Autonomic Classification of IP Traffic in an NFV-based Network

Trabajo de Grado

María Camila Martínez Ordoñez

Juliana Alejandra Vergara Reyes

Advisor: PhD. Oscar Mauricio Caicedo Rendón

Departamento de Telemática
Facultad de Ingeniería Electrónica y Telecomunicaciones
Universidad del Cauca
Popayán, Cauca, 2017

Autonomic Classification of IP Traffic in an NFV-Based Network

María Camila Martínez Ordoñez
Juliana Alejandra Vergara Reyes

Trabajo de grado presentado a la Facultad de Ingeniería
Electrónica y Telecomunicaciones de la
Universidad del Cauca para obtener el título de:
Ingeniero en Electrónica y Telecomunicaciones

Advisor: PhD. Oscar Mauricio Caicedo Rendón

Departamento de Telemática
Facultad de Ingeniería Electrónica y Telecomunicaciones
Universidad del Cauca
Popayán, Cauca, 2017

Acknowledgements

Firstly, we want to thank God for our lives and all the blessings we have received. Sincerely, we really appreciate all the effort our families have done to achieve our goals. Their love and support have been an important tool to reach our dreams. For that reason, we want to dedicate our work as gratefulness for everything they have done throughout our education. Furthermore, we want to thank our academic advisor Professor Oscar Mauricio Caicedo Rendon for guiding us and supporting us during the development of our undergraduate work. His support and innovate ideas helped us to solve different problems happened during this work.

Abstract

Network Function Virtualization (NFV) is an emerging solution that improves the flexibility, efficiency, and manageability of networks by leveraging virtualization and cloud computing technologies to run networked devices in software. The implementation of NFV presents issues such as the introduction of new software components, bottleneck performance and monitoring of hidden traffic. A considerable amount of NFV traffic is invisible using traditional monitoring strategies because it does not hit a physical link. The implementation of autonomous management and supervised algorithms of Machine Learning (ML) become a key strategy to manage this hidden traffic.

In this undergraduate work, we focus on analyzing NFV traffic features in two test environments with different components and traffic generation. We perform a benchmarking of the performance of supervised ML algorithms concerning its efficiency; considering that the efficiency of the algorithms depends on the trade-off between the time-response and the precision achieved in the classification. The results show that the NaiveBayes and C4.5 algorithms reach values greater than 90.68 % in a response time range between 0.37 sec and 3 sec.

Contents

List of Figures

List of Tables

List of Snippet

Chapter 1

Introduction

1.1 Problem Statement

The Network Functions Virtualization (NFV) describes and defines how network services are designed, constructed, and deployed using virtualized software components and how these are decoupled from the hardware upon which they execute [2]. Organizations need to implement NFV to address dynamic user requirements, growing workloads, and the complexity of agile development. However, there are a lot of issues to implement NFV such as the introduction of new software components (e.g., hypervisor and management/orchestration elements), the management of bottleneck, and monitoring the hidden traffic [3]. In NFV-based networks, a considerable amount of traffic communicates among virtual machines (VMs) running inside a physical host [4]. It changes how and where monitoring network virtual networks, in fact, the NFV hidden traffic is commonly referred to as the blind spots of east-west traffic as it never hits a physical interface [5]. It is to highlight that the hidden traffic could lead to difficulties in diagnosing network performance and detecting malicious agents within a virtualized data center [6].

The growth of networks, the hidden traffic in NFV, and the need for advanced monitoring architectures are challenges that human administrators face to carry out their daily management tasks [7]. The autonomic management assists to cope

with these challenges by automating and distributing decision-making processes. This type of the management also aids to improve security, prevention, detection, control and error handling, and allows tracking the evolution of network applications [8, 9, 10, 11, 12, 13]. Within the field of autonomic management, the Machine Learning (ML) is an important technique that has been successfully applied to deal with problems such as the accurate classification of traffic on traditional networks, detection of network anomalies [14], medical diagnostics [15], and data mining [16]. Indeed, traffic classification is a fundamental block needed to enable any traffic management operation, such as differentiating traffic pricing and treatment (*e.g.,* policing and shaping) and security (*e.g.,* firewall, filter and anomaly detection) [17].

Traffic classification has been extensively worked on traditional networks by using ML algorithms. The works [18, 19, 20, 21, 22, 23, 24] reveal that the supervised algorithms have an excellent behavior in traffic classification. However, none of such works have focused on classifying traffic in NFV. This classification differs from traditional networks because, in NFV-based networks, the traffic passes by physical links but sometimes the traffic only flows by virtual links. To the best our knowledge, in NFV-based networks, there are few works that perform traffic classification [25, 26, 27]. These works focus on classifying the traffic by using ML algorithms as virtual network functions, analyzing the traffic in the deployment of NFV in data center networks, and evaluating the traffic of NFV middleboxes. Unlike the above works, in this paper, we perform a benchmarking to analyze the behavior of several supervised ML algorithms in the IP traffic classification in NFV-based networks.

To sum up, ML algorithms have been widely used for classifying traffic in traditional networks. However, there is little research on their behavior in NFV-based networks. Consequently, we propose the following research question:

How do supervised algorithms behave in autonomic classification of IP traffic in NFV-based networks?

To answer this research question, we present the following objectives.

1.2 Objectives

1.2.1 General

- Performing an analysis of the behavior of supervised algorithms during the autonomic classification of IP traffic in an NFV-based network.

1.2.2 Specifics

- Adapting an IP traffic dataset, already used for studies in traditional networks, to an NFV-based network to verify the changes in their behavior in those networks.
- Selecting the most efficient decision tree and Bayesian algorithms for the classification of traffic in traditional networks.
- Evaluating qualitatively and quantitatively the efficiency of the algorithms selected for the autonomic classification of IP traffic in NFV.

1.3 Research Contributions

The main contributions provided in this work are mentioned below:

- A dataset of IP traffic adapted to the features of NFV-based networks. This dataset allows the analysis and observation of the hidden traffic that travels by the virtual components of the network.
- A benchmarking of the behavior of the most efficient algorithms in the autonomic classification of IP traffic in traditional and NFV-based networks. The efficiency of the algorithms is presented regarding the time-response and the precision of each algorithm.

- A quantitative and qualitative evaluation of the autonomic classification of IP hidden traffic in NFV-based networks.

1.4 Methodology and Activities

To organize the activities developed during our undergraduate work, we take as a reference the Documentary Research Model (MID) and Building Solutions Model (MCS), defined as follows:

- MID used for the construction of the state-of-the-art of the project that focuses on the autonomic traffic management of an NFV-based network with automatic learning techniques [28].
- MCS used as a methodological reference for the construction of solutions aimed at classifying of traffic in NFV- based networks using automatic learning classifiers [29].

Activities undertaken in our work are shown below.

Initial knowledge basis generation

- Review of related work. The different works carried out in the field of traffic classification, autonomic management, and NFV were collected and analyzed.
- Synthesis. The research problem is determined, consolidating and relating the different works studied.
- Generation of the theoretical basis. Knowledge was expanded in the area of autonomic traffic classification in virtualized networks.

Dataset for traffic classification

- Data and features recognition of a dataset in a traditional network. Some features of the traffic in traditional networks, used in the task of traffic classification, were taken as a basis to construct a new dataset.
- Create test NFV-based networks. NFV-based networks are proposed for the analysis and data collection.
- Feature detection in an NFV-based network. Features were determined that allowed to know the route and behavior of the hidden traffic in this type of networks.
- Generate and collect data. Traffic was generated and captured in the test network for further analysis.

Benchmarking

The benchmarking process was performed by using the methodology called Training, Validation and Testing [30].

- Data adaptation. The data collected for the training of supervised classification algorithms (*i.e.,* C4.5, NaiveBayes, and BayesNet) were prepared.
- Training, validating and testing. Datasets adapted to train and test supervised classification algorithms were used to determine their efficiency.

Publishing

- Papers writing. We carried out and submitted a journal paper named "A benchmarking of the efficiency of supervised ML algorithms in the NFV traffic classification", and a conference paper named "IP traffic classification in NFV: a benchmarking of supervised Machine Learning algorithms".
- Monograph writing. We wrote the monograph as a highly detailed study of the work done.

1.5 Publications

The work presented in this monograph was reported to the scientific community through paper submissions.

- **Juliana Alejandra Vergara Reyes, Maria Camila Martinez Ordoñez,** Armando Ordonez, Oscar Mauricio Caicedo Rendon. **IP traffic classification in NFV: a benchmarking of supervised Machine Learning algorithms** IEEE Colombian Conference on Communications and Computing (COLCOM 2017), August 16th - 18th, Cartagena, Colombia.
  - Status: Accepted
  - Classification: H1 (SCIMAGO)
- **Juliana Alejandra Vergara Reyes, Maria Camila Martinez Ordoñez,** Oscar Mauricio Caicedo Rendon. **A benchmarking of the efficiency of supervised ML algorithms in the NFV traffic classification.** INGENIERÍA E INVESTIGACIÓN *journal*.
  - Status: Submitted
  - Classification: A1 (COLCIENCIAS)

1.6 Document Structure

This document has been divided into chapters described below.

- Chapter 1 presents the **Introduction** that contains the Problem Statement, Objectives, Research Contributions, Methodology and Activities, Publications and the structure of this document .
- Chapter 2 presents the **Background** about the relevant topics concerning our research. These topics include Autonomic Management, Machine Learning, and Network Functions Virtualization.

- Chapter 3 presents the **Related Work** that describes the researches that are closer to our proposal.
- Chapter 4 presents the **Dataset construction and Benchmarking Process.** Initially, we present the process of constructing a dataset for training supervised ML algorithms. Subsequently, the process of generating and collecting data, the method of training the algorithms to determine its efficiency. Finally, the benchmarking process that we follow to determine the behavior of the algorithms according to their efficiency.
- Chapter 5 presents the **Benchmarking results** from two case studies, where we present the results of precision and time-response obtained in the training, validation, and testing of each algorithm. Finally, we present the analysis of such results to determine the efficiency of the algorithms evaluated.
- Chapter 6, we present **Conclusions** and **Future work**, where we provide the main conclusions of our work and important implications for future works.

Chapter 2

Background

In this chapter, we present the concepts necessary for developing our work. The initial description is about the autonomic management and its applications, after we introduce the concept of the machine learning for traffic classification and finally, we present NFV and its components.

2.1 Autonomic Management

The autonomic management is a systematic approach to handle computer-based systems without human intervention [31]. In networks, an autonomic environment needs characteristics such as self-configuring, self-healing, self-optimizing and self-protecting. Self-configuring allows a system to adapt to dynamic environments (*e.g.,* the deployment and removal of new network components) and refers to the dynamic configuration capability [32]. Self-healing is to discover, diagnose and react to disruptions becoming networks more resilient [33]. Self-optimizing allows automatically monitoring resources to improve their overall utilization and perform tasks promptly [34]. Self-protecting is to anticipate, detect, identify, and protect against attacks making networks less vulnerable [35].

Figure 2.1 presents the architecture of an autonomic manager and its loop divided into monitoring, analysis, planning and execution fuctions. These functions share

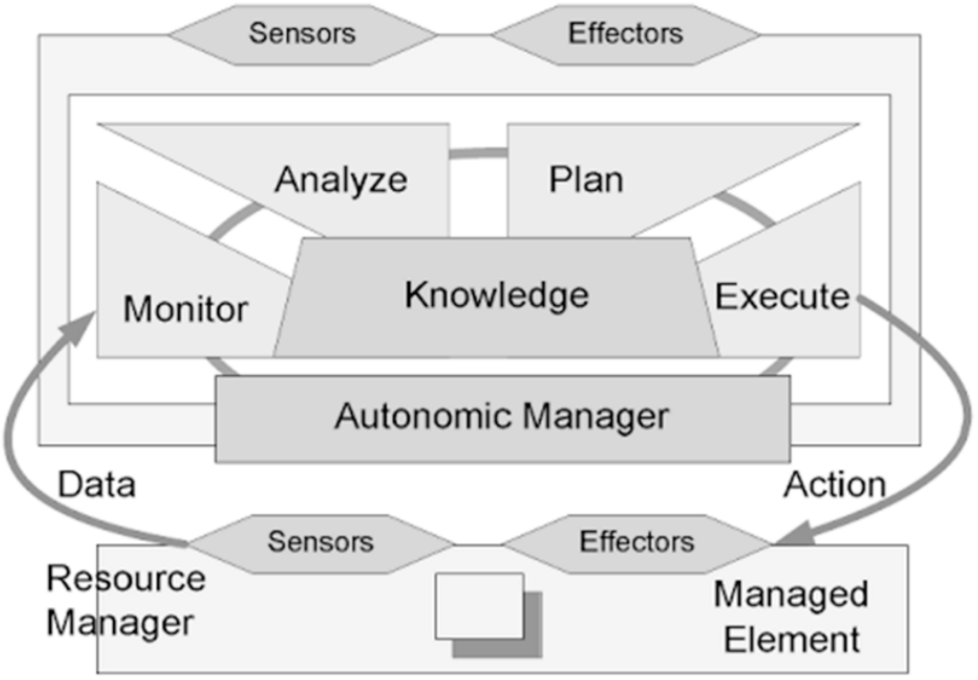

Figure 2.1: Architecture of Autonomic Manager
Source: [35]

knowledge and work together to provide the functionality of this loop [35]. An autonomic manager is an application that manages to automate some of the primary management functions according to the behavior defined by the management interfaces. The main fuctions of an autonomic manager are:

- Monitoring Function is to provide the collection and filtering mechanisms of the system and informs details collected from the monitored resource.
- Analysis Function is to allow the regional administrator to learn about the environment and help predict future situations.
- Planning Function is to provide the mechanisms necessary to achieve the goals and objectives of the system, makes use of the guidance information to perform this work.
- Execution Function is to provide the mechanisms that control the execution of a plan with considerations for dynamic updates.

Autonomic management functions (*e.g.,* monitoring and analysis) may be performed by using ML techniques as part of self-configuring and self-healing characteristics that allow systems, for instance, to identify network features, determine errors in run time, improve networks deployment, overcome issues, and discover knowledge regarding network traffic [8].

2.2 Machine Learning

ML is an evolving branch of computational algorithms that are designed to emulate human intelligence by learning from the surrounding environment [36]. In this sense, ML is an important technique that has been successfully applied to deal with problems such as the precise classification of traffic on traditional networks [24], detection of network anomalies [14], medical diagnostics [37], and data mining [16]. ML can learn from experience, analytical observation, and other means, resulting in a system that can be improved continuously and automatically, providing greater efficiency and effectiveness.

In this work, we argue that ML is fundamental to networks because it allows to use and mix clustering, regression, anomaly detection and artificial neural networks to perform complex tasks, such as reduce training time and computational complexity in some network processes [21]. We used ML algorithms because they provide methods useful to classify flows based on independent statistical features, such as packet length and arrival intervals [19].

ML algorithms are classified, in a general way, into Supervised and Unsupervised algorithms which has been also used in traffic classification [38]. Supervised learning (*e.g.,* a decision tree or Bayesnet) designs a model from a training dataset used to classify unseen data. Unsupervised learning creates groups with similar characteristics of data without prior knowledge, and the groups are transformed into a classification model [21].In this undergraduate work, we focus on analyzing the behavior of these algorithms in NFV-based networks [18] [21] [39] [40].

2.2.1 Supervised Algorithms

The supervised classification consists mainly of three stages [41] (see Figure 2.2). First, training stage. In this stage, the algorithm learns and trains with a set of data designed for that, which contains the initial reference data. Second, classification stage. At this stage the algorithm receives new samples, by way of a test, those samples may or may not hold the test data together. In this stage it is necessary to know if the data and the information are correct, for this reason, the third stage corresponds to the validation. In this stage, the results obtained in the classification stage are compared with the data of the initial set of reference used in the training stage. This final stage allows evaluating and identifying the performance of the algorithms in the data classification and estimating their behavior on a real operating network.

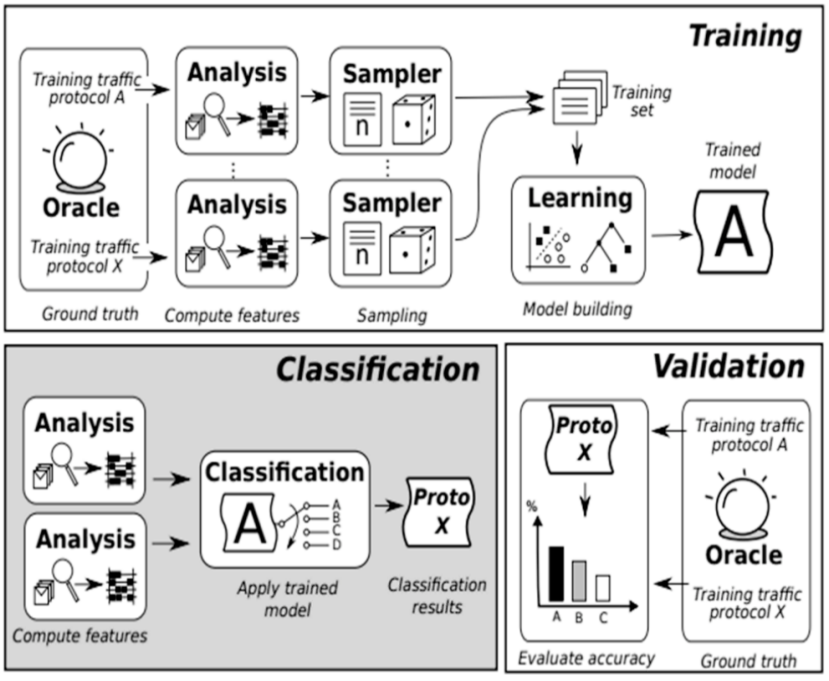

Figure 2.2: Workflow of Supervised Classification
Source: [41]

Supervised classification algorithms base their performance on the construction of pre-tagged models for self-training. These models allow them to learn and make

decisions, to ensure good behavior and precision when implementing them with other datasets. The following are different classes of supervised algorithms according to their mode of operation [42]:

The Bayesian algorithms. These algorithms are the focus on the problems of classification and regression by applying the Bayes Theorem. Within this group of algorithms exist two relevant algorithms that are using in the context of traditional networks, these algorithms are:

- Naïve Bayes. This algorithm is known as a probabilistic classifier that uses the Bayesian theorem and some hypothesis, with which it can determine an independence between the predictive variables of a set of data. This algorithm requires a small amount of data for its training and thus to be able to estimate certain parameters for the classification of another set of data.
- Bayesian Network. This algorithm model a phenomenon using a previous set of variables and their respective dependency ratios. With this model, we can estimate the probability of possible unknown variables, based on known variables.

The algorithm of Artificial Neural Network. This type of algorithm is inspired by how the neurons function in the human brain. These algorithms start from a significant set of data to teach the network the desired properties.

The Decision Tree Algorithms. These algorithms take a set of data and generate logical diagrams that serve to categorize a series of conditions and then make decisions. In this category one of the most used algorithms for the classification of traffic by its high values of efficiency reached is the algorithm C4.5.

- C4.5 Decision Tree. This algorithm is based on the structure of a tree, where the nodes represent the features and the branches represent the values of connection between such features. C4.5 constructs a tree with the criterion of dividing the gain ratio based on entropy.

2.3 Network Functions Virtualization

NFV is the virtualization of processes and internal functions of different components of the network, such as routers, switches, and storage equipment. NFVs primary function is to bring the network functions from hardware to software to reduce Operational Expenditure (OPEX) and Capital Expenditure (CAPEX), in addition to facilitating the proper deployment of new services more quickly and efficiently [43]. Virtualizing the network functions allows service providers to flexibility, among other things, traffic management by enabling them to host the network functions in virtual machines [44].

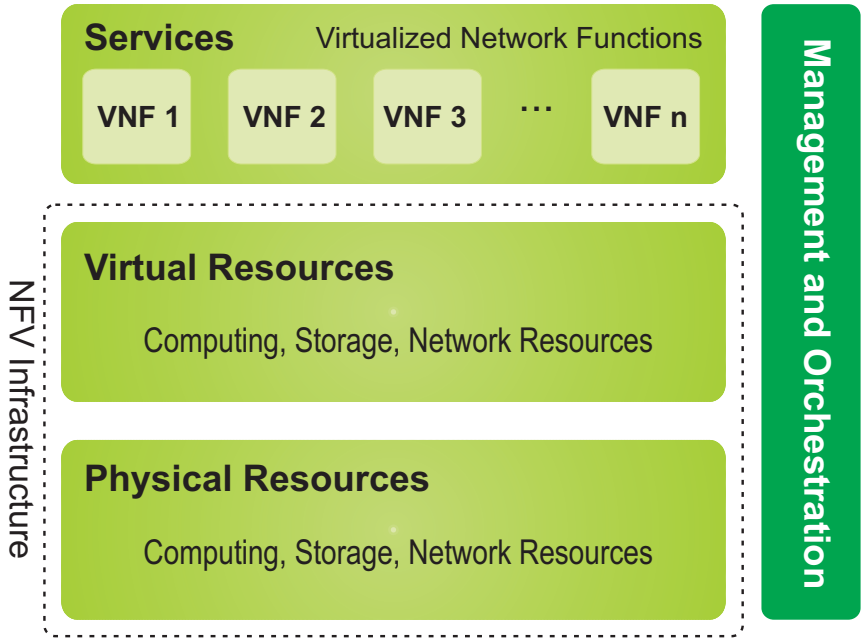

Figure 2.3: NFV Architecture
Source: [45]

According to ETSI, the NFV architecture (see Figure 2.3) has three key parts: Network Function Virtualization Infrastructure (NFVI), Virtual Network Functions (VNFs) and NFV Management and Orchestration (MANO) [45]. NFVI is the combination of physical and virtual resources (hardware and software) that form the environment in which VNFs are deployed. Physical resources include commercial resources outside the software platform, storage, and network that provide treat-

ment, storage, and connectivity to VNFs. In turn, virtual resources are abstractions of computing, storage, and network resources [46].

A Network Function (NF) is a functional block within a network infrastructure, which is accessible through external interfaces. NFs can be elements of a home network, such as DHCP servers, firewalls, and routers. A VNF is an implementation of an NF that uses virtual resources as a VM. A single VNF can be composed of multiple internal components and can be deployed in multiple VMs, where each VM hosts a single component of the VNF [47].

NFV MANO provides the necessary functionality for provisioning VNFs, and related operations such as configuring and deploying those functions and focuses on all management and virtualization tasks within the NFV framework. Also, MANO defines the interfaces for the communication between the different components of the architecture, as well as the coordination with traditional network management systems such as Operations Support System (OSS) and Business Support Systems (BSS) to allow the management of VNFs [43].

2.4 Traffic Classification on NFV

NFV is an excellent tool for deploying next generation networks and services, which brings benefits in scalability, high level of flexibility, efficient utilization of network resources, and reduction of costs and power consumption [48]. The NFV traffic flows by distinct ways in different scenarios. First, the traffic goes via a physical link in three different cases:

- VMs are connected to the same Virtual Switch (vSwitch), different port group, and distinct Virtual Local Area Network (VLAN)
- VMs are linked with separate vSwitch but same port group
- VMs run on different hosts and are connected to different vSwicthes and port groups

Second, the traffic never hits a physical link if two VMs are connected to the same vSwitch, port group, VLAN, and both are running on the same physical host. These VMs communicate within the vSwitch [49].

The most of the NFV traffic communicates from VM to VM and this traffic does not pass by a physical link becoming invisible to traditional monitoring and creates a big blind spot for network operations [4]. Since this traffic increases the network speed and reduces network latency, it is so important to classify it [6]. In fact, traffic classification allows automatically recognizing the application that has generated a stream or individual packets. This classification has become a fundamental issue for network management that is of extreme interest to carriers, Internet Service Providers and network administrators [17].

Chapter 3

Related Works

In this chapter, we introduce the closest works to our benchmarking in traffic classification. The first section describes the works focused on classifying traffic in traditional networks. Subsequently, we present the works that include a traffic classification in networks based on NFV. Finally, Table 3.1 is presented as a summary of the works mentioned.

3.1 Traffic Classification in Traditional Networks with ML Algorithms

The autonomic management of traffic in NFV-based networks has been little studied by the research published so far. However, different authors in their work have performed the classification of traffic in traditional networks by using ML algorithms. Some of the most relevant proposals are described below:

The work "Practical machine learning based multimedia traffic classification for distributed QoS management" presents the desing, implementation and performance evaluation of a distributed, ML-based traffic classification and control system for FreeBSD's IP Firewall (IPFW) [18]. On an Intel Core i7 2.8 GHz PC the system can classify up to 400.000 packets per second using only one core and their system scales

well to up to 100.000 simultaneous flows. Also, the corresponding implementation allows controlling subsequent traffic shaping or blocking at multiple (potentially lower performance) routers or gateways distributed around the network. It is to note that this work analyzed and compared the classification speeds of different ML algorithms (based on the Java implementations of WEKA). Measured speeds range from $27,000$ flows/sec to $60,000$ flows/sec, on PCs with different CPUs.

In the work "IP traffic classification based on machine learning", the authors use the ML-based classification method to identify the classes of the unknown flows using the payload-independent statistical features such as packet length and arrival-interval [19]. To improve the efficiency of the classification methods and refine the selected features the authors adopted the feature reduction techniques. They compare and evaluate the ML classification algorithms (*e.g.,* NaiveBayes, Decision Tree, Nearest Neighbor and Support Vector Machine) based on the BRASIL data source regarding overall precision, average precision, and average recall. The experiments performed by the authors reveals that the decision-tree algorithm is the best for IP traffic classification and is able to construct the real-time classification system.

The work "A method for classification of network traffic based on C5.0 Machine Learning Algorithm" presents the process of collecting data, the arguments used in the classification process, introduces the C5.0 classifier, and finally evaluates and compares the obtained results. The authors proposed the usage of C5.0 Machine Learning Algorithm (MLA) to overcome the drawbacks of existing methods for traffic classification [20]. The authors of this work, constructed a boosted classifier, which can distinguish between seven different applications (*i.e.,* Skype, FTP, torrent, web browser traffic, web radio, interactive gaming, and SSH) with an average precision ranking from 99.3% to 99.9%.

In the work "A Near Real-time IP Traffic Classification Using Machine Learning" a real-time internet traffic dataset was developed using packet capturing tool for 2-second packet capturing duration and other datasets have been developed by reducing a number of features of 2-second duration dataset using Correlation and Consistency based Feature Selection (FS) Algorithms [21]. The authors of this work, employed five ML algorithms MLP, RBF, C4.5, BayesNet and NaïveBayes

for IP traffic classification with the datasets above mentioned. The results reveals that BayesNet gives very high precision with smaller training time making it more suitable for near real-time IP traffic classification with a reduction in a number of features characterizing each internet application using correlation-based feature selection algorithm.

In the work "Comparative analysis of machine learning algorithms along with classifiers for network intrusion detection", several classification techniques and machine learning algorithms have been considered to categorize the network traffic [22]. The authors found nine classifiers like BayesNet, Logistic, IBK, C4.5, PART, JRip, Random Tree, Random Forest and REPTree. The comparison of these algorithms has been performed using the WEKA tool, the NSL-KDD based dataset, and a 10-fold cross validation. The results reveals that the Bayes-Net and Random Forest algorithms are the most suitable for intrusion detection and classification.

In the work "WeChat: Text and Picture Messages Service Flow Traffic Classification Using Machine Learning Technique", text and picture messages traffics are classified using two different types of datasets (*i.e.,* Harbin Institute of Technology (HIT) and Dorm13) and 4 well-known machine learning algorithms (*i.e.,* C4.5 decision tree, BayesNet, NaiveBayes and Support Vector Machine) [23]. Experimental result analysis reveals that using HIT dataset all the applied machine learning classifiers classify WeChat text and picture messages traffic very accurately as compared to Dorm13 dataset. Using HIT dataset, all ML classifier perform very well, but C4.5 and Support Vector Machine are the ones that give very effective precision results of 99.91% and 99.57%, respectively, as compared to other ML classifiers.

In the work "Network Traffic Classification techniques and comparative analysis using Machine Learning algorithms", the authors discuss network traffic classification techniques. Also, they developed a real time internet dataset using network traffic capture tool (*i.e.,* Wireshark Tool), then to perform a comparative analysis by applying four machine learning classifiers (*i.e.,* Support Vector Machine, C4.5 decision tree, NaiveBays, and BayesNet) [24]. The experimental result reveals that C4.5 classifier gives high precision as compared to other machine learning classifiers which are 78.91 %.

3.2 Traffic Classification in NFV-based Networks

To the best our knowledge, in NFV-based networks, there are few works that perform traffic classification [25, 26, 27].These works are described in the next paragraphs.

In the work "vTC: Machine Learning Based Traffic Classification As a Virtual Network Function", the authors propose a design of virtual network functions (vTC) to flexibly select and apply ML classifiers on run time [25]. This work analyzes and observes the effectiveness of several ML classifiers, such as K-Nearest Neighbors, Support Vector Machine, Decision Tree, Adaptive Boosting, Naive Bayes and Multi-Layer Perception, that have been helpful in classification problems.

The results of evaluating these classifiers disclose that their effectiveness depends highly on the analyzed protocols as well as the features collected from network data. The proposed vTC for traffic classification can improve the precision by up to 13%. Although the authors do not analyze the main features of hidden NFV traffic, they improve the efficiency of traffic classification on NFV-based networks.

The authors of "Traffic-Aware Placement of NFV Middleboxes" proposed an algorithm that processes and filters the traffic among middleboxes [26]. They analyze the traffic changing effects and study the efficient deployment of NFV middleboxes in Software-Defined Networks (SDN) to reduce the traffic. Although the authors analyze the effects of traffic changing with an algorithm, they do not focus on the features that compose this traffic.

In the work "Efficient NFV deployment in data center networks", the authors focus on deploying NFV in data center networks proposing a heuristic algorithm for allocating Virtual Network Functions (VNFs) [27]. This algorithm was trained by using collected data from a simulated data center supporting multiple VNFs deployed in VMs. The proposed algorithm is a solution to slow down the growth of the east-west traffic and reduce the waste of data center computation resources. Also, it allows the scaling of the networks in a more agile way in comparison with other algorithms.

3.3 Summary

Table 3.1 presents a summary of the related work. Unlike the works mentioned in the previous sections, we carry out a benchmarking to analyze the behavior of ML supervised algorithms in the IP traffic classification in NFV. Consequently, first, a traditional dataset was adapted from traditional networks to the conditions of NFV to classify traffic. Second, the behavior of three supervised algorithms (*i.e.,* C4.5, NaiveBayes, and BayesNet) that have been used on traditional networks was compared for traffic classification. Third, the algorithms were tested and their efficiency in terms of time-response and precision was analyzed.

Table 3.1: Related Work

Reference	**Focus**	**Network**		**Type of Traffic**	**Techniques**
		Traditional	**NFV**		
[18]	Classification and Control	✓		IP	Supervised algorithms
[19]	Classification	✓		IP	Supervised algorithms
[20]	Classification	✓		SSH, Skype	Supervised algorithms
[21]	Classification	✓		IP	Supervised algorithms
[22]	Comparative Analysis	✓		HTTP	Supervised algorithms
[23]	Classification	✓		Text , Picture	Supervised algorithms
[24]	Classification	✓		IP	Supervised algorithms
[25]	Classification		✓	NFV	Supervised algorithms
[26]	Deployment		✓	NFV	Heuristic algorithm
[27]	Allocation		✓	NFV	Authors algorithm
This work	Classification		✓	IP	Supervised algorithms

Chapter 4

Dataset Construction

To characterize the hidden traffic that travels by NFV-based networks, we propose the adaptation and construction of the structure of a dataset that contains the information of these networks.

In this chapter, we describe the process for constructing the dataset in the NFV classification and present the test NFV-based networks (*i.e.,* a Software-defined network (SDN) and an LTE EPC network) that allow us to obtain the characterization of the initial datasets (*i.e.,Traditional* and *Virtual*).

4.1 Dataset Construction Process

To classify the traffic in NFV-based networks is needed a dataset that represents their intrinsic features. Therefore, to structure this dataset is needed to select features that reflect the behavior of NFV traffic including the applications or types of services involved in the application layer. Figure 4.1 presents the process that we follow to build up the dataset proposed for traffic classification in NFV-based networks. To develop this process, we take as a basis the methodology developed in [50], that proposes a cascade model (*i.e.,* each step has a step as a prerequisite).

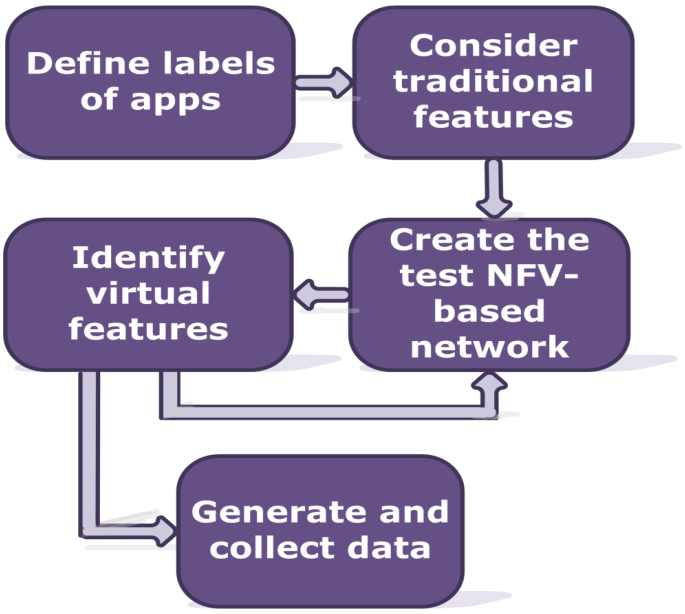

Figure 4.1: Process for dataset creation

In particular, our process is formed by five steps as follows.

Define labels of apps is to categorize a set of applications that use the TCP protocol in the transport layer and that are used to classify the traffic. In this step we take as a base 13 labels of applications that were used in the work [1] for classifying their TCP traffic. The authors defined applications for each label with a deep analysis of their traffic. In our case, we determine each application with the analysis of the protocols extrated from the NFV-based network traffic. These applications allow us to categorize our TCP traffic (See Table 4.1).

Consider traditional features is to take as a basis the features of the dataset proposed in [50]. This dataset has 10 features such as frame length, IPv4 packet size, client and server IP and TCP ports, length of IP and TCP headers in bytes, IP protocol, TCP sequence number, and the size of the TCP window on reception. These features (Table 4.2) provide information about the TCP traffic and allow the authors to classify it in normal and anomalous. These features form the structure of our *Traditional* dataset.

Create the test NFV-based network aims to emulate an NFV-based network,

Table 4.1: Labels of applications [1]

Class	Applications
WEB	Web browsers, web applications
MAIL	IMAP, POP, SMTP
BULK	FTP, wget
ATTACK	Port scans, worms, viruses, sql injections
CHAT	MSN Messenger, Yahoo IM, Jabber
P2P	Napster, Kazaa, Gnutella, eDonkey, BitTorrent
DATABASE	MySQL, dbase, Oracle
MULTIMEDIA	Windows Media Player, Real, iTunes
VOIP	Skype
SERVICES	X11, DNS, IDENT, LDAP, NTP
INTERACTIVE	SSH, TELNET, VNC, GotoMyPC
GAMES	Microsoft Direct Play
GRID	Grid computing

where we can generate and collect data for their subsequent analysis.

Identify virtual features is to determine the particular features that characterize the traffic in NFV-based networks. Here, to identify the virtual features, we collect and analyze the traffic in different points of the network for identifying the main features that determine the behavior and route of such traffic using Wireshark [50].

Generate and collect data aims to build up and fill the structure of datasets created, using traffic-generated tools and traffic analyzers such as D-ITG, Iperf, and Wireshark.

4.1.1 Case 1: Dataset construction on NFV-based SDN

Initially, with the emulation of a test NFV-based SDN network, we follow the dataset construction process:

(1) We consider the features described in Table 4.2 to construct the structure of the dataset called *Traditional*.

Table 4.2: Structural Traditional Features

Network	Feature	Description	Example
Traditional	frame.len	Frame length on the wire	60bytes-(480bits)
	ip.src	Source address	192.168.0.24
	ip.dst	Destination address	192.168.0.27
	ip.hdr_len	Header length	5-(20 bytes)
	tcp.srcport	Source port	43266
	tcp.dstport	Destination port	22
	tcp.seq	Secuence number	37
	tcp.hdr_len	Header length	32
	tcp.window_size	Calculated window size	1247
	ip.proto	Protocol	6-(TCP)17-(UDP)

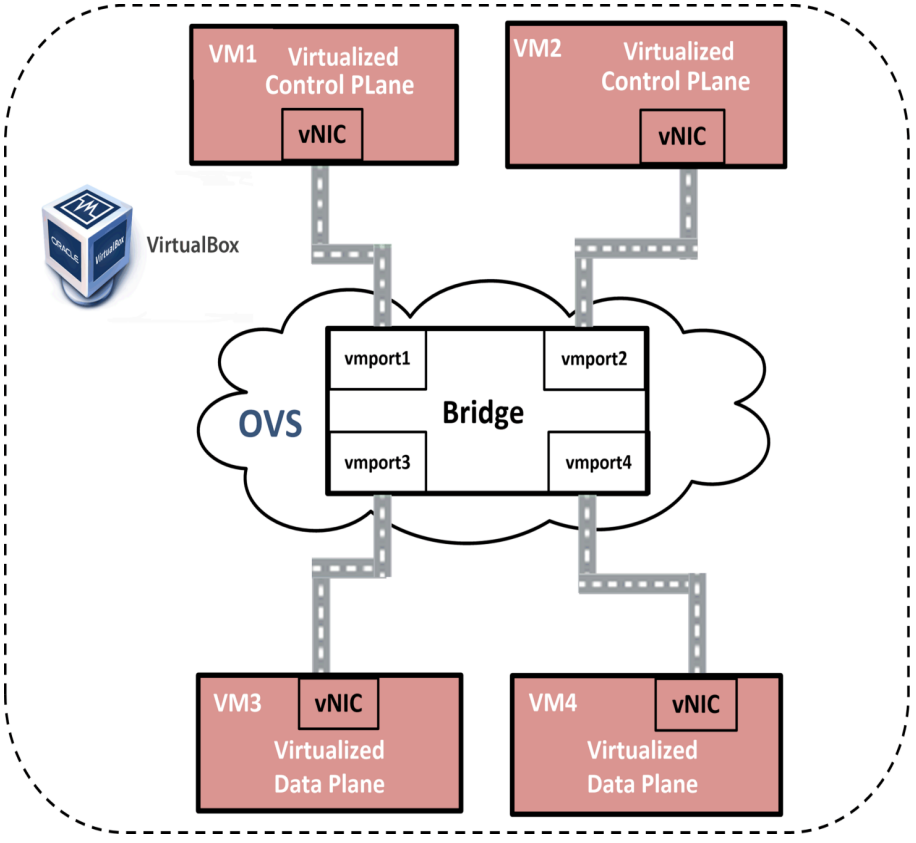

Figure 4.2: NFV-based SDN

(2) We propose the test NFV-based network of Figure 4.2 that is composed by four VMs, two running a Ryu controller [51] (*i.e.*, a controller to increase

network agility, facilitating the traffic management) and the other ones running Open vSwitch [52] (OVS, which is used to forward traffic between VMs on the same physical host, and between VMs and the physical network) emulated in Mininet [53]. The characteristics of this network are listed in Table 4.3.

Table 4.3: Configuration of NFV-based SDN

Name	Processor	Memory	Disk	OS	Virtualization
Ryu	Intel Core i3-2328M CPU 2.20 GHz	2.0 GiB	6.2 GB	Ubuntu 16.04	VM VrtualBox
Mininet					
OVS	Intel Core i3-2328M CPU 2.20 GHzx4	5.6 GiB	66.0 GB		

In particular, in the above mentioned environment, the Ryu controller was executed on two VMs (*i.e.*,VM_1 and VM_3), an OVS was deployed (to enable communication among the VMs) on a server, and in this OVS, we create a bridge with four virtual ports to connect the VMs (see Snippet 4.1).

Snippet 4.1: Bridge configuration

```
93763522-282c-41fa-83eb-a116c5e2c64e
Bridge bridge
  Port "enp2s0"
      Interface "enp2s0"
  Port "vmport1"
      Interface "vmport1"
  Port "vmport2"
      Interface "vmport2"
  Port "vmport3"
      Interface "vmport3"
  Port "vmport4"
      Interface "vmport4"
  Port bridge
      Interface bridge
          type: internal
```

```
ovs_version: "2.5.2"
```

The Snippet 4.1 presents the bridge state, the first line refers to the Unique Universal ID (UUI) of our environment according to RFC 4122 [54]. Next, we see the name of our bridge "bridge" and its respective ports *enp2s0, vmport1, vmport2, vmport3,* and *vmport4.* Finally, the interface "bridge" that is of internal type and is recognized like another interface of the system.

(3) To identify virtual features, the traffic was analyzed in each VM, getting similar features than in traditional networks. Then, the traffic was analyzed in the bridge, observing that it varies when passes by virtual ports. In these ports, traffic presents changes in the origin and destination addresses. The traffic that is before the virtual port has the addresses of VMs. However, when traffic passes by the port and it is on the bridge, the addresses are modified and correspond to the addresses of such a port. Therefore, the indicators of Protocol in the Frame (*i.e., frame.protocols*) and Ethernet Type (*i.e., eth.type*) were selected to characterize the traffic in the bridge. Finally, the four features above mentioned (*i.e.,* source and destination Ethernet addresses, frame protocols and Ethernet type) were selected as the basis for building the structure of the *Virtual* dataset (Table 4.4).

Table 4.4: Structural Virtual Features

Network	Feature	Description	Example
Virtual SDN	frame.protocols	Protocols in frame	eth:ethertype:arp
	eth.dst	Destination	68:a0:f6:b0:14:84
	eth.src	Source	08:00:27:f0:dc:db
	eth.type	Type	0x0800-Ipv4 0x86DD-Ipv6

(4) In this case, to generate traffic, we use the Distributed Internet Traffic Generator (D-ITG) tool [55]. Snippet 4.2 depicts the command line for traffic generation with D-ITG. In our case, we regularly change the parameter "**-t**" (from 5 sec to 30 minutes), in order to generate traffic with variations of time between the hosts of the networks emulated.

Snippet 4.2: Configuration D-ITG commands

```
-a <dest_address> -C <rate [pkts/s]> -c <pkt_size>\\
-t <duration [ms]> -T <protocol>
```

In the data collection process, we used two networks (one per machine, VM_2 and VM_4) handled by the Ryu Controller deployed in VM_3 (*i.e.*, a tree topology of depth 3, 21 switches, and 64 hosts; see Figure 4.3). The IP addresses assigned to each are presented in Table 4.5.

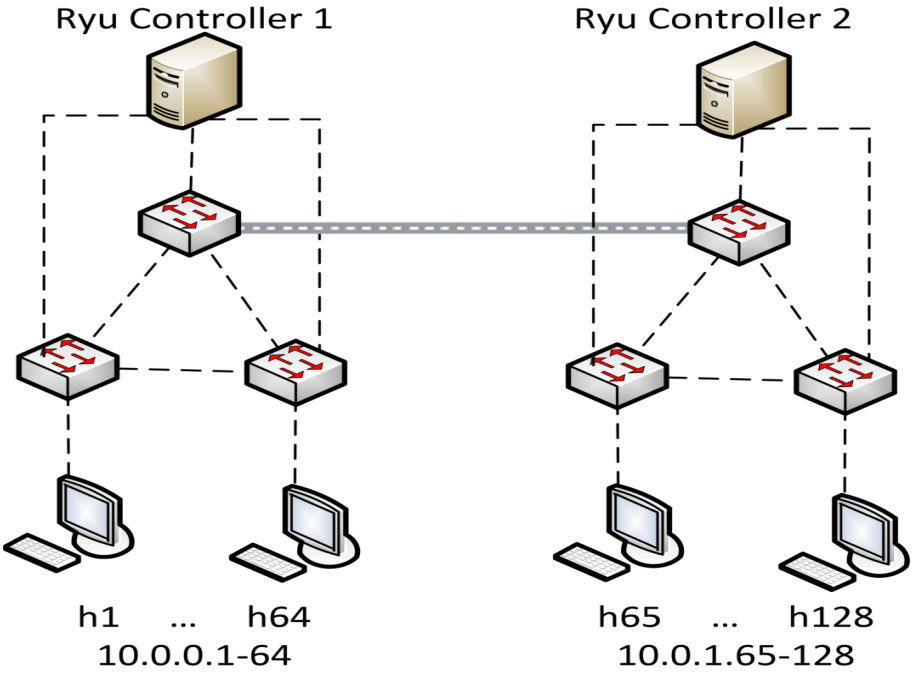

Figure 4.3: Mininet networking configuration

Table 4.5: Assignment of IP addresses

	Host Range	**IP address**
Network 1	h1 - h64	10.0.0.1-64
Network 2	h65 - h128	10.0.1.65-128
Ryu Controller 1	-	10.0.0.112
Ryu Controller 2	-	10.0.0.113

The collection was performed as follows.

- We put sniffers on two points of reference: Point A (*vmport2* and bridge) and Point B (*vmport4* and bridge).
- At Point A, we captured ten random periods of 30 minutes, and in the Point B we captured one period of 30 minutes.
- We collected the data in a .pcapng file using Wireshark.
- We use the T-shark command [50] of Snipper 4.3 for the conversion of the data to .csv (for managing in Weka [56]) and the extraction of the features of Tables 4.2 and 4.4.

Snippet 4.3: T-shark command to extract data

```
tshark -r PointA.pcapng -T fields -e frame.len\\
       -e frame.protocols -e eth.dst -e eth.src\\
       -e eth.type -e ip.proto -e ip.src -e ip.dst\\
       -e ip.hdr_len -e tcp.srcport -e tcp.dstport\\
       -e tcp.seq -e tcp.hdr_len -e tcp.window_size\\
       -E header=y -E separator=, -E quote=d \\
       -E occurrence=f  > PointA.csv
```

In this command, the "**-r**" parameter reads the "PointA.pcapng" file previously extracted using Wireshark, "**-T**" as a text file with "**-e**" fields representing the characteristic of Table 4, "**-E**" indicates the print option fields of the data; and ">" followed by the file name and its extension ("PointA.csv") allows saving that new document.

As a summary the *traditional* and *Virtual* datasets in the NFV-based SDN network are constructed as follows:

Table 4.6: Structure of Traditional and Virtual datasets

Network	Feature	Description	Example
Traditional	frame.len	Frame length on the wire	60bytes-(480bits)
	ip.src	Source address	192.168.0.24
	ip.dst	Destination address	192.168.0.27
	ip.hdr_len	Header length	5-(20 bytes)
	tcp.srcport	Source port	43266
	tcp.dstport	Destination port	22
	tcp.seq	Secuence number	37
	tcp.hdr_len	Header length	32
	tcp.window_size	Calculated window size	1247
	ip.proto	Protocol	6-(TCP)17-(UDP)
Virtual SDN	frame.protocols	Protocols in frame	eth:ethertype:arp
	eth.dst	Destination	68:a0:f6:b0:14:84
	eth.src	Source	08:00:27:f0:dc:db
	eth.type	Type	0x0800-Ipv4 0x86DD-Ipv6
Number of instances	1.048.575		

4.1.2 Case 2: Dataset construction on NFV-based LTE EPC

4.1.3 Test Environment

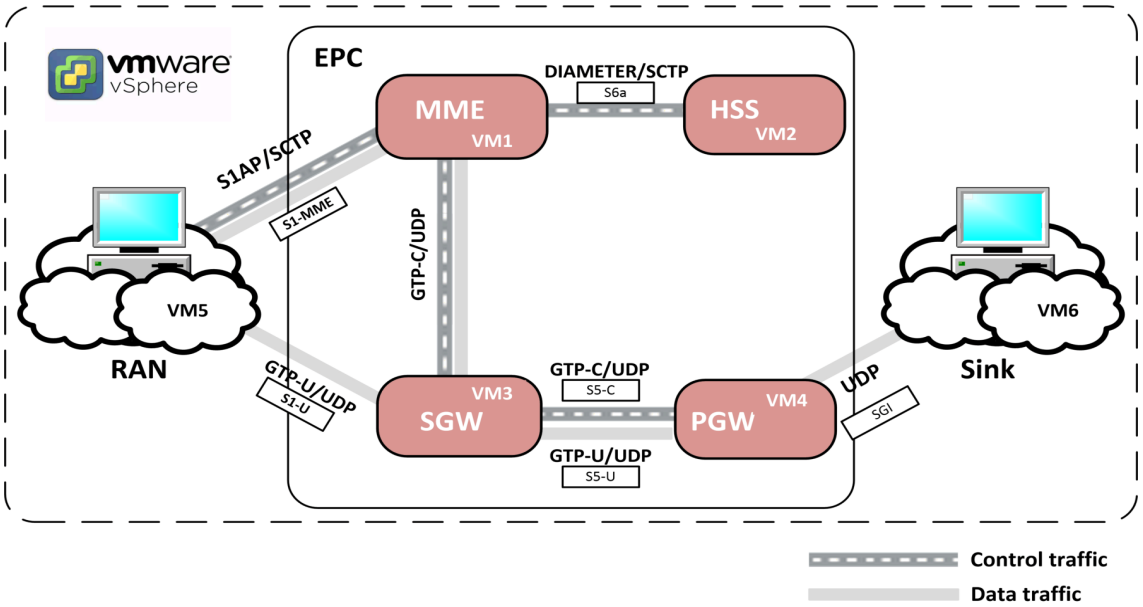

Figure 4.4: NFV-based LTE EPC Network
Source: [57]

We consider mobile networks as a second scenario for classifying traffic in NFV, since these networks, nowadays, are faced with an increase in the traffic that passes through them. And this increase presents a problem for the operators in the deployment of new services and the location of new hardware devices in the network.

For emulating a mobile network, we take as the basis the emulation of an NFV-based LTE EPC network, proposed in [57] (see Figure 4.4). This emulation is based on the client-server paradigm, in which, the server is multi-threaded to serve all client requests.

In this network, the authors present six entities, including the main network functions of EPC (*i.e.,* MME, HSS, SGW, and PGW), a RAN simulator to generate traffic in EPC, and a Sink module responsible for receiving the traffic generated (uplink) and generate acknowledgment data (downlink). These functions are pre-

sented as software modules running on VMs hosted on a private cloud. These VMs have specific characteristics listed in Table 4.7 and were deployed in VMware vSphere [58].

Table 4.7: Configuration of NFV-based EPC modules

Entity	No.Cores	RAM	Disk	OS	Virtualization
RAN	8	4 GB	10 GB	Ubuntu 14.04	VMware vSphere
MME, SGW, PGW, HSS	2	2 GB			
Sink	4	2 GB			

The implementation of this emulation is based on the object-oriented paradigm in which each entity uses a server or a client object in each of its ports. These ports correspond to the interfaces described by 3GPP (*i.e.,* S1-MME, S1-U, S11, S5-C, S5-U, S6a, and SGi). The communication by these objects occurs by sending requests and responses between different entities of LTE EPC. The description of the EPC entities [57] is presented below:

The **Mobility Management Entity (MME)** maintains a global map throughout the communication to store the connection information and the status related to the user equipment (UE). MME communicates with other system entities using three different ports corresponding to three different standard interfaces: S1-MME, S6a, and S11. On the S1-MME interface, it uses an SCTP server object to communicate with eNodeB. On the S6a interface, it uses an SCTP client object to communicate with HSS to obtain information related to the UE and the updating of their locations. At interface S11, it uses a UDP client object to communicate with SGW for session/bearer related procedures.

The **Home Subscriber Server (HSS)** uses a MySQL object for database-related operations and an SCTP client object for communicate with MME over the S6a interface. Multiple instances are created in the MySQL database where UE information is stored, such as authentication, subscription profile, and location tracking.

The **Serving Gateway (SGW)** manages control plane communication with MME and PGW on interfaces S11 and S5-C, respectively. On the S11 interconnect port, it uses a UDP server object and on the S5-C interface port, it uses a client UDP

object. On the other hand, it manages the data plane communication with eNodeB and PGW in the interface S1-U and S5-U respectively. On the S1-U interface, it uses a Uplink object from the UDP server and a UDP client downlink object.

The **Packet Data Network Gateway (PGW)** handles control plane communication with SGW on interface port S5-C through a UDP server object. It manages the data plane communication with SGW and Sink on interface ports S5-U and SGi, respectively. On the S5-U interface, it uses a Uplink object from the UDP server and a UDP client downlink object. A separate IP table map is also used to assign a static IP address for each UE being connected to the EPC.

The **Radio Access Network (RAN) Simulator** proposed in [57] combines the functionalities of UE and eNodeB to generate control and data traffic to EPC. This RAN simulator performs multiple threads and creates a RAN object for each wire, which in turn controls the control plane and data plane communication with MME and SGW respectively. For the tunneling process during UE data transfer, separate threads are generated to monitor and retrieve UE packets from the core stack. For control plane communication, an SCTP client object is used on the S1-MME interface port. For data plane communication, it uses separate UDP client/server objects on interface port S1-U for uplink/downlink data transfer. A global array of RAN contexts is used to maintain all information related to UEs and eNodeBs.

The **Sink module** represents the Packet Data Network (PDN) server. It is responsible for receiving the generated uplink traffic and returning the acknowledgment data as downlink traffic. As in the case of the RAN simulator, several Sink objects are created, each in their thread.

For the specific dataset construction in the NFV-based LTE EPC:

(1) We identified three labels for this dataset **MOBILE** (*i.e.,* gsm protocols), **SERVICES** (*i.e.,* SLL and X11), and **CONTROL** (*i.e.,* only TCP/IP protocol).

(2) We consider the features of Tables 4.2 and 4.4, for the construction of the structure of new *Traditional* and *Virtual* datasets, respectively.

(3) We deploy the network of Figure 4.4 with its respective characteristics (Table 4.7).

(4) We tested the EPC implementation using data traffic, determining a specific number of UEs to EPC from the RAN simulator to the Sink module. For the traffic generation between EPC modules, we use the Iperf3 [59] tool. Iperf3 allows sending TCP data with different bandwidths and duration. In this case, the particular Iperf3 Client process is started in the RAN simulator, with a required input data rate. And the Iperf3 operation in Server mode in the Sink module. The Snippet 4.4 present the command used for generating traffic by Iperf3.

Snippet 4.4: Iperf3 Command Line

```
iperf3 -B <client_IP> -c <server_IP> -p <server_port> \\
       -b <data_rate> -M <mtu> -t <time[sec]> \\
       -S <type_of_services>
```

In this command line, the "**-B**" link Host Interface (*i.e.,* 172.16.0.2), "**-c**" run iperf3 in Client mode, "**-p**" configure the Server port to listen, and it is the same as the Client (*i.e.,* 55000), "**-b**" sets the transmission bit rate per second (default 1 Mbps), "**-M**" maximun segment size TCP/STCP (default MTU - 40 bytes), "**-t**" time in seconds to transmit (default 10 sec); and "**-S**" set the IP service type.

In this second case study, we performed eleven captures of thirty minutes each one in the MME entity, to know the behavior of TCP traffic in the EPC. For the analysis of data traffic, the authors in [57] propose to modify the speed of traffic generation in Iperf3. For this reason, we vary the value of parameter "**-b**", between 20 Mbps and 110 Mbps.

As a summary the *traditional* and *Virtual* datasets in the NFV-based LTE-EPC network are constructed as follows:

Table 4.8: Structure of Traditional and Virtual datasets

Network	Feature	Description	Example
Traditional	frame.len	Frame length on the wire	60bytes-(480bits)
	ip.src	Source address	192.168.0.24
	ip.dst	Destination address	192.168.0.27
	ip.hdr_len	Header length	5-(20 bytes)
	tcp.srcport	Source port	43266
	tcp.dstport	Destination port	22
	tcp.seq	Secuence number	37
	tcp.hdr_len	Header length	32
	tcp.window_size	Calculated window size	1247
	ip.proto	Protocol	6-(TCP)17-(UDP)
Virtual SDN	frame.protocols	Protocols in frame	eth:ethertype:arp
	eth.dst	Destination	68:a0:f6:b0:14:84
	eth.src	Source	08:00:27:f0:dc:db
	eth.type	Type	0x0800-Ipv4 0x86DD-Ipv6
Number of instances	1.048.575		

Chapter 5

Benchmarking Results

This chapter presents the process for benchmarking the efficiency of the supervised ML algorithms (*i.e.,* C4.5, NaiveBayes y BayesNet) in the traffic classification in the two NFV-based test networks presented in the chapter previous.

In this process, we introduce the preparation of the data collected in the *Dataset Construction* process and its adaptation, in addition to the training, validation, and testing processes of the three supervised algorithms.

5.1 Benchmarking Process

Benchmarking is a systematic process useful for evaluating services or tasks within a system. This allows performing a comparison for generating changes in the performance or development of such a system [60]. In this work, the Benchmarking allows comparing the efficiency of supervised ML algorithms in NFV-based networks. Here, the efficiency is related to the trade-off between time-response (*i.e.,* time spent by algorithms for traffic classification) and precision (*i.e.,* the number of class members classified correctly over the total number of instances classified for a given class) [1].

The benchmarking was performed by using the methodology called Training, Validation and Testing [30]. For data classification and prediction processes, this method-

ology proposes a division of the overall dataset into two subsets. The first one for training and validation (2/3 of the data) and the second subset for testing (1/3 of the data), see Figure 5.1.

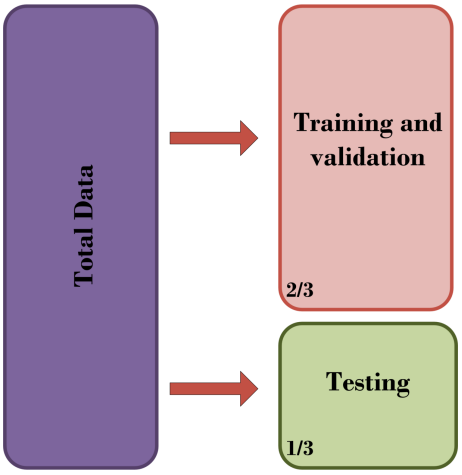

Figure 5.1: Division of data

The previously labeled training subset is used by the algorithms to create their previous classification model and to validate it, and the subset of tests is used to verify the classification models in a final stage, such that this unlabeled subset is subjected to classification.

In this sense, we propose the benchmarking process composed of four steps, as shown in Figure 5.2.

In this process, the input data are the *Traditional* and *Virtual* dataset, which will be subjected to a selection of features to determine the features with which we will create a third dataset called *Combined*. For the whole process that follows, we use the Weka tool [56] that allows performing an adequate preparation of the data.

Dataset Adaptation is to prepare the Traditional and Virtual data sets to train supervised classification algorithms. In this preparation, we perform two processes:

(1) Lebeling process: We analyze the protocols found in the frames of the collected data streams and based on the labels that the authors propose in [1], we identify for each dataset the necessary labels.

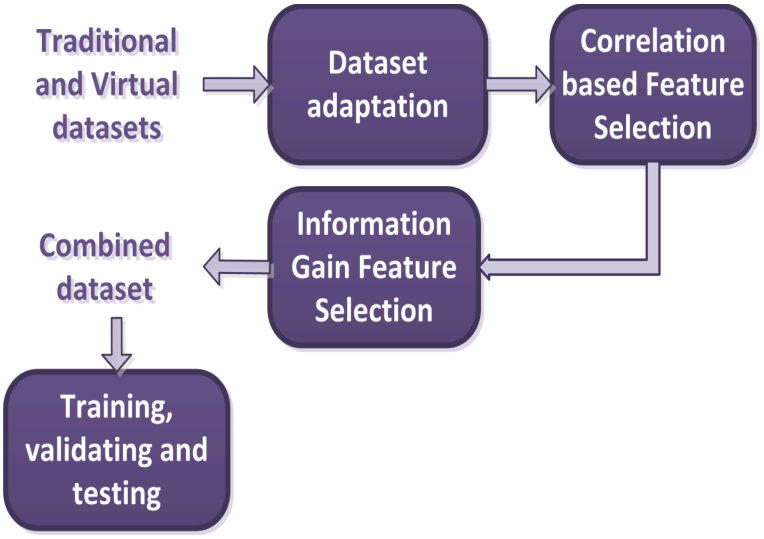

Figure 5.2: Benchmarking process

(2) Filtering process: We use Weka data management options such as filters. It is necessary to use some filters to organize the data so that the algorithms can perform a correct use of them, in this sense we used the following filters:

- **Discretize** is to transform features with continuous values to features with a finite range of values. We disallowed Bayesian algorithms to have better data handling because they work better in their training using categorical data.
- **Normalize** is to scale an attribute to a particular range generally -1 to 1, or 0 to 1. We normalized to avoid that some attributes with higher values gain a significantly more significant weight in the final model. (*e.g., frame.length, tcp.window_size*).
- **Class balancer** allows adjusting the weights of the data labels keeping the same weight. We use a class balancer because the selected algorithms (*i.e.,* C4.5, NaiveBayes, and BayesNet) are designed to optimize overall precision without considering the distribution of each class, and as a result, tend to ignore small classes in classifying.

Correlation based Feature Selection is to identify the features that have greater relation with the feature Class, the labels of apps. From this process, the features

with a correlation upper to 0,4 are chosen. This selection was made as follows:

(1) The *Traditional* and *Virtual* datasets were opened in Weka.

(2) *GreedyStepwise* was selected as the search method and *CfsSubsetEval* as the evaluator in Weka (select attributes option).

(3) The selection was executed to determine the features.

(4) The most correlated features were selected, for the *Traditional* dataset, and for the *Virtual* dataset.

Information Gain Feature selection consists of identifying the features that contribute a significant amount of information to determine the corresponding label in *Traditional* and *Virtual* datasets. This selection was carried out by selecting the following combination:

- We selected the *Ranker* search method to classify the features according to an individual evaluation.
- We used the evaluator *InfoGainAttributeEval* to determine the relation of gain information per feature and the types of application.

At the end of the two feature selections, we obtain the most relevant data from the Traditional and Virtual dataset for the construction of the Combined data set.

Training, Validation and Testing aim to divide the *Traditional*, *Virtual* and *Combined* datasets into the two respective subsets of data proposed for the Methodology. Re-sampling extracted these subsets of data: the first for training corresponding to 60% of the total data and the second subset for the test corresponding to the remaining 40%.

Here, the supervised algorithms C4.5, NaiveBayes, and BayesNet are trained and evaluated since they have been successfully (achieving high results of precision) used to classify traffic in traditional networks. Initially, in the training stage, the training data subset is used for preparing (*i.e.,* to generate a previous classification model)

each algorithm. Then, in the validation stage, the algorithms improve their corresponding classification model by using *Cross-validation*. Finally, in the testing step, the testing subset is used for obtaining the final classification model of algorithms above mentioned.

To train the supervised algorithms with the data sets extracted in each case study proposed in Chapter 4, we perform the benchmarking process in each one.

5.2 Case 1: Traffic Classification on NFV-based SDN

In this case, we obtained the following benchmarking results:

(1) In the *Dataset Adaptation* step, by analyzing the protocols, we identify three of the thirteen labels we took as the basis of the work [1] (*i.e.,* INTERACTIVE, SERVICES, and WEB). To categorize the applications on the OpenFlow protocol we define a fourth label that we call OPENFLOW, and finally to tag other types of protocols found we propose a fifth label that we call OTHER.

(2) Continuing with the two *Features Selection* processes, we identified the most relevant features. For the *Traditional* dataset we have *tcp.srcport*, *tcp.dstport*, *tcp.window_size*, *ip.src* and *ip.dst*, and for *Virtual* dataset only *frame.protocols*.

(3) Finally, we continued with *Training, Validation and Testing* processes (Section 5.1) and the corresponding results are presented below.

5.2.1 Results

In this case, the algorithms C4.5, NaiveBayes, and BayesNet were trained by using the *Traditional*, *Virtual* and *Combined* datasets. Initially, in the validation step, the algorithms perform their first classification (they construct a classification model) where we obtain initial values of time-response and precision. Then, when the

algorithms test their model, we extract another value for those two metrics. Finally, to define the behavior of each algorithm and its variation in time-response and precision, we calculate the standard deviation using the results of the validation and test processes, this analysis is performed in each capture of thirty minutes.

Table 5.1: Precision comparison

Point A			
Dataset	**C4.5**	**NaiveBayes**	**BayesNet**
Traditional	$92,535 \pm 0,185$	$80,802 \pm 0,034$	$91,118 \pm 0,034$
Virtual	$99,998 \pm 0,001$	$99,998 \pm 0,001$	$99,998 \pm 0,001$
Combined	$99,998 \pm 0,001$	$98,673 \pm 0,006$	$99,225 \pm 0,002$
Point B			
Dataset	**C4.5**	**NaiveBayes**	**BayesNet**
Traditional	$88,854 \pm 0,018$	$79,500 \pm 0,375$	$90,234 \pm 0,116$
Virtual	$99,998 \pm 0,001$	$99,915 \pm 0,012$	$99,953 \pm 0,350$
Combined	$99,998 \pm 0,001$	$90,174 \pm 0,380$	$99,890 \pm 0,005$

Table 5.1 depicts the precision results when C4.5, NaiveBayes, and BayesNet are used for classifying traffic in NFV-based SDN, of the Figure 4.2. These results reveal diverse facts. First, the overall precision is higher than 79,5%, being a similar effect to that achieved by the same algorithms in the traffic classification in traditional networks, which proves that they are a good choice for the classification in virtual networks [1][21]. Second, C4.5 and BayesNet are more accurate than NaiveBayes; this behavior depends on the model that creates each algorithm. On the one hand, C4.5 analyzes and selects as nodes (in its tree model) the features with the highest gain of information which makes it more precise with values between 88,8% - 99,9%. While the NaiveBayes and BayesNet algorithms consider the independence and dependence between the data, respectively, in this way their models perform a longer and more complex data processing than C4.5. Third, the results obtained in the classification for the *Combined* dataset for Point A (98 % - 99 %) show an increase of more than 7 % compared to the results in the classification of the *Traditional* dataset. And in Point B there is an increase of more than 9 % in the same case. This increase is directly related to the features that make up each dataset since they provide a different type of information to the algorithms for creating the classification models. And in this case, the *Combined* dataset is composed of the

features with a greater information gain and correlation of the *Traditional* and *Virtual* datasets. Therefore, when C4.5, NaiveBayes, and BayesNet process and classify the *Combined* dataset, they build a more effective classification model than with the *Traditional* dataset.

Table 5.2: Time-response comparison

Point A			
Dataset	**C4.5**	**NaiveBayes**	**BayesNet**
Traditional	$39,88 \pm 0,968$	$1,540 \pm 0,113$	$11,21 \pm 1,718$
Virtual	$0,240 \pm 0,980$	$0,254 \pm 0,063$	$0,235 \pm 0,091$
Combined	$1,565 \pm 0,063$	$1,480 \pm 0,834$	$7,345 \pm 1,732$
Point B			
Dataset	**C4.5**	**NaiveBayes**	**BayesNet**
Traditional	$0,405 \pm 0,120$	$0,120 \pm 0,070$	$0,350 \pm 0,140$
Virtual	$0,040 \pm 0,028$	$0,030 \pm 0,028$	$0,060 \pm 0,560$
Combined	$0,145 \pm 0,035$	$0,007 \pm 0,000$	$0,680 \pm 0,210$

Table 5.2 present the time-response results of C4.5, NaiveBayes and BayesNet when used to classify traffic in the Points A and B (Figure 4.2), respectively. These results disclose that, first, the algorithms in Point A require more time (from 0,2 sec to 39,8 sec) than in Point B (from 0.007 sec to 0.68 sec), for classifying the datasets. This variation between the time ranges is related to the difference between the number of flows captured at each point, in Point A there are 1,048.575 flows and in Point B 52,264 data flows. Second, in both points, the three algorithms present their lowest time-response of classification when using the *Virtual* dataset, this is presented by the number of features that compose such a dataset (only five features). Third, in both points, the NaiveBayes algorithm presents the smallest variation between the time-responses in the classification of each dataset. And this is because NaiveBayes requires a minimum amount of information, to estimate the data needed for classifying traffic, regardless of the number of dataset features.

To sum up, although C4.5 and BayesNet achieve a precision greater than 88%, they have significant variations (0,04 sec - 39 sec) in their time-response results, because their classification models depend on the amount of data contained in each dataset. While NaiveBayes reaches a precision greater than 79,5%, and its time-responses

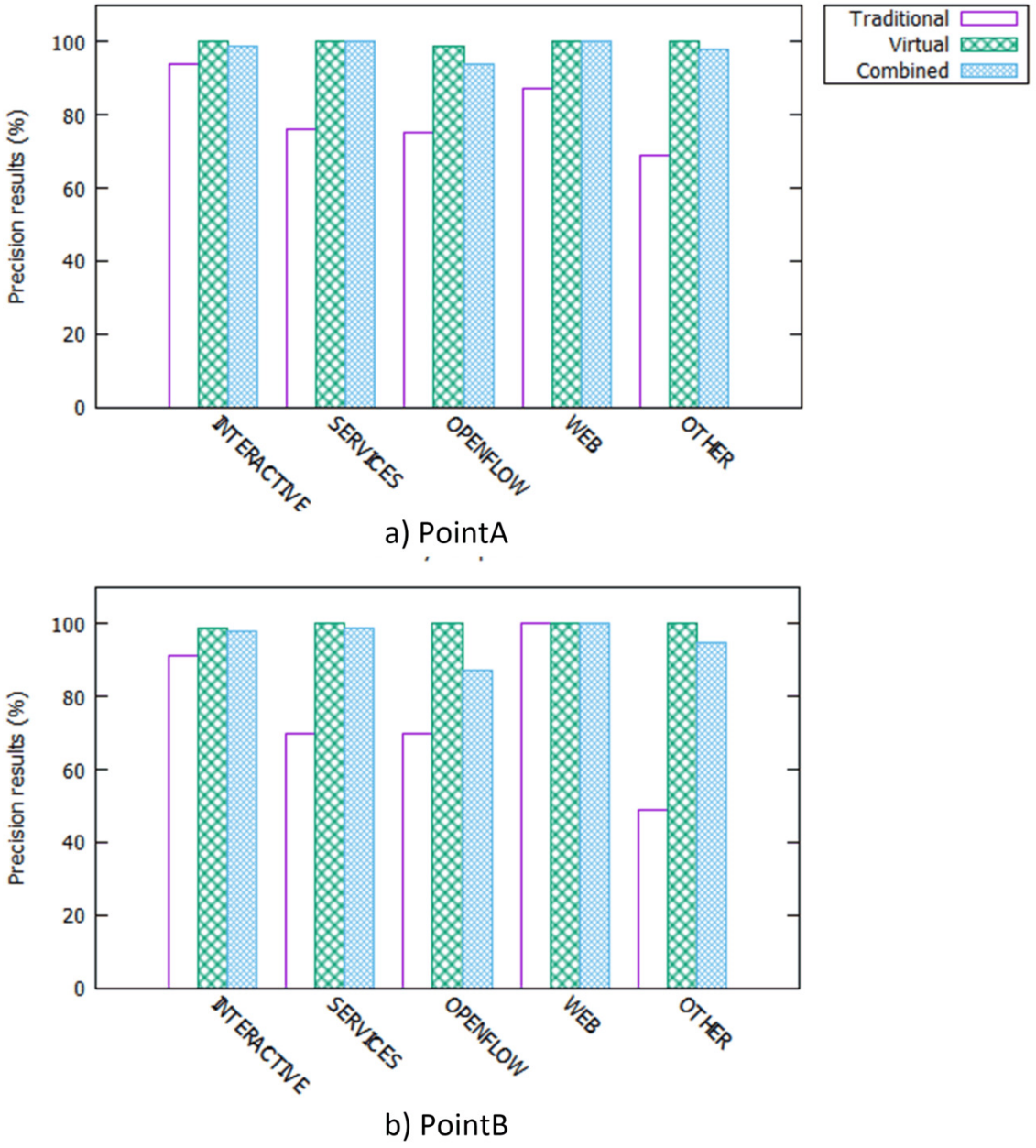

Figure 5.3: Precision per-class in NaiveBayes Classification

results range from 0,007 sec to 2,5 sec. The difference between C4.5, as the algorithm with the highest level of precision, and NaiveBayes, as the most constant in time, is approximately 5,5% in precision; a difference that NaiveBayes compensates, without losing precision, reducing the time-response about 6 sec.

Therefore, we argue than NaiveBayes is the best algorithm to classify traffic in NFV-based SDN since it presents a better trade-off between precision and time-response. It is to highlight that this algorithm has also a high performance for classifying traffic with the *Traditional* dataset achieving a precision range upper 79,5% and time-responses lesser than 1,54 sec.

Figure 5.3 presents the precision results, achieved by NaiveBayes, when classifies the application labels INTERACTIVE, SERVICES, OPENFLOW, WEB, and OTHER. These results reveal first, in both points, NaiveBayes presents its lowest values of precision (from 49,8% to 76,5%) in the classification of some labels (*i.e.,* SERVICES, OPENFLOW, and OTHER). This reduction in precision indicates that *i)* the distribution of these labels in the training subset is not uniform; and *ii)* these labels are composed of a less number of members, on the other labels. Second, in the classification using the *Virtual* dataset the precision values are greater than 99,6% for the five classes, with the *Combined* dataset the rank is upper than 94,9%, whereas the classification with the *Traditional* dataset is greater than 70%. These values indicate again that the classification is linked to the quantity and types of features that compose each dataset.

5.3 Case 2: Traffic Classification on NFV-based LTE EPC

In this case, we obtained the following benchmarking results:

(1) In the *Dataset Adaptation* process, we identify only one of the thirteen labels we took as the basis (*i.e.,* SERVICES). In this case, as in case 1, we propose new tags to categorize other types of applications. In this case, we propose a label called CONTROL to identify frames that contain only the TCP. And another label called MOBILE, to categorize the basic applications of an LTE network that are part of the gsm protocol.

(2) After processing the *Traditional* and *Virtual* datasets in the *Features Selection* processes, we obtained the following: for the *Traditional* dataset we have *frame.len tcp.srcport*, *tcp.dstport*, *tcp.seq*, *tcp.hdr_len*, and *tcp.window_size*, and for *Virtual* dataset only *frame.protocols*.

(3) Finally, With the three datasets created (*i.e., Traditional, Virtual,* and *Combined*) we train the three supervised ML algorithms (*i.e.,* C4.5, NaiveBayes, and BayesNet). The respective results of these processes are presented below.

5.3.1 Results

Table 5.3 presents the precision results obtained by each algorithm. These results reveal that, first, the precision values obtained are higher than 70,7%, which shows a similar behavior of the algorithms that in Case 1. Second, C4.5 and BayesNet achieve higher precision values than the NaiveBayes algorithm, with a difference greater than 1 %. This difference in precision is related to the construction of the classification model of each algorithm. Third, the behavior of Case 1 is repeated in which the three algorithms present their lowest percentage of precision when classifying the *Traditional* dataset, this is because of the amount of information provided by the features of such a dataset during the classification. In this sense, a variation greater

than 4% between the classification precision between the *Traditional* and *Combined* dataset is identified.

Table 5.3: Precision comparison - LTE EPC System

Dataset	**C4.5**	**NaiveBayes**	**BayesNet**
Traditional	$96,178 \pm 0,020$	$70,770 \pm 2,512$	$94,913 \pm 0,029$
Virtual	100 ± 0	100 ± 0	100 ± 0
Combined	100 ± 0	$98,802 \pm 0,002$	$99,569 \pm 0,021$

Table 5.4 presents the results of time-response for classifying each dataset. These results reveal that, first, in this second case study, we have a range of values between 0,035 sec to 0,3 s, being a range of values with a smaller difference, than the difference obtained in Case 1. This range of values is related to the quantity of data that contains each dataset, and in this case, the three datasets are composed of 24,105 data streams. Likewise, we note that the variations between the results of each algorithm are minimal and somewhat similar. Second, we identify a behavior similar to that of Case 1, the three algorithms (*i.e.,* C4.5, NaiveBayes, and BayesNet) take a little more time in the classification of the *Traditional* dataset, whereas for the *Virtual* dataset they present the least time for classifying, this because of the six more features that compose the *Traditional* dataset. Third, although the values are very close, Naive Bayes presents the lowest values, with an approximate average of 0.04 sec for classifying the three datasets (*i.e., Traditional, Virtual,* and *Combined*).

Table 5.4: Time-response comparison - LTE EPC System

Dataset	**C4.5**	**NaiveBayes**	**BayesNet**
Traditional	$0,300 \pm 0,056$	$0,045 \pm 0,007$	$0,135 \pm 0,007$
Virtual	$0,045 \pm 0,007$	$0,010 \pm 0$	$0,050 \pm 0,042$
Combined	$0,085 \pm 0,021$	$0,035 \pm 0,007$	$0,090 \pm 0,028$

To determine the most efficient algorithm, we consider that, first, the difference in precision between C4.5 and BayesNet is less than 2 %, reaching higher precision levels than NaiveBayes. Second, regarding the time-response results, NaiveBayes has the lowest variation in its results and the lowest average time-response, approximately 0,04 sec. Third, the difference in response time results indicates that C4.5 and

BayesNet require more time, 0,1 sec and 0,05 sec respectively than Naive Bayes in the classification. With these small differences in time-response, we argue that the BayesNet algorithm is more efficient than NaiveBayes, thus guaranteeing precision values in a higher range with a small increase of 0,05 sec in the time-response.

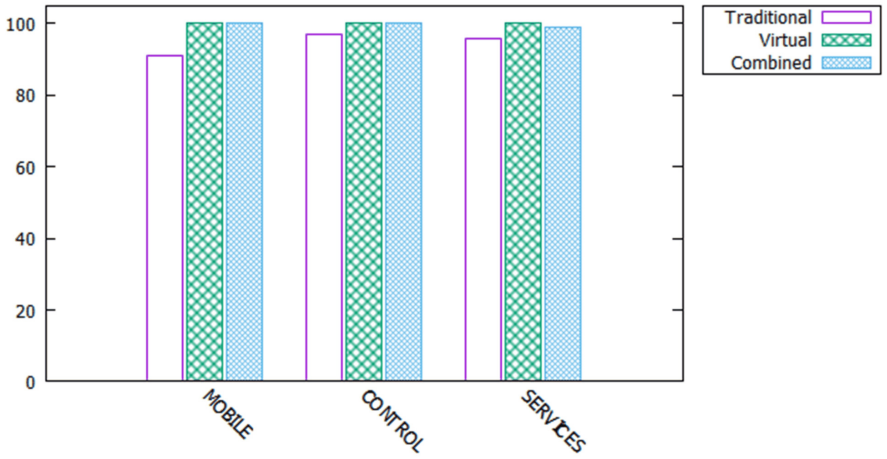

Figure 5.4: Precision per-class in BayesNet Classification

Figure 5.4 presents the precision results, achieved by BayesNet, when classifies the defined application labels MOBILE, CONTROL and SERVICES. With these results, we note, first, BayesNet delivers higher results than 80 %, being excellent values for the classification of IP traffic. Second, for the three labels, the algorithm behaves very similar to classification precision averages. Third, it is relevant that for the *Virtual* dataset the classification percentages are the highest, not only for the labels presented above but also in the ranking with the other two algorithms. These significant results are presented by the amount of data in the *Virtual* dataset and the information they provide for the classification.

Chapter 6

Conclusions and Future Work

In this chapter, we start by answering the proposed research question. Then, we provide the main comparative conclusions obtained by the evaluation. Finally, we expose directions and implications for future work.

6.1 Conclusions

This work presented the investigation perform to answer the research question:

How do supervised algorithms behave in autonomic classification of IP traffic in NFV-based networks?.

To answer this question, we carried out a benchmarking of some supervised ML algorithms (*i.e.,* C4.5, NaiveBayes, and BayesNet) in the traffic classification in a NFV-based SDN and in a NFV-based LTE EPC.

Performing our benchmarking, it was needed to analyze certain features and the behavior of traffic in NFV networks, to adapt a traditional dataset to a NFV-based network, and to train and validate the algorithms with the adapted data for determining their behavior. In this sense, the behavior of C4.5, NaiveBayes and BayesNet was evaluated in two aspects, first, quantitatively as follows:

In this aspect it is necessary to emphasize that the behavior is related to the efficiency achieved in the classification that depends on the trade-off between time-response and percent of precision. The results obtained reveal that:

- In the Case 1, we find out that the supervised algorithms C4.5 and NaiveBayes has a range of precision from 80% to 99% for traffic classification in traditional and NFV-based SDN. And NaiveBayes algorithm is the most efficient classifier, since it has a precision value range upper than 80,8%, similar to C4.5, and between response times in the classification of each dataset, has minimal variations.
- In Case 2, the results obtained by the evaluated algorithms increased regarding the Case 1. C4.5 presented the best precision in the classification of traffic with values higher than 96,17 %, Naive Bayes and Bayes Net likewise have values, higher than 70,7 % and 94,9 %, respectively. Although there is a percentage difference upper than 1 % between these results, the three algorithms handle a similar time-response range between 0,03 sec and 0,3 sec, being a smaller and more balanced range than Case 1.

Considering the above results, we can state that supervised ML algorithms have a good performance in the classification of NFV traffic, and they can improve their precision in this type of classification concerning the classification in traditional networks.

Second, qualitatively, we emphasize that the new features identified for NFV-based networks allow: *i)* monitoring hidden traffic circulating by VMs, *ii)* improving the management of such a traffic; and *iii)* increasing the precision levels in the classification. Furthermore, these features allow the algorithms to improve their classification models which lead to possible improvements in the precision and reduction in time-response.

Finally, we can highlight the importance and contributions that network virtualization currently represents. NFV not only provides scalability and cost reduction in hardware resources, but also brings benefits in the management and control of traffic.

6.2 Future work

According to the work carried out for developing this undergraduate project, we expose some ideas to continue it. These ideas are outlined below.

- UDP traffic classification. We propose to make a comparative analysis of these algorithms with UDP traffic. This review offers a different characterization of the dataset and some different labels of apps.
- Real-time traffic capture. This proposal would be aimed at attracting traffic in a data center, where the amount of data and applications would be greater. In this way, the training of the algorithms would be more robust and the results more accurate.

Bibliography

[1] W. Li, M. Canini, A. W. Moore, and R. Bolla, "Efficient application identification and the temporal and spatial stability of classification schema," *Computer Networks*, vol. 53, pp. 790 – 809, 2009.

[2] K. Gray and T. D. Nadeau, *Network Function Virtualization*. Morgan Kaufmann, 2016.

[3] M. D. Firoozjaei, J. P. Jeong, H. Ko, and H. Kim, "Security challenges with network functions virtualization," *Future Generation Computer Systems*, vol. 67, pp. 315–324, 2017.

[4] Ixia. (2016) network function virtualization (nfv): 5 major risks.

[5] U. Shankara, "Communication between virtual machines," Mar. 16 2007.

[6] D. Cotroneo *et al.*, "Network function virtualization: Challenges and directions for reliability assurance," in *IEEE ISSREW*, Naples, Italy, 2014, pp. 37–42.

[7] R. Weingärtner, G. B. Bräscher, and C. B. Westphall, "A distributed autonomic management framework for cloud computing orchestration," in *IEEE SERVICES*, San Francisco, CA, USA, 2016, pp. 9–17.

[8] B. Solomon, D. Ionescu, M. Litoiu, and G. Iszlai, "Designing autonomic management systems for cloud computing," in *ICCC-CONTI*, Timisoara, Romania, 2010, pp. 631–636.

[9] H. Mearns and J. Leaney, "The use of autonomic management in multi-provider telecommunication services," in *IEEE ECBS*, Phoenix, AZ, USA, 2013, pp. 129–138.

[10] J. O. Kephart and D. M. Chess, "The vision of autonomic computing," *Computer*, vol. 36, pp. 41–50, 2003.

[11] N. Agoulmine, *Autonomic Network Management Principles: From Concepts to Applications*. Elsevier Science, 2010.

[12] K. Tsagkaris *et al.*, "Customizable autonomic network management: integrating autonomic network management and software-defined networking," *IEEE Vehicular Technology Magazine*, vol. 10, no. 1, pp. 61–68, 2015.

[13] V. Carela-Español *et al.*, "An autonomic traffic classification system for network operation and management," *Journal of Network and Systems Management*, vol. 23, no. 3, pp. 401–419, 2015.

[14] A. Valdes, R. Macwan, and M. Backes, "Anomaly detection in electrical substation circuits via unsupervised machine learning," in *IRI*. IEEE, 2016, pp. 500–505.

[15] I. Maglogiannis, *Emerging Artificial Intelligence Applications in Computer Engineering: Real Word AI Systems with Applications in EHealth, HCI, Information Retrieval and Pervasive Technologies*. IOS Press, 2007.

[16] I. H. Witten, E. Frank, M. A. Hall, and C. J. Pal, *Data Mining: Practical machine learning tools and techniques*. Morgan Kaufmann, 2016.

[17] N. Al Khater and R. E. Overill, "Network traffic classification techniques and challenges," in *IEEE ICDIM*. Jeju Islands, South Korea: IEEE, 2015, pp. 43–48.

[18] S. Zander and G. Armitage, "Practical machine learning based multimedia traffic classification for distributed qos management," in *IEEE LCN*, Bonn, Germany, 2011, pp. 399–406.

[19] D. Qin, J. Yang, J. Wang, and B. Zhang, "Ip traffic classification based on machine learning," in *IEEE ICCT*, Jinan, China, 2011, pp. 882–886.

[20] T. P. J. M. Bujlow, Tomasz; Riaz, "A method for classification of network traffic based on c5.0 machine learning algorithm," in *IEEE ICNC*, Maui, Hawaii, USA, 2012, pp. 237–241.

[21] K. Singh, S. Agrawal, and B. Sohi, "A near real-time ip traffic classification using machine learning," *International Journal of Intelligent Systems and Applications*, vol. 5, no. 3, p. 83, 2013.

[22] S. Choudhury and A. Bhowal, "Comparative analysis of machine learning algorithms along with classifiers for network intrusion detection," in *ICSTM*, Tamil Nadu, India, 2015, pp. 89–95.

[23] M. Shafiq *et al.*, "Wechat text and picture messages service flow traffic classification using machine learning technique," in *IEEE HPCC/SmartCity/DSS*, Sydney, NSW, Australia, 2016, pp. 58–62.

[24] M. Shafiq, X. Yu, A. A. Laghari, L. Yao, N. K. Karn, and F. Abdessamia, "Network traffic classification techniques and comparative analysis using machine learning algorithms," in *Computer and Communications (ICCC), 2016 2nd IEEE International Conference on*. IEEE, 2016, pp. 2451–2455.

[25] L. He, C. Xu, and Y. Luo, "vtc: Machine learning based traffic classification as a virtual network function," in *ACM*, ser. SDN-NFV Security, New Orleans, Louisiana, USA, 2016, pp. 53–56.

[26] W. Ma, C. Medina, and D. Pan, "Traffic-aware placement of nfv middleboxes," in *IEEE GLOBECOM*, San Diego, CA, USA, 2015, pp. 1–6.

[27] P. W. Chi, Y. C. Huang, and C. L. Lei, "Efficient nfv deployment in data center networks," in *IEEE ICC*, London, UK, 2015, pp. 5290–5295.

[28] C. Hoyos, "Un modelo para la investigación documental. guia teórico práctica sobre la construcción del estado del arte," *Medellin*, 2000.

[29] C. E. Serrano, "Modelo para la construcción de soluciones," *Universidad del Cauca*, 2002.

[30] O. Chapelle, P. Haffner, and V. N. Vapnik, "Support vector machines for histogram-based image classification," *IEEE Transactions on Neural Networks*, vol. 10, no. 5, pp. 1055–1064, Sep 1999.

[31] S. B. S. B. D. O. F. M. D. W. S. J. Jennings, B.; van der Meer, "Towards autonomic management of communications networks," *IEEE Communications Magazine*, vol. 45, 2007.

[32] P. Neves *et al.*, "The selfnet approach for autonomic management in an nfv/sdn networking paradigm." SAGE PublicationsSage UK: London, England, 2016, pp. 1–17.

[33] A. Dhraief *et al.*, "Self-healing and optimizing of the hip-based m2m overlay network," in *IEEE ICAC*, San Jose, CA, USA, 2013, pp. 183–192.

[34] A. Ramakrishnan *et al.*, "Learning deployment trade-offs for self-optimization of internet of things applications," in *IEEE ICAC*. San Jose, CA, USA: ACM, 2013, pp. 213–224.

[35] A. Helmy, B. Jennings, L. Murphy, and T. Pfeifer, *Autonomic Management of Mobile Multimedia Services: 9th IFIP/IEEE International Conference on Management of Multimedia and Mobile Networks and Services, MMNS 2006, Dublin, Ireland, October 25-27, 2006, Proceedings*. Springer, 2006, vol. 4267.

[36] I. El Naqa and M. J. Murphy, "What is machine learning?" in *Machine Learning in Radiation Oncology*. Springer, 2015, pp. 3–11.

[37] I. Maglogiannis, *Emerging Artificial Intelligence Applications in Computer Engineering: Real Word AI Systems with Applications in EHealth, HCI, Information Retrieval and Pervasive Technologies*. IOS Press, 2007.

[38] T. Kalaiselvi and P. Shanmugaraja, "Internet traffic classification using supervised learning algorithms–a survey," vol. 03, 2016.

[39] S. Huang, K. Chen, C. Liu, A. Liang, and H. Guan, "A statistical-feature-based approach to internet traffic classification using machine learning," in *2009 International Conference on Ultra Modern Telecommunications Workshops*, Oct 2009, pp. 1–6.

[40] R. Alshammari and A. N. Zincir-Heywood, "Machine learning based encrypted traffic classification: Identifying ssh and skype," in *2009 IEEE Symposium on Computational Intelligence for Security and Defense Applications*, July 2009, pp. 1–8.

[41] S. Valenti, D. Rossi, A. Dainotti, A. Pescapè, A. Finamore, and M. Mellia, "Reviewing traffic classification," in *Data Traffic Monitoring and Analysis*. Springer, 2013, pp. 123–147.

[42] Y. Wang and S. Z. Yu, "Machine learned real-time traffic classifiers," in *Intelligent Information Technology Application, 2008. IITA '08. Second International Symposium on*, vol. 3, Dec 2008, pp. 449–454.

[43] J. Elias, F. Martignon, S. Paris, and J. Wang, "Efficient orchestration mechanisms for congestion mitigation in nfv: Models and algorithms," *IEEE Transactions on Services Computing*, vol. PP, no. 99, pp. 1–1, 2015.

[44] B. Han, V. Gopalakrishnan, L. Ji, and S. Lee, "Network function virtualization: Challenges and opportunities for innovations," *Communications Magazine, IEEE*, vol. 53, no. 2, pp. 90–97, 2015.

[45] R. Mijumbi, J. Serrat, J. L. Gorricho, N. Bouten, F. D. Turck, and R. Boutaba, "Network function virtualization: State-of-the-art and research challenges," *IEEE Communications Surveys Tutorials*, vol. 18, no. 1, pp. 236–262, 2016.

[46] R. Mijumbi, J. Serrat, and J.-L. Gorricho, "Self-managed resources in network virtualisation environments," in *IM*, 2015.

[47] H. Hawilo, A. Shami, M. Mirahmadi, and R. Asal, "Nfv: state of the art, challenges, and implementation in next generation mobile networks (vepc)," *Network, IEEE*, vol. 28, no. 6, pp. 18–26, 2014.

[48] A. Al-Quzweeni, A. Lawey, T. El-Gorashi, and J. M. H. Elmirghani, "A framework for energy efficient nfv in 5g networks," in *2016 ICTON*, July 2016, pp. 1–4.

[49] V. communities. (2014) understand how virtual machine traffic routes. [Online]. Available: https://communities.vmware.com/docs/DOC-25426

[50] H. R. Chishti, "A traffic classification method using machine learning algorithm," 2013.

[51] (2017) ryu sdn framework. [Online]. Available: http://osrg.github.io/ryu/

[52] Ovs - openv vswitch. [Online]. Available: http://openvswitch.org/

[53] M. Team. (2017) Mininet: An instant virtual network on your laptop (or other pc) - mininet. [Online]. Available: http://mininet.org

[54] M. M. Leach, P. and R. Salz. (2005, July) A universally unique identifier (uuid) urn namespace. [Online]. Available: http://www.rfc-editor.org/info/rfc4122

[55] A. Botta, A. Dainotti, and A. Pescapè, "A tool for the generation of realistic network workload for emerging networking scenarios," *Computer Networks*, pp. 3531 – 3547, 2012.

[56] E. Frank *et al.*, *Weka-A Machine Learning Workbench for Data Mining.* Boston, MA, USA: Springer US, 2010, pp. 1269–1277.

[57] M. V. Sadagopan N S. (2016, June) Virtualized evolved eacket core for lte networks. [Online]. Available: https://github.com/networkedsystemsIITB/NFV_LTE_EPC

[58] vmware. [Online]. Available: https://www.vmware.com/

[59] iperf - the ultimate speed test tool for tcp, udp and sctp. [Online]. Available: https://iperf.fr/

[60] J. Zhu, *Quantitative models for performance evaluation and benchmarking: data envelopment analysis with spreadsheets.* Springer, 2014, vol. 213.