A Framework for QoS-aware Traffic Classification Using Semi-supervised Machine Learning in SDNs

Pu Wang*,Shih-Chun Lin†, and Min Luo‡
*Department of Electrical Engineering and Computer Science, Wichita State University, Wichita, KS 67260
†BWN Lab, School of Electrical and Computer Engineering, Georgia Institute of Technology, Atlanta, GA 30332
‡ Shannon Lab, Huawei Technologies Co., Ltd. Santa Clara
Email: pu.wang@wichita.edu; slin88@ece.gatech.edu; min.ch.luo@huawei.com

***Abstract*—In this paper, a QoS-aware traffic classification framework for software defined networks is proposed. Instead of identifying specific applications in most of the previous work of traffic classification, our approach classifies the network traffic into different classes according to the QoS requirements, which provide the crucial information to enable the fine-grained and QoS-aware traffic engineering. The proposed framework is fully located in the network controller so that the real-time, adaptive, and accurate traffic classification can be realized by exploiting the superior computation capacity, the global visibility, and the inherent programmability of the network controller. More specifically, the proposed framework jointly exploits deep packet inspection (DPI) and semi-supervised machine learning so that accurate traffic classification can be realized, while requiring minimal communications between the network controller and the SDN switches. Based on the real Internet data set, the simulation results show the proposed classification framework can provide good performance in terms of classification accuracy and communication costs.**

Keywords: Traffic classification, SDN, QoS, Semi-supervised Machine Learning

I. Introduction

Currently, software defined networking (SDN) is envisioned as an emerging and promising networking paradigm with the promise to dramatically improve network resource utilization, simplify network management, reduce operating costs and promote innovation and evolution [1]. As the key feature of SDN, the separation between the control plane and the data plane necessitates the revisit and redesign of the traffic engineering (TE) solutions so that the promising features of SDN, such as openness, programmability, and global visiability, can be actively exploited. In particular, traffic engineering in SDN focuses on efficiently and effectively identifying, regulating, and forwarding traffic flows in the whole network [2].

As SDN manages the network traffic on the basis of "flows", the accuracy and efficiency of the traffic classification (TC) engine plays a crucial role in SDN. Different from the most of the existing work which focuses on identifying the applications that generate the traffic flows, we aim to propose a novel QoS-aware TC framework capable of identifying the QoS class, such as interactive video gaming or bulky data transfer, for different traffic flows in a real-time and cost-efficient fashion. On one hand, providing desired QoS for different traffic flows is an essential part of the traffic engineering in SDN. Thus, the TC engine has to identify the QoS class for the particular traffic flows so that the suitable routing paths can be chosen. On the other hand, the conventional traffic classification solutions, which aim to identify the exact application of every traffic flow, is not an effective way to identify the QoS class of the traffic flows because many different applications may belong to the same QoS class, which demands the similar QoS requirements. Moreover, as many new applications appear every day, it is time-consuming and impractical to maintain the real-time update of the list of all applications existing within the Internet.

To counter the above-mentioned challenge, we jointly apply machine learning with DPI, in a novel framework that can be fully implemented in a SDN controller. The proposed framework consists of two components: (i) the local traffic identification component at SD switches at the network edge and (ii) the global traffic classifier at the network controller. The former one aims to detect the long-lived, i.e., "elephant" flows among the new incoming ones, while the latter part performs the QoS-aware traffic classification for identifying the QoS class of the traffic flow through a mapping function. The mapping function is simply a function that takes a few features of the traffic flow, e.g., the average packet interarrival time, Hurst parameter and port number, as the inputs and gives the QoS class of the traffic flow as the output. The global traffic classifier at the controller is responsible for learning, building and refining the mapping function based on the historical traffic information.

The proposed traffic classification system has three advantages. First, the SD-switches are kept as simple as possible by only incorporating light-weight elephant flow identification module. Second, the network controller is utilized to guarantee the accuracy and the adaptability of the QoS traffic classifier. This is achieved by exploiting the global view of the network flows to build the accurate mapping functions through time-consuming but accurate DPI along with the semi-supervised machine learning that only requires limited information, e.g., first 20 packets, from the elephant flows detected at the SD switches. Third, the whole framework follows a modular design principle so that every component in the framework can be improved at any time. The proposed QoS traffic classification solution can be adopted to enable fine-grained QoS-aware traffic engineering in SDN [3].

978-1-5090-2628-9/16 $31.00 © 2016 IEEE
DOI 10.1109/SCC.2016.133

The rest of the paper is organized as follow: in Section II, the related work is provided; the details of the QoS-aware traffic classification framework can be found in Section III; Section IV gives the description of evaluation process and provides the simulation results; and a conclusion of this paper is given in Section V.

II. Related Work

Historically, traffic classification relies on TCP or UDP port matching [4]. However, because of the increasing usage of the private and dynamic ports, the port-based techniques are not effective any more. To counter this challenge, two schemes are introduced, i.e., Deep packet inspection (DPI) and Machine Learning (ML). DPI inspects the packet payload and searches for known signatures to infer the most likely application [5]. DPI is of high accuracy but fails with the encrypted payload. Meanwhile, ML-based approaches [6] learn from the big data and use statistical properties of the traffic flow to infer the application used. However, most of the related work [7] [8] [9] were using supervised ML, which means all available traffic traces are labeled with known applications, but in reality, limited information can be captured and the ever-changing new applications makes supervised ML less effective. To address such issue, unsupervised ML was investigated [10] [11], which exploits unlabeled database as the training set. However, it is difficult for unsupervised ML-based approaches to realize a good performance with low complexity. Then, semi-supervised ML was proposed [12], which combines the benefits of both supervised or unsupervised ML. However, unsupervised ML-based algorithms cannot be directly applied in SDN without exploiting and conforming the decoupled control and data planes. In this paper, we aim to develop a traffic classification engine which takes into account the unique architectural features of SDN.

III. A QoS-aware traffic classification framework

Our proposed framework aims to classify a traffic flow into a QoS category in a real-time and adaptive fashion without the need of identifying the exact application from which the traffic flow is generated. On one hand, real-time traffic classification is of critical important, which, however, has not been properly explored. Many existing algorithms use flow-level features for traffic classification, which is "a posteriori" method and may generate classification results when the flow ends. On the other hand, an adaptive traffic classification system is required, which should be adapted to the ever-changing network applications by periodically re-training the classifiers.

A. Overall framework

To conform the SDN architecture, our traffic classification (TC) engine is located within the centralized SDN controller. The purposed TC engine performs: (1) efficient network monitoring with low-overhead and minimal switch changes; (2) detection of QoS-significant (i.e., "elephant") flows; (3) QoS-aware traffic classification, and (4) enables services such as application detection using DPI ruining in the network controller.

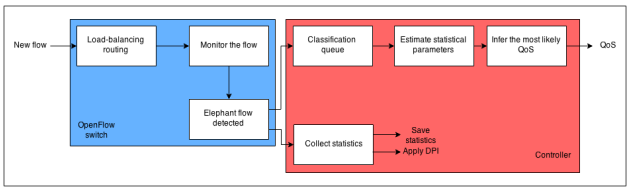

Fig. 1. Traffic classification framework scheme

As shown in Fig. 1, the system consists of two main parts. The first component is responsible of detecting the QoS-significant flows in the new incoming flows. The second component performs the QoS-aware traffic classification and the related network management tasks. The main process of the operation of the system is explained as follow:

1) "Elephant" flows detection: After the network flows go through the edge switches, those switches detect the QoS-significant "elephant" flows. Several approaches have been proposed to detect "elephant" flows in literature. Since the SDN controller can monitor the network, the detection criterion used is that if a flow uses more that K% of the link bandwidth, it is recognized as an "elephant" flow, where K could go from 1% to 10% depending on the bandwidth of that link. The OpenFlow defaultly supported pull-based statistics can be utilized in this detection process. By receiving the traffic statistics sent from the switch, the controller gets to know the existence of "elephant" flows.

2) Statistics collection and features extraction: In network controller, the ML algorithms build a mapping function $g(\vec{\mathbf{x}}) = y$ where $\vec{\mathbf{x}}$ corresponds to measurable statistical properties of an "elephant" flow while y refers to the most likely QoS that flow needs. Therefore, before classifying a flow, two steps need to be done. One is to obtain the measured vector $\vec{\mathbf{x}}$, and the other is to train the ML-based classifier. As long as an "elephant" flow is detected by a SDN switch, the flow information needs to be sent to the controller through the least congested path immediately. Once the controller gets the flow information, it starts to calculate the statistical properties and group them in $\vec{\mathbf{x}}$.

Information is firstly gathered from the packet trace of the flow and then the statistical features are extracted out from that information. The features used to train the ML algorithm and to classify the flows are categorized into the following classes:

1) Time information: inter-arrival period;
2) Packet information, e.g., packet length and direction of the packet;
3) Protocol information, e.g., IP/Port of source/destination and transport protocol;
4) stochastic information, e.g., Hurst parameter

To build a real-time classifier only the information of first N ($N = 20$ in our work) packets in a flow is used to calculate

$\vec{\mathbf{x}}$. The simulation results proof that using only 20 packets is enough to have a good classification. As the socket information is always the same, the information of the packet belonging to the same socket should be gathered only once.

3) QoS-aware traffic classification: The QoS classifier exploits the semi-supervised ML algorithm, e.g., Laplacian SVM which is hosted inside the centralized SDN controller. Before performing the actual traffic classification, the classifier needs to be trained by following the steps below:

1) Setting up a database storing network traffic traces;
2) Filtering out the "elephant" flows from the database;
3) Applying DPI to find out the application of each remained "elephant" flows (i.e., labeling the flows). Note that a significant portion of all flows are still unlabeled due to limited information;
4) Defining QoS classes based on the applications found. Delay, jitter, and loss rate are mainly concerned factors. The corresponding detected applications are assigned to each class as its representative applications. For instance,
   - **Voice**: GoogleVoice
   - **Video conference**: Skype, GoogleTalk
   - **Streaming**: USstream, Sopcast
   - **Bulk data transfer**: FTP, Mega
   - **Interactive data**: SSH, Telnet
   - **Best-effort traffic**: default class

   Notice that not all kinds of application are considered in the QoS class definition, because the only "elephant" flows enters our TC engine ;
5) Labelling all flows with their corresponding QoS classes to complete the "labeling" process;
6) Measuring the statistical parameters of each flow and calculating the feature vector $\vec{\mathbf{x}}$ used in the ML algorithm;
7) Training the classifier using semi-supervised learning. In particular, Laplacian SVM is adopted.

Instead of performing fine-grained application classification, coarse-grained classification with better generalization properties is used here. It is assumed that applications that require the same QoS, tend to exhibit similar statistical properties. This is a typical semi-supervised learning assumption [16], called the cluster assumption.

Semi-supervised ML algorithms use labeled, $\mathcal{X}_L$, and unlabeled, $\mathcal{X}_U$, data to infer the classification model. In semi-supervised learning, the data set consists of n vectors $\mathcal{X} = (x_i)_{i\in[n]}$ and two subsets depending on the label. The first one $\mathcal{X}_L = (\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_l)$ are labeled data set with the labels $\mathcal{Y}_L = (\mathbf{y}_1, \mathbf{y}_2, ..., \mathbf{y}_l)$. The second one $\mathcal{X}_U = (\mathbf{x}_{l+1}, \mathbf{x}_{l+2}, ..., \mathbf{x}_{l+u})$ are unlabeled ones. Considering $\mathcal{X} = \mathcal{X}_L \cup \mathcal{X}_U$ and $l+u = n$, on one hand, when $l = 0$ we do not have labeled data and $\mathcal{X}$ is used for unsupervised learning. On the other hand, when $u = 0$ all samples are labeled and $\mathcal{X}$ is used for supervised learning. Each element $\mathbf{x}_i$ is composed of a series of features. Specifically, each $\mathbf{x}_i$ has m features $\mathbf{f}_i$ so that $\mathbf{x}_i = (\mathbf{f}_1, \mathbf{f}_2, ..., \mathbf{f}_m)$. Fig. 2 summarizes the structure of the data.

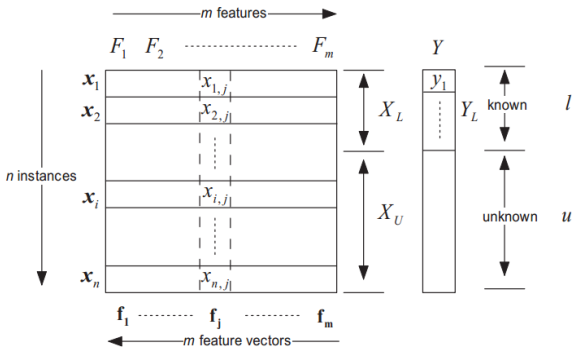

Fig. 2. Data structure for semi-supervised learning

In our scheme, the graph-based semi-supervised learning [17] is adopted which utilizes a graph representation of the data, with each node corresponding to a labelled or unlabelled sample. The graph may be constructed using domain knowledge or similarity of examples, while two common methods are used, which connect each data point either to its k nearest neighbours or to nodes within some distance ϵ. The weight W_{ij} of an edge between $\mathbf{x}_i$ and $\mathbf{x}_j$ is typically set to $e^{\frac{-||x_i - x_j||^2}{\sigma}}$.

Using the principles behind manifold regularization [18] and the regularization formulation of SVM [19], the SVM can be formulated with the manifold regularaizer as:

$$f = \arg\min_{f\in\mathcal{H}} \left\{ C\sum_{i=1}^{n} max(0, 1 - yf(x)) + \gamma_{\mathcal{H}}\|f\|^2_{\mathcal{H}} + \gamma_{\mathcal{I}}\|f\|^2_I \right\}$$

We adopt the similar methods in [20] to solve the The procedure of solving above problem. The detailed procedures are omitted for brevity.

The training phase of the ML-based classifier is usually done offline. With the labeled traffic traces obtained through DPI in 3)–5), when a calculated feature vector $\vec{\mathbf{x}}$ is fed into the classifier, the coefficients of the ML algorithm will be adjusted based on the difference (i.e., error) between the temporary outcome and the actual labeled result. After training with a number of the labeled traces, the coefficients are adjusted well enough to stop the training phase (i.e., the temporary outcome and the actual labeled result match well).

As explained before, since not all applications can be detected and the labeled traces are not enough considering the existence of unknown applications, supervised ML cannot provide good performance in the real world. As semi-supervised ML learns from both labeled and unlabeled data traces simultaneously, those unlabeled flows will be classified into the same QoS class as the labeled ones, because they share the most similar statistical features. Even if the application that yields the unlabeled flows cannot be identified, fortunately, those unlabeled traffic flows can still be categorized into certain QoS classes based on the statistical correlation for the proper traffic regulating or forwarding.

4) Groud truth update and classifier re-trainning: It is known that machine learning learns from the big data and builds a mapping function $g(\vec{\mathbf{x}}) = y$. Thus, having good and meaningful data traces as references is important to build classifiers with good accuracy and generality. However, different types of applications and traffic patterns arise depending on the specific purpose of a network; for example, the applications used in residential networks are different from those used in research networks or data enterprise networks. Furthermore, new applications keep emerging every day and even the current ones may be experiencing updates which change their functionalities and consequently change the statistical properties. Therefore, it is necessary to deploy a policy to gather new data from the network and update the ground truth database periodically, so that it can be used to re-train the QoS classifier after a duration of t_{update}.

Basically, it is required to save N (e.g., $N = 20$) packets of various "elephant" flows flowing through the network so that later we can update the classifiers used. For real-time requirement, it is desirable that a small number of packets is needed to estimate the statistical features because the parameter estimation should be fast. The following method is proposed to gather the data needed:

1) When an "elephant" flow is detected, its information will be stored with a probability of p, the value of which depends on the network state;
2) If an "elephant" flow is selected to be stored, then one of the intermediate SDN switch along the path of the flow will be randomly selected and send the information (i.e., N packets) of the flow to the controller. Those packets received by the controller will be stored in a historical database.

B. Key features of the proposed architecture

(1) A modular approach is used, so that each block could be updated in the future;
(2) The framework is a generic one, which could be deployed in various kinds of networks such as campus network, enterprise network and etc.;
(3) DPI and machine learning are combined into a single framework, which takes the benefits of both worlds. DPI provides accurate dection for known application, while machine learning provides a superior classification performance for unknown applications ;
(4) A periodic network statistics gathering scheme is proposed so that it is possible to update the "ground truth" database and re-train the classifier to make them adapt to future changes;
(5) This is the first engine in SDN that utilizes truly semi-supervised ML algorithms.

IV. Evaluation

To evaluate our design, we conducted traffic classification simulations on the real internet data, which was captured by the Broadband Communication Research Group in UPC, Barcelona, Spain. The data set is a 59GB traffic trace file in which the packets of the internet traffic flows on the campus were stored. Note that, the payload of most packets are not stored for the consideration of storage space and privacy issues. Before starting the traffic classification engine, several preprocessing stages need to be done in order to trim the traffic trace file into a set of feature vectors of the traffic samples, which are fed into the classifier.

First, for the data set in use, the flow information was also collected during its capturing process. The packets in the data set has already been categorized into more than 760000 traffic flows. The flow information is stored in a .info file with a format shown in Fig. 3. After going through all the flows, we found that more than half of the flows were torrent traffic, so that from the perspective of building a more diverse data set, 440000 torrent flows were removed which finally produced the 59GB .csv file.

flow_id	start_time	end_time	local_ip
remote_ip	local_port	remote_ port	transport_p rotocol
operating_ system	process_ name	urls	content_ types

Fig. 3. Data structure used for flows in the .info file

Secondly, based on the design of our traffic classification framework, "elephant flow" detection/filtering was conducted on all the flows in the data set to extract all elephant flows which were the objective flows in our classification simulation. As a result, there are 3377 flows satifying the requirement.

The next step was to label each flows selected in the previous step. All the flows went through a DPI module which outputs the label of each flow. The labeled flows were then categoried into 4 QoS classes including voice/video conference, interactive data, streaming, and bulk data transfer. Examples of the mapping policy between the application and the QoS class is shown in Table 1. Because it is hard for us to put all the existing applications on our list, there exist a large number of unlabeled flows within all the flows. Moreover, as new applications appear every day, it is not possible to have all flows labeled in a real traffic classification application. In the data set used, there were 1508 unlabeled flows among all 3377 flows. And the data set was further cut into two groups, the training set and the testing set. The ratio between the size of the training set and the size of the testing set is 7.82:1. Note that, to correctly run the testing process, all the flows in

QoS Class	Applications
Voice/Video Conference	Skype, QQ, Google Hangout, ...
Interactive Data	Gaming, Web, Http Services, ...
Streaming	PPStream, Vimeo, SopCast, Putlocker, ...
Bulk Data Transfer	FTP, Torrent, Dropbox, ...

TABLE I
EXAMPLE OF QOS CLASSES AND CORRESPONDING APPLICATIONS

Total Flows		Training Set		Testing Set			
L	U	L	U	L			U
1869	1508	1486	1508	383			0
				C1	C2	C3	C4
				11	29	141	202

TABLE II
THE INGREDIENTS OF THE DATA SET USED

the testing set should be labeled, becasue the unlabeled flows cannot be verfied with an unknown application after going through the classifier.

The fourth step was extracting features from each flow. In the evaluation process, 60 features were extracted from the raw flow data, and as we mentioned before, these 60 features can be categorized into four groups, including time information, packet information, protocol information, and Hurst parameters. However, as a general issue existing in all machine learning based traffic classification research, 60 features are way too many for all being used in the training process of the classifier, because the dimension is too high while the training set is too small as compared with the level of the dimension. This would lead to the severe overfit of the classifier and make the classifier with a bad generalization ability, which means both the accuracy (bias) and the variance would not be good. So that we conducted a feature selection algorithm (i.e., Wrapper) in which the forward selection was employed. Since the Laplacian SVM classifier used has two parameters (i.e.,λ and σ) affect the performance of the classifier a lot, a semi-greedy search on (λ, σ) pairs with two steps (i.e., coarse search and fine search) was used to find the classifier with the best performance. Firstly, in coarse search, the log spaces of the two parameters were used, and the result could provide a small region of (λ, σ) where the accuracy of the classifier is better than elsewhere. Based on our simulation, the area around ($\lambda = 0.00005, \sigma = 0.25$) was expected. Secondly, a fine search among $\lambda = 0.00001 : 0.0001, \sigma = 0.21 : 0.23$ was conducted to search the classifier with the highest accuracy.

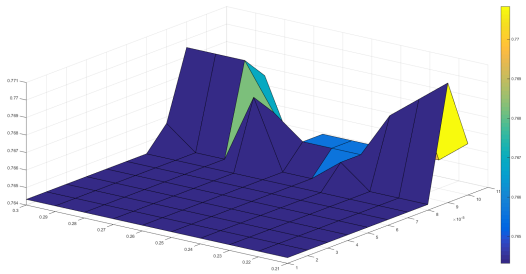

Fig. 4. The accuracy of the classifier with 60 features

Based on the accuracy of the test results on different subset of features, we finally chose a subset of 9 features among all 60 features, considering the complexity of the classification system. The 9 features are listed in the following TABLE III.

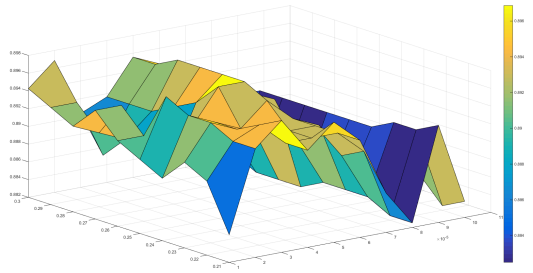

Fig. 5. The accuracy of the classifier with 17 features

Feature	Explaination
HPktLenSD	Entropy of the packet length from Src. to Dst.
dstPort	Dst. port
srcPort	Src. port
HPktLenDS	Entropy of the packet length from Dst. to Src.
avgPktLenDS	Average length of packets from Dst. to Src.
avgPktLenSD	Average length of packets from Src. to Dst.
rspndPkt	Packets to respond from Src. to Dst.
minPktLenDS	Minimum length of packets from Dst. to Src.
pktIntDegree	Packet interactivity degree from Src. to Dst.
medPktLenSD	Median of the packet length from Src. to Dst.

TABLE III
FINAL SUBSET OF FEATURES USED

The comparison between the performance of our classifier and the existing K-means algorithm based classifier in terms of testing accuracy is shown in Fig 6. As we can see, our TC framework using Laplacian SVM as the semi-supervised machine learning algorithm outperforms the previous semi-supervised machine learning scheme using K-means algorithm [12]. In addition, it can bee seen that when the number of features selected into the subset reaches 7-9, the test accuracy exceeds 90% which is an acceptable value in the traffic classification area.

V. CONCLUSION

A QoS-aware traffic classification framework for SDN is proposed in this work. Using this framework, traffic flows could be categorized into different QoS classes. The QoS parameters inferred may be used to re-route efficiently "elephant" flows to meet the resource utilization goals. Semi-supervised machine learning is employed in the QoS classifier to deal with the traffic with unknown applications. Since the feature extraction only uses the first several packets of a flow, the engine runs in a real-time fashion. Moreover the traffic classification framework is adaptive to different kinds of networks by periodically re-training the groud truth database and the QoS classifier.

REFERENCES

[1] Fundation, Open Networking. "Software-defined networking: The new norm for networks." ONF White Paper (2012).

[2] Akyildiz, Ian F., et al. "A roadmap for traffic engineering in SDN-OpenFlow networks." Computer Networks 71 (2014): 1-30.

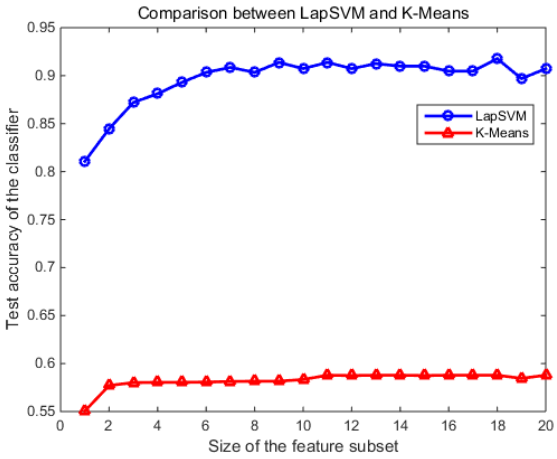

Fig. 6. The accuracy of the classifier with 60 features

[3] S.-C. Lin, P. Wang , and L. Min, "Jointly optimized QoS-aware virtualization and routing in software defined networks," Computer Networks Journal 29 (2016), 69 - 78.

[4] http://www.iana.org/assignments/service-names-port-numbers/service-names-port-numbers.xhtml

[5] Moore, Andrew W., and Konstantina Papagiannaki. "Toward the accurate identification of network applications." Passive and Active Network Measurement. Springer Berlin Heidelberg, 2005. 41-54.

[6] Bishop, Christopher M. Pattern recognition and machine learning. Vol. 4. No. 4. New York: springer, 2006.

[7] Roughan, Matthew, et al. "Class-of-service mapping for QoS: a statistical signature-based approach to IP traffic classification." Proceedings of the 4th ACM SIGCOMM conference on Internet measurement. ACM, 2004.

[8] Moore, Andrew W., and Denis Zuev. "Internet traffic classification using bayesian analysis techniques." ACM SIGMETRICS Performance Evaluation Review. Vol. 33. No. 1. ACM, 2005.

[9] Nguyen, Thuy TT, and Grenville J. Armitage. "Training on multiple sub-flows to optimise the use of Machine Learning classifiers in real-world IP networks." LCN. 2006.

[10] McGregor, Anthony, et al. "Flow clustering using machine learning techniques." Passive and Active Network Measurement. Springer Berlin Heidelberg, 2004. 205-214.

[11] Erman, Jeffrey, Martin Arlitt, and Anirban Mahanti. "Traffic classification using clustering algorithms." Proceedings of the 2006 SIGCOMM workshop on Mining network data. ACM, 2006.

[12] Erman, Jeffrey, et al. "Offline/realtime traffic classification using semi-supervised learning." Performance Evaluation 64.9 (2007): 1194-1213.

[13] Li, Sanping, Eric Murray, and Yan Luo. "Programmable network traffic classification with OpenFlow extensions." Network Innovation through OpenFlow and SDN: Principles and Design (2014): 269.

[14] Lockwood, John W., et al. "NetFPGA–An Open Platform for Gigabit-Rate Network Switching and Routing." Microelectronic Systems Education, 2007. MSE'07. IEEE International Conference on. IEEE, 2007.

[15] Farhadi, Hamid, and Akihiro Nakao. "Rethinking Flow Classification in SDN." Cloud Engineering (IC2E), 2014 IEEE International Conference on. IEEE, 2014.

[16] Duda, Richard O., Peter E. Hart, and David G. Stork. Pattern classification. John Wiley & Sons,, 1999.

[17] Camps-Valls, Gustavo, T. Bandos Marsheva, and Dengyong Zhou. "Semi-supervised graph-based hyperspectral image classification." Geoscience and Remote Sensing, IEEE Transactions on 45.10 (2007): 3044-3054.

[18] Belkin, Mikhail, Partha Niyogi, and Vikas Sindhwani. "Manifold regularization: A geometric framework for learning from labeled and unlabeled examples." The Journal of Machine Learning Research 7 (2006): 2399-2434.

[19] Scholkopf, Bernhard, and Alexander J. Smola. Learning with kernels: support vector machines, regularization, optimization, and beyond. MIT press, 2002.

[20] Melacci, Stefano, and Mikhail Belkin. "Laplacian support vector machines trained in the primal." The Journal of Machine Learning Research 12 (2011): 1149-1184.