2015 IEEE 14th International Symposium on Network Computing and Applications

Identification and Selection of Flow Features for Accurate Traffic Classification in SDN

Anderson Santos da Silva, Cristian Cleder Machado,
Rodolfo Vebber Bisol, Lisandro Zambenedetti Granville, Alberto Schaeffer-Filho
Institute of Informatics – Federal University of Rio Grande do Sul – Porto Alegre, Brazil
Email: {assilva, ccmachado, rvbisol, granville, alberto}@inf.ufrgs.br

***Abstract*—Software-Defined Networking (SDN) aims to alleviate the limitations imposed by traditional IP networks by decoupling network tasks performed on each device in particular planes. This approach offers several benefits, such as standard communication protocols, centralized network functions, and specific network elements, for example, controller devices. Despite these benefits, there is still a lack of adequate support for performing tasks related to traffic classification, because (*i*) there are traffic profiles that are very similar, which makes their classification difficult (*e.g.*, both HTTP and DNS flows are characterized by packet bursts); (*ii*) OpenFlow, the key SDN implementation today, only offers native flow features, such as packet and byte count, that do not describe intrinsic traffic profiles; and (*iii*) there is a lack of support to determine what is the optimal set of flow features to characterize different types of traffic profiles. In this paper, we introduce an architecture to collect, extend, and select flow features for traffic classification in OpenFlow-based networks. The main goal of our solution is to offer an extensive set of flow features that can be analyzed and refined and to be capable of finding the optimal subset of features to classify different types of traffic flows. The experimental evaluation of our proposal shows that some features emerge as meaningful, occupying the top positions for the classification of distinct flows in different experimental scenarios.**

***Keywords*—Flow Feature; Feature selection; Traffic Classification; Software-Defined Networking.**

I. Introduction

Traffic classification assists network providers in guaranteeing quality of service (QoS), detecting malicious attacks, reallocating network resources, and performing traffic modeling [1]. In order to achieve more accurate traffic classification, it is important to retrieve precise information about individual traffic flows features, which include, for example, the average packet transmission time. Retrieving relevant network information in traditional IP networks presents several challenges. First, heterogeneous network devices, such as switches and routers, often expose flow information through proprietary management interfaces. Second, to tune how monitoring data is handled internally to devices, one needs to configure each device individually. Finally, because of the distributed control state in forwarding devices, the features of a flow can be changed along its transmission, *e.g.*, ending up with modifications on TCP/IP header fields or facing packet drops, thus causing unanticipated network behavior in subsequent hops.

Software-Defined Networking (SDN) offers a reformulation of the network control logic and alleviate the limitations imposed by traditional IP networks [2]. In SDN, the control plane is decoupled from the forwarding plane, *i.e.*, part of the control logic is moved from the forwarding devices to a logically centralized device often referred to as the network controller. In this approach, every control decision is taken by the network controller, while network devices become simple packet forwarders, programmable through a standardized protocol, such as OpenFlow [3]. However, despite the benefits brought by SDN, there is still an important lack of adequate support for performing tasks related to traffic classification in OpenFlow-based networks. Achieving high accuracy of traffic classification in SDN is difficult for several reasons: (*i*) there are traffic profiles that are very similar, which makes their classification difficult, *e.g.*, both HTTP and DNS flows are characterized by packet bursts; (*ii*) the native flow features available in OpenFlow, such as packet and byte counts, do not convey sufficient information to accurately distinguish between some types of flows; and (*iii*) there is a lack of support to determine what is the optimal set of flow features to characterize different types of traffic profiles.

Considering the aforementioned issues, we introduce in this paper an architecture to identify, extend, and select sets of flow features derived from native OpenFlow counters as well as from mathematical statistics, *e.g.*, mean, variance, maximum and minimum values, 1^{st} and 3^{rd} quartiles, and Fourier Transform. Our main objective is to offer an architecture capable of defining the optimal subset of flow features to identify different types of flows, thus helping to improve the accuracy of network tasks, such as anomaly detection and QoS enforcement. The contribution of this paper is threefold: (*i*) to devise a module capable of gathering and identifying network flow characteristics; (*ii*) to create a set of advanced flow features to characterize traffic profiles; and (*iii*) to investigate the use of two different feature selection techniques, the *Principal Component Analysis* (PCA) and the *Genetic Algorithm* (GA), to determine the subset of flow features that is more appropriate for classifying a particular traffic profile. Some of the traffic features proposed in this paper are indeed more meaningful than the native flow counters provided by OpenFlow, and allow a more accurate classification of different types of traffic. Further, our experimental results show that different subsets of flow features can yield a more accurate traffic classification depending on the traffic profile.

The paper is organized as follows. In Section II, we discuss the related work. In Section III, we present our proposed solution for the identification and selection of traffic flow features in OpenFlow-based networks. In Section IV, we describe the experiments and initial results. Finally, in Section V, we present concluding remarks and outline the future work.

978-1-5090-1849-9/15 $31.00 © 2015 IEEE
DOI 10.1109/NCA.2015.12

II. Background and Related Work

Traffic classification algorithms use as input the flows to be classified as well as flows' descriptive features. The accuracy of classification algorithms depends not only on the quality of collected features but also on the amount of features considered too. Too few features hinders more precise classifications; too many features hinders that as well, because of noise caused by an excessive number of variables (*i.e.*, features) being considered. As such, there is an optimal number of features that should be taken into account, and finding such an optimal number is a research challenge per se. To achieve that, feature selection techniques are necessary, and they have been investigated for many years already [4] [5]. However, network applications change continuously and new traffic profiles are hard to predict. As a result, traffic classification and feature selection become even more challenging.

Blum *et al.* [6] describe several solutions to deal with large amounts of irrelevant features. The authors discuss solutions for feature selection based on (*i*) heuristics methods, (*ii*) filtering of irrelevant data, and (*iii*) weight assignment to features. Fahad *et al.* [7] investigate the advantages of using techniques to select traffic features, such as information gain (IG), gain ratio (GR), principal component analysis (PCA), correlation based feature selection (CBFS), Chi-square, and consistency-based-search (CBS). The performance of each technique was evaluated using metrics such as stability (*i.e.*, selection of the same set of flow features despite producing variations in traffic classification) and similarity (*i.e.*, how similar is each subset of features generated by a technique). Finally, Lanzi *et al.* [8] present feature selection using heuristics, including a solution based on genetic algorithms.

Previous research on feature selection investigated the use of techniques and algorithms to assist traffic classification in traditional IP networks. These research efforts suffered from numerous limitations imposed by traditional IP networks, such as (*i*) distributed control state in forwarding devices, (*ii*) proprietary decision-making logic, and (*iii*) heterogeneous protocols and interfaces. These limitations have an impact on solutions that aim, for example, to obtain an overall view of the network traffic since it is difficult and expensive to collect decentralized and non-standard information. The emergence of the SDN paradigm brought new possibilities to assist traffic classification and feature selection. As an example, flow statistics from each switch can be gathered through standard interfaces and protocols, such as OpenFlow. However, specific solutions to feature selection specifically in SDN environments have not been investigated yet. Thus, our goal is to explore the native collection of flow statistics and extend it to obtain more appropriate features for traffic classification.

III. An Architecture for Flow Feature Identification and Selection in SDN

In this section, we propose an architecture to collect, extend, and select flow features for traffic classification in Software-Defined Networking (SDN). Its main goal is to offer an extensive set of flow features that can be analyzed and refined, and to be capable of finding the optimal subset of features to classify different types of traffic flows. We consider a flow as a 5-tuple consisting of source IP address, destination IP address, TCP/UDP source port, TCP/UDP destination port, and transport protocol identification. It is important to emphasize that this tuple can be adjusted according to the needs of each network infrastructure. Thus, some approaches can ignore information such as destination TCP/UDP port when the network infrastructure performs only one type of service, *e.g.*, HTTP.

A. Architecture Overview

Our proposed architecture is depicted in Figure 1. It comprises two main components: (*i*) *Flow Feature Manager*, which is responsible for handling network information and for computing additional flow features; and (*ii*) *Flow Feature Selector*, which is responsible for analyzing each flow feature exposed by the Flow Feature Manager, and for selecting the most meaningful ones to be applied in traffic classification. Note that the implementation of these components, which execute in the *Application Plane*, does not require modifications to the SDN controller implementation. Further, these components can communicate with other SDN applications (*e.g.*, monitoring or management systems) regardless of the SDN controller implementation.

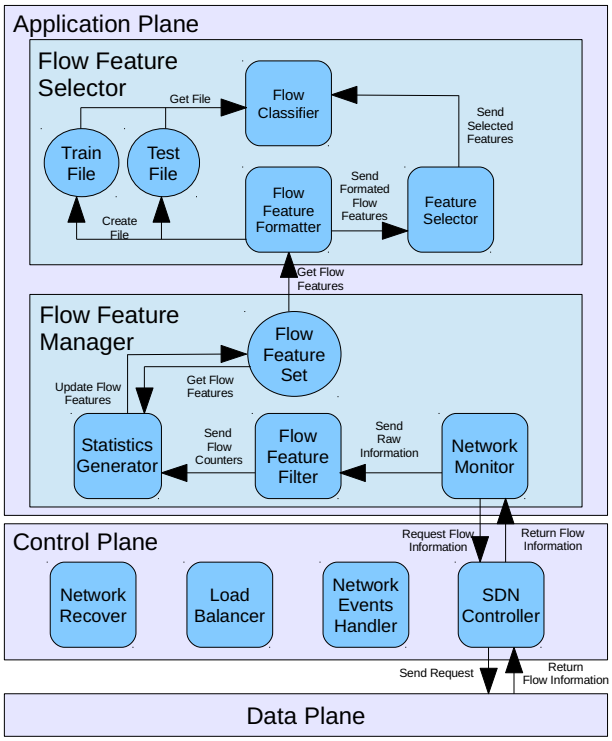

Fig. 1: Architecture for feature identification and selection.

In the following, we describe these components in details:

- **Flow Feature Manager** comprises three modules: (*i*) the Network Monitor, responsible for gathering network information from the network through requests sent to the controller; (*ii*) the Flow Feature Filter, which filters the information received from the Network Monitor and sends it to the Statistics

Generator; and (*iii*) the Statistics Generator, which extends the information gathered in a set of more complex flow features, as discussed later on in this section.

- **Flow Feature Selector** comprises three modules: (*i*) the Flow Feature Formatter, which organizes the data for the feature selection algorithms and automatically creates the training and testing sets for the traffic classification algorithms; (*ii*) the Feature Selector, which is responsible for creating the optimal subset of flow features using a specific technique, *e.g.*, Principal Component Analysis (PCA); and (*iii*) the Flow Classifier that evaluates the flow feature subsets created using traffic classification accuracy as metric.

These components operate as follows. Periodically, the Network Monitor sends to the controller a request for traffic information. The time interval for this request can be configured on demand. For example, flows with short duration and frequent bursts require a smaller time interval between requests, while a longer time interval can be used for flows with longer duration and more constant behavior. The controller then gathers the traffic information from the flow tables of each switch[1] and replies to the Network Monitor. This information is then sent to the Flow Feature Filter, selecting only the native counters, *i.e.*, byte and packet counters. The Statistics Generator uses the native counters gathered from the switches to calculate new flow features (*e.g.*, packet length mean), extend the flow features, and store these in a data structure that we call *Flow Feature Set*.

Subsequently, the process of analyzing the full set of features and selecting the optimal features subset is started. First, the Flow Feature Formatter receives the flow feature set sent by the Flow Feature Manager, and performs two operations: (*i*) it creates a training and a test set that will be used by the Flow Classifier; and (*ii*) it organizes the full flow feature set in the specific format employed by the Flow Feature Selector. The formatted data is used as input to the feature selection algorithms implemented in this module, which are able to identify the most meaningful features for flow classification. Finally, the Flow Classifier module classifies the flows using the feature subset yielded by the Feature Selector to evaluate the fitness of each subset for some traffic profile. Similarly to the Feature Selector module, the Flow Classifier also supports the implementation of a range of different algorithms.

In our proof-of-concept implementation, the Flow Feature Selector is instantiated with (*i*) *Principal Component Analysis* (PCA) and (*ii*) *Genetic Algorithm* (GA). The PCA algorithm [10] evaluates the relationships between a set of variables, determining the variability associated with each of them. PCA is used as a strategy to reduce the number of variables by removing correlated values. The result of PCA is a subset of the original variables, called principal component, which accounts for the most significant variance in the original dataset. The Genetic Algorithm mimics natural evolution and combines current best solutions for producing new solutions, which are then analyzed to assess their quality. Briefly, the algorithm creates an initial population representing a set of solutions. During each iteration, called generation, it selects the fittest solutions and creates new ones through a combination (crossover) or mutation [11]. A series of parameters must be determined a priori, such as population size and mutation probability. Although we used two well-known algorithms for realizing the Flow Feature Selector module, we emphasize that this module can support a wide-range of other algorithms.

[1]In the OpenFlow switch, the flow table stores actions for every incoming packet [9]. The flow table is organized in a structure containing a subset of packets headers associated with each flow information.

B. Extended Flow Features

OpenFlow is currently the most important protocol for SDN implementation [12]. Although traffic classification can be performed using the native traffic counters provided by OpenFlow (*e.g.*, packet counts), this approach presents several limitations, since these counters do not allow detecting atypical traffic behavior, such as packet bursts. Because of the wide range of possible traffic profiles, more descriptive traffic discriminators are necessary. In this paper, we propose an extension of the native OpenFlow counters through statistics analysis, with the goal of producing more descriptive information about the traffic behavior, such as bytes per second mean and packets per second variance. More specifically, we rely on the standard byte and packet counts retrieved from the network controller, but create an enhanced flow feature set. We introduce a third type of counter, named *samples counter*, which indicates the number of times each flow appears during feature polling. This counter is maintained by the Statistics Generator and is essential to compute some of the more advanced flow features, as discussed below.

The extended set of flow features introduced in this paper is classified into three categories: (*i*) *statistical features* – mean, variance, first, and third quartiles; (*ii*) *scalar features* – maximum and minimum values, flow size, and flow duration; and (*iii*) *complex feature* – Discrete Fourier Transform (DFT) of packet inter-arrival-time. Table I presents the complete set comprising 33 new flow features. We advocate that these extended flow features can improve the performance of classification schemes in SDN. First, the statistical features are suitable to summarize the temporal behavior of traffic profiles. For example, the mean packet count represents the central value of a set of observations. Thus, they are better descriptors in comparison to SDN native packet counters. Second, scalar features are convenient to indicate the instantaneous profile of traffic flows, such as duration. As a result, they are important in the detection of short-lived communications or packet bursts, since statistical features alone are not sensitive to these traffic profiles. Finally, complex features are useful to anticipate the need for more sophisticated traffic information. Different from the previous categories, complex features can be used to compress a wide range of traffic observations. For example, Discrete Fourier Transform can be used to refine several measurements of packet's inter-arrival-time into a set of frequency components, which is a more distinguished representation. Some of these features were investigated by Auld *et al.*[13] and Moore *et al.* [14].

OpenFlow's native counters were used as the basis for deriving two new flow features, namely packet length P_l and

TABLE I: Extended set comprising 33 new flow features.

Statistical Features	Scalar Features	Complex Features
Bytes per second mean	Bytes per second maximum value	Packet inter-arrival-time Fourier Transform 1^{st} Component
Bytes per second variance	Bytes per second minimum value	Packet inter-arrival-time Fourier Transform 2^{nd} Component
Packets per second mean	Packets per second maximum value	Packet inter-arrival-time Fourier Transform 3^{rd} Component
Packets per second variance	Packets per second minimum value	Packet inter-arrival-time Fourier Transform 4^{th} Component
Packets length mean	Packets length maximum value	Packet inter-arrival-time Fourier Transform 5^{th} Component
Packets length variance	Packets length minimum value	Packet inter-arrival-time Fourier Transform 6^{th} Component
Packets length 1^{st} quartiles	Packets inter-arrival-time maximum value	Packet inter-arrival-time Fourier Transform 7^{th} Component
Packets length 3^{rd} quartiles	Packets inter-arrival-time minimum value	Packet inter-arrival-time Fourier Transform 8^{th} Component
Packet inter-arrival-time mean	Flow duration	Packet inter-arrival-time Fourier Transform 9^{th} Component
Packet inter-arrival-time variance	Flow size in packets	Packet inter-arrival-time Fourier Transform 10^{th} Component
Packet inter-arrival-time 1^{st} quartiles	Flow size in bytes	
Packet inter-arrival-time 3^{rd} quartiles		

packet inter-arrival-time[2] P_{it}. These are estimations of the real packet length and real inter-arrival-time, since calculating their precise values would require deep packet inspection and the monitoring of every packet in the network, which are expensive tasks. Consequently, we apply sampling strategies to estimate these features and use Equation 1 and Equation 2 to calculate packet length and inter-arrival, respectively. B_c represents byte count, P_c represents packet count between two requests, and T is the time-interval between two requests.

$$P_l = \frac{B_c}{P_c} \tag{1}$$

$$P_{it} = \frac{T}{P_c} \tag{2}$$

To calculate mean and variance, the Statistics Generator updates the mean μ and the variance σ^2 for the flow features every time a new information request is processed. Mean is calculated using Equation 3, where μ_n is the updated mean, μ_o is the old mean, θ is the new sample, and n is the samples counter.

$$\mu_n = \frac{n * \mu_o + \theta}{n+1} \tag{3}$$

The updated variance σ_n^2 is calculated using Equation 4, where σ_o^2 is the old variance.

$$\sigma_n^2 = \frac{n}{n+1} * (\sigma_o^2 + \frac{(\theta - \mu_n) * (\theta - \mu_o)}{n}) \tag{4}$$

In this work, we calculate DFT of packet inter-arrival-time samples and use the top ten components as flow features according to Auld *et al.* [13] work. DFT represents the samples of a variable in the frequency domain. It is defined by Equation 5, where N is the number of samples, x_n is a sample, and X_k is the resulting component.

$$X_k = \sum_{n=0}^{N-1} x_n . e^{-2\pi i k n / N},\ k \,\epsilon\, \mathbb{Z} \tag{5}$$

IV. Prototype and Experimental Evaluation

In this section, we describe a proof-of-concept implementation, comprising the actual algorithms used to instantiate the modules described in the previous section. We also present our experimental evaluation and discuss initial results.

A. Prototype Implementation

We developed a prototype for the Flow Feature Manager and the Flow Feature Selector components in Python. The sets of flow features are stored in a *dictionary* data structure. To select the optimal flow feature subset, the Feature Selector module implements two distinct algorithms: Principal Component Analysis (PCA) [10] and Genetic Algorithm (GA) [11]. In addition, to analyze the classification accuracy of each subset of flow features, the Flow Classifier module implements the well-known Support Vector Machine (SVM) algorithm

[2]We use packet length and packet inter-arrival-time statistics because individual values do not usually give substantial information about the traffic profile.

for traffic classification [15]. The PCA, GA, and SVM were implemented using R[3] libraries (psych, genalg, and e1071, respectively). Still, each module can be customized with other classification algorithms and feature selection techniques as needed. Finally, for managing and monitoring the network infrastructure, we chose the Floodlight Controller[4] version 1.0, as it offers a suitable communication interface between our application and the controller through a REST API.

B. Experiments and Initial Results

Our goal is to measure the accuracy of the resulting traffic classification using specific subsets of traffic features, discovered via the Genetic Algorithm (GA) and the Principal Component Analysis (PCA), as opposed to using the full-blown set of features. Accuracy is defined as the percentage of correctly classified instances among the total number of instances [1]. To evaluate the proposed architecture, we used a standard tree topology with three levels comprising seventeen switches, and sixty-four hosts. We defined four types of traffic profiles in our experiments: (*i*) DDoS attack, (*ii*) FTP traffic, (*iii*) video streaming using VLC Media Player[5], and (*iv*) background traffic generated using Scapy[6]. We composed these types of traffic into three scenarios defined in Table II. For each scenario we use the traffic classification accuracy as metric to define the optimal flow feature set. For comparison purposes, we use as benchmark the classification accuracy for each scenario obtained by using the complete set of flow features: 94.67% (scenario A), 92% (scenario B), and 85.33% (scenario C).

TABLE II: Experimental scenarios.

Scenario	Type of flow
A	DDoS attacks (60%) with Scapy flow (40%)
B	FTP traffic (35%) with Scapy flows (65%)
C	Video streaming (50%) with Scapy flows (50%)

In order to produce more accurate results, the Genetic Algorithm (GA) must be initialized with a few parameters. We set the *population size* to 200 individuals randomly generated and the *crossover percentage* to 20%. We also must configure two other parameters: (*i*) *number of iterations* and (*ii*) *mutation probability*. In order to do so, we analyzed the classification accuracy for a number of iterations, namely 10, 50, 75, 100, 150, and 200, until the accuracy stabilized. Figure 2 presents the resulting accuracy obtained for a given number of iterations. As a result, we set the number of iterations to 100, since it is the smallest value with the highest accuracy.

As can be observed in Figure 3, the *mutation probability* does not have an impact in classification accuracy, and we use 0.01 as standard mutation probability. Because of the amount of time taken to execute this algorithm (nearly 88 minutes), we chose scenario C to set these parameters, since it has the lowest classification accuracy when using all flow features. The classification accuracies obtained by using GA to find the optimal subset of flow features for each scenario were: 98% (scenario A), 94.67% (scenario B), and 91.33% (scenario C).

[3]www.r-project.com
[4]http://www.projectfloodlight.org/
[5]http://www.videolan.org/vlc/index.html
[6]http://www.secdev.org/projects/scapy/

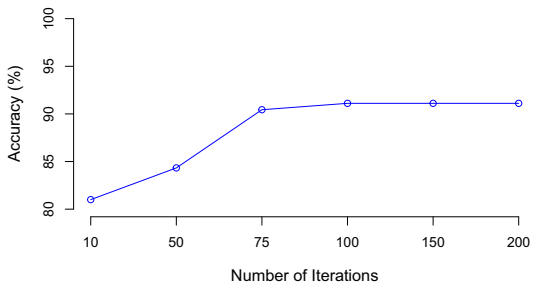

Fig. 2: Accuracy for each number of iterations.

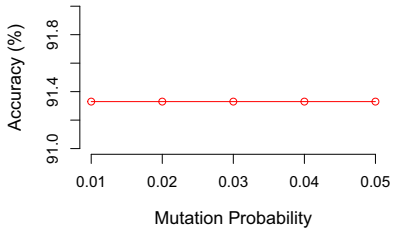

Fig. 3: Accuracy for each mutation probability.

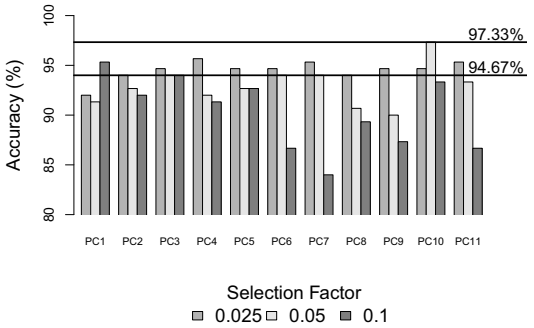

Fig. 4: Accuracy of the 11 most meaningful principal components selected by PCA (for each factor) in scenario A.

To find the optimal subset of flow features, we also applied PCA. PCA determines the most important features by creating one principal component to match each variable (feature), *i.e.*, in our experiments it creates 33 principal components. It works as follows: initially, only the components that jointly represent 90% of the features variability are chosen, resulting in 11 components; then, since a principal component is created as a weighed sum of the features, a subset is created for each component including the features with higher weight than a specific *factor*. In our experiments, we used 0.025, 0.05, and 0.1 as factors. As a result, we have three different subsets for each component. Figure 4 shows the classification accuracy for

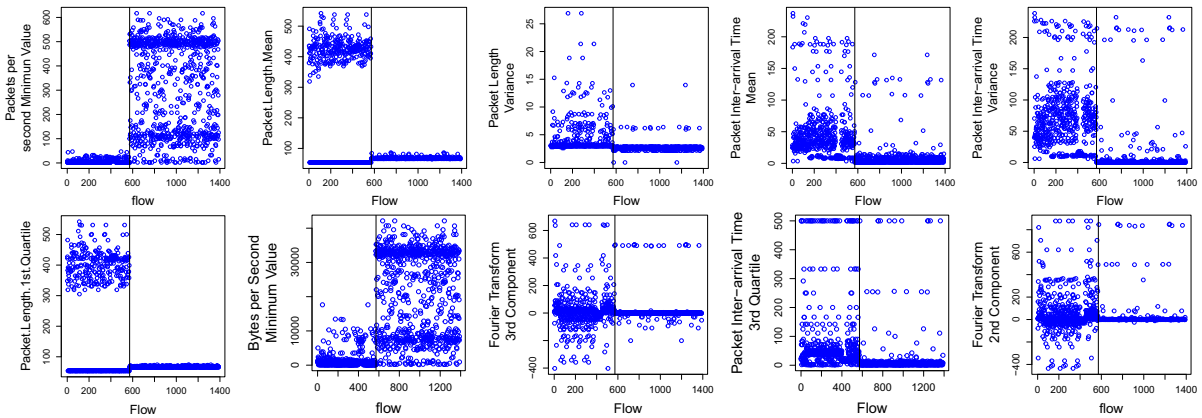

Fig. 5: Ten most meaningful flow features according to PCA analysis in Scenario A (DDoS).

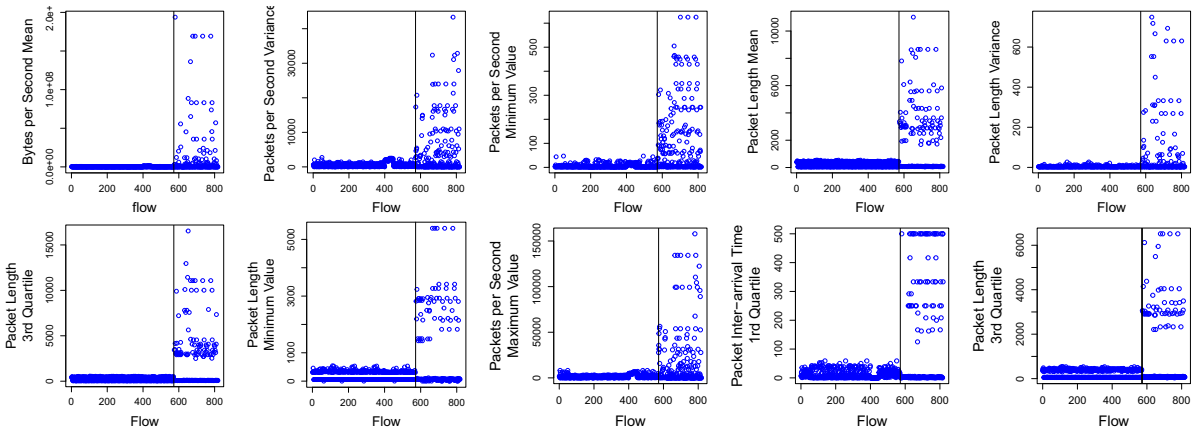

Fig. 6: Ten most meaningful flow features according to PCA analysis in Scenario B (FTP).

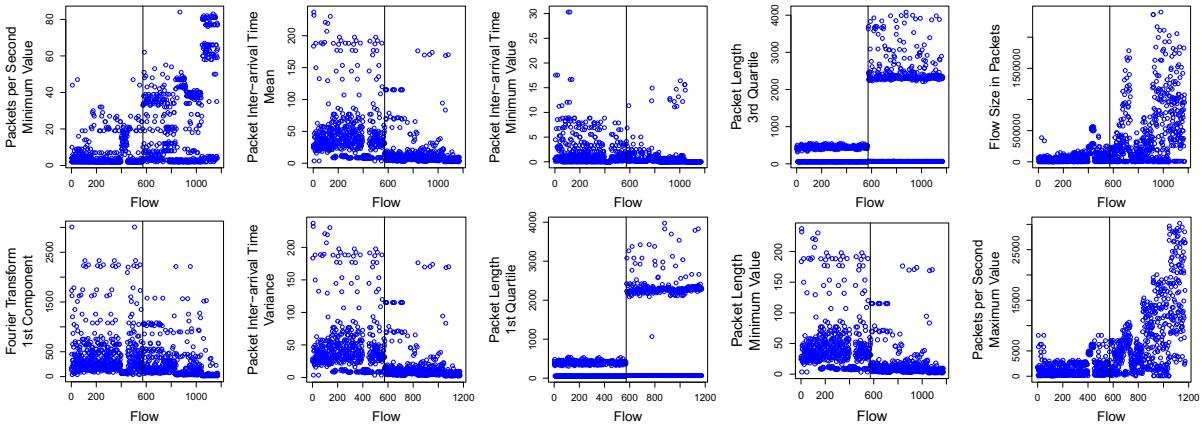

Fig. 7: Ten most meaningful flow features according to PCA analysis in Scenario C (Video Streaming).

each subset in scenario A. We can draw several conclusions from this: (*i*) not all subsets lead to a higher accuracy compared to the accuracy obtained by using the complete flow feature set (94.67%); (*ii*) factor 0.025 seems to lead to the best solutions in most cases, although the optimal solution is a subset of features selected using factor 0.05 in the 10^{th} principal component (97.33%); and (*iii*) factor 0.1 cannot be ruled out because it leads to the highest accuracy in the 1^{st} principal component,

TABLE III: Optimal flow features subsets selected for each experiment.

Scenario A		Scenario B		Scenario C	
PCA	**GA**	**PCA**	**GA**	**PCA**	**GA**
Bytes per second mean	Bytes per second mean	Bytes per second mean	Bytes per second variance	Bytes per second Minimum value	Bytes per second variance
Bytes per second Minimum value	Bytes per second variance	Bytes per second variance	Bytes per second Maximum value	Packets per second mean	Bytes per second Maximum value
Packets per second Minimum value	Bytes per second Maximum value	Bytes per second Maximum value	Minimum value of Bytes per second	Packets per second variance	Bytes per second Minimum value
Packets Length mean	Bytes per second Minimum value	Bytes per second Minimum value	Packets per second Mean	Packets per second Maximum value	Packets per second variance
Packets Length variance	Packets per second mean	Packets per second mean	Packets per second Variance	Packets per second Minimum value	Packets per second Minimum value
Packet Length Maximum value	Packets per second Maximum value	Packets per second variance	Packets per second Maximum value	Packet Length variance	Packet Length mean
Packet Length Minimum value	Packets per second Minimum value	Packet per second Maximum value	Packet per second Minimum value	Packet Length Minimum value	Packet Length variance
Packet Inter-arrival time mean	Packet Length mean	Packet per second Minimum value	Packet Length Mean	Packet Inter-arrival time mean	Packet Length Maximum value
Packet Inter-arrival time variance	Packet Length variance	Packet Length mean	Packet Length Maximum value	Packet Inter-arrival time variance	Packet Length Minimum value
Packet Inter-arrival time Maximum value	Packet Length Maximum value	Packet Length variance	Packet Length Minimum value	Packet Inter-arrival time Maximum value	Packet Inter-arrival time mean
Packet Inter-arrival time 1^{st} quartile	Packet Length Minimum value	Packet Length Maximum value	Packet inter-arrival time Mean	Packet Inter-arrival time Minimum value	Packet Inter-arrival time Maximum value
Packet Length 1^{st} quartile	Packet Inter-arrival time mean	Packet Length Minimum value	Packet inter-arrival time Variance	Flow Size in Packets	Flow Size in Bytes
Packet Length 3^{rd} quartile	Packet Inter-arrival time Minimum value	Packet Inter-arrival time variance	Flow Size in Bytes	Packet Inter-arrival time 1^{st} quartile	Packet Inter-arrival time 1^{st} quartile
Fourier Transform 2^{nd} Component	Flow Duration	Packet Inter-arrival time Maximum value	Flow Size in packets	Packet Inter-arrival time 3^{rd} quartile	Packet Inter-arrival time 3^{rd} quartile
Fourier Transform 3^{rd} Component	Flow Size in Bytes	Packet Inter-arrival time Minimum value	Packet inter-arrival time 1^{st} quartile	Packet Length 1^{st} quartile	Packet Length 3^{rd} quartile
Fourier Transform 6^{th} Component	Flow Size in Packets	Flow Size in Bytes	Packet Length 1^{st} quartile	Fourier Transform 1^{st} Component	Fourier Transform 1^{st} Component
	Packet Inter-arrival time 3^{rd} quartile	Packet Inter-arrival time 1^{st} quartile	Packet Length 3^{rd} quartile	Fourier Transform 2^{nd} Component	Fourier Transform 2^{nd} Component
	Packet Length 1^{st} quartile	Packet Inter-arrival time 3^{rd} quartile	Fourier Transform 2^{nd} component	Fourier Transform 4^{th} Component	Fourier Transform 3^{rd} Component
	Fourier Transform 1^{st} Component	Packet Length 1^{st} quartile	Fourier Transform 4^{th} component	Fourier Transform 5^{th} Component	Fourier Transform 4^{th} Component
	Fourier Transform 2^{nd} Component	Packet Length 3^{rd} quartile	Fourier Transform 5^{th} component	Fourier Transform 6^{th} Component	Fourier Transform 5^{th} Component
	Fourier Transform 3^{rd} Component	Fourier Transform 2^{nd} Component	Fourier Transform 7^{th} component	Fourier Transform 7^{th} Component	Fourier Transform 8^{th} Component
	Fourier Transform 4^{th} Component	Fourier Transform 3^{rd} Component	Fourier Transform 10^{th} Component	Fourier Transform 8^{th} Component	Fourier Transform 9^{th} Component
	Fourier Transform 5^{th} Component	Fourier Transform 4^{th} Component		Fourier Transform 9^{th} Component	Fourier Transform 10^{th} Component
	Fourier Transform 6^{th} Component	Fourier Transform 5^{th} Component		Fourier Transform 10^{th} Component	
	Fourier Transform 9^{th} Component	Fourier Transform 7^{th} Component			
		Fourier Transform 8^{th} Component			
		Fourier Transform 9^{th} Component			
		Fourier Transform 10^{th} Component			

thus can also result in high accuracy in a different scenario. Based on these three observations, it is not possible to identify which factor gives the best subset for any given scenario, consequently all three factor must be considered.

Because of space constraints, we illustrate only the 10 most meaningful features (in order of importance) selected through PCA in each scenario, depicted in Figure 5, Figure 6, and Figure 7. In each scatter plot, the vertical line separates the types of traffic being simulated: the left side represents the background traffic, and the right side represents DDoS traffic (in Figure 5), FTP flows (in Figure 6), and Video Streaming (in Figure 7). Classification accuracies for the optimal subset of features selected with PCA for each scenario were: 97.33% (scenario A), 94% (scenario B), and 88.67% (scenario C).

Figure 8 summarizes the classification accuracy in each scenario using all features, as well as using the optimal subsets selected with PCA and GA. In all scenarios, the accuracy was improved by using the optimal feature subsets generated by GA and PCA, when compared with the accuracy obtained by using the complete set of flow features. Table III details the feature subsets selected by PCA and GA in each scenario.

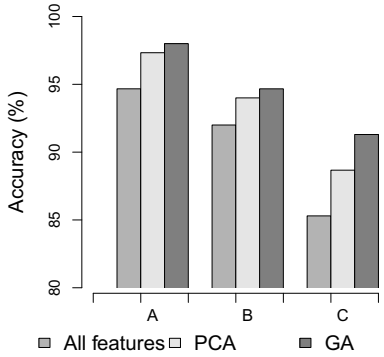

Fig. 8: Traffic classification accuracy for each flow feature set in all three scenarios.

V. Concluding Remarks

In this paper, we introduced an architecture to collect, extend, and select a set of flow features for traffic classification in Software-Defined Networking (SDN). We also demonstrated that for particular types of traffic, an optimal subset of flow features can be selected using both principal component analysis and genetic algorithm.

Our experimental results show that some of the traffic features proposed in this paper are indeed more meaningful than the native flow counters provided by OpenFlow, and allow a more accurate classification of different types of traffic. Also, our results show that these features can be combined, according to each type of flow, and be used for indicating the profile of a particular flow. Furthermore, in all scenarios the subset of flow features selected by either GA or PCA allows a more accurate traffic classification when compared to classification accuracy obtained with the complete set of flow features.

As future work, we will investigate additional flow features, and analyze other types of flow services. Finally, we plan to explore and apply other algorithms and mechanisms for traffic classification and compare them with the results we have collected so far.

Acknowledgement

This work is partially supported by ProSeG - Information Security, Protection and Resilience in Smart Grids, a research project funded by MCTI/CNPq/CT-ENERG #33/2013.

References

[1] T. Nguyen and G. Armitage, "A survey of techniques for internet traffic classification using machine learning," *IEEE Communications Surveys Tutorials*, vol. 10, no. 4, pp. 56–76, Fourth 2008.

[2] J. A. Wickboldt, W. P. de Jesus, P. H. Isolani, C. B. Both, J. Rochol, and L. Z. Granville, "Software-Defined Networking: Management Requirements and Challenges," *IEEE Communications Magazine*, vol. 53, no. 1, pp. 278–285, Jan 2015.

[3] N. McKeown, T. Anderson, H. Balakrishnan, G. Parulkar, L. Peterson, J. Rexford, S. Shenker, and J. Turner, "OpenFlow: Enabling Innovation in Campus Networks," *SIGCOMM Comput. Commun. Rev.*, vol. 38, no. 2, pp. 69–74, Mar. 2008. [Online]. Available: http://doi.acm.org/10.1145/1355734.1355746

[4] C. Pascoal, M. Rosario de Oliveira, R. Valadas, P. Filzmoser, P. Salvador, and A. Pacheco, "Robust feature selection and robust pca for internet traffic anomaly detection," in *Proceedings of IEEE INFOCOM*, March 2012, pp. 1755–1763.

[5] M. Mantere, M. Sailio, and S. Noponen, "Feature Selection for Machine Learning Based Anomaly Detection in Industrial Control System Networks," in *IEEE International Conference on Green Computing and Communications (GreenCom)*, Nov 2012, pp. 771–774.

[6] A. L. Blum and P. Langley, "Selection of Relevant Features and Examples in Machine Learning," *Journal Artificial Intelligence - Special issue on relevance*, vol. 97, no. 1-2, pp. 245–271, Dec. 1997. [Online]. Available: http://dx.doi.org/10.1016/S0004-3702(97)00063-5

[7] A. Fahad, Z. Tari, I. Khalil, I. Habib, and H. Alnuweiri, "Toward an efficient and scalable feature selection approach for internet traffic classification," *Computer Networks*, vol. 57, no. 9, pp. 2040 – 2057, 2013.

[8] P. Lanzi, "Fast feature selection with genetic algorithms: a filter approach," in *IEEE International Conference on Evolutionary Computation*, Apr 1997, pp. 537–540.

[9] Open Networking Foundation, "OpenFlow Switch Specification - Version 1.0.0 (Wire Protocol 0x01)," December 31 2009, available at: <https://www.opennetworking.org/sdn-resources/onf-specifications/openflow>. Accessed: August 2014.

[10] K. Pearson., "LIII. On lines and planes of closest fit to systems of points in space," *Philosophical Magazine Series 6*, vol. 2, no. 11, pp. 559–572, 1901. [Online]. Available: http://dx.doi.org/10.1080/14786440109462720

[11] I.-S. Oh, J.-S. Lee, and B.-R. Moon, "Hybrid genetic algorithms for feature selection," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 26, no. 11, pp. 1424–1437, Nov 2004.

[12] N. Feamster, J. Rexford, and E. Zegura, "The Road to SDN," *Queue - Large-Scale Implementations*, vol. 11, no. 12, pp. 20:20–20:40, Dec. 2013. [Online]. Available: http://doi.acm.org/10.1145/2559899.2560327

[13] T. Auld, A. Moore, and S. Gull, "Bayesian Neural Networks for Internet Traffic Classification," *IEEE Transactions on Neural Networks*, vol. 18, no. 1, pp. 223–239, Jan 2007.

[14] A. W. Moore and D. Zuev, "Internet Traffic Classification Using Bayesian Analysis Techniques," *SIGMETRICS Perform. Eval. Rev.*, vol. 33, no. 1, pp. 50–60, Jun. 2005. [Online]. Available: http://doi.acm.org/10.1145/1071690.1064220

[15] H. Kim, K. Claffy, M. Fomenkov, D. Barman, M. Faloutsos, and K. Lee, "Internet Traffic Classification Demystified: Myths, Caveats, and the Best Practices," in *Proceedings of the 2008 ACM CoNEXT Conference*, ser. CoNEXT '08. New York, NY, USA: ACM, 2008, pp. 11:1–11:12.