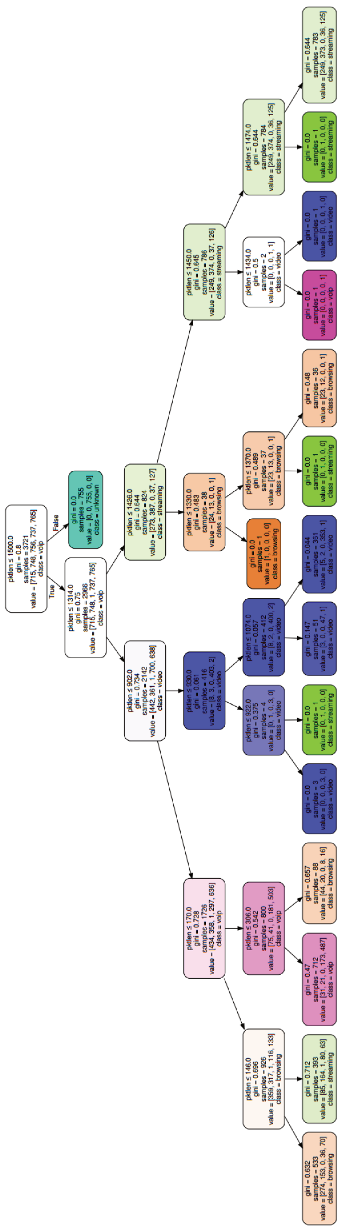


Figure A.1: Decision tree udp and udp_sec.

Appendix B

Classification system settings

We proposed different scenarios to test the reliability of our framework. In this section we present the experimental set up for the ILM.

B.1 Offline training

In this particular case, it is necessary to evaluate the accuracy of the semi-supervised approach proposed. We will analyze the overlapping areas found by our approach, and the new organization proposed by ILM. The model settings are described in Table B.1.

Parameter	**Description**	**Value**
k	number of nearest neighbors	k= 3
mc	number of micro-clusters built for each class	$mc = 20$
α	maximum size of the batch	$\alpha = 100$
MA	it holds the name of the majority classes, the user can set this number	$\|MA\| = 3$

Table B.1: Setting defined to test the Incremental learning system

B.2 ILM testbed

In order to reproduce a dynamic learning or an evolutionary process, we manually remove samples of one or more applications specially from the minority class. For instance, from the Streaming class, we can consider the application YouTube or NetFlix as new incoming behavior or class distribution change. This new dataset will be called online validation set. The setting established for each of the case studies is shown as follows,

- PAM
  - Streaming: YouTube and NetFlix applications
  - VoIP: Skype application
  - Browsing : Wikipedia application
  - P2P : eDonkey application
- VPN-nonVPN

- Streaming: YouTube and Netflix applications
- Browsing : Yahoo, Amazon and others
- VPN-Streaming: YouTube application
- VPN-VoIP : Hangouts application

It is worth mentioning that we selected these applications because they are the most used and popular on Internet. In addition, we consider the key classes for improving the QoS such as Streaming, VoIP and P2P classes.

Appendix C

Ensemble and ILM analysis

The Internet traffic classification counts with some constraints in terms of implementation. One of the most important ones is the time of response, which has to be very rapid. However, reducing the response time in ML models can lead to decreasing the classification performance. In this topic, the work in [64] present an efficient implementation of RF models for Internet classification, where the prediction time is reduced to 2μs per flow. Due to we are dynamically adding classifiers to our ensemble, the RF might or might not be the best solution. In this sense, we first find an equivalent low cost model, that can offer the same performance. We aim at building a model that can be easily implemented along with nDPI, because this tool is used to process the Internet packets, to build the flows and to compute the statistical features.

In order to achieve this, we propose to evaluate our proposal with a greedy and more simple approach using RF as base classifiers. Therefore, we will try to find an approximate model that can keep the performance obtained with the previous models. For this particular objective, we will use DTs as base classifiers, which in turn represent a simple classification model that can be denoted as lighter and, in some cases, weaker than RF.

This section is organized as follow. Section C.1 will present the performance of a greedy and simple models using RF as base classifiers. In section C.2, we envisage to find a configuration that approximate the results given in the previous section; in this sense, we can offer a lightweight model that can be implemented in the Internet traffic classification application by using DTs as base classifiers.

C.1 Random forests as base classifiers

We propose two configurations that provide the best average performance. CS_w counts with three RF base classifier, where the number of estimators by trees is 10, and the maximum depth of the trees is 20. On the other hand, CS_s have also 3 base classifiers but with 100 estimators for the RFs, and not limit for the depth of the trees. These trees use the entropy metric as the split criterion. In Table C.1, we show the accuracy (acc), the Area Under the Receiver Operating Characteristic Curve (ROC-AUC) score, the f-scores of the classes of interest in this investigation. From the table, we can notice that is not necessary to create a greedy model CS_s

to obtain good average performance, the model CS_w offers the same classification performance.

		PAM		VPN-nonVPN	
		CS_w	CS_s	CS_w	CS_s
Acc		0.9765	0.9773	0.9685	0.9704
ROC-AUC		0.9928	0.9937	0.9927	0.9954
f-score	Browsing	0.9840	0.9845	0.9824	0.9795
	P2P	0.9882	0.9887	0.9303	0.9371
	Streaming	0.8073	0.8140	0.8354	0.8476
	VoIP	0.9650	0.9681	0.8467	0.8688

Table C.1

The performance given by the model CS_s is taken as reference value for finding an approximation with DTs that can guarantee a low cost implementation. It is important to mention that the motivation of this approach is also supported by the dynamic construction base classifiers. In this sense, we have to assure that our approach is sustainable.

C.2 Decision trees as base classifiers

In order to find an approximate model, that disposes a lightweight configuration maintaining the performance given by the reference values, we propose to evaluate different configurations until find the closest to the expected values. This new configuration can be potentially implemented in a real word application for traffic classification.

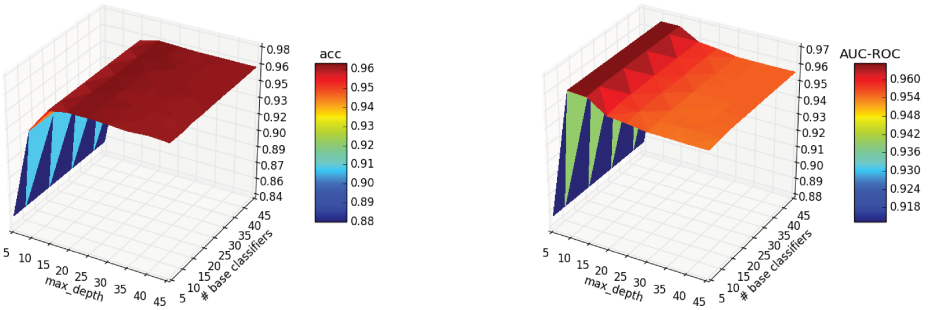
We variate several parameters of the dynamic ensemble using DTs as base classifiers. Therefore, we build a grid graphic that will serve as the validation tool of the new configuration. All the results here presented are obtained using the PAM dataset, and the VPN-nonVPN dataset.

For the PAM case, we summarize and analyze the results as follows:

- Figure C.1 shows the acc and ROC-AUC values after varying the number of base classifiers and the maximum depth of the trees. We can notice that with a maximum depth from 10 to 20 the results are good and comparable with the reference values. The ROC-AUC scores decreased however the accuracy is almost the same as the reference values.

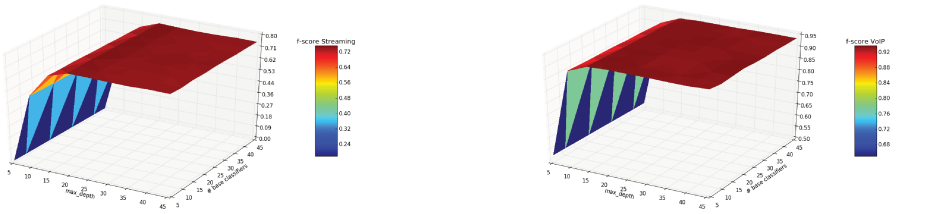
- The f-score of the majority classes in Figure C.2 is high and close to the reference values. However, we notice that for the minority class in Figure C.3, the f-scores decreased with a relative error to the reference value equal to 8% and 4% for the Streaming and VoIP classes. Although, this deference is significant, it is expected due to the imbalance property of the application. We do not consider it negligible, but acceptable for the objectives of this proposal.

To conclude this section, the model that fits all the requirements is given by an ensemble with 5 DTs as base classifiers and maximum depth equal to 15.



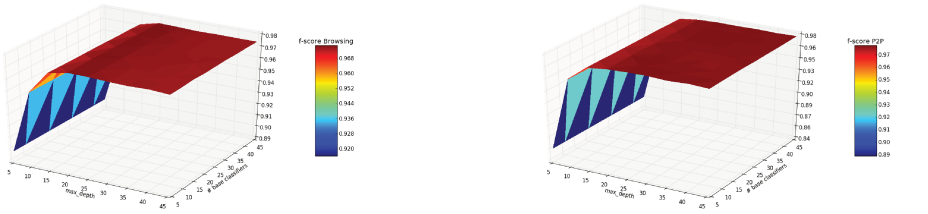
(a) Acc (b) ROC-AUC

Figure C.1: Acc and ROC-AUC of the dynamic ensemble varying the number of base classifiers and the maximum depth of the DTs



(a) Streaming (b) VoIP

Figure C.2: F-score of the dynamic ensemble varying the number of base classifiers and the maximum depth of the DTs



(a) Browsing (b) P2P

Figure C.3: F-score of the dynamic ensemble varying the number of base classifiers and the maximum depth of the DTs

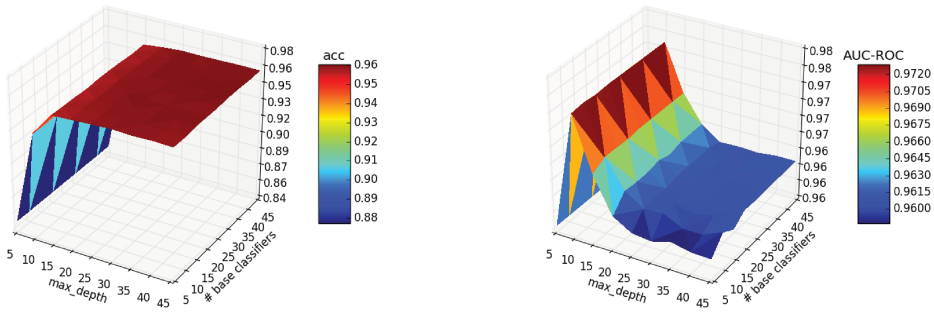
For the VPN-nonVPN dataset, we varied several parameters of the dynamic ensemble using DTs as base classifiers. The new lightweight configuration is based on the following results:

- Figure C.4 shows the acc and ROC-AUC values after varying the number of base classifiers and the maximum depth of the threes. We can notice that with a maximum depth upper than 15 the results are good and comparable with the

reference values. However, the best ROC-AUC scores are obtained between 10 and 15 as a maximum depth.

- The f-score values of the majority classes Browsing and VPN-VoIP in Figure C.5 are high and close to the reference values. However, we notice that for the minority classes Figure C.6 and Figure **??** the f-scores decreased with a relative error to the reference value lower or equal than 6% the Streaming, VoIP and P2P classes.
- Finally, the time of training and prediction are reduced as it was the case for the PAM dataset.

To conclude this section, the model that fits all the requirements is given by an ensemble with 5 DTs as base classifiers and maximum depth equal to 15.



(a) Acc (b) ROC-AUC

Figure C.4: Acc and ROC-AUC of the dynamic ensemble varying the number of base classifiers and the maximum depth of the DTs

(a) Browsing (b) VPN-VoIP

Figure C.5: F-score of the dynamic ensemble varying the number of base classifiers and the maximum depth of the DTs

Implementation results

Taking the best DT configuration, we repeated the tests in Section 7.4 to demonstrate that new knowledge is being discovered by this new ensemble. In Table C.2, the

(a) Streaming (b) VoIP

Figure C.6: F-score of the dynamic ensemble varying the number of base classifiers and the maximum depth of the DTs

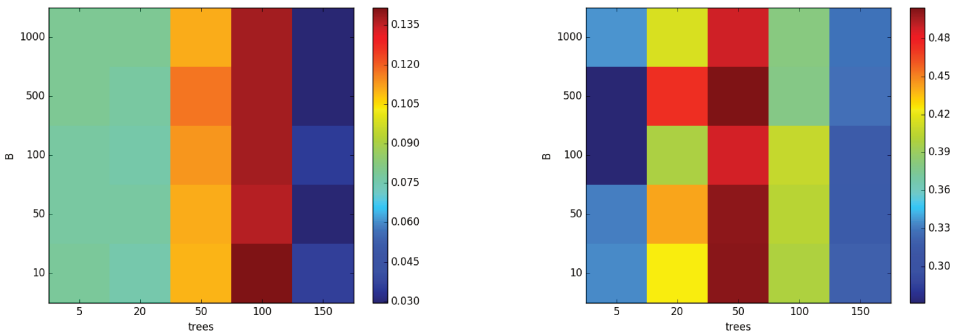
performance of both proposals is shown, where the DTs as base classifiers give even better results in some cases.

	PAM		VPN-NonVPN	
	DTs	RFs	DTs	RFs
Steaming	0.1313	0.1112	0.6127	4180
VoIP	0.5017	0.5413	0.3131	0.2543
Browsing	0.8980	0.9921	0.9412	0.97
P2P	0.3011	0.3550	-	-

Table C.2: Accuracy of the evolutionary test

C.3 Further analysis

As an additional test, we evaluate the effect of varying the size of the batch and the number of base classifiers. The batch's size will be dependent on the available input samples. For instance, for the PAM case study, the number of samples available was between 3000 and 3500 for the YouTube and Skype applications, respectively. However, from Figure C.7, we can notice that it is the amount of base classifiers which considerably affects the performance of the framework. For both applications, the number of base classifiers that give the best performance are between 50 and 100 DTs.



(a) YouTube (b) Skype

Figure C.7: Heat map varying the number of base classifiers and batch size

Appendix D

Tree based model implementation in C

This appendix shows how to parse a tree based model from python to C. In this sense, we show in Figure D.1 the scheme followed.

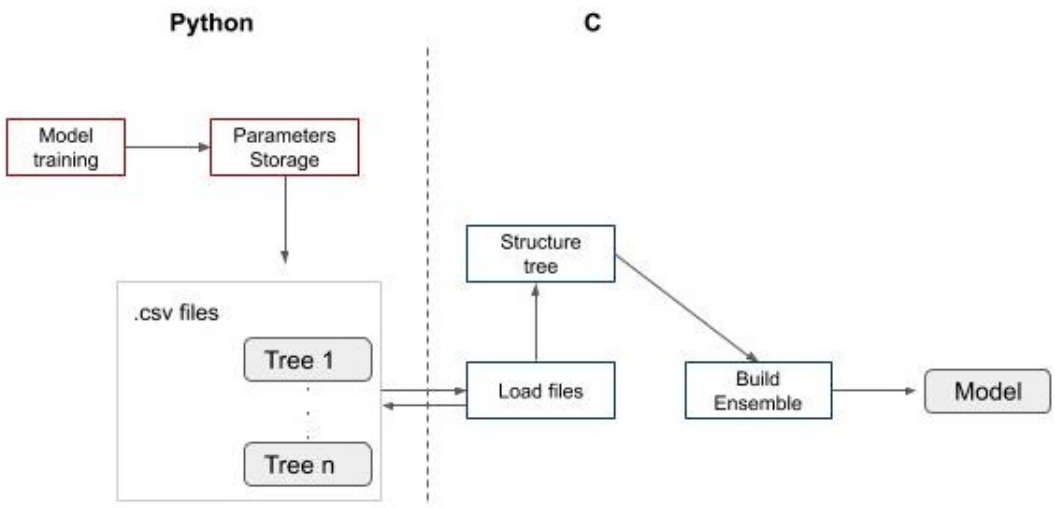


Figure D.1: Schema to parse a tree based model from python to C.

We describe in Table D.1 the steps performed in python. Each node in the three T has the following attributes: left child, right child, feature, threshold and Class. It is important to take into account:

- The nodes are saved in order, meaning that the first element in the .csv file is the root of the tree.
- The properties left child and right child denote where are the child of the current node with a number (the row position in the .csv file).
- Feature is the position i of the attribute to be evaluated in an input vector x. If $i < 0$, it is necessary to stop the inspection of tree's nodes.
- Threshold takes the value to be compared with $x[i]$.
- Class is vector with the class' membership probability of x.

Macro algorithm

Input data: $\{X, y\}$
Procedure:

1. Train the model with $\{X, y\}$
2. For each three T in the model,
   2.1. Get the nodes of the trees
   2.2. Save the nodes into a .csv format

Output: T in .csv format

Table D.1: Pseudo code for building the model in python

Following, in C, we load each tree of the ensemble and build the structure of the tree with the parameters previously defined. Once the trees are loaded, the prediction can be made by following the steps in Table D.2, using as combination method the classical voting process. However, the combination method adapted by scikit-learn is the mean terminal leaf probability across all trees. We also have the possibility of using this method by considering two new properties for each tree: samples which is the total amount of samples fell in the node, and value a vector with the class values of those samples.

Macro algorithm

Input data: x
Procedure:

1. For each tree $T[i]$ in the ensemble,
   1.1. pos = 0
   1.2. For True
      1.2.1. If $tree[pos].feature > 0$,
         - if $x[tree[pos].feature] <= tree[pos].threshold$, then $pos = tree[pos].left\_child$
         - else, $pos = tree[pos].right\_child$
      1.2.2. else, $pred[i] = tree[pos].Class$ and break the for
2. Apply a combination method over the vector $pred$ and get $final\_pred$

Output: $final\_pred$

Table D.2: Pseudo code for reconstructing the model in C

For a multiclass classification, the combination method is equivalent to the mayority vote by the trees weighted by their probability estimates. On the other hand, the multilabel classification result is the classes with an average probability estimates by the trees upper than a threshold values (in our case, fixed to 0.6).

References

[1] Giuseppe Aceto et al. "Multi-Classification Approaches for Classifying Mobile App Traffic". In: *Journal of Network and Computer Applications* (2017).

[2] Rahul Agrawal et al. "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. Rio de Janeiro, Brazil, 2013, pp. 13–24. ISBN: 978-1-4503-2035-1.

[3] Leman Akoglu, Hanghang Tong, and Danai Koutra. "Graph based anomaly detection and description: a survey". In: *Data Mining and Knowledge Discovery* 29.3 (2015), pp. 626–688.

[4] *Alexa*. https://www.alexa.com/topsites. Accessed: 2018-12-26.

[5] Mashael AlSabah, Kevin S. Bauer, and Ian Goldberg. "Enhancing Tor's performance using real-time traffic classification". In: *ACM Conference on Computer and Communications Security*. 2012.

[6] Riyad Alshammari and A. Nur Zincir-Heywood. "Can encrypted traffic be identified without port numbers, IP addresses and payload inspection?" In: *Computer Networks* 55.6 (2011), pp. 1326 –1350. ISSN: 1389-1286. DOI: https://doi.org/10.1016/j.comnet.2010.12.002. URL: http://www.sciencedirect.com/science/article/pii/S1389128610003695.

[7] Rafael Antonello et al. "Deep packet inspection tools and techniques in commodity platforms: Challenges and trends". In: *Journal of Network and Computer Applications* 35.6 (2012), pp. 1863 –1878.

[8] Rafael Antonello et al. "Deep packet inspection tools and techniques in commodity platforms: Challenges and trends". In: *Journal of Network and Computer Applications* 35.6 (2012), pp. 1863 –1878.

[9] Z. Aouini, A. Kortebi, and Y. Ghamri-Doudane. "Traffic monitoring in home networks: Enhancing diagnosis and performance tracking". In: *2015 International Wireless Communications and Mobile Computing Conference (IWCMC)*. 2015, pp. 545–550.

[10] D. J. Arndt and A. N. Zincir-Heywood. "A Comparison of three machine learning techniques for encrypted network traffic analysis". In: *2011 IEEE Symposium on Computational Intelligence for Security and Defense Applications (CISDA)*. 2011, pp. 107–114.

[11] Manuel Baena-Garćıa et al. "Early Drift Detection Method ?" In: 2006.

[12] Arian Baer et al. "DBStream: A holistic approach to large-scale network traffic monitoring and analysis". In: *Computer Networks* 107 (2016), pp. 5 –19.

[13] T. Bakhshi and B. Ghita. "On Internet Traffic Classification: A Two-Phased Machine Learning Approach". In: *Journal of Computer Networks and Communications* 2016.2048302 (2016). DOI: http://dx.doi.org/10.1155/2016/2048302.

[14] Roni Bar Yanai et al. "Realtime Classification for Encrypted Traffic". In: *Proceedings of the 9th International Conference on Experimental Algorithms*. SEA'10. Springer-Verlag, 2010, pp. 373–385. ISBN: 3-642-13192-1, 978-3-642-13192-9. DOI: 10.1007/978-3-642-13193-6_32.

[15] Gustavo E. A. P. A. Batista, Ronaldo C. Prati, and Maria Carolina Monard. "A Study of the Behavior of Several Methods for Balancing Machine Learning Training Data". In: *SIGKDD Explor. Newsl.* 6.1 (June 2004), pp. 20–29. ISSN: 1931-0145.

[16] L. Bertaux et al. "Software defined networking and virtualization for broadband satellite networks". In: *IEEE Communications Magazine* 53.3 (2015), pp. 54–60.

[17] A. Bittau et al. *Cryptographic protection of TCP Streams (tcpcrypt)*. https://datatracker.ietf.org/doc/rfc8548/. Internet Engineering Task Force (IETF).

[18] Enrico Bocchi et al. "MAGMA network behavior classifier for malware traffic". In: *Computer Networks* 109, Part 2 (2016), pp. 142 –156.

[19] Alessio Botta, Alberto Dainotti, and Antonio Pescapé. "A tool for the generation of realistic network workload for emerging networking scenarios". In: *Computer Networks* 56.15 (2012), pp. 3531 –3547.

[20] George Edward Pelham Box and Gwilym Jenkins. *Time Series Analysis, Forecasting and Control*. Holden-Day, Incorporated, 1990. ISBN: 0816211043.

[21] L. Breiman. "Random forests". In: *Machine Learning* 45.1 (2001), pp. 5–32.

[22] Tomasz Bujlow, Valentín Carela-Espanol, and Pere Barlet-Ros. "Independent comparison of popular DPI tools for traffic classification". In: *Computer Networks* 76 (2015), pp. 75 –89.

[23] Julian Andres Caicedo-Muñoz et al. "QoS-Classifier for VPN and Non-VPN traffic based on time-related features". In: *Computer Networks* 144 (2018), pp. 271 –279. ISSN: 1389-1286.

[24] CAIDA. *CAIDA data*. http://www.caida.org/data/. Accessed: 2017-09-27.

[25] Arthur Callado et al. "Better network traffic identification through the independent combination of techniques". In: *Journal of Network and Computer Applications* 33.4 (2010), pp. 433 –446.

[26] *Capella*. https://www.polarsys.org/capella/index.html. Accessed: 2018-12-17.

[27] Valentín Carela-Español et al. "Analysis of the Impact of Sampling on NetFlow Traffic Classification". In: *Comput. Netw.* 55.5 (2011), pp. 1083–1099.

[28] Valentín Carela-Español, Tomasz Bujlow, and Pere Barlet-Ros. "Is Our Ground-Truth for Traffic Classification Reliable?" In: *Passive and Active Measurement*. Ed. by Michalis Faloutsos and Aleksandar Kuzmanovic. Cham: Springer International Publishing, 2014, pp. 98–108.

[29] M. Cermák, T. Jirsík, and M. Laštovička. "Real-time analysis of NetFlow data for generating network traffic statistics using Apache Spark". In: *NOMS 2016 - 2016 IEEE/IFIP Network Operations and Management Symposium*. 2016, pp. 1019–1020.

[30] M. Cermák et al. "A performance benchmark for NetFlow data analysis on distributed stream processing systems". In: *NOMS 2016 - 2016 IEEE/IFIP Network Operations and Management Symposium*. 2016, pp. 919–924.

[31] Girish Chandrashekar and Ferat Sahin. "A survey on feature selection methods". In: *Computers & Electrical Engineering* 40.1 (2014), pp. 16–28.

[32] Kenjiro Cho, Koushirou Mitsuya, and Akira Kato. "Traffic Data Repository at the WIDE Project". In: *Proceedings of the Annual Conference on USENIX Annual Technical Conference*. ATEC '00. USENIX Association, 2000, pp. 51–51.

[33] *Cisco Visual Networking Index: Forecast and Trends, 2017–2022*. https://www.cisco.com/c/en/us/solutions/collateral/service-provider/visual-networking-index-vni/white-paper-c11-741490.html. Accessed: 2019-03-27.

[34] B. Claise, B. Trammell, and 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.

[35] Brian Corrie et al. "Towards quality of experience in advanced collaborative environments". In: *in Third Annual Workshop on Advanced Collaborative Environments*. 2003.

[36] A. Dainotti, A. Pescape, and K. C. Claffy. "Issues and future directions in traffic classification". In: *IEEE Network* 26.1 (2012), pp. 35–40.

[37] Alberto Dainotti et al. "Internet traffic modeling by means of Hidden Markov Models". In: *Computer Networks* 52.14 (2008), pp. 2645 –2662.

[38] Fannia Pacheco Daniel Sanchez and E. Exposito. "Emulating Application and User Behavior on a Cloud Platform for Later Traffic Analysis". Projet de recherche, 2eme ann'ee. Master Sciences et Technologies, Mention Informatique, Parcours Industry 4.0, UPPA, 2019.

[39] *DARPA Intrusion Detection Data Sets*. https://ll.mit.edu/ideval/data/. Accessed: 2017-09-27.

[40] *Data Plane Development Kit*. https://www.dpdk.org/. Accessed: 2019-07-26.

[41] Jonathan J. Davis and Andrew J. Clark. "Data preprocessing for anomaly based network intrusion detection: A review". In: *Computers & Security* 30.6 (2011), pp. 353 –375.

[42] R&D Département : Business Line Telecommunication. *Application du machine learning au Satcom*. Tech. rep. Thales Alenias Space, 2019.

[43] R&D Département : Business Line Telecommunication. *Application du machine learning au Satcom*. Tech. rep. Thales Alenias Space, 2019.

[44] R&D Département : Business Line Telecommunication. *Architecture de QoS basée sur les DPI*. Tech. rep. Thales Alenias Space, 2018.

[45] L. Deri et al. "nDPI: Open-source high-speed deep packet inspection". In: *2014 International Wireless Communications and Mobile Computing Conference (IWCMC)*. 2014, pp. 617–622.

[46] T. Dierks and E. Rescorla. *The Transport Layer Security (TLS) Protocol*. https://tools.ietf.org/html/rfc8446. Internet Engineering Task Force (IETF).

[47] Dinil Mon Divakaran et al. "SLIC: Self-Learning Intelligent Classifier for network traffic". In: *Computer Networks* 91 (2015), pp. 283 –297.

[48] L. Dixon, T. Ristenpart, and T. Shrimpton. "Network Traffic Obfuscation and Automated Internet Censorship". In: *IEEE Security Privacy* 14.6 (2016), pp. 43–53.

[49] Gerard Draper-Gil et al. "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*, INSTICC. SciTePress, 2016, pp. 407–414.

[50] Ran Dubin et al. "Real Time Video Quality Representation Classification of Encrypted HTTP Adaptive Video Streaming - the Case of Safari". In: *CoRR* abs/1602.00489 (2016). arXiv: 1602.00489. URL: http://arxiv.org/abs/1602.00489.

[51] M. Dunham. *Data Mining: Introductory and Advanced Topics*. Prentice Hall, 2003.

[52] Margaret H. Dunham. *Data Mining: Introductory and Advanced Topics*. Upper Saddle River, NJ, USA: Prentice Hall PTR, 2002.

[53] *Elasticsearch*. https://www.elastic.co/. Accessed: 2019-03-17.

[54] Fatih Ertam and Engin Avcı. "A new approach for internet traffic classification: GA-WK-ELM". In: *Measurement* 95 (2017), pp. 135 –142.

[55] Adil Fahad et al. "An optimal and stable feature selection approach for traffic classification based on multi-criterion fusion". In: *Future Generation Computer Systems* 36 (2014), pp. 156 –169.

[56] Adil Fahad et al. "Toward an efficient and scalable feature selection approach for internet traffic classification". In: *Computer Networks* 57.9 (2013), pp. 2040 –2057.

[57] Elaine R. Faria, João Gama, and André C. P. L. F. Carvalho. "Novelty Detection Algorithm for Data Streams Multi-class Problems". In: *Proceedings of the 28th Annual ACM Symposium on Applied Computing*. SAC '13. Coimbra, Portugal, 2013, pp. 795–800.

[58] Tom Fawcett. "An introduction to ROC analysis". In: *Pattern Recognition Letters* 27.8 (2006), pp. 861 –874.

[59] R. Ferrús et al. "SDN/NFV-enabled satellite communications networks: Opportunities, scenarios and challenges". In: *Physical Communication* 18 (2016). Special Issue on Radio Access Network Architectures and Resource Management for 5G, pp. 95 –112. ISSN: 1874-4907.

[60] Pawel Foremski. "On different ways to classify Internet traffic: a short review of selected publications". In: *Theoretical and Applied Informatics* 25.2 (2013).

[61] David Freedman. *Statistical Models : Theory and Practice*. Cambridge University Press, 2005. ISBN: 0521854830.

[62] A. Freier, P. Karlton, and P. Kocher. *The Secure Sockets Layer (SSL) Protocol Version 3.0*. https://tools.ietf.org/html/rfc6101. Internet Engineering Task Force (IETF).

[63] João Gama et al. "Learning with Drift Detection". In: *Advances in Artificial Intelligence – SBIA 2004*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2004, pp. 286–295.

[64] J. Garcia et al. "Towards Video Flow Classification at a Million Encrypted Flows Per Second". In: *2018 IEEE 32nd International Conference on Advanced Information Networking and Applications (AINA)*. 2018, pp. 358–365.

[65] I. Ghafir et al. "A Survey on Network Security Monitoring Systems". In: *2016 IEEE 4th International Conference on Future Internet of Things and Cloud Workshops (FiCloudW)*. 2016, pp. 77–82.

[66] Heitor M. Gomes et al. "Adaptive random forests for evolving data stream classification". In: *Machine Learning* 106.9 (2017), pp. 1469–1495.

[67] Paulo M. Gonçalves et al. "A comparative study on concept drift detectors". In: *Expert Systems with Applications* 41.18 (2014), pp. 8144 –8156.

[68] L. Grimaudo et al. "SeLeCT: Self-Learning Classifier for Internet Traffic". In: *IEEE Transactions on Network and Service Management* 11.2 (2014), pp. 144–157.

[69] F. Gringoli et al. "GT: Picking Up the Truth from the Ground for Internet Traffic". In: *SIGCOMM Comput. Commun. Rev.* 39.5 (Oct. 2009), pp. 12–18.

[70] Santiago Egea Gómez et al. "Ensemble network traffic classification: Algorithm comparison and novel ensemble scheme proposal". In: *Computer Networks* 127 (2017), pp. 68 –80. ISSN: 1389-1286.

[71] Katie Hafner and Matthew Lyon. *Where Wizards Stay Up Late: The Origins of the Internet*. 1st. New York, NY, USA: Simon & Schuster, Inc., 1996. ISBN: 0684812010.

[72] Guo Haixiang et al. "Learning from class-imbalanced data: Review of methods and applications". In: *Expert Systems with Applications* 73 (2017), pp. 220 – 239.

[73] Trevor Hastie, Robert Tibshirani, and Jerome H. Friedman. *The elements of statistical learning: data mining, inference, and prediction, 2nd Edition*. Springer series in statistics. Springer, 2009. ISBN: 9780387848570.

[74] M. Z. Hayat and M. R. Hashemi. "A DCT based approach for detecting novelty and concept drift in data streams". In: *2010 International Conference of Soft Computing and Pattern Recognition*. 2010, pp. 373–378.

[75] H. He and E. A. Garcia. "Learning from Imbalanced Data". In: *IEEE Transactions on Knowledge and Data Engineering* 21.9 (2009), pp. 1263–1284.

[76] B. Hestnes et al. "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*. 2003, pp. 3–10.

[77] M. Hirvonen and M. Sailio. "Two-phased method for identifying SSH encrypted application flows". In: *2011 7th International Wireless Communications and Mobile Computing Conference*. 2011, pp. 1033–1038.

[78] R. Hofstede et al. "Flow Monitoring Explained: From Packet Capture to Data Analysis With NetFlow and IPFIX". In: *IEEE Communications Surveys Tutorials* 16.4 (2014), pp. 2037–2064.

[79] Jae-Hyun Ham Hyun-Min An Su-Kang Lee and Myung-Sup Kim. "Traffic Identification Based on Applications using Statistical Signature Free from Abnormal TCP Behavior". In: *Journal of Information Science and Engineering* 31 (2015), pp. 1669–1692.

[80] Marios Iliofotou et al. "Graption: A graph-based P2P traffic classification framework for the internet backbone". In: *Computer Networks* 55.8 (2011), pp. 1909 –1920.

[81] *Internet Assigned Numbers Authority (IANA)*. https://www.iana.org/. Accessed: 2019-09-27.

[82] *Internet Engineering Task Force (IETF)*. https://www.ietf.org/. Accessed: 2018-12-26.

[83] *iPerf - The ultimate speed test tool for TCP, UDP and SCTP*. https://iperf.fr/. Accessed: 2017-07-18.

[84] ITU-T. *End-user multimedia QoS categories*. Tech. rep. TELECOMMUNICATION STANDARDIZATION SECTOR OF ITU, 2001.

[85] M. Arumaithurai J. Korhonen H. Tschofenig and A. Lior. *Traffic Classification and Quality of Service (QoS): Attributes for Diameter*. https://tools.ietf.org/html/rfc5777. Internet Engineering Task Force (IETF).

[86] Silva J. M. C., P. Carvalho, and Lima Solange Rito. "Inside packet sampling techniques: exploring modularity to enhance network measurements". In: *International Journal of Communication Systems* 30.6 (2017), e3135–n/a. ISSN: 1099-1131.

[87] Yu Jin, Esam Sharafuddin, and Zhi-Li Zhang. "Unveiling Core Network-wide Communication Patterns Through Application Traffic Activity Graph Decomposition". In: *SIGMETRICS Perform. Eval. Rev.* 37.1 (June 2009), pp. 49–60. ISSN: 0163-5999.

[88] Yu Jin et al. "A Modular Machine Learning System for Flow-Level Traffic Classification in Large Networks". In: *ACM Trans. Knowl. Discov. Data* 6.1 (Mar. 2012), 4:1–4:34. ISSN: 1556-4681.

[89] T. Jirsik et al. "Toward Stream-Based IP Flow Analysis". In: *IEEE Communications Magazine* 55.7 (2017), pp. 70–76.

[90] Jan Jusko and Martin Rehak. "Identifying peer-to-peer communities in the network by connection graph analysis". In: *International Journal of Network Management* 24.4 (2014), pp. 235–252.

[91] Ram Keralapura, Antonio Nucci, and Chen-Nee Chuah. "A novel self-learning architecture for P2P traffic classification in high speed networks". In: *Computer Networks* 54.7 (2010), pp. 1055 –1068.

[92] J. Khalife, A. Hajjar, and J. Diaz-Verdejo. "A multilevel taxonomy and requirements for an optimal traffic classification model". In: *International Journal of Network Management* 24 (2014), pp. 101 –120.

[93] N. Al Khater and R. E. Overill. "Network traffic classification techniques and challenges". In: *2015 Tenth International Conference on Digital Information Management (ICDIM)*. 2015, pp. 43–48.

[94] Donald E. Knuth. *Seminumerical Algorithms*. Third. Vol. 2. The Art of Computer Programming. Reading, Massachusetts: Addison-Wesley, 1998.

[95] Dragi Kocev et al. "Ensembles of Multi-Objective Decision Trees". In: *Machine Learning: ECML 2007*. Ed. by Joost N. Kok et al. Berlin, Heidelberg: Springer Berlin Heidelberg, 2007, pp. 624–631.

[96] A. Kortebi et al. "Home Networks Traffic Monitoring Case Study: Anomaly Detection". In: *2016 Global Information Infrastructure and Networking Symposium (GIIS)*. 2016, pp. 1–6.

[97] Bartosz Krawczyk, Michal Wozniak, and Francisco Herrera. "On the usefulness of one-class classifier ensembles for decomposition of multi-class problems". In: *Pattern Recognition* 48.12 (2015), pp. 3969 –3982. ISSN: 0031-3203. DOI: https://doi.org/10.1016/j.patcog.2015.06.001. URL: http://www.sciencedirect.com/science/article/pii/S0031320315002216.

[98] Bartosz Krawczyk et al. "Dynamic ensemble selection for multi-class classification with one-class classifiers". In: *Pattern Recognition* 83 (2018), pp. 34 –51. ISSN: 0031-3203.

[99] Bartosz Krawczyk et al. "Ensemble learning for data stream analysis: A survey". In: *Information Fusion* 37 (2017), pp. 132 –156.

[100] Riad Sadok Kristell Aguilar and Fannia Pacheco. "Emulating Application and User Behavior on a Cloud Platform for Later Traffic Analysis". Projet initiation à la recherche, 2eme ann'ee. Master Sciences et Technologies, Mention Informatique, Parcours SIGLIS, UPPA, 2018.

[101] James F. Kurose and Keith W. Ross. *Computer Networking: A Top-Down Approach (6th Edition)*. 6th. Pearson, 2012. ISBN: 0132856204, 9780132856201.

[102] Yingxu Lai et al. "On monitoring and predicting mobile network traffic abnormality". In: *Simulation Modelling Practice and Theory* 50 (2015), pp. 176 –188.

[103] K.V. Lalitha and V.R. Josna. "Traffic Verification for Network Anomaly Detection in Sensor Networks". In: *Procedia Technology* 24 (2016). International Conference on Emerging Trends in Engineering, Science and Technology (ICETEST - 2015), pp. 1400 –1405.

[104] D Larose. *k-Nearest Neighbor Algorithm, in Discovering Knowledge in Data: An Introduction to Data Mining*. Hoboken NJ USA: John, Wiley and Sons ,Inc, 2004.

[105] Vincent Lemaire, Christophe Salperwyck, and Alexis Bondu. "A Survey on Supervised Classification on Data Streams". In: *Business Intelligence: 4th European Summer School, eBISS 2014, Berlin, Germany, July 6-11, 2014, Tutorial Lectures*. Ed. by Esteban Zimányi and Ralf-Detlef Kutsche. Cham: Springer International Publishing, 2015, pp. 88–125.

[106] S. Leroux et al. "Fingerprinting encrypted network traffic types using machine learning". In: *NOMS 2018 - 2018 IEEE/IFIP Network Operations and Management Symposium*. 2018, pp. 1–5.

[107] Georgiy Levchuk. "Function and activity classification in network traffic data: existing methods, their weaknesses, and a path forward". In: vol. 9850. 2016, pp. 9850 –9850 –13. DOI: 10.1117/12.2225949.

[108] *Libpcap*. https://www.tcpdump.org/. Accessed: 2019-03-17.

[109] Guan-Zhou Lin et al. "Network traffic classification based on semi-supervised clustering". In: *The Journal of China Universities of Posts and Telecommunications* 17 (2010), pp. 84 –88.

[110] Huan Liu and Hiroshi Motoda. *Computational Methods of Feature Selection*. Boca Raton, Florida: Chapman, Hall, Taylor, and Francis Group, 2008.

[111] Zhen Liu, Ruoyu Wang, and Deyu Tang. "Extending labeled mobile network traffic data by three levels traffic identification fusion". In: *Future Generation Computer Systems* 88 (2018), pp. 453 –466. ISSN: 0167-739X.

[112] Zhen Liu et al. "A class-oriented feature selection approach for multi-class imbalanced network traffic datasets based on local and global metrics fusion". In: *Neurocomputing* 168.Supplement C (2015), pp. 365 –381.

[113] Mohammad Lotfollahi et al. "Deep Packet: A Novel Approach For Encrypted Traffic Classification Using Deep Learning". In: *CoRR* abs/1709.02656 (2017). arXiv: 1709.02656. URL: http://arxiv.org/abs/1709.02656.

[114] James M. Lucas et al. "Exponentially Weighted Moving Average Control Schemes: Properties and Enhancements". In: *Technometrics* 32.1 (Jan. 1990), pp. 1–29.

[115] Gjorgji Madjarov et al. "An extensive experimental comparison of methods for multi-label learning". In: *Pattern Recognition* 45.9 (2012). Best Papers of Iberian Conference on Pattern Recognition and Image Analysis (IbPRIA'2011), pp. 3084 –3104.

[116] A.K. Marnerides, A. Schaeffer-Filho, and A. Mauthe. "Traffic anomaly diagnosis in Internet backbone networks: A survey". In: *Computer Networks* 73 (2014), pp. 224 –243.

[117] Peter Megyesi, Geza Szabó, and Sandor Molnár. "User behavior based traffic emulator: A framework for generating test data for DPI tools". In: *Computer Networks* 92 (2015), pp. 41 –54.

[118] Stuart E. Middleton and Stefano Modafferi. "Scalable classification of QoS for real-time interactive applications from IP traffic measurements". In: *Computer Networks* 107 (2016), pp. 121 –132.

[119] S. Miller, K. Curran, and T. Lunney. "Multilayer Perceptron Neural Network for Detection of Encrypted VPN Network Traffic". In: *CiberSA*. 2018.

[120] S. Molnar, P. Megyesi, and G. Szabo. "Multi-functional traffic generation framework based on accurate user behavior emulation". In: *2013 IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*. 2013, pp. 13–14.

[121] Andrew Moore et al. *Discriminators for use in flow-based classification*. Tech. rep. University of London, 2005.

[122] B. Moore. *Policy Core Information Model (PCIM) Extensions*. https://tools.ietf.org/html/rfc3460. Internet Engineering Task Force (IETF).

[123] B. Moore et al. *Policy Core Information Model – Version 1 Specification*. https://tools.ietf.org/html/rfc3060. Internet Engineering Task Force (IETF).

[124] V. Moreno et al. "Commodity Packet Capture Engines: Tutorial, Cookbook and Applicability". In: *IEEE Communications Surveys Tutorials* 17.3 (2015), pp. 1364–1390.

[125] D. Muelas et al. "Dictyogram: A statistical approach for the definition and visualization of network flow categories". In: *2015 11th International Conference on Network and Service Management (CNSM)*. 2015, pp. 219–227.

[126] D. Naboulsi et al. "Large-Scale Mobile Traffic Analysis: A Survey". In: *IEEE Communications Surveys Tutorials* 18.1 (2016), pp. 124–161.

[127] Taoufik En Najjary and Guillaume 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, June 2010.

[128] A. Neumann et al. "Towards monitoring of hybrid industrial networks". In: *2017 IEEE 13th International Workshop on Factory Communication Systems (WFCS)*. 2017, pp. 1–4.

[129] B. Ng, M. Hayes, and 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)*. 2015, pp. 1–9.

[130] C. Niephaus, M. Kretschmer, and G. Ghinea. *QoS Provisioning in Converged Satellite and Terrestrial Networks: A Survey of the State-of-the-Art*. 2016.

[131] *OpenBACH*. https://www.openbach.org/. Accessed: 2017-12-18.

[132] *OpenSAND*. http://opensand.org/. Accessed: 2019-03-17.

[133] *OpenVPN*. https://openvpn.net/. Accessed: 2019-03-17.

[134] F. Pacheco et al. "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)*. 2018, pp. 1–8.

[135] F. Pacheco et al. "Towards the deployment of Machine Learning solutions in network traffic classification: A systematic survey". In: *IEEE Communications Surveys Tutorials* (2018), pp. 1–1. DOI: 10.1109/COMST.2018.2883147.

[136] Fannia Pacheco et al. "An Autonomic Traffic Analysis Proposal Using Machine Learning Techniques". In: *Proceedings of the 9th International Conference on Management of Digital EcoSystems*. MEDES '17. Bangkok, Thailand, 2017, pp. 273–280. ISBN: 978-1-4503-4895-9.

[137] W. Pak and Y. Choi. "High Performance and High Scalable Packet Classification Algorithm for Network Security Systems". In: *IEEE Transactions on Dependable and Secure Computing* 14.1 (2017), pp. 37–49.

[138] F. Pedregosa et al. "Scikit-learn: Machine Learning in Python". In: *Journal of Machine Learning Research* 12 (2011), pp. 2825–2830.

[139] Lizhi Peng et al. "Effectiveness of Statistical Features for Early Stage Internet Traffic Identification". In: *International Journal of Parallel Programming* 44.1 (2016), pp. 181–197.

[140] Marcin Pietrzyk et al. "Challenging Statistical Classification for Operational Usage: The ADSL Case". In: *Proceedings of the 9th ACM SIGCOMM Conference on Internet Measurement*. IMC '09. Chicago, Illinois, USA, 2009, pp. 122–135. ISBN: 978-1-60558-771-4.

[141] Marco Pimentel et al. "A review of novelty detection". In: *Signal Processing* 99 (2014), pp. 215 –249. ISSN: 0165-1684.

[142] R. Polikar. "Ensemble based systems in decision making". In: *IEEE Circuits and Systems Magazine* 6.3 (2006), pp. 21–45.

[143] David M. W. Powers. "Evaluation : from Precision , Recall and F-measure to Roc". In: *Journal of Machine Learning Technologies* 2.1 (2006), pp. 37 –63.

[144] *Proxmox*. https://www.proxmox.com/en/. Accessed: 2019-01-27.

[145] *Qosmos*. https://www.qosmos.com/. Accessed: 2019-03-17.

[146] Arun Raghuramu et al. "Uncovering the footprints of malicious traffic in wireless mobile networks". In: *Computer Communications* 95 (2016), pp. 95 –107.

[147] Jesse Read et al. "Classifier chains for multi-label classification". In: *Machine Learning* 85.3 (2011), p. 333.

[148] J. Riihijarvi and P. Mahonen. "Machine Learning for Performance Prediction in Mobile Cellular Networks". In: *IEEE Computational Intelligence Magazine* 13.1 (2018), pp. 51–60.

[149] Pedro M. Santiago del Rio et al. "Wire-speed Statistical Classification of Network Traffic on Commodity Hardware". In: *Proceedings of the 2012 Internet Measurement Conference*. IMC '12. Boston, Massachusetts, USA, 2012, pp. 65–72. ISBN: 978-1-4503-1705-4.

[150] Antonello Rizzi et al. "A low complexity real-time Internet traffic flows neuro-fuzzy classifier". In: *Computer Networks* 91 (2015), pp. 752 –771.

[151] Werner Robitza et al. "Challenges of future multimedia QoE monitoring for internet service providers". In: *Multimedia Tools and Applications* (2017).

[152] Joung Woo Ryu et al. "An Efficient Method of Building an Ensemble of Classifiers in Streaming Data". In: *Big Data Analytics*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2012, pp. 122–133.

[153] A. Saffari et al. "On-line Random Forests". In: *2009 IEEE 12th International Conference on Computer Vision Workshops, ICCV Workshops*. 2009, pp. 1393–1400.

[154] *Selenium*. https://www.seleniumhq.org/. Accessed: 2019-03-17.

[155] Yan Shi, Dezhi Feng, and Subir Biswas. "A Natural Language-Inspired Multi-label Video Streaming Traffic Classification Method Based on Deep Neural Networks". In: abs/1906.02679 (2019). URL: https://arxiv.org/abs/1906.02679.

[156] M. Siller and J. C. Woods. "QoS arbitration for improving the QoE in multimedia transmission". In: *2003 International Conference on Visual Information Engineering VIE 2003*. 2003.

[157] H. Singh. "Performance Analysis of Unsupervised Machine Learning Techniques for Network Traffic Classification". In: *2015 Fifth International Conference on Advanced Computing Communication Technologies*. 2015, pp. 401–404.

[158] Raman Singh, Harish Kumar, and R.K. Singla. "An intrusion detection system using network traffic profiling and online sequential extreme learning machine". In: *Expert Systems with Applications* 42.22 (2015), pp. 8609 –8624.

[159] Marina Sokolova and Guy Lapalme. "A systematic analysis of performance measures for classification tasks". In: *Information Processing & Management* 45.4 (2009), pp. 427 –437.

[160] "Source identification of encrypted video traffic in the presence of heterogeneous network traffic". In: *Computer Communications* 129 (2018), pp. 101 – 110.

[161] Murat Soysal and Ece Guran Schmidt. "Machine learning algorithms for accurate flow-based network traffic classification: Evaluation and comparison". In: *Performance Evaluation* 67.6 (2010), pp. 451 –467.

[162] R. Stankiewicz and A. Jajszczyk. "A survey of QoE assurance in converged networks". In: *Computer Networks* 55.7 (2011). Recent Advances in Network Convergence, pp. 1459 –1473.

[163] Ray Stanton. "Securing VPNs: comparing SSL and IPsec". In: *Computer Fraud & Security* 2005.9 (2005), pp. 17 –19.

[164] Alok Tongaonkar et al. "Towards self adaptive network traffic classification". In: *Computer Communications* 56 (2015), pp. 35 –46.

[165] *Traffic Classification Using nDPI over DPDK*. https://www.ntop.org/ndpi/traffic-classification-using-ndpi-over-dpdk/. Accessed: 2019-07-26.

[166] I. Trestian et al. "Googling the Internet: Profiling Internet Endpoints via the World Wide Web". In: *IEEE/ACM Transactions on Networking* 18.2 (2010), pp. 666–679.

[167] ETSI TS. *Digital Video Broadcasting (DVB); Second Generation DVB Interactive Satellite System (DVB-RCS2); Part 1: Overview and System Level specification*. Tech. rep. Digital Video Broadcasting, 2014.

[168] Grigorios Tsoumakas and Ioannis Katakis. "Multi-label classification: An overview". In: *Int J Data Warehousing and Mining* 2007 (2007), pp. 1–13.

[169] Sergey Tulyakov et al. "Review of Classifier Combination Methods". In: *Machine Learning in Document Analysis and Recognition*. Ed. by Simone Marinai and Hiromichi Fujisawa. Berlin, Heidelberg: Springer Berlin Heidelberg, 2008, pp. 361–386.

[170] *ULAKNET data*. http://ulakbim.tubitak.gov.tr/en/hizmetlerimiz/data-warehouse. Accessed: 2017-09-27.

[171] L. Vassio, I. Drago, and M. Mellia. "Detecting user actions from HTTP traces: Toward an automatic approach". In: *2016 International Wireless Communications and Mobile Computing Conference (IWCMC)*. 2016, pp. 50–55.

[172] P. Velan and V. Pus. "High-density network flow monitoring". In: *2015 IFIP/IEEE International Symposium on Integrated Network Management (IM)*. 2015, pp. 996–1001.

[173] Petr Velan et al. "A survey of methods for encrypted traffic classification and analysis". In: *International Journal of Network Management* 25.5 (2015), pp. 355–374.

[174] W. Wang et al. "End-to-end encrypted traffic classification with one-dimensional convolution neural networks". In: *2017 IEEE International Conference on Intelligence and Security Informatics (ISI)*. 2017, pp. 43–48.

[175] Y. Wang et al. "A novel semi-supervised approach for network traffic clustering". In: *2011 5th International Conference on Network and System Security*. 2011, pp. 169–175.

[176] Yu Wang et al. "Internet traffic clustering with side information". In: *Journal of Computer and System Sciences* 80.5 (2014), pp. 1021 –1036.

[177] T. Wellem et al. "A hardware-accelerated infrastructure for flexible sketch-based network traffic monitoring". In: *2016 IEEE 17th International Conference on High Performance Switching and Routing (HPSR)*. 2016, pp. 162–167.

[178] MAWI Working Group of the WIDE Project. *MAWI data*. http://mawi.wide.ad.jp/mawi/. Accessed: 2017-09-27.

[179] I. Witten and E. Frank. *Data mining: practical machine learning, tools and techniques*. Boston: Morgan Kaufman, 2005.

[180] Charles V. Wright et al. "Generating Client Workloads and High-Fidelity Network Traffic for Controllable, Repeatable Experiments in Computer Security". In: *Recent Advances in Intrusion Detection*. Ed. by Somesh Jha, Robin Sommer, and Christian Kreibich. Berlin, Heidelberg: Springer Berlin Heidelberg, 2010, pp. 218–237. ISBN: 978-3-642-15512-3.

[181] K. Xu, F. Wang, and L. Gu. "Behavior Analysis of Internet Traffic via Bipartite Graphs and One-Mode Projections". In: *IEEE/ACM Transactions on Networking* 22.3 (2014), pp. 931–942.

[182] Zhen Xu, Lin Ma, and Jun Sun. "Efficient tri-ary search tree based packet classification algorithm". In: *IET Conference Proceedings* (2007), 833–836(3).

[183] R. Yavatkar, D. Pendarakis, and R. Guerin. *A Framework for Policy-based Admission Control*. https://tools.ietf.org/html/rfc2753. Internet Engineering Task Force (IETF).

[184] T. Yildirim and P. Radcliffe. "VoIP traffic classification in IPSec tunnels". In: *2010 International Conference on Electronics and Information Engineering*. Vol. 1. 2010, pp. V1–151–V1–157.

[185] ChengGuo Yin, ShuangQing Li, and Qi Li. "Network traffic classification via HMM under the guidance of syntactic structure". In: *Computer Networks* 56.6 (2012), pp. 1814 –1825.

[186] Ruixi Yuan et al. "An SVM-based machine learning method for accurate internet traffic classification". In: *Information Systems Frontiers* 12.2 (2010), pp. 149–156.

[187] Chaoyun Zhang, Paul Patras, and Hamed Haddadi. "Deep Learning in Mobile and Wireless Networking: A Survey". In: *CoRR* abs/1803.04311 (2018). URL: http://arxiv.org/abs/1803.04311.

[188] Hongli Zhang et al. "Feature selection for optimizing traffic classification". In: *Computer Communications* 35.12 (2012), pp. 1457 –1471.

[189] J. Zhang et al. "Network Traffic Classification Using Correlation Information". In: *IEEE Transactions on Parallel and Distributed Systems* 24.1 (2013), pp. 104–117.

[190] Jun Zhang et al. "Unsupervised traffic classification using flow statistical properties and IP packet payload". In: *Journal of Computer and System Sciences* 79.5 (2013), pp. 573 –585.

[191] Min-Ling Zhang and Zhi-Hua Zhou. "ML-KNN: A lazy learning approach to multi-label learning". In: *Pattern Recognition* 40.7 (2007), pp. 2038 –2048.