Sistemas & Telemática
ISSN: 1692-5238
EditorSyT@icesi.edu.co
Universidad ICESI
Colombia

Vergara Reyes, Juliana Alejandra; Martínez Ordoñez, María Camila; Caicedo Rendón, Oscar Mauricio

A benchmarking of the efficiency of supervised ML algorithms in the NFV traffic classification

Sistemas & Telemática, vol. 15, núm. 42, 2017, pp. 47-67
Universidad ICESI
Cali, Colombia

Available in: http://www.redalyc.org/articulo.oa?id=411553241002

Original research / Artículo original / Pesquisa original - Tipo 1

A benchmarking of the efficiency of supervised ML algorithms in the NFV traffic classification

Juliana Alejandra Vergara Reyes / julianavergara@unicauca.edu.co / http://orcid.org/0000-0003-1147-4415

María Camila Martínez Ordoñez / macamartinez@unicauca.edu.co / http://orcid.org/0000-0002-1522-9241

Oscar Mauricio Caicedo Rendón, Ph.D / omcaicedo@unicauca.edu.co / http://orcid.org/0000-0003-2223-947X

Universidad del Cauca, Facultad de Ingeniería Electrónica y Telecomunicaciones, Departamento de Telemática, Grupo de Ingeniería Telemática (GIT), Popayán, Colombia

ABSTRACT The implementation of NFV allows improving the flexibility, efficiency, and manageability of networks by leveraging virtualization and cloud computing technologies to deploy computer networks. The implementation of autonomic management and supervised algorithms from Machine Learning [ML] become a key strategy to manage this hidden traffic. In this work, we focus on analyzing the traffic features of NFV-based networks while performing a benchmarking of the behavior of supervised ML algorithms, namely J48, Naïve Bayes, and Bayes Net, in the IP traffic classification regarding their efficiency; considering that such an efficiency is related to the trade-off between time-response and precision. We used two test scenarios (an NFV-based SDN and an NFV-based LTE EPC). The benchmarking results reveal that the Naïve Bayes and Bayes Net algorithms achieve the best performance in traffic classification. In particular, their performance corroborates a good trade-off between precision and time-response, with precision values higher than 80 % and 96 %, respectively, in times less than 1,5 sec.

KEYWORDS IP traffic; NFV; machine learning; supervised algorithms.

Comparación de la eficiencia de algoritmos de ML supervisados en la clasificación de tráfico NFV

RESUMEN La implementación de NFV permite mejorar la flexibilidad, eficiencia y gestión de redes al emplear tecnologías de virtualización y computación en la nube para desplegar nuevas redes de computadores. La implementación de procesos de gestión autónomos, junto con algoritmos de aprendizaje supervisado en la rama del conocimiento denominada aprendizaje de máquina (ML, Machine Learning) se ha convertido en una estrategia clave para gestionar tráfico en segundo plano. En este documento se presenta un proyecto de investigación que analiza características de tráfico de redes basadas en NFV al realizar una comparativa de la eficiencia (benchmarking) del comportamiento de algoritmos de aprendizaje supervisado para ML. Se analizaron los algoritmos J48, Naïve Bayes y Bayes Net y se analizó la clasificación de tráfico IP respecto a su eficiencia, la que está relacionada con la compensación entre el tiempo de respuesta y la precisión del algoritmo. Se emplearon dos escenarios de prueba (una SDN basada en NFV y un EPC LTE basado en NFV). Los resultados del benchmarking revelan que los algoritmos Naïve Bayes y Bayes Net obtuvieron mejor desempeño en la clasificación del tráfico. En particular, estos valores corroboran una adecuada compensación entre precisión y tiempo de respuesta, con valores de precisión mayores a 80% y 96%, respectivamente, en tiempos menores a 1.5 segundos.

PALABRAS CLAVE Algoritmos supervisados; aprendizaje de máquina; NFV; tráfico IP.

Benchmarking da eficiência dos algoritmos supervisionados de ML na classificação de tráfego NFV

RESUMO A implementação de NFV permite melhorar a flexibilidade, a eficiência e a capacidade de gerenciamento das redes aproveitando a virtualização e as tecnologias da computação em nuvem para implantar redes informáticas. A implementação de gerenciamento autônomo e algoritmos supervisionados de Aprendizado de Máquinas (Machine Learning - ML) tornam-se uma estratégia chave para gerenciar esse tráfego oculto. Neste trabalho, nosso foco é a análise das características do tráfego em redes baseadas em NFV, ao mesmo tempo em que realizamos uma avaliação comparativa do comportamento dos algoritmos supervisionados de ML, isto é, J48, Naïve Bayes e Bayes Net na classificação de tráfego IP em relação à sua eficiência; considerando que essa eficiência está relacionada ao equilíbrio entre o tempo de resposta e precisão. Foram utilizados dois cenários de teste (um SDN baseado em NFV e um LTE EPC baseado em NFV). Os resultados da avaliação comparativa revelam que os algoritmos Naïve Bayes e Bayes Net têm o melhor desempenho na classificação do tráfego. Em particular, seu desempenho corrobora um bom equilíbrio entre a precisão e o tempo de resposta, com valores de precisão superiores a 80% e 96%, respectivamente, para tempos inferiores a 1,5 segundos.

PALAVRAS-CHAVE Tráfego IP; NFV; Aprendizado de máquinas; algoritmos supervisionados.

Received / Recepción: Agosto 31, 2017 - Accepted / Aceptación: Septiembre 19, 2017 *doi:10.18046/syt.v15i42.2539*

I. Introduction

The Network Functions Virtualization [NFV] establishes how network services are implemented using virtualized software components and how they decouple from the hardware on which they execute (Gray, 2016). Organizations need to implement NFV to meet user requirements, increasing workloads, and the complexity of agile development. However, the implementation of NFV raises several challenges, such as the inclusion of new software components (e.g., hypervisor and management/orchestration elements) and hidden traffic control (Firoozjaei, Jeong, Ko, & Kim, 2017). The hidden traffic of NFV is commonly referred to as east-west traffic, since it never goes through a physical network interface (Shankara, 2007). In this sense, we highlight that if there is no hidden traffic management, this could lead to difficulties in diagnosing or improving network performance (Cotroneo et al., 2014). In NFV-based networks, much of the traffic is communicated only between virtual machines [VMs] located on the same physical host (Ixia, 2016).

The constant growth of networks, the management of hidden traffic in NFV and the need for scalable architectures are challenges faced by system administrators and operators (Weingärtner, Bräscher, & Westphall, 2016). The implementation of autonomic management within the systems helps to address these challenges, automating and distributing processes within the network. This type of management also allows to improve security, prevention, detection, control and error handling and gives way to the evolution of network applications (Solomon, Ionescu, Litoiu, & Iszlai, 2010; Mearns & Leaney, 2013; Kephart & Chess, 2003; Agoulmine, 2010; Tsagkaris et al., 2015; Carela-Español, Barlet-Ros, Mula-Valls, & Sole-Pareta, 2015). Within the concept of automatic management, Machine Learning [ML] plays a significant role in solving problems such as accurate classification of traffic in traditional networks (Shafiq et al., 2016a), data mining (Witten, Frank, Hall, & Pal, 2016), detection of network anomalies (Valdes, Macwan, & Backes, 2016), and in medical processes for diagnosis (Maglogiannis, 2007).

Traffic classification has been performed in traditional networks by using ML algorithms. The works by Shafiq et al., (2016a; 2016b); Singh, Agrawal, and Sohi (2013); Zander and Armitage (2011); Bujlow, Riaz, and Pedersen (2012); Qin et al., (2011); and Choudhury and Bhowal (2015) reveal that the supervised ML algorithms have an adequate performance in traffic classification tasks. However, none of such works has focused on classifying traffic in NFV-based networks. This type of classification differs from the classi-

I. Introducción

La virtualización de funciones de red [NFV, *Network Functions Virtualization*] establece la manera en que los servicios de red son implementados utilizando componentes software virtualizados, además de cómo se desacoplan del hardware en donde se ejecutan (Gray, 2016). Las organizaciones requieren implementar las NFV para cumplir con los requerimientos del usuario, incrementando la carga de trabajo y la complejidad del desarrollo ágil. Sin embargo, la implementación de NFV acarrea diversos retos, como la inclusión de nuevos componentes software (hipervisor y elementos de gestión), además del control de tráfico oculto (Firoozjaei, Jeong, Ko, & Kim, 2017). El tráfico oculto de las NFV hace referencia a tráfico "este-oeste", puesto que nunca circula por interfaces físicas de red (Shankara, 2007). Por consiguiente, es preciso insistir en que, en ausencia de gestión de este tráfico, pueden surgir dificultades en el diagnóstico o mejora del rendimiento de la red (Cotroneo et al., 2014). En redes basadas en NFV, la mayoría del tráfico es entre máquinas virtuales [VM, *Virtual Machines*] ubicadas en el mismo host físico (Ixia, 2016).

El constante crecimiento de las redes, la gestión del tráfico oculto en las NFV y la necesidad de arquitecturas escalables son retos que enfrentan los administradores y operadores de sistemas actualmente (Weingärtner, Bräscher, & Westphall, 2016). La implementación de mecanismos autónomos de gestión en dichos sistemas ayuda a abordar los anteriores retos al automatizar y distribuir procesos en la red. Este tipo de gestión también mejora la seguridad, prevención, detección, control y gestión de errores, y traza un camino para la evolución de las diversas aplicaciones en red (Solomon, Ionescu, Litoiu, & Iszlai, 2010; Mearns & Leaney, 2013; Kephart & Chess, 2003; Agoulmine, 2010; Tsagkaris et al., 2015; Carela-Español, Barlet-Ros, Mula-Valls, & Sole-Pareta, 2015). Dentro del concepto de gestión automática, el aprendizaje de máquina [ML, *Machine Learning*] juega un rol significativo en la resolución de problemas como una clasificación adecuada del tráfico en redes tradicionales (Shafiq et al., 2016a), minería de datos (Witten, Frank, Hall, & Pal, 2016), detección de anomalías de red (Valdes, Macwan, & Backes, 2016) y procesos médicos para realizar diversos diagnósticos (Maglogiannis, 2007).

La clasificación de tráfico se ha realizado en redes tradicionales utilizando algoritmos de ML. Los trabajos realizados por Shafiq et al., (2016a; 2016b); Singh, Agrawal y Sohi (2013); Zander y Armitage (2011); Bujlow, Riaz, y Pedersen (2012); Qin et al., (2011); y Choudhury and Bhowal (2015) indican que los algoritmos ML de aprendizaje supervisado presentan un desempeño adecuado en tareas de clasificación de tráfico. No obstante, ninguno de los anteriores trabajos se ha enfocado en la clasificación de tráfico en redes basadas en NFV. Este tipo de clasificación difiere de la clasificación de redes tradicionales porque en las redes basadas en NFV, en ciertas ocasiones, el tráfico fluye únicamente por enlaces virtuales. En la revisión de la literatura realizada,

se encontró que pocos trabajos consideran el análisis de la clasificación de tráfico en redes basadas en NFV (He, Xu & Luo, 2016; Ma, Medina, & Pan, 2015; Chi, Huang, & Lei, 2015). Estos trabajos se enfocan en la clasificación del tráfico al utilizar algoritmos de ML como funciones virtuales de red al analizar el tráfico en el despliegue de NFV en centros de datos; además de evaluar el tráfico NFV en *middleboxes*. A diferencia de los trabajos anteriores, este documento presenta un benchmarking para analizar el comportamiento de diversos algoritmos ML de aprendizaje supervisado en la clasificación de tráfico IP en redes tradicionales basadas en NFV.

En particular, este documento presenta un análisis de las características de tráfico en una red NFV, la adaptación de un dataset con dichas características y otras tradicionales, y el entrenamiento de algoritmos ML de aprendizaje supervisado (J48, Naïve Bayes y Bayes Net) para determinar su eficiencia respecto de la respuesta en el tiempo y la clasificación en la precisión. Se emplearon dos escenarios: SDN basado en NFV y EPC LTE basado en NFV. Los resultados revelan que los algoritmos Naïve Bayes y J48 son los de mejor desempeño respecto de la clasificación del tráfico en este tipo de redes. Puntualmente, el desempeño de estos algoritmos demuestra una adecuada compensación entre precisión y respuesta en el tiempo, con valores de precisión mayores a 80% y 96% para cada algoritmo respectivamente. Lo anterior en tiempos menores a 1.5 segundos.

El resto de este documento se estructura de la siguiente manera: en la sección 2 se presentan los trabajos relacionados; en la Sección 3 se introduce la adaptación de dataset y la metodología de *benchmarking* y se presenta una descripción de los algoritmos ML de aprendizaje supervisado empleados; en la sección 4 se presenta la evaluación y análisis de los dos casos de estudio ejecutados para el benchmarking, y en la sección 5, las conclusiones y el trabajo futuro que se puede derivar de lo que se presenta en este documento.

II. Trabajos relacionados

Algunos autores han llevado a cabo investigaciones para clasificar el tráfico en redes tradicionales utilizando algoritmos ML de aprendizaje supervisado, pero no existe mucha información acerca del comportamiento de dichos algoritmos en redes basadas en NFV. Shafiq et al., (2016a; 2016b); Singh et al., (2013); Zander y Armitage (2011); Bujlow et al., (2012); Qin et al., (2011); and Choudhury y Bhowal (2015) realizan clasificaciones del tráfico IP en redes tradicionales utilizando algoritmos de ML (Bayes Net, árboles de decisión (*Decision Trees*), *Random Forest y Naïve Bayes*) con el fin de identificar el más preciso.

Como se mencionó, pocos trabajos han realizado clasificaciones en redes basadas en NFV (He et al., 2016; Ma et al., 2015; Chi et al., 2015). Dichos autores proponen un diseño de funciones virtuales de red [vTC] para seleccionar y aplicar clasificadores ML en tiempo de ejecución y analizan la efectividad de diversos clasificadores ML como *K-Nearest*

fication of traditional networks because, in NFV-based networks, sometimes the traffic only flows by virtual links. To the best our knowledge, few works perform traffic classification in NFV-based networks (He, Xu & Luo, 2016; Ma, Medina, & Pan, 2015; Chi, Huang, & Lei, 2015). 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 traditional and NFV-based networks.

In particular, in this paper, we analyze the NFV network traffic features, adapt a dataset with these features and some traditional ones, and train supervised ML algorithms (i.e., J48, Naïve Bayes, and Bayes Net) for determining their efficiency regarding time-response and classification precision. We used two scenarios, namely an NFV-based SDN and an NFV-based LTE EPC. The results reveal that the Naïve Bayes and J48 algorithms have the best performance in traffic classification in this type of networks. In particular, the performance of these algorithms demonstrates a good trade-off between precision and time-response, with precision values higher than 80 % and 96 %, respectively, in times less than 1,5 sec.

The remainder of this paper is organized as follows. Section 2 presents the related work. Section 3 introduces the Dataset Adaptation, the benchmarking methodology and the description of supervised ML algorithms used in this work. Section 4 presents the Evaluation and Analysis of the two cases of study performed for the benchmarking. Section 5 concludes this paper and presents implications for future work.

II. Related work

Some authors have carried out works for classifying traffic in traditional networks by using supervised ML algorithms, but there is a bit of information about the behavior of these algorithms in NFV-based networks. Shafiq et al., (2016a; 2016b); Singh et al., (2013); Zander and Armitage (2011); Bujlow et al., (2012); Qin et al., (2011); and Choudhury and Bhowal (2015) perform classifications of IP traffic in traditional networks using ML algorithms (e.g., Bayes Net, Decision Tree, Random Forest, and Naïve Bayes) to identify the most precise.

As mentioned, few works perform traffic classification in NFV-based (He et al., 2016; Ma et al., 2015; Chi et al.,

Table 1. Related work / Trabajos relacionados

Pa Author	Pa Focus	Network		Type of traffic	Techniques
		Traditional	NFV		
Zander & Armitage (2011)	Classification and control	✔		IP	Supervised algorithms
Qin et al. (2011)	Classification	✔		IP	Supervised algorithms
Bujlow et al. (2012)	Classification	✔		SSH, Skype	Supervised algorithms
Singh et al. (2013)	Classification	✔		IP	Supervised algorithms
Choudhury & Bhowal (2015)	Comparative analysis	✔		HTTP	Supervised algorithms
Shafiq et al. (2016 b)	Classification	✔		Text, Picture	Supervised algorithms
Shafiq et al. (2016a)	Classification	✔		IP	Supervised algorithms
He et al. (2016)	Classification		✔	NFV	Supervised algorithms
Ma et al. (2015)	Deployment		✔	NFV	Heuristic algorithms
Chi et al. (2015)	Allocation		✔	NFV	Authors algorithms
Our work	Benchmarking classification		✔	IP	Supervised algorithms

2015). They propose a design of virtual network functions (vTC) to select and apply ML classifiers on run time. This work analyses the effectiveness of several ML classifiers, such as K-Nearest Neighbors, Support Vector Machine, Decision Tree, Adaptive Boosting, Naïve Bayes and Multi-Layer Perception, which have been used 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 contribute by investigating the efficiency of traffic classification on NFV-based networks.

Ma et al., (2015) proposed an algorithm that processes and filters the traffic among middleboxes. They 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 comprise this traffic and less in analyzing the hidden one.

Chi et al., (2015) concentrate their work in deploying NFV in data center networks proposing a heuristic algorithm for allocating Virtual Network Functions [VNF]. 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 scaling data centers in a more agile way in comparison with other algorithms.

Table 1 presents a summary of the related work. Such a works, we carry out a benchmarking to analyze the behavior

Neighbors, Support Vector Machines, Decision Trees, Adaptive Boosting, Naïve Bayes y Multi-Layer Perception, algoritmos típicamente empleados en problemas de clasificación. Los resultados de evaluar dichos clasificadores revelan que su efectividad depende, en gran medida, de los protocolos analizados y de las características acumuladas de los datos en la red. El vTC propuesto para la clasificación de tráfico puede mejorar la precisión hasta un 13%. Aunque los autores no analizan las principales características de tráfico NFV oculto, sí contribuyen al investigar la eficiencia en la clasificación de tráfico en redes basadas en NFV.

Ma et al., (2015) proponen un algoritmo que procesa y filtra el tráfico entre *middleboxes*. Estudian el despliegue eficiente de middleboxes NFV en redes definidas por software [SDN, *Software-Defined Networks*] para reducir el tráfico. Aunque los autores analizan los efectos del tráfico cambiante con un algoritmo, no se enfocan en las características que componen dicho tráfico ni analizan el tráfico oculto.

Chi et al., (2015) concentran su trabajo en desplegar NFVs en un datacenter de una red determinada. proponiendo un algoritmo heurístico para ubicar las funciones virtuales de red [VNF, *Virtual Network Functions*]. Su algoritmo fue entrenado para utilizar datos recolectados de un datacenter simulado que soporta múltiples VNF desplegadas en VM. El algoritmo propuesto es una solución para reducir el crecimiento del tráfico este-oeste y reducir la inutilización de recursos de computación en el *datacenter*. Además, su propuesta permite escalar datacenters de manera ágil, comparado con otros algoritmos.

La **Tabla 1** presenta un resumen del trabajo relacionado. Como dichos trabajos, en este documento llevamos a cabo un *benchmarking* o comparativa para analizar el comportamiento de algoritmos ML de aprendizaje supervisado en la clasificación de tráfico IP en una NFV. Por consiguiente, como primera medida se empleó un dataset tradicional y se adaptó a condiciones puntuales presentes en las NFV

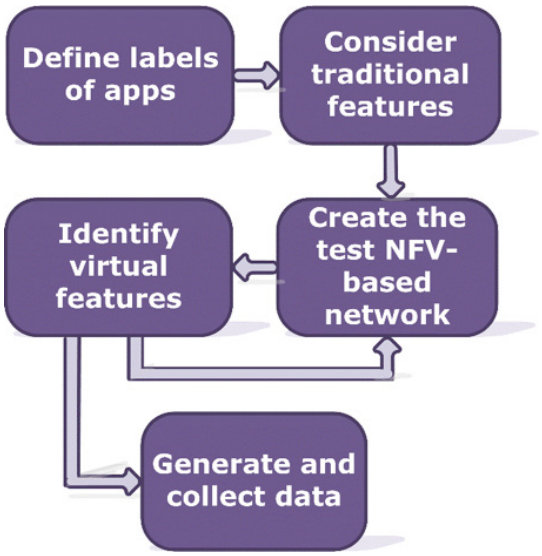

Figure 1. Process for dataset creation / Proceso para la creación del dataset

para clasificar el tráfico; después, se realizó una comparación entre tres algoritmos de aprendizaje supervisado (J48, Naïve Bayes y Bayes Net), comúnmente utilizados en redes tradicionales para clasificación de tráfico; finalmente, los algoritmos fueron probados y se analizó su eficiencia respecto de la respuesta en el tiempo y precisión.

III. Construcción del dataset y proceso de benchmarking

En esta sección se describe el proceso de construcción del dataset para la clasificación de tráfico en redes NFV, además del proceso de *benchmarking* para el entrenamiento, validación y pruebas de los algoritmos ML de aprendizaje supervisado analizados.

A. Construcción del dataset

Con el fin de clasificar el tráfico presente en redes basadas en NFV, es necesario conocer el comportamiento de las características intrínsecas a este tipo de tráfico. Por ende, se estructuró un dataset que refleja el desempeño del tráfico NFV incluyendo las aplicaciones o tipos de servicio presentes en la capa de aplicación. La **Figura 1** presenta el proceso que se siguió para construir dicho dataset.

Para realizar este proceso, se consideró como base la metodología descrita en Chishti (2013). En particular, el proceso propuesto se conforma de cinco pasos, como se detalla a continuación.

Paso 1. Definir las etiquetas de las aplicaciones

Consiste en categorizar un conjunto de aplicaciones (en la capa de aplicación) del protocolo TCP (capa de transporte) empleados para clasificar el tráfico (Li, Canini, Moore, & Bolla, 2009). Algunas aplicaciones del protocolo TCP pueden ser WEB (aplicaciones web), correo (servicios de correo electrónico, protocolos como IMAP, POP, SMTP), multimedia (servicios multimedia, iTunes) y VOIP (Skype).

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., J48, Naïve Bayes, and Bayes Net) that have been used on traditional networks was compared for traffic classification. Third, the algorithms were tested, and their efficiency regarding time-response and precision was analyzed.

III. Dataset construction and benchmarking process

In this section, we describe the process of constructing the dataset for NFV traffic classification, and the benchmarking process for the training, validation, and testing of supervised ML classification algorithms.

A. Dataset Construction

To classify the traffic in NFV-based networks, it is necessary to know the behavior of the intrinsic characteristics of this type of traffic. Therefore, we structure a dataset that reflects the performance of NFV traffic including the applications or types of services involved in the application layer. **Figure 1** presents the process that we follow to build up this dataset.

To develop this process, we take as a basis the methodology developed in (Chishti, 2013). In particular, our process is formed by five steps as follows.

Step 1:Defines labels of apps

It is to categorize a set of applications (application layer) of the TCP protocol (transport layer) used to classify the traffic (Li, Canini, Moore, & Bolla, 2009). Some applications on the TCP protocol can be WEB (i.e., web applications), MAIL (i.e., email services, protocols such as IMAP, POP, SMTP), MULTIMEDIA (e.g., multimedia services, iTunes), VOIP (e.g., Skype).

Step 2: Consider traditional features

In this step, it is to take as a basis a traditional dataset. We recommend the use of the dataset proposed by Chishti (2013) because it has been successfully used in previous investigations. Such a dataset has features that provide information about the TCP traffic (see **Table 2**) and allow classifying it in normal and anomalous, in traditional computer networks.

Step 3: Create the test NFV-based Network

It is to simulate, emulate or create a real NFV-based network for generating and collecting data for their subsequent analysis.

Table 2. Structural traditional features / Características tradicionales estructurales

Feature / *Característica*	Description / *Descripción*	Example / *Ejemplo*
frame.len	Frame length on the wire / *Longitud de la trama en el cable*	60 bytes-(480bits)
ip.src	Source address / *Dirección fuente*	192.168.0.24
ip.dst	Destination address / *Dirección de destino*	192.168.0.27
ip.hdr_len	Header length / *Longitud del encabezado*	5 – (20 bytes)
tcp.srcport	Source port / *Puerto fuente*	43266
tcp.dstport	Destination port / *Puerto de destino*	22
tcp.seq	Sequence number / *Número de secuencia*	37
tcp.hdr_len	Header length / *Longitud del encabezado*	32
tcp.window_size	Calculated window size / *Tamaño de ventana calculado*	1247
ip.proto	Protocol / *Protocolo*	6 – TCP
		17 -UDP

Step 4: Identify virtual features

It is to determine the particular features that characterize the traffic, including the hidden one, in NFV-based networks. Here, to identify the virtual features, it is needed to capture the traffic, which circulates between VMs in where are running the VNFs. A useful tool to perform this step is Wireshark (Chishti, 2013) that can be located at different points of the NFV-based network.

Step 5: Generate and collect data

It is to build up and fill the structure of datasets created, using traffic-generated tools. In this step, some TCP traffic generation tools were used, such as The Distributed Internet Traffic Generator [D-ITG] (Botta, Dainotti, & Pescapé, 2012) and Iperf3 (iPerf, 2017). With these tools and similar ones, it is posible to characterize generated traffic.

B. Benchmarking Process

This is a process used for evaluating services or tasks within a system; it allows having a comparison point for generating changes in the performance of such system (Zhu, 2014). In our evaluation, the benchmarking allows comparing the efficiency of supervised ML algorithms in NFV-based networks. Considering that 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) (Li et al., 2009).

We perform the benchmarking by using the methodology called Training, Validation and Testing (Chapelle, Haffner, & Vapnik, 1999). This methodology proposes a division of the overall dataset into two parts. The first one for training and validation (2/3 of the data) and the second part for testing (1/3 of the data), see **Figure 2**.

Paso 2. Considerar características tradicionales

Este paso hace referencia a tomar como base un *dataset* tradicional. Se recomienda el uso del *dataset* propuesto por Chishti (2013), puesto que ha sido usado satisfactoriamente en investigaciones previas. Dicho dataset presenta características que proveen información acerca del tráfico TCP (ver **Tabla 2**) y permite clasificarlo como normal y anómalo en redes de computación tradicionales.

Paso 3. Crear la red de prueba basada en NFV

Esto es, similar, emular o crear una red real basada en NFV para generar y recolectar datos para su posterior análisis.

Paso 4. Identificar características virtuales

Hace referencia a determinar las características particulares que presenta el tráfico —incluso el oculto— en redes basadas en NFV. Se requiere capturar dicho tráfico, el cual circula entre VMs donde se ejecutan las VNF para identificar sus características virtuales. Una herramienta útil para realizar este paso es Wireshark (Chishti, 2013), la cual puede ser ubicada en diferentes puntos de la red NFV.

Paso 5. Generar y recolectar datos

Esto se hace al construir y llenar la estructura de los datasets creados utilizando herramientas generadoras de tráfico. En este paso se utilizaron algunas herramientas de generación de tráfico TCP, como el generador de tráfico de Internet distribuido [D-ITG, Distributed Internet Traffic Generator] (Botta, Dainotti, & Pescapé, 2012) y el denominado Iperf3 (iPerf, 2017). Con estas herramientas y otras similares es posible caracterizar el tráfico generado.

B. Proceso de benchmarking

El *benchmarking* se utiliza para evaluar servicios o tareas en un sistema y permite tener un punto de comparación para generar cambios en el rendimiento de dicho sistema (Zhu, 2014). En la evaluación presentada en este documento, el *benchmarking* permite comparar la eficiencia de algoritmos de ML de aprendizaje supervisado en redes basadas en NFV

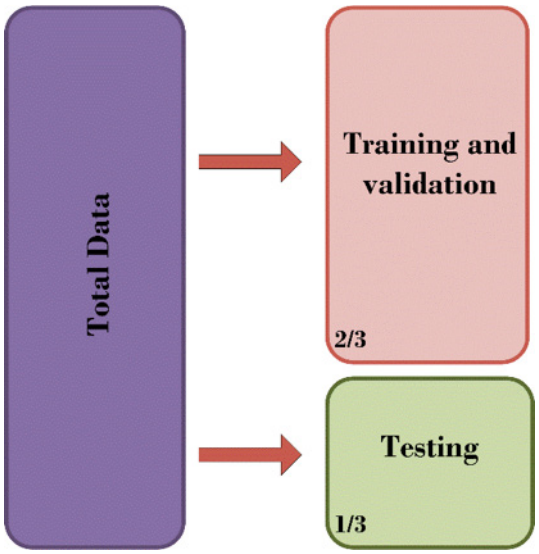

Figure 2. Division of data / División de los datos

al considerar que la eficiencia se relaciona con la compensación entre la respuesta en el tiempo –el tiempo empleado por los algoritmos para clasificar el tráfico– y la precisión –el número de miembros de clase clasificados correctamente en relación al número total de instancias clasificadas para una clase dada– (Li et al., 2009).

Se ejecutó el *benchmarking* utilizando la metodología denominada entrenamiento, validación y pruebas (Chapelle, Haffner, & Vapnik, 1999), la cual propone una división del dataset en dos partes: la primera para entrenamiento y validación (2/3 de los datos); y la segunda para pruebas (1/3 de los datos). En la **Figura 2** se presenta esta división con mayor detalle.

La división de los datos permite a los algoritmos crear un modelo previo de clasificación con los subconjuntos de entrenamiento y validación. Por ende, este modelo es empleado por algoritmos para realizar la clasificación final de los datos empleando el subconjunto de pruebas. En la **Figura 3** se presenta el proceso de *benchmarking*, conformado por cuatro pasos que se describen a continuación.

Paso 1. Selección de características

Esta selección puede realizarse utilizando Weka u otra herramienta similar bajo el siguiente orden: i) se abre cada dataset; ii) se selecciona la opción *GreedyStepwise* como método de búsqueda y *CfsSubsetEval* como evaluador; iii) se ejecutan los elementos seleccionados para determinar las características; y iv) se seleccionan las características con mayor correlación.

Paso 2. Creación del dataset

En este paso se requiere crear un dataset con la información necesaria para caracterizar el tráfico en redes basadas en NFV. Este *dataset* se denominó *combinado* y su correspondiente característica de selección puede ser ejecutada con los siguientes pasos: i) las características son clasificadas de

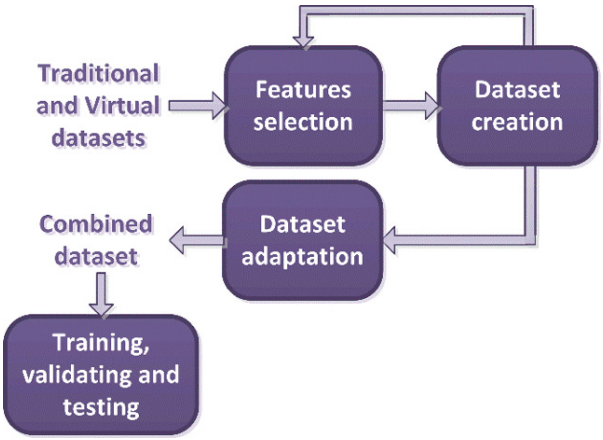

Figure 3. Benchmarking process / Proceso de benchmarking

The data division allows the algorithms to create a previous classification model with the training and validation subsets. Then, this model is used by algorithms to perform the final data classification with the testing subset. **Figure 3** presents the benchmarking process formed by four steps as follows.

Step 1: Features selection

This selection can be performed by Weka or a similar tool as follows: i) each dataset is opened; ii) GreedyStepwise is selected as the search method and CfsSubsetEval as the evaluator; iii) the selection is executed to determine the features; and iv) the most correlated features are selected.

Step 2: Dataset creation

In this step, it is needed to create a dataset with the necessary information to characterize the traffic in NFV-based networks. This dataset is called Combined, and its corresponding feature selection may be carried out by following the next steps: i) the features are classified according to an individual evaluation, using the Ranker search method; and ii) to determine the relation of gain information per-feature and the types of application, the evaluator InfoGainAttributeEval is used.

Step 3: Dataset adaptation

In this adaptation, initially, it is needed to apply some filters: i) normalize aims to transform values to a given values range; ii) discretize is to transform features with continuous values to features with a finite range of values; and iii) class balancer allows adjusting the weights of the data labels keeping the same weight. Finally, two data subsets are extracted by re-sampling: the first one for training and the second data subset for testing.

Step 4: Training, validation and testing

In this step, the data subsets extracted from the previous process are used for training and testing the supervised classification algorithms and to determine their efficiency. The supervised algorithms J48, Naïve Bayes, and Bayes Net are trained and evaluated since they have been successfully 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 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.

IV. Classification algorithms

The different techniques of classifying network traffic are divided into two classes: deterministic –hard– and probabilistic –soft– (Li et al., 2009). The deterministic classification assigns data points to one of the several mutually exclusive traffic-classes. On the other hand, a probabilistic classification method classifies the data by assigning a per-class membership probability, which can be based on an artificial allocation of probabilities or an a priori experience.

In this paper, we focus on determining the efficiency of three probabilistic methods, such as the J48, Naïve Bayes and Bayes Net algorithms. To determine the efficiency, we consider the trade-off between the percentage of precision and the time-response of each algorithm.

A. J48

J48 is a well-known decision tree algorithm where the classification will be definitive (to assign each data point one of the mutual-exclusive classes) (Li et al., 2009; Zander & Armitage, 2011; Choudhury & Bhowal, 2015). The input to J48 consists of a collection of training cases, each one having a tuple of values for a fixed set of features $F=F_1,F_2\ldots,F_k$ and a class. A feature F_a can be described as continuous or discrete according to whether its values are numeric or nominal. The class C is discrete and has values $C_1,C_2\ldots,C_x$.

The goal is to learn from the training cases a function:

$$DOM(A_1)\times DOM(A_2)\times \ldots DOM(A_k) \rightarrow DOM(C) \qquad (1)$$

That maps from the feature values to a predicted class.

As such the decision tree is a recursive structure where a leaf node is labeled with a class value, and a test node that has two or more outcomes, each linked to a sub tree. To classify

acuerdo a una evaluación individual utilizando el método de búsqueda Ranker; y ii) se emplea el evaluador *InfoGainAttributeEval* para determinar la relación de ganancia de información por característica y por tipo de aplicación.

Paso 3. Adaptación del dataset

Esta adaptación requiere inicialmente la aplicación de algunos filtros: i) normalizar los objetivos para transformar los valores en un rango dado y fijo; ii) discretizarlos para transformar las características con valores continuos a características con un rango finito de valores; y iii) balancear las clases es importante para ajustar los pesos de las etiquetas de datos manteniendo el mismo peso. Finalmente, dos subconjuntos de datos se extraen por re-muestreo: el primero para entrenamiento y el segundo para pruebas.

Paso 4. Entrenamiento, validación y pruebas

En este paso, los subconjuntos obtenidos en el paso anterior son utilizados para entrenar y probar los algoritmos de clasificación de aprendizaje supervisado y determinar su eficiencia. Los algoritmos J48, Naïve Bayes y Bayes Net son entrenados y evaluados, puesto que han sido utilizados satisfactoriamente para clasificar tráfico en redes tradicionales. Inicialmente, en la etapa de entrenamiento, el subconjunto de datos de entrenamiento se utiliza para preparar (esto es, generar un modelo de clasificación previo) cada algoritmo. A continuación, en la etapa de validación, los algoritmos mejoran su modelo de clasificación utilizando validación cruzada (cross-validation). Finalmente, en la etapa de pruebas, el subconjunto de pruebas se emplea para obtener el modelo final de clasificación de cada algoritmo mencionado.

IV. Algoritmos de clasificación

Las diferentes técnicas de clasificación de tráfico de red se dividen en dos clases: determinísticas —duras— y probabilísticas —suaves— (Li et al., 2009). La clasificación determinística asigna puntos de datos a una de las muchas clases de tráfico mutuamente excluyentes, mientras que el método probabilístico clasifica los datos asignando una membresía de probabilidad por clase, la cual se basa en la ubicación artificial de probabilidades o en una experiencia a priori.

En este documento, el trabajo realizado se enfocó en determinar la eficiencia de tres métodos probabilísticos, como son los algoritmos J48, Naïve Bayes y Bayes Net. Con el fin de determinar la eficiencia, se consideró la compensación entre el porcentaje de precisión y la respuesta en el tiempo de cada algoritmo.

A. J48

J48 es un conocido algoritmo de árboles de decisión en donde la clasificación será definitiva, al asignar cada punto de datos a clases mutuamente excluyentes (Li et al., 2009; Zander & Armitage, 2011; Choudhury & Bhowal, 2015). La entrada del J48 consiste en una colección de casos de entrenamiento, cada uno con una tupla de valores por cada conjunto fijo de características $F=F_1,F_2\ldots,F_k$ y una clase. Una característica F_a puede describirse como continua o discreta

dependiendo de si sus valores son numéricos o nominales. La clase C es discreta y presenta valores $C_1, C_2, \ldots, C_x$.

El objetivo es aprender de los casos de entrenamiento por función:

$$DOM(A_1) \times DOM(A_2) \times \ldots DOM(A_k) \rightarrow DOM(C) \qquad (1)$$

Esto mapea desde los valores de cada característica a una clase predecible.

Dado que los árboles de decisión son una estructura recursiva, donde cada rama es etiquetada con un valor de una clase con cada nodo de pruebas que tenga dos o más salidas, cada una enlazada con un sub-árbol, para clasificar un objeto utilizando J48, éste objeto se encuentra inicialmente en la parte más alta del árbol. De ahí, dicho objeto irá iterativamente a un sub-árbol hasta que alcance un nodo en una rama:

- si está en un nodo hoja, la etiqueta asociada a dicha hoja se convierte en la clase predicha;
- si el objeto se encuentra en un nodo de pruebas, se moverá al tope del sub-árbol cuando la salida de la prueba sea determinada.

B. Naïve Bayes

Naïve Bayes está basado en el teorema elemental de Bayes. Este algoritmo puede alcanzar valores relativamente buenos en desempeño cuando se emplea en tareas de clasificación (Novakovic, 2016; Qin et al., 2011). El clasificador Naïve Bayes simplifica ampliamente el aprendizaje al asumir que las características son independientes dada una variable de clase. Más formalmente, este clasificador se define por funciones discriminantes (2).

$$f_i(X) = \prod_{j=1}^{N} P(x_j|c_i)P(c_i) \qquad (2)$$

Donde $X=(X_1, X_2, \ldots, X_N)$ denota un vector de características y $C_j, j=1,2,\ldots,N$ indica las posibles etiquetas de la clase.

La fase de entrenamiento de un clasificador para aprendizaje consiste en estimar probabilidades condicionales $P(x_j|c_i)$ y anteriores $P(c_i)$. Aquí, los valores $P(c_i)$ son estimados al contar los ejemplos de entrenamiento que caen en la clase c_i y posteriormente dividir dicho resultado por el tamaño del set de entrenamiento. Similarmente, las probabilidades condicionales son estimadas simplemente observando la distribución de frecuencia de la característica X_j en el subconjunto de entrenamiento etiquetado como clase C_i.

C. Bayes Net

Una red Bayesiana (Muralidharan & Sugumaran, 2012; Choudhury & Bhowal, 2015; Singh et al., 2013) consiste de un conjunto de variables $V=\{A\_1, A\_2, \ldots, A\_N\}$ y un conjunto de aristas dirigidas E, las cuales forman un grafo acíclico dirigido [DAG, *Directed Acyclic Graph*] $(DAG)G=(V,E)$, donde una distribución conjunta de variables se representa

an object using J48, the object to be classified is initially at the top (root) of the tree. The object will go iteratively into a sub tree until it reaches a leaf node:

- If it is at a leaf node, the label associated with that leaf becomes the predicted class;
- If it is at a test node, when the outcome of the test is determined, it is moved to the top of the subtree for that outcome.

B. Naïve Bayes

Naïve Bayes is based on the elementary Bayes Theorem. It can achieve relatively good performance on classification tasks (Novakovic, 2016; Qin et al., 2011). Naïve Bayes classifier greatly simplifies learning by assuming that features are independent given a class variable. More formally, this classifier is defined by discriminant functions:

$$f_i(X) = \prod_{j=1}^{N} P(x_j|c_i)P(c_i) \qquad (2)$$

Where $X=(X_1, X_2, \ldots, X_N)$ denotes a feature vector and C_j $j=1,2,\ldots,N$, denote possible class labels.

The training phase of a classifier for learning consists of estimating conditional probabilities $P(x_j|c_i)$ and prior probabilities $P(c_i)$. Here, $P(c_i)$ are estimated by counting the training examples that fall into class ci and then dividing the resulting count by the size of the training set. Similarly, conditional probabilities are estimated by simply observing the frequency distribution of feature X_j within the training subset that is labeled as class C_i.

C. Bayes Net

A Bayesian network (Muralidharan & Sugumaran, 2012; Choudhury & Bhowal, 2015; Singh et al., 2013) consists of a set of variables, $V=\{A_1, A_2, \ldots, A_N\}$ and a set of directed edge, E, between variables, which form a directed acyclic graph $(DAG)G=(V,E)$ where a joint distribution of variables is represented by the product of conditional distributions of each variable given its parents. Each node, $Ai \in V$ represents a random variable and a directed edge from A_i *to* A_j, $(A_i, A_j) \in V$ represents the conditional dependency between A_i and A_j. In a Bayesian networks, each variable is independent on its non-descendants, given a value of its parents in G. This independence encoded in G reduces the number of parameters which is required to characterize a joint distribution, so that posterior distribution can be efficiently inferred. In a Bayesian network over $V=\{A_1, A_2, \ldots, A_N\}$, the joint distribution P(V)

is the product of all conditional distributions specified in the Bayesian network such as:

$$P(A_1, A_2 \ldots, A_N) = \prod_{i=1}^{N} P\,(A_i|Pa_i) \qquad (3)$$

Where $P(A_i \mid Pa_i)$ is the conditional distribution of A_i, given Pa_i which denotes the parent set of A_i. A conditional distribution for each variable has a parametric form that can be learned by the maximum likelihood estimation.

V. Benchmarking

This section presents the two case studies performed in our benchmarking. The first one was developed by using a n NFV-based Software-Defined Network [SDN]. The second case study was carried out by using an NFV-based LTE network. In these networks, the supervised ML algorithms (i.e., J48, Naïve Bayes, and Bayes Net) were evaluated as follows.

- The two NFV-based networks are configured and deployed.
- Different tools for generating and collecting data were used for each particular NFV-based network.
- An evaluation of the efficiency of supervised algorithms J48, Naïve Bayes and Bayes Net in the classification of the traffic was performed.

It is to note that in the two case studies, the Dataset Construction and Benchmarking Process were used to generate an independent analysis.

A. Case 1: Traffic classification on NFV-based SDN

Initially, in the dataset construction process:

- We defined the labels INTERACTIVE (i.e., SSH and Telnet), SERVICES (i.e., X11 and SSL), OPENFLOW (i.e., a protocol that allows a software server to determine the path of packet forwarding on a switched network.), WEB (i.e., HTTP) and OTHER. These labels compose a feature called Class.
- We consider the features described in **Table 2** to construct the structure of the dataset called Traditional.
- We propose the test NFV-based SDN of **Figure 4** that is composed by four VMs, two running a Ryu controller (RYU SDN framework, 2017) (i.e., a controller to increase network agility, facilitating the traffic management) and the other ones running Open vSwitch (OVS - Openv vSwitch) (OVS, which is used to forward traffic between VMs on the same physical host, and between VMs and the physical network) emulated

por el producto de las distribuciones condicionales de cada variable dados sus padres. Cada nodo $A_i \in V$ representa una variable aleatoria y una arista dirigida desde A_i hacia A_j. $(A_i,A_j) \in V$ representa la dependencia condicional entre A_i y A_j. En las redes Bayesianas, cada variable es independiente de sus no-descendientes dado un valor de sus padres en G. Esta codificación independiente en G reduce el número de parámetros requeridos para caracterizar una distribución conjunta, por lo que posteriores distribuciones pueden ser inferidas eficientemente. En una red Bayesiana sobre $V=\{A_1,A_2,\ldots,A_N\}$, la distribución conjunta $P(V)$ es el producto de todas las distribuciones condicionales especificadas en la red Bayesiana:

$$P(A_1, A_2 \ldots, A_N) = \prod_{i=1}^{N} P\,(A_i|Pa_i) \qquad (3)$$

Donde $P(A_i \mid Pa_i)$ es la distribución condicional de A_i dado Pa_i, el cual denota el conjunto padre de A_i. Una distribución condicional para cada variable presenta una forma paramétrica que puede aprender por la máxima estimación de probabilidad.

V. Benchmarking

Esta sección presenta los dos casos de estudio evaluados durante la comparación descrita en este documento. El primero fue desarrollado utilizando una SDN basada en NFV y el segundo en una red LTE basada en NFV. En dichas redes, los algoritmos de ML J48, Naïve Bayes y Bayes Net de aprendizaje supervisado fueron evaluados de la siguiente manera:

- las dos redes basadas en NFV fueron configuradas y desplegadas;
- se emplearon diferentes herramientas para generar y recolectar datos en cada red basada en NFV; y
- se ejecutó la evaluación de la eficiencia de los algoritmos de aprendizaje supervisado J48, Naïve Bayes y Bayes Net en la clasificación del tráfico de la red.

Es importante destacar que en los dos casos de estudio el proceso de construcción y benchmarking del dataset fue utilizado para generar un análisis independiente.

A. Caso 1: Clasificación de tráfico en la SDN basada en NFV

El proceso de construcción del dataset incluyó los puntos descritos a continuación.

- Se definieron las etiquetas INTERACTIVO (SSH y Telnet), SERVICIOS (X11 y SSL), OPENFLOW (un protocolo que permite a un servidor software determinar la ruta del enrutamiento de paquetes en una red conmutada), WEB (HTTP) y OTROS. Estas etiquetas componen una característica llamada Clase.
- Se consideraron las características descritas en la **Tabla 2** para construir la estructura del dataset, la cual se denominó Tradicional.
- Se propone la SDN basada en NFV de la **Figura 4**, la cual está compuesta de cuatro VMs, dos ejecutando

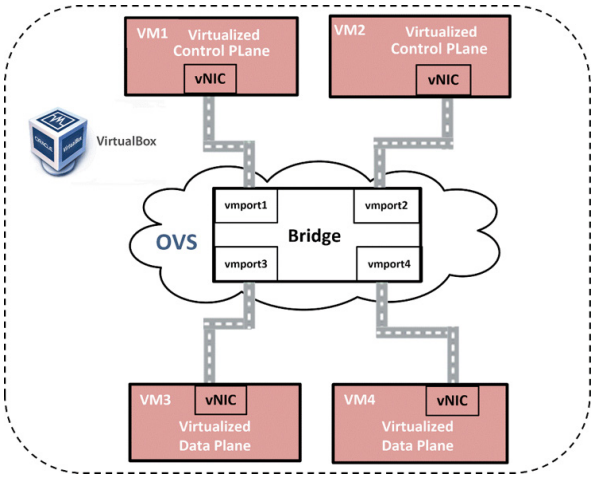

Figure 4. NFV-based Test SDN / SDN de prueba basada en NFV

un controlador Ryu (RYU SDN framework, 2017), un controlador para incrementar la agilidad de la red, facilitando la gestión del tráfico; mientras que las otras dos VMs ejecutan Open vSwitch (OVS - Openv vSwitch), el cual es utilizado para reenviar tráfico entre VMs en el mismo host físico y entre VMs en la red física. Se emuló esta red en Mininet, el cual es una red virtual instantánea disponible en una computadora ("Mininet: An instant…", 2017). En particular en el escenario mencionado, el controlador Ryu fue ejecutado en dos VMs (VM_1 and VM_3), mientras que una OVS fue configurada para permitir la comunicación entre las VM. En dicho OVS, se creó un *bridge* (unión virtual) con cuatro puertos virtuales ($vmport_1$, $vmport_2$, $vmport_3$ y $vmport_4$) para conectar las VM.

Con el fin de identificar características virtuales, el tráfico fue analizado en cada VM, obteniendo características similares en redes tradicionales. Por ende, se decidió analizar el tráfico en el bridge, observando que éste presenta variaciones cuando atraviesa puertos virtuales. En dichos puertos, el tráfico presenta cambios en las direcciones de origen y destino: el tráfico antes del puerto virtual contiene las direcciones de las VM. Sin embargo, cuando el tráfico pasa por un puerto y alcanza el bridge, se modifican las direcciones correspondiendo a las de dicho puerto. Por consiguiente, los indicadores de protocolo en la trama (denominados frame. protocol) y tipo de Ethernet (eth.type) fueron seleccionados para caracterizar el tráfico en el bridge. Finalmente, las cuatro características mencionadas (direcciones Ethernet de origen y destino, protocolos de trama y tipo de Ethernet) se seleccionaron como la base para construir la estructura del dataset virtual (ver **Tabla 3**).

En este caso, para la generación de tráfico se empleó la herramienta denominada D-ITG (Botta et al., 2012). En el proceso de recolección de datos se emplearon dos redes (una por máquina, VM_2 *y* VM_4) dirigidas por el controlador Ryu desplegado en VM_1 y dos puntos de referencia: punto A: $vmport_2$ y *bridge*; y punto B: $vmport_4$ y *bridge* (ver **Figura 4**).

in Mininet ("Mininet: An instant…", 2017). In particular, in the above mentioned environment, the Ryu controller was executed on two VMs (VM_1 and VM_3), an OVS was configured to enable communication among these VMs, and in this OVS, we create a bridge with four virtual ports (i.e. $vmport_1$, $vmport_2$, $vmport_3$, and $vmport_4$), to connect the VMs.

- 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 3**).
- In this case, to generate traffic, we used the D-ITG tool (Botta et al., 2012). In the data collection process, we used two networks (one per machine, VM_2 and VM_4) handled by a Ryu Controller deployed in VM_1, and two points of reference: Point A ($vmport_2$ and *bridge*) and Point B ($vmport_4$ and *bridge*), see **Figure 4**.

The collection was performed as follows: i) we put sniffers on the Point A and Point B, ii) at Point A, we captured 10 random periods of 30 minutes, and in the Point B we captured 1 period of 30 minutes; and iii) we collected the data in a .pcapng file using Wireshark, which was converted into .csv file by T-shark commands (Chishti, 2013) for managing in Weka (Frank, 2010).

Table 4 depicts the precision results when J48, Naïve Bayes, and Bayes Net are used for classifying traffic in the test NFV-based SDN (**Figure 4**). From these results, we noticed, 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. Second, J48 and Naïve Bayes are more precise than Bayes Net; this behavior depends on the model that creates each algorithm. On the one hand, J48 analyzes and selects as nodes the features with the highest gain of information which makes it more pre-

Table 3. Structural virtual features / Funciones virtuales estructurales

Network	Features	Description	Example
NFV-based SDN	frame.protocols	Protocols in frame	Eth:ethertype:arp
	eth.dst	Destination	68:a0:f6:b0:14:84
	eth.sr	Source	08:00:27:f0:dc:db
	eth.type	Type	0X0800-Ipv4
		43266	0x86DD-Ipv6

Table 4. Precision comparison / Comparación de precisión

Point	Dataset	J48	Naïve Bayes	Bayes Net
A	Traditional	92,535 ± 0,185	80,802 ± 0,034	91,118 ± 0,034
	Virtual	99,998 ± 0,001	99,998 ± 0,001	99,998 ± 0,001
	Combined	99,998 ± 0,001	98,673 ± 0,006	99,225 ± 0,002
B	Traditional	88,854 ± 0,018	79,5 ± 0,375	90,234 ± 0,116
	Virtual	99,998 ± 0,001	99,915 ± 0,012	99,953 ± 0,350
	Combined	99,998 ± 0,001	90,174 ± 0,380	99,890 ± 0,005

Table 5. Time-response comparison / Comparación de tiempos de respuesta

Point	Dataset	J48	Naïve Bayes	Bayes Net
A	Traditional	39,885 ± 0,968	1,54 ± 0,113	11,215 ± 1,718
	Virtual	0,24 ± 0,98	0,254 ± 0,063	0,235 ± 0,091
	Combined	1,565 ± 0,063	1,48 ± 0,834	7,345 ± 1,732
B	Traditional	0,405 ± 0,120	0,12 ± 0,070	0,35 ± 0,14
	Virtual	0,04 ± 0,028	0,03 ± 0,028	0,06 ± 0,56
	Combined	0,145 ± 0,035	0,007 ± 0	0,68 ± 0,21

cise with values between 88,8 % - 99,9 %. While the Naïve Bayes and Bayes Net algorithms consider the independence and dependence between the data, respectively, in this way their models do a longer and more complex data processing than J48. Third, the results obtained in the classification for the *Combined* dataset (90 % - 99 %) increase by up 10 % compared with the results of the Traditional dataset (79 % - 92 %). This increase is directly related to the features that compose each dataset since they provide a different type of information to the algorithms for the creation of classification models.

Table 5 present the time-response results of J48, Naïve Bayes and Bayes Net when used to classify traffic in the Points A and B (Figure 4), respectively. These results disclose that, first, the algorithms in Point A spend more time (from 0,2 sec to 39,8 sec) than in Point B (from 0,007 sec to 0,68 sec), for the classification of datasets. This time variation is related to the difference between the number of flows cap-

La captura de datos se realizó de la siguiente manera: i) se pusieron sniffers (sensores que capturan datos) en los puntos A y B; ii) en A se capturaron diez periodos aleatorios de 30 minutos y en el punto B se capturó un periodo de 30 minutos; y iii) se guardaron los datos en un archivo con extensión .pcapng utilizando Wireshark, el cual fue convertido a tipo .csv vía comandos de T-shark (Chishti, 2013), para gestionar los datos en Weka (Frank, 2010).

La **Tabla 4** describe los resultados de precisión cuando los algoritmos J48, Naïve Bayes y Bayes Net se utilizaron para la clasificación de tráfico en la SDN basada en NFV (Figura 4). De estos resultados puede deducirse que la precisión general es mayor a 79.5%, un efecto similar al alcanzado por los mismos algoritmos en tareas de clasificación de tráfico en redes tradicionales. Además, los algoritmos J48 y Naïve Bayes son más precisos que el Bayes Net. Este comportamiento depende del modelo que crea cada algoritmo; por un lado, el J48 analiza y selecciona como nodos a las características con la mayor ganancia de información, lo cual hace posibles valores de precisión entre el 88.8% y el 99.9%. Mientras que, por otra parte, los algoritmos Naïve Bayes y Bayes Net

consideran la independencia y dependencia entre los datos, respectivamente. Por consiguiente, sus modelos realizan tareas más extensas y complejas cuando procesan datos en comparación al J48.

Los resultados obtenidos en la clasificación para el dataset combinado (90% a 99%) son mayores en un 10% comparados con los del dataset tradicional (79% a 92%). Este incremento está directamente relacionado con las características que constituyen cada dataset, puesto que proveen diferentes tipos de información a los algoritmos para la creación de los modelos de clasificación.

La **Tabla 5** presenta los resultados de respuesta en tiempo de los algoritmos evaluados cuando clasificaron tráfico en los puntos A y B (ver **Figura 4**), respectivamente. Estos resultados revelan que los algoritmos en el punto A requieren más tiempo (de 0.2 s a 39.8 s) que en el punto B (de 0.007 s a 0.68 s) para la clasificación de los datasets. Esta variación en los tiempos se relaciona con la diferencia entre el número de flujos de datos capturados en cada punto: en el punto A existen 1,048,575 flujos, mientras que en el B hay únicamente 52,264. Además, en ambos puntos los tres algoritmos presentaron sus valores más bajos de respuesta en tiempo cuando se empleó el dataset virtual, lo que ocurrió principalmente por la cantidad de características que conforman cada dataset (únicamente cinco).

En ambos puntos el algoritmo Naïve Bayes presentó la menor variación en la respuesta en el tiempo durante la clasificación de cada dataset, esto es principalmente porque dicho algoritmo requiere de una cantidad mínima de datos para estimar la clasificación del tráfico. Por ende, sin importar el número de características, el modelo Naïve Bayes clasifica los datos en tiempos similares.

Para resumir, aunque los algoritmos J48 y Bayes Net alcanzaron una precisión mayor al 88%, conllevan variaciones significativas (0.04 s a 39 s) en su respuesta en el tiempo, puesto que sus modelos de clasificación dependen de la cantidad de datos presente en cada dataset. Individualmente, Naïve Bayes alcanzó una precisión mayor a 79.5%, con un rango de respuesta en el tiempo de 0.007 s hasta 2.5 s. Por consiguiente, la diferencia entre J48 —el algoritmo más preciso— y Naïve Bayes –el más constante en el tiempo– es de aproximadamente 5.5% en precisión, una diferencia que Naïve Bayes compensa sin reducir la precisión al reducir su respuesta en el tiempo en alrededor de seis segundos.

La **Figura 5** presenta los resultados de precisión obtenidos por Naïve Bayes en la clasificación de cada etiqueta de aplicación (INTERACTIVO, SERVICIOS, OPENFLOW, WEB y OTROS) definida inicialmente. Estos resultados revelan que en ambos puntos (A y B), Naïve Bayes presenta sus valores más bajos de precisión (de 49.8% a 76.5%) en la clasificación de algunas etiquetas (SERVICIOS, OPENFLOW Y OTROS). Esta reducción en precisión indica que la distribución de dichas etiquetas en el subconjunto de entrenamiento no es uniforme, y que dichas etiquetas se componen de un menor número de miembros respecto a las otras.

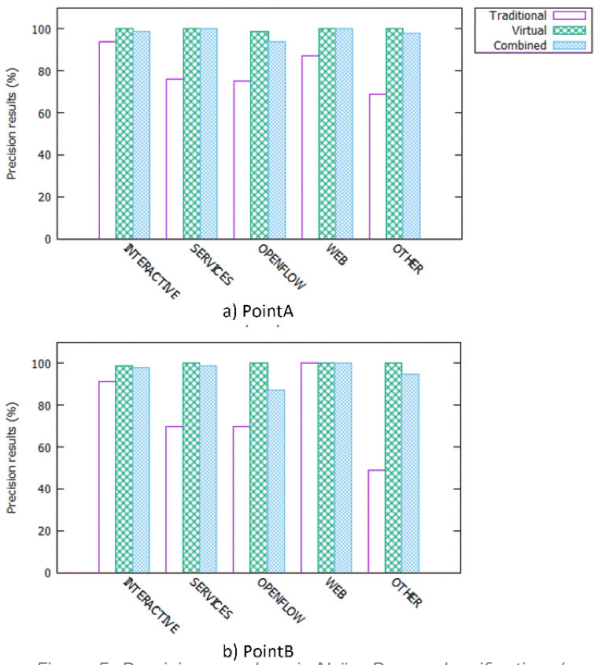

Figure 5. Precision per-class in Naïve Bayes classification / Precisión por clase en la clasificación Naïve Bayes

tured 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 happens because of the number of features forming such a dataset (only five features). Third, in both points, the Naïve Bayes algorithm presents the smallest variation in time-response during the classification of each dataset; and this is because Naïve Bayes needs a minimum amount of data, to estimate the data required for classifying traffic. Therefore, regardless the number of dataset features, the Naïve Bayes model classifies the data in a similar time.

To sum up, although J48 and Bayes Net 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. In turn, Naïve Bayes reaches a precision greater than 79,5 %, with a time-response ranging from 0,007 sec to 2,5 sec. Therefore, the difference between J48, as the most accurate algorithm, and Naïve Bayes, as the most constant in time, is approximately 5,5 % in precision; a difference that Naïve Bayes compensates, without reducing the precision, reducing the time-response around 6 sec.

Figure 5 presents the precision results, achieved by Naïve Bayes, in the classification for each application labels (i.e., INTERACTIVE, SERVICES, OPENFLOW, WEB, and OTHER), initially defined. These results reveal first, in both

Table 6. Recall and precision per-class in Naïve Bayes classification / Recall y precisión por clase in la clasificación Naïve Bayes

Point	Class	Traditional		Virtual		Combined	
		Precision%	Recall%	Precision%	Recall%	Precision%	Recall%
A	Interactive	95,5	89,1	100	100	99,8	99,9
	Services	74	79,5	100	100	100	98,9
	Openflow	76,1	98,1	99,9	100	95	100
	Web	89,1	96,7	100	100	100	100
	Other	65,5	40,3	100	99,9	98,8	94,5
B	Interactive	92,6	97,3	99,6	100	97,6	97,3
	Services	71,3	77,8	100	100	99,9	96,1
	Openflow	69,8	100	100	100	70	100
	Web	100	100	100	100	100	100
	Other	49,5	21	100	99,6	91,6	56,1

points Naïve Bayes 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, as expected, that the classification is linked to the quantity and types of features that compose each dataset.

Another aspect that can be considered to verify the behavior of the algorithms is the recall (also called sensitivity). The Recall identifies the fraction of data retrieved correctly on the amount of data selected as relevant for classification. According to recall values (see **Table 6**), In the classification of the Virtual dataset the Naïve Bayes algorithm accomplishes exceptional values of recalculation for the five labels (values greater than 99,6%) indicating that this algorithm works very well for classifying these services. On the other hand, for the Traditional dataset the algorithm presents low values in the classification of the OTHER label in both points, indicating that the selection of the features continues to play a significant role in the classification.

B. Case 2: Traffic classification on NFV-based LTE EPC

We consider mobile networks as a second scenario for classifying traffic since these networks, nowadays, are faced with an exponential increase in the traffic that passes by them. This increase represents a challenge for telco operators in the deployment of new services and the location of new

Por otra parte, en la clasificación utilizando el *dataset* virtual, los valores de precisión fueron mayores a 99.6% para las cinco clases; con el dataset combinado, dicho valor fue de 94.9% y la clasificación con el dataset tradicional fue mayor al 70%. Estos datos indican de nuevo —como se esperaba— que la clasificación está relacionada con la cantidad y tipo de características que componen cada *dataset*.

Otro aspecto que puede considerarse al verificar el comportamiento de los algoritmos es la exhaustividad (también llamada sensibilidad). Dicho parámetro identifica la fracción de datos correctamente recuperada de la cantidad de datos seleccionada como relevante para la clasificación. De acuerdo con los valores de exhaustividad (ver **Tabla 6**), en la clasificación del *dataset* virtual, el algoritmo Naïve Bayes alcanzó valores excepcionales de recálculo para las cinco etiquetas (valores mayores a 99.6%). Esto indica que este algoritmo funciona muy bien para clasificar dichos servicios. Por otra parte, para el *dataset* tradicional, el algoritmo presentó valores bajos en la clasificación de la etiqueta OTROS en ambos puntos, indicando que la selección de características juega un rol importante en la clasificación.

B. Caso 2: Clasificación de tráfico en un EPC LTE basado en NFV

Para este caso se consideró una red móvil como segundo escenario para la clasificación de tráfico, puesto que di-

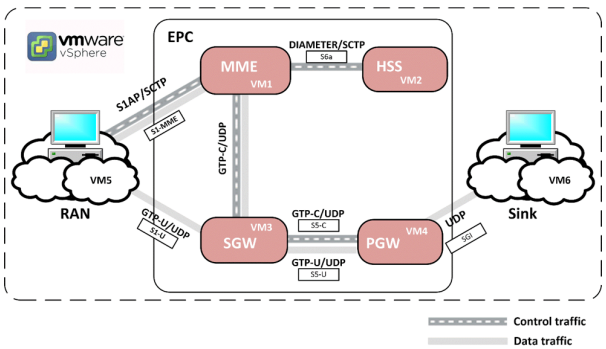

Figure 6. LTE EPC net based in NFV / La red LTE EPC basada en NFV

Table 7. Configuration of NFV-based EPC modules / Configuración de los módulos EPC basados en NFV

Entity	Cores (#)	RAM (GB)	Disk (GB)	OS	Virtualization
RAN	8	4	10	Ubuntu 16.04	
MME, SGW, PGW, HSS	2	2			
Sink	4	2			VMware vSphere

chas redes se enfrentan a un crecimiento exponencial en el tráfico que circula por ellas. Dicho incremento representa un reto para los operadores de telecomunicaciones en el despliegue de nuevos servicios y en la ubicación de nuevos dispositivos hardware en dichas redes. Algunas soluciones propuestas a este reto están enfocadas en la arquitectura del núcleo evolucionado de paquetes [EPC, *Evolved Packet Core*] y en la implementación de NFV en él (Kumar, Satapaphy, Sadagopan, & Vutukuru, 2016). Por consiguiente, es posible considerar a la virtualización de componentes hardware del EPC, brindando al operador móvil la oportunidad de prestar nuevos servicios y aumentar la escalabilidad de la red.

Para simular y desplegar una red móvil se tomó como base el esquema de un sistema EPC LTE basado en NFV propuesto por (Kumar et al., 2016) (ver **Figura 6**). Este sistema se basa en el paradigma cliente-servidor, donde el servidor es capaz de procesar múltiples hilos para servir todas las peticiones del cliente. En él existen seis entidades, incluidas las funciones de red del EPC principal (MME, HSS, SGW, PGW), un simulador de RAN que genera tráfico y un módulo sumidero responsable de recibir el tráfico generado (*uplink*) y generar datos acuse de recibido (*downlink*). En particular, se desplegaron las seis entidades (simulador RAN, MME, HSS, SGW, PGW, módulo sumidero) en distintas VMs corriendo VMware vSphere (VMware, 2017). Las VM desplegadas presentan las características específicas listadas en la **Tabla 7**.

En el proceso de construcción del *dataset* se tuvo en cuenta los puntos descritos a continuación.

- Se identificaron tres etiquetas denominadas MOVIL (protocolos de servicio GSM como el punto de acceso a internet [IAP, *Internet Access Point*], SERVICIOS (SSL para encriptación de tráfico y X11 para servicios de interacción gráficos) y CONTROL (únicamente protocolo TCP/IP).
- Se emplearon las características de las **Tablas 2** y **3** para la construcción de la estructura de los nuevos *datasets*
- Se desplegó el EPC LTE basado en NFV de la **Figura 6** con sus respectivas características (**Tabla 6**).
- Se evaluó la implementación EPC utilizando datos de tráfico y determinando un número específico de UEs por EPC desde el simulador RAN hasta el módulo sumidero. Para la generación del tráfico entre módulos EPC se utilizó la herramienta Iperf3 (iPerf, 2017), la cual permite enviar datos TCP con diferentes anchos de banda y duración. En este caso, el proceso del

hardware devices in the network. Some existing solutions to this challenge focus on the architecture of the underlying Evolved Packet Core [EPC] and the implementation of NFV (Kumar, Satapaphy, Sadagopan, & Vutukuru, 2016). In this sense it is possible to consider the virtualization of the hardware components of the EPC, giving the operator the opportunity to provide new services and to have a scalable network.

For simulating and deploying a mobile network, we take as the basis the scheme of an NFV-based LTE EPC system, proposed by (Kumar et al., 2016) (see **Figure 6**). This system is based on the client-server paradigm, where the server is multi-threaded to serve all client requests. In this system, there are six entities, including the main EPC network functions (i.e., MME, HSS, SGW, PGW), a RAN simulator to generate traffic, and a Sink module responsible for receiving the traffic generated (uplink) and generate acknowledgment data (downlink). In particular, we deploy the six entities (i.e., simulator RAN, MME, HSS, SGW, PGW, Sink module) on separate VMs running on VMware vSphere (VMware, 2017). The deployed VMs have specific characteristics listed in **Table 7**.

In the dataset construction process:

- We identified three labels, namely, MOBILE (i.e., GSM service protocols, such as the Internet Access Point - IAP), SERVICES (i.e., SSL For encrypted information traffic and X11 for graphical interaction services), and CONTROL (i.e., only TCP/IP protocol).
- We used the features of **Tables 2** and **3**, for the construction of the structure of new *Traditional* and *Virtual* datasets, respectively.
- We deploy the NFV-based LTE EPC of **Figure 6** with its respective characteristics (**Table 6**).
- 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 (iPerf, 2017) 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.

For the benchmarking process:

- In the *Feature Selection*, the features with greater predictive capacity were identified. For the Traditional dataset, *frame.len* and *tcp.hdr_len* were obtained, and the *frame.protocols* feature for the Virtual dataset.
- For creating the Combined dataset, we developed the feature selection described in step Dataset Creation of the Benchmarking process. In this case, the new dataset consists of the features *frame.len, tcp.srcport, tcp.dstport, tcp.seq, tcp.hdr_len* and *tcp.window_size* of the Traditional dataset, and has only the *frame.protocols* feature of the Virtual dataset. These seven characteristics present a significant gain relation with the feature Class, for the traffic classification.
- With the three datasets created (i.e., Traditional, Virtual, and Combined), we train the three supervised ML algorithms (i.e., J48, Naïve Bayes, and Bayes Net). The respective results of these processes are presented below.

Table 8 presents the precision results obtained by each algorithm. These results reveal, first, the precision values obtained are higher than 70,7 %, which reflects a similar behavior of the algorithms that in Case 1. Second, J48 and Bayes Net are more precise than the Naïve Bayes, with a difference of approximately 8 %, a difference that is significant to decide what the most efficient algorithm is. 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. Regardless of the type of services identified and the number of data streams collected, the features of that dataset directly affect the precission of the algorithms. In this sense, a variation of approximately 12 % between the classification precision between the Traditional and Combined dataset is identified.

Table 9 presents the results of time-response for classifying each dataset. These results reveal, first, in Case 2 the time-response results have a less difference range (between 0,035 sec to 0,3 sec) than in Case 1 (between 0,007 sec to 39 sec). The obtained range is related to the quantity of data per dataset, and in this case, the three datasets are composed of 24,105 data streams. Second, we identify a behavior similar in time to that of Case 1, the three evaluated algorithms (i.e., J48, Naïve Bayes, and Bayes Net) spend a little more time in

cliente Iperf3 se inicia en el simulador RAN requiriendo una tasa de datos de entrada. La operación del Iperf3 se configuró en modo servidor en el módulo sumidero.

Para el proceso de *benchmarking* se tuvo en cuenta los puntos descritos a continuación.

- En la selección de características se identificaron aquellas con mayor capacidad predictiva. Para el *dataset* tradicional se obtuvieron las denominadas *frame.len* y *tcp.hdr_len*, mientras que en el *dataset* virtual se obtuvo la llamada *frame.protocols*.
- Para la creación del dataset combinado se desarrolló la selección de características descrita en el paso "Creación del dataset para el proceso de *benchmarking*". En este caso, el nuevo dataset consiste de las características *frame.len, tcp.srcport, tcp.dstport, tcp.seq, tcp.hdr_len* y *tcp.window_size* del dataset tradicional, y tiene únicamente la característica frame.protocols del dataset virtual. Estas siete características presentan una relación de ganancia significativa con la característica Clase para la clasificación del tráfico.
- Con los tres *datasets* creados (tradicional, virtual y combinado), se procedió a entrenar los tres algoritmos ML de aprendizaje supervisado (J48, Naïve Bayes y Bayes Net). Los resultados de estos procesos se presentan a continuación.

La **Tabla 8** presenta los resultados de precisión para cada algoritmo. Dichos resultados revelan que los valores de precisión obtenidos son mayores a 70.7%, lo cual refleja un comportamiento similar a los obtenidos en el caso 1. También, J48 y Bayes Net son más precisos que Naïve Bayes, con una diferencia de aproximadamente 8%; este valor es significativo para decidir cuál es el algoritmo más eficiente, dicha diferencia se relaciona con la construcción de los modelos de clasificación para cada algoritmo. Por otra parte, el comportamiento en el caso 1 se repite, puesto que los tres algoritmos presentaron sus valores más bajos de precisión cuando clasificaron el dataset tradicional. Sin importar el tipo de servicio identificado y el número de cadenas de datos recolectadas, las características de dicho *dataset* afectan directamente la precisión de los algoritmos. Por consiguiente, se identificó una variación de aproximadamente 12% entre la precisión al clasificar los *datasets* tradicional y combinado.

La **Tabla 9** presenta los resultados de la respuesta en el tiempo al clasificar cada *dataset*. Dichos resultados revelan que en el caso 2 los resultados de respuesta en el tiempo presentan un menor rango de diferencias (entre 0.035 s a 0.3 s) que en el caso 1 (entre 0.007 s a 39 s). Dicho rango se relaciona con la cantidad de datos por *dataset*; en este caso, los tres *datasets* tienen un total de 24,105 cadenas de datos. Por otra parte, se logró identificar un comportamiento similar en tiempo que en el caso 1: los tres algoritmos evaluados requieren un poco más de tiempo en la clasificación del dataset tradicional, mientras que el *dataset* virtual es clasificado en el menor tiempo, lo que ocurre porque el dataset tradicional conlleva la clasificación de seis características adicionales.

Table 8. Precision comparison – LTE EPC System (point A) / Comparación en precisión – EPC LTE

Dataset	J48	Naïve Bayes	Byes Net
Traditional	96,178 ± 0,020	70,770 ± 2,512	94,913 ± 0,029
Virtual	100 ± 0	100 ± 0	100 ± 0
Combines	100 ± 0	98,802 ± 0,002	99,569 ± 0,021

Table 9. Time-response comparison – LTE EPC System (point A) / Comparación en la respuesta en el tiempo – LTE

Dataset	J48	Naïve Bayes	Byes Net
Traditional	0,300 ± 0,056	0,045 ± 0,007	0,135 ± 0,007
Virtual	0,045 ± 0,007	0,010 ± 0,000	0,050 ± 0,042
Combines	0,085 ± 0,021	0,035 ± 0,007	0,090 ± 0,028

Además, aunque los valores de respuesta en el tiempo son cercanos, Naïve Bayes presentó los valores más bajos con aproximadamente un promedio de 0.04 s para clasificar los tres tipos de *datasets*.

De acuerdo con los resultados mencionados arriba, se identificaron comportamientos entre los algoritmos como que la diferencia en precisión entre J48 y Bayes Net es de aproximadamente 0.5%, lo cual hace más preciso al primero que el Naïve Bayes, este último con una diferencia de casi el 8% respecto a J48. Por otra parte, respecto de los resultados en la respuesta en el tiempo, Naïve Bayes mantiene un balance entre sus resultados con una respuesta promedio de aproximadamente 0.04 s. La diferencia en respuesta en el tiempo indica que J48 y Bayes Net requieren más tiempo para clasificar que Naïve Bayes (0.1 s y 0.05 s, respectivamente). Con estas pequeñas diferencias en la respuesta en el tiempo, se concluye que el algoritmo Bayes Net es más eficiente, puesto que garantiza aproximadamente 8% más de precisión en la clasificación, con un incremento de únicamente 0.05 s en el tiempo de respuesta del algoritmo.

La **Figura 7** presenta los resultados de precisión obtenidos por Bayes Net cuando clasificó las etiquetas MOVIL, CONTROL y SERVICIOS. Con dichos resultados, es posible inferir que este algoritmo proporciona valores de precisión mayores a 91.7%, un valor bastante adecuado para clasificar tráfico IP. Por otra parte, para las tres etiquetas, este algoritmo se comporta de manera similar; en promedio, la clasificación

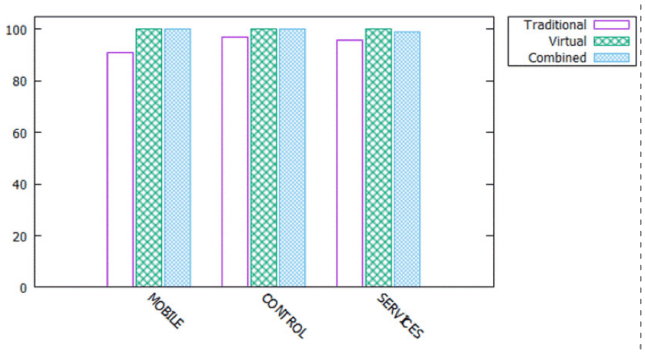

Figure 7. Precision per-class in J48 classification / Precisión por clase en la clasificación realizada por J48

the classification of the Traditional dataset, whereas for the Virtual dataset they present the least time for classifying, it is because of the six more features that compose the Traditional dataset. Third, although the time-response values are very close, Naïve Bayes presents the lowest values, with an approximate average of 0,04 sec for classifying the Traditional, Virtual, and Combined datasets.

According to the results mentioned above, we identified behaviors among the algorithms as, first, the precision difference between J48 and Bayes Net is approximately 0,5%, which makes them more precise than Bayes Naïve with a difference of almost 8%. Second, concerning time-response results, Naïve Bayes maintains a balanced level between its results and an average response time of approximately 0,04 sec. Third, the difference in response time results indicates that J48 and Bayes Net require more time for sorting, about 0,1 sec, and 0,05 sec, respectively than Naïve Bayes. With these small differences in response time, we argue that the Bayes Net algorithm is more efficient, which guarantees approximately 8% more in the precision of the classification, with an increase of 0,05 sec in the time-response of the algorithm.

Figure 7 presents the precision results, achieved by Bayes Net, when classifies the labels: MOBILE, CONTROL and SERVICES. With these results, we note, first, Bayes Net delivers precision values upper to 91,7%, being excellent results for classify IP traffic. Second, for all three tags, this algorithm behaves very similarly, that is, on average, the rating is higher than 97% for each case. Third, it is relevant that in the classification by class using the Virtual dataset the percentages are greater than in the other two datasets. This difference not only occurs in the classification of the three labels but also in the general classification of the datasets. These significant results are accomplished because the amount of data

Table 10. Recall and precision per-class using J48 algorithm (point A) / Exhaustividad y precisión por clase utilizando el algoritmo J48

Class	Traditional		Virtual		Combined	
	Precision%	Recall%	Precision%	Recall%	Precision%	Recall%
MOBILE	91,8	99,6	100	100	100	100
CONTROL	99,2	88,6	100	100	100	100
SERVICES	97,8	99,8	100	100	100	100

in the Virtual dataset and the information they provide for performing the corresponding classification.

To fully describe the J48 performance, Table 10 presents its recall values in the traffic classification in the NFV-based LTE EPC. These results reveal that J48 can accurately classify each label, reaching recall values upper to 88%. For the three labels (MOBILE, CONTROL and SERVICES), J48 behaves more precise than the other two algorithms. In the case of the Control label, the precision value is greater than the recall value indicating that during the classification time, other types of service of this class could have been recognized.

VI. Conclusions and futute work

In this paper, we carried out a benchmarking of supervised ML algorithms (i.e., J48, Naïve Bayes, and Bayes Net) for classify traffic in NFV. For performing it, it was needed to analyze certain characteristics and the behavior of traffic in NFV-based networks, adapt a traditional network dataset to a network virtualized environment, and train and validate the aforementioned algorithms with the adapted data, to determine their behavior.

In the Case 1, NFV-based SDN, we find out that the supervised algorithms J48 and Naïve Bayes achieve a range of precision from 80% to 99%. However, we consider to Naïve Bayes as the most efficient classifier, since it has a precision value range upper to 79,5%, similar to J48 but presenting minimal time-response variations.

In case 2, NFV-based on LTE EPC, we found a difference between precision results, where the J48 and Bayes Net algorithms are approximately 8 % more precise than Naive Bayes, although they have a similar range of response time. However, we selected Bayes Net as the most efficient, because it requires less time for classification than J48.

As future work, we intend to generate and collect traffic data in a data center, where the amount of data and applications would be greater. Also, we are interested in performing a comparative analysis with UDP traffic. ST

fue mayor al 97% para cada caso. Además, es relevante indicar que en la clasificación por clase utilizando el dataset virtual, los porcentajes son mayores que con los otros dos datasets. Esta diferencia no sólo ocurre en la clasificación de las tres etiquetas, sino que también aparece en la clasificación general de los datasets. Estos resultados se complementan entre sí, puesto que la cantidad de datos en el dataset virtual y la información que proveen para realizar la correspondiente clasificación concordaron.

Para describir totalmente el desempeño del algoritmo J48, la **Tabla 10** presenta los valores de exhaustividad en la clasificación de tráfico sobre el EPC LTE basado en NFV. Dichos resultados revelan que J48 puede clasificar correctamente cada etiqueta, obteniendo valores de exhaustividad superiores al 88%. Para las etiquetas MOVIL, CONTROL y SERVICIOS, J48 es más preciso que los otros dos algoritmos evaluados. Para el caso puntual de la etiqueta CONTROL, el valor de precisión es mayor que el valor de exhaustividad, lo que indica que durante el tiempo de clasificación, otros tipos de servicio de esta clase pudieron haber sido reconocidos.

VI. Conclusiones y trabajo futuro

En este documento se llevó a cabo una comparación (benchmarking) de algoritmos de ML con aprendizaje supervisado (J48, Naïve Bayes y Bayes Net) para la clasificación de tráfico en un NFV. Para analizar su desempeño fue necesario examinar ciertas características y el comportamiento del tráfico en redes basadas en NFV, adaptar un dataset tradicional a un ambiente de red virtualizado, y entrenar y validar los algoritmos anteriores con los datos adaptados para determinar su comportamiento.

En el caso 1 —una SDN basada en NFV—, se encontró que los algoritmos J48 y Naïve Bayes lograron un rango de precisión desde 80% hasta 99%. Sin embargo, se consideró a Naïve Bayes como el más eficiente, puesto que presentó un rango de precisión mayor a 79.5%, similar a J48, pero con mínimas variaciones en la respuesta en el tiempo.

En el caso 2 —un EPC LTE basado en NFV— se encontraron diferencias entre los valores de precisión, donde J48 y Bayes Net son aproximadamente 8% más precisos que Naïve Bayes, aunque tienen un rango similar en la respuesta en el tiempo. Sin embargo, se concluyó que Bayes Net fue el clasificador más eficiente porque requiere menos tiempo de clasificación que J48

Como trabajo futuro se planea generar y recolectar datos en un datacenter donde la cantidad de datos y aplicaciones es mucho mayor. También se muestra interés en realizar un análisis comparativo utilizando tráfico UDP. ST

References / Referencias

Agoulmine, N. (2010). *Autonomic network management principles: From concepts to applications.* Amsterdam, The Netherlands: Elsevier.

Botta, A., Dainotti, A., & Pescapé, A. (2012). A tool for the generation of realistic network workload for emerging networking scenarios. *Computer Networks, 56*(15), 3531-3547.

Bujlow, T., Riaz, T., & Pedersen, J. M. (2012, January). A method for classification of network traffic based on C5. 0 Machine Learning Algorithm. In *Computing, Networking and Communications (ICNC), 2012 International Conference on* (pp. 237-241). IEEE.

Carela-Español, V., Barlet-Ros, P., Mula-Valls, O., & Sole-Pareta, J. (2015). An autonomic traffic classification system for network operation and management. *Journal of Network and Systems Management, 23*(3), 401-419.

Chapelle, O., Haffner, P., & Vapnik, V. (1999). Support vector machines for histogram-based image classification. *IEEE transactions on Neural Networks, 10*(5), 1055-1064.

Chi, P. W., Huang, Y. C., & Lei, C. L. (2015, June). Efficient NFV deployment in data center networks. In *Communications (ICC), 2015 IEEE International Conference on* (pp. 5290-5295). IEEE.

Chishti, H. R. (2013). *A traffic classification method using machinelearning algorithm* [thesis]. Luton, UK: University of Bedfordshire.

Choudhury, S., & Bhowal, A. (2015). Comparative analysis of machine learning algorithms along with classifiers for network intrusion detection. In *Smart Technologies and Management for Computing, Communication, Controls, Energy and Materials (ICSTM), 2015 International Conference on* (pp. 89-95). IEEE.

Cotroneo, D., De Simone, L., Iannillo, A. K., Lanzaro, A., Natella, R., Fan, J., & Ping, W. (2014). Network function virtualization: Challenges and directions for reliability assurance. In *Software Reliability Engineering Workshops (ISSREW), 2014 IEEE International Symposium on* (pp. 37-42). IEEE

Firoozjaei, M. D., Jeong, J. P., Ko, H., & Kim, H. (2017). Security challenges with network functions virtualization. *Future Generation Computer Systems, 67,* 315-324.

Frank, E. (2010). Weka-A machine learning workbench for data mining. In *Data mining and knowledge discovery handbook* (pp. 1269-1277). Boston, MA: Springer.

Gray, K. (2016). *Network function virtualization.* Boston, MA: Morgan Kaufmann.

He, L., Xu, C., & Luo, Y. (2016). VTC: Machine learning based traffic classification as a virtual network function. In *Proceedings of the 2016 ACM International Workshop on Security in Software Defined Networks & Network Function Virtualization* (pp. 53-56). New York, NY: ACM.

iPerf. (2017). *iPerf - The ultimate speed test tool for TCP, UDP and SCTP.* Retrieved from https://iperf.fr/

Ixia. (2016). *Network function virtualization (nfv): 5 major risks.* Retrieved from https://www.ixiacom.com/resources/network-function-virtualization-nfv-5-major-risks

Kephart, J. O. & Chess D. M. (2003). The vision of autonomic computing. *Computer, 36*(1), 41-50.

Kumar, J., Satapaphy, P., Sadagopan, N., & Vutukuru, M. (2016). *Virtualized evolved eacket core for LTE networks.* Retrieved from: https://github.com/networkedsystemsIITB/NFV_LTE_EPC

Li, W., Canini, M., Moore, A. W., & Bolla, R. (2009). Efficient application identification and the temporal and spatial stability of classification schema. *Computer Networks, 53*(6), 790-809.

Mearns, H., & Leaney, J. (2013, April). The use of autonomic management in multi-provider telecommunication services. In *Engineering of Computer Based Systems (ECBS), 2013 20th IEEE International Conference and Workshops on the* (pp. 129-138). IEEE.

Ma, W., Medina, C., & Pan, D. (2015, December). Traffic-aware placement of NFV middle boxes. In *Global Communications Conference (GLOBECOM), 2015 IEEE* (pp. 1-6). IEEE.

Maglogiannis, I. (Ed.) (2007). *Emerging artificial intelligence applications in computer engineering: real word ai systems with applications in ehealth, hci, information retrieval and pervasive technologies.* Amsterdam, The Netherlands: IOS.

Mininet: An instant virtual network on your laptop (or other PC). (2017). Retrieved from http://mininet.org

Muralidharan, V., & Sugumaran, V. (2012). A comparative study of Naïve Bayes classifier and Bayes net classifier for fault diagnosis of monoblock centrifugal pump using wavelet analysis. *Applied Soft Computing, 12*(8), 2023-2029.

Novakovic, J. (2016). Toward optimal feature selec-tion using ranking methods and classi-fication algorithms. *Yugoslav Journal of Operations Research, 21*(1), 119-135.

OVS - Openv vSwitch. (n.d.). Retrieved from http://openvswitch.org/

Qin, D., Yang, J., Wang, J., & Zhang, B. (2011, September). IP traffic classification based on machine learning. In *Communication Technology (ICCT), 2011 IEEE 13th International Conference on* (pp. 882-886). IEEE.

RYU SDN framework. (2017). Retrieved from http://osrg.github.io/ryu/

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

Shafiq, M., Yu, X., Laghari, A., Yao, L., Karn, N., Abdesssamia, F., & Salahuddin, S. (2016b). We chat text and picture messages service flow traffic classification using machine learning Technique. In *IEEE HPCC/SmartCity/DSS* (pp. 58-62).

Shankara, U. (2007). *Patent No. 20070220217*. Bengalooru, IN.

Singh, K., Agrawal, S., & Sohi, B. S. (2013). A near real-time IP traffic classification using machine learning. *International Journal of Intelligent Systems and Applications, 5*(3), 83.

Solomon, B., Ionescu, D., Litoiu, M., & Iszlai, G. (2010, May). Designing autonomic management systems for cloud computing. In *Computational Cybernetics and Technical Informatics (ICCC-CONTI), 2010 International Joint Conference on* (pp. 631-636). IEEE.

Sugumaran, V. M. (2012). A comparative study of Naïve Bayes classifier and Bayes net classifier for fault diagnosis of monoblock centrifugal pump using wavelet analysis. *Applied Soft Computing, 12*(8), 2023 - 2029.

Tsagkaris, K., Logothetis, M., Foteinos, V., Poulios, G., Michaloliakos, M., & Demestichas, P. (2015). Customizable autonomic network management: integrating autonomic network management and software-defined networking. *IEEE Vehicular Technology Magazine, 10*(1), 61-68.

Valdes, A., Macwan, R., & Backes, M. (2016). anomaly detection in electrical substation circuits via unsupervised machine learning. In *Information Reuse and Integration (IRI), 2016 IEEE 17th International Conference on* (pp. 500-505). IEEE.

VMware. (2017). Retrieved from https://www.vmware.com/

Weingärtner, R., Bräscher, G. B., & Westphall, C. B. (2016, June). A distributed autonomic management framework for cloud computing orchestration. *In Services (SERVICES), 2016 IEEE World Congress on* (pp. 9-17). IEEE.

Witten, I. H., Frank, E., Hall, M. A., & Pal, C. J. (2016). *Data Mining: Practical machine learning tools and techniques.* Amsterdam, The Netherlands: Elsevier.

Zander, S., & Armitage, G. (2011, October). Practical machine learning based multimedia traffic classification for distributed QoS management. In *Local Computer Networks (LCN), 2011 IEEE 36th Conference on* (pp. 399-406). IEEE.

Zhu, J. (2014). *Quantitative models for performance evaluation and benchmarking: data envelopment analysis with spreadsheets.* New York, NY: Springer.

CURRICULUM VITAE

Juliana Alejandra Vergara Reyes Electronics and Telecommunications Engineer from the Universidad del Cauca (Popayán, Colombia). She has made emphasis in Telecommunications on her bachelor studies. She is an ISOC and IEEE Communications Society member. Her main interest is oriented to NFV, network management, control systems and related works to telecommunications engineering / Ingeniera en Electrónica y Telecomunicaciones de la Universidad del Cauca (Popayán, Colombia) con énfasis en Telecomunicaciones. Es miembro de ISOC e IEEE. Sus principales áreas de interés son NFV, manejo de redes, sistemas de control y telecomunicaciones.

Maria Camila Martinez Ordonez Electronics and Telecommunications Engineer from the Universidad del Cauca (Popayán, Colombia). She has made emphasis in Telecommunications on her bachelor studies. She is an ISOC and IEEE Communications society member. Her main interest is oriented to NFV, network management, optical fiber networks, wireless communications and related works to telecommunications engineering / Ingeniera en Electrónica y Telecomunicaciones de la Universidad del Cauca (Popayán, Colombia) con énfasis en Telecomunicaciones. Es miembro de ISOC e IEEE. Sus principales áreas de interés son NFV, manejo de redes, redes de fibra óptica, comunicación inalámbrica y telecomunicaciones.

Oscar Mauricio Caicedo Rendón Full professor at the Telematics Department at the Universidad del Cauca [Unicauca] (Popayán, Colombia). As researcher, he is part of the Telematics Engineering Group at Unicauca and the Computer Networks Group at Universidade Federal do Rio Grande do Sul [UFRGS] (Porto Alegre, Brasil). He holds a Ph.D. in Computer Science from the Institute of Informatics of UFRGS, a Master in Telematics and a Bachelor in Telecommunications from Unicauca. He has been an IETF Fellowship and a traveler grant from ACM Sigcomm. Furthermore, he has published in prominent journals as Computer Networks and Computer Communications, and in relevant conferences as IEEE Globecom, AINA, COMPSAC, ISCC, and CNSM / Profesor de planta del Departamento de Telemática de la Universidad del Cauca [Unicauca] (Popayán, Colombia). Es miembro del Grupo de Investigación en Ingeniería Telemática (Unicauca) y del Grupo de investigación Redes de Computación de la Universidade Federal do Rio Grande do Sul [UFRGS] (Porto Alegre, Brasil). Es Ingeniero de Telecomunicaciones y Máster en Telemática de la Universidad del Cauca y Doctor en Ciencias de la Computación del Instituto de Informática de la UFRGS. Fue becario de la IETF y ACM Sigcomm. Ha publicado en destacadas revistas, como Computer Networks y Computer Communications, y participado en reconocidas conferencias, tales como: IEEE Globecom, AINA, COMPSAC, ISCC y CNSM.