Network Traffic Classification using Machine Learning Techniques over Software Defined Networks

Mohammad Reza Parsaei, Mohammad Javad Sobouti, Seyed Raouf khayami, Reza Javidan
Department of Computer Engineering and Information Technology
Shiraz University of Technology (SUTECH), Shiraz, Iran

***Abstract*—Nowadays Internet does not provide an exchange of information between applications and networks, which may results in poor application performance. Concepts such as application-aware networking or network-aware application programming try to overcome these limitations. The introduction of Software-Defined Networking (SDN) opens a path towards the realization of an enhanced interaction between networks and applications. SDN is an innovative and programmable networking architecture, representing the direction of the future network evolution. Accurate traffic classification over SDN is of fundamental importance to numerous other network activities, from security monitoring to accounting, and from Quality of Service (QoS) to providing operators with useful forecasts for long-term provisioning. In this paper, four variants of Neural Network estimator are used to categorize traffic by application. The proposed method is evaluated in the four scenarios: feedforward; Multilayer Perceptron (MLP); NARX (Levenberg-Marquardt) and NARX (Naïve Bayes). These scenarios respectively provide accuracy of 95.6%, 97%, 97% and 97.6%.**

Keywords—Software defined networks; openflow; traffic classification; neural network; multilayer perceptron

I. INTRODUCTION

SDN is a new paradigm in telecommunications and computer networks. The main goal of SDN is to meet challenges existing in IP-based networks, such as complex management. In today's networks, administrators must apply many overwhelming changes to the network configurations in case of a little change in network policies, rules or topology, testing new protocols, to have a dynamic network management [1]-[4]. SDN as a comprehensive concept separates data plan (which is responsible for forwarding data packets) and control plan (which is responsible for routing, traffic engineering, and management policies) to confront limitations and challenges of today's networking [5]-[8]. Fig. 1 shows SDN architecture. OpenFlow (OF) protocol is one of the most important and practical communication protocols, which enables controller to interact with the network switches. This protocol is a standard interface that is used mostly in SDN. OF switches contain one or multiple flow tables with flow entries. Each entry consists of paired rules and actions. The tables are filled by the controller. Each rule consists of fields related to headers data, such as source and destination MAC and IP addresses, port numbers and other necessary information. Each action determines instruction(s) to be executed on the packet in order to match the entry's rule [9]-[10].

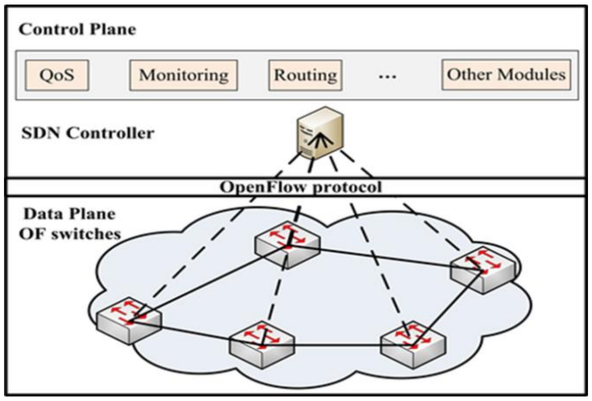

Fig. 1. SDN architecture.

Separation of data plan and control plan, gives ability to network administrators to make programmable policies and easily manage data plan via the controller [11]. SDN also makes it easy to have a dynamic management, configuration, troubleshooting and even testing new protocols and ideas in the network without troubles [12]. An important case in network management for having high availability and efficiency is traffic classification. There are methods for applying traffic classification in networks [13]:

- Using port numbers to determine application and application layer protocols. However, these methods are not completely accurate.
- Deep Packet Inspection (DPI) is used. These methods have high accuracy, however, there are some issues regarding its implementation while dynamic ports and encrypted traffics are not supported in current networks yet. It also causes high overhead to the system and violates user privacy.

These methods have their own problems, so, researches have been recently focusing on machine learning techniques, which take advantage of statistical properties for traffic classification.

Although there are many challenges in current networks for traffic classification, global view of controllers in SDN improves network management while its concept is simple and easy to use for extracting statistical data of network traffic from switches [14].

Hanigan et al. [15] used traffic classification methods based on DPI to inspect flows over SDN to distinguish application protocols in runtime. Their goal is to enable controller to distinguish and isolate different application flows, managing and programming flows to guaranty QoS for delay sensitive applications. When the network is loaded, a great part of the controllers' processing resources must be dedicated to DPI tools. Thus, the performance of the entire network is affected.

Arsalan et al. [16] proposed a framework to determine the application type of existing flows in a wireless network, which consist of several mobile devices connected to an OF switch. In control plan, a machine learning-based trainer receives the information. On the other hand, the OF switch gathers the properties of different flows and sends them to the control layer for creating a model for application layer recognition. After model creation, when a host joins the network, the OF switch sends the device flows properties to the traffic classification model, based on the machine learning technique. Then, the application of source flow is determined. The traffic classification model is based on C5.0 decision tree algorithm.

Jang et al. [17] proposed a method that uses flows properties, gathered in a dataset, as K-means algorithm input, in learning phase, for clustering. These clusters are used to implement a traffic classification model. The clusters with similar features are aggregated based on the information obtained from the content of the packets. Although the accuracy rate of this method is 89 percent, it reduces the need of investigating the packet contents and accurate diagnosis of encrypted packets. Most researches present traffic classification on a set of statistics from stored flows in an offline manner. The high time complexity and processing overhead are two challenges of online traffic classification. Current approaches also cause an overwhelming overhead on the system. The goal of this paper is to classify the traffic over SDN using information in the header of packets received from OF switches and statistics in the controller. By considering protocol capabilities on extracting flows statistics and neural networks variant such as feedforward, MLP, NARX (Levenberg-Marquardt) and NARX (Naïve Bayes), a framework for online traffic classification based on application layer protocol is proposed. The overall accuracy of this model in traffic classification for feed-forward, MLP, NARX (Levenberg-Marquardt) and NARX (Naïve Bayes) algorithms with value of 95.6%, 97%, 97% and 97.6%, respectively.

The highest accuracy of the previous methods was 94% [18] but the accuracy of the proposed method is 97.6%. Advantages of this method over current methods are low processing overhead, low network overhead and low runtime execution. The following sections of this paper are as follows: Section 2 presents the proposed method for online traffic classification in SDN. Section 3 provides the implementation and performance evaluation of proposed method. Finally, discuss and analyze the results in Section 4.

I. The Proposed Method

In this paper, a new method is used for classification of network traffic proportional to SDN architecture.

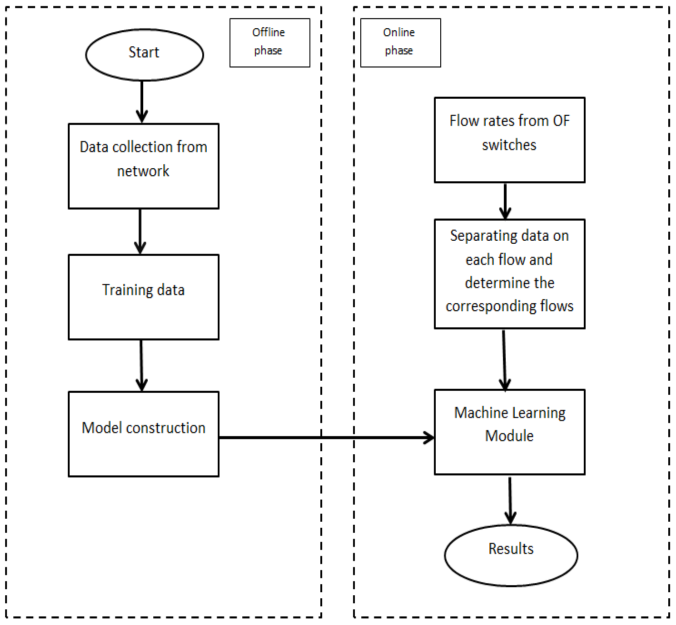
Offline phase
Start
Data collection from network
Training data
Model construction
Online phase
Flow rates from OF switches
Separating data on each flow and determine the corresponding flows
Machine Learning Module
Results

Fig. 2. The outline of the proposed method.

In such networks, all switches are connected to a central controller that may be lower cost than current networks. The protocol of each flow can be identified by classification of traffic based on application layer in the level of control. Fig. 2 shows the outline of the proposed approach.

The proposed method consists of offline and online phases that are as follows:

A. *Offline phase*

In offline phase, making data collection and model classifier are discussed. This means that floodlight uses a web-based graphical interface to connect users to the controller. We use below URI to receive raw data of the required set of training data to get statistics on all existing traffic flows in the switches:

http://localhost:8080/wm/core/switch/All/flow/Json.

Information is extracted for each flow by the mentioned URI. The structure of API data exchange is JSON. They are identifiable after calling up information in the browser. The flows are separated for using of massive data that obtained from all flows in switches.

For this purpose used five attribute (IP source, IP destination, the source port, the destination port, transport layer protocol), which specifies a unique flow. In the next step, after separating flows from each other, associated return flows with them that are constituent a 2-way flow are combined and done pre-processing on them for making a sample (a raw of sets of training data). Eventually, each row of the training data set that represents a two-way flow, contains the characteristics (see Table 1). Last item the APPpro (application layer protocol) represents a class variable in the training data set that is completed manually.

TABLE I. CHARACTERISTICS ASSOCIATED WITH EACH FLOW IN TRAINING DATA SET

Attribute	Description
srcIp	Source IP
dstIp	Destination IP
SrcPort	Transport Layer port in source
DstPort	Transport Layer port in destination
reqPro	Transport layer protocol , flow
respro	Transport layer protocol , backflow
reqAvgSz	The average size of the flow pockets
resAvgSz	The average size of the backflow pockets
reqPktperSec	Mean number of pockets per second on the flow
resPktperSec	Mean number of pockets per second on the backflow
reqBytperSec	Mean number of bytes per second on the flow
resBytperSec	Mean number of bytes per second on the backflow
APPpro	Application layer protocol

A classification model is used after classification algorithm on training data set, which is used to create traffic classification module [19-21] for floodlight controller.

B. *Online phases*

In online phase, ML module that is added to floodlight controller classifies the network traffic operation by the help of developed model in offline phase. This means that received statics flows in the switch and the application of layer protocol obtains each of them. The results obtained this module and management of bandwidth, security and management issues in order to supply QOS goals provided for javaAPI and restAPI. They order both of them are used to communicate with other modules and applications. ML module evaluated in the four scenarios which following algorithm is used:

1) *Feed-forward Neural Network*

In feed-forward neural network, connections between units do not form a cycle so it is different from recurrent network. Feed-forward network is the first and most simple type of neural network algorithms. The flow of information always moves forward from input to output. A supervised technique called backpropagation is used to improve its performance. It propagates backward from output to input in the network, decreases errors and optimizes performance by correcting the weight of edges, connected to nodes. The weights can be corrected by Gradient Descent method using (1) for calculating the change of each edge's weight.

$$\Delta W_i \; = \; -\eta \; \partial E / \partial W_i \qquad (1)$$

In this equation ɳ is the learning rate which its value is considered equal to 0.1. Also, the expected value is obtained from (2), in which   is the target value and   is the perceptron's output.

$$\begin{aligned} &Expected\ value \ = \partial / \partial w_i \ 1/2 \ S_d \left(t_d - o_d\right)^2 \\ &= \partial / \partial w_i \ 1/2 \ S_d (t_d - S_i w_i x_i)^2 \qquad (2) \\ &= S_d \left(t_d - \ o_d\right) \ (-x_i) \end{aligned}$$

2) *Multilayer Perceptron (MLP)*

MLP is a type of neural network, which maps a set of input data to one or more output, based on learning from previous samples. Because of its strong nonlinear approximation behavior, MLP is the most useful model in neural networks and is used almost in every scientific field. An MLP is consisting of multi layers of nodes in a directive graph, in which each layer is fully connected to the next layer. MLP also uses backpropagation for training network. MLP is the modified version of perceptron and can recognize nonlinear data [22]. Using gradient descent, find changes in each weight according to (3), where is the output of the previous neuron and is calculated by (4):

$$\Delta w_{ji}(n) = -\eta \frac{\partial \varepsilon(n)}{\partial v_j(n)} y_i(n) \qquad (3)$$

$$y(v_i) = \tanh(v_i) \; and \; y(v_i) = (1+e^{-v_i})^{-1} \qquad (4)$$

Here V_i is the weighted sum of the input synapses. $\varepsilon(n)$ in (3) is calculated through (5) and the error in output node *j* in the *nth* training example is calculated by (6), where *d* is the target value and *y* is the value produced by the perceptron:

$$\varepsilon(n) = \frac{1}{2}\sum_j e_j^2(n) \qquad (5)$$

$$e_j(n) = d_i(n) - y_j(n) \qquad (6)$$

3) *Non-linear Autoregressive Exogenous Multilayer Perceptron (NARX)*

The NARX model is generally used for predicting time series by approximation of nonlinear relationships between exogenous variables and the predictor variable, as defined in (7):

$$\begin{aligned} &y(t) = f(x(t-1), x(t-2)\ldots x(t-d); \\ &y(t-1), y(t-2)\ldots y(t-d) \end{aligned} \qquad (7)$$

y(t) is the predictor variable and *x(t)* denotes the Exogenous time series. Usually function *f* is a nonlinear polynomial. For modeling function *f* in NARX, it is also possible to create a dynamic MLP network by assuming in time *t*, *d* is the previous variable of predictor variable and the predictor variable should be accessible. This configuration is based on delay and is without feedback, called open loop. This model is used for one-step-ahead predictions, because the estimation is based on previous knowledge of real past values for target series and is not based on prediction that produces errors in results. In this model two functions are used for learning:

- Levenberg-Marquardt Algorithm

One of the learning functions of this method is retrieved from Levenberg-Marquardt Algorithm. This algorithm is a way to find the minimum of a nonlinear polynomial function and is a standard method for solving Minimum Square of nonlinear function problem. This algorithm is used to minimize square curve fitness problem. The β parameter in the curve model of $f(x,\beta)$ is from experimental data set of dependent and independent variables $\left(x_i, y_i\right)$, which sum of their square of derivation is minimum, as shown in (8):

$$\hat{\beta} = \arg\min S\left(\beta\right) \equiv \arg\min \sum_{i=1}^{m}\left[y_i - f\left(x_i, \beta\right)\right]^2 \quad (8)$$

In each step of iteration, the β parameter vector is replaced with a new approximate value of $\beta + \delta$. For δ calculation, the functions *f* are estimated by linearization as in (9). In this equation J_i is the gradient of function *f* with respect to β which calculated as in (10):

$$f(\mathrm{x}_i, \beta + \delta) \approx f(x_i, \beta) + J_i\delta \qquad (9)$$

$$J_i = \frac{\partial f(X_i, \beta)}{\partial \beta} \qquad (10)$$

The sum of squares $S(\beta)$ at its minimum has a zero gradient with respect to β. Equation (11) shows the first approximation of $f(x_i, \beta + \delta)$ and its vector notations are in (12):

$$S(\beta + \delta) \approx \sum_{i=1}^{m}(y_i - f(x_i, \beta) - J_i\delta)^2 \qquad (11)$$

$$\begin{aligned} &S(\beta + \delta) \approx \| y - \mathrm{f}(\beta) - J\delta \|^2 = \\ &[\mathrm{y} - \mathrm{f}(\beta) - \mathrm{J}\,\delta]^T[\mathrm{y} - \mathrm{f}(\beta) - \mathrm{J}\,\delta] = \\ &[\mathrm{y} - \mathrm{f}(\beta)]^T[\mathrm{y} - \mathrm{f}(\beta)] - \\ &[\mathrm{y} - \mathrm{f}(\beta)]^T J\delta - (\mathrm{J}\,\delta)^T[\mathrm{y} - \mathrm{f}(\beta)] + \\ &\delta^T J^T J\delta = [\mathrm{y} - \mathrm{f}(\beta)]^T[\mathrm{y} - \mathrm{f}(\beta)] - \\ &2[\mathrm{y} - \mathrm{f}(\beta)]^T J\delta + \delta^T J^T J\delta. \end{aligned} \qquad (12)$$

Equation (13) is obtained from setting the derivation of $S\left(\beta + \delta\right)$ to zero:

$$(J^T J)\delta = J^T[y - f(\beta)] \qquad (13)$$

In this equation, J_i is the *ith* row of the Jacobian matrix and Y_i and $f\left(x_i, \beta\right)$ are the *ith* component of *Y* and *f* vectors respectively. Levenberg's contribution is to replace this equation by a "damped version" as show in (14):

$$(J^T J + \lambda I)\delta = J^T[y - f(\beta)] \qquad (14)$$

Besides, Marquardt replaced the identity matrix "I" with diagonal matrix consist of diagonal elements of $\mathrm{J}^T\mathrm{J}$, resulting in Levenberg–Marquardt algorithm as in (15):

$$[J^T J + \lambda diag(J^T J)]\delta = J^T[y - f(\beta)] \qquad (15)$$

- Naïve Bayes

Another learning function is from Naïve Bayes classifiers family, which do classifies using Bays probability theorem. Naïve Bayes is a common technique for creating classifiers. One of its positive points is its efficiency in solving probability problems. It is also very flexible and needs only a few learning data for estimating the required parameters [23]. The mathematical form of Bayes theorem is as (16):

$$p(C_k \mid x) = \frac{p(C_k)\,p(x \mid C_k)}{p(x)} \qquad (16)$$

II. PERFORMANCE EVALUATION

A SDN network with one software switch, one controller and two hosts, is configured for our experiments. In order to implement software defined networks, OVS (open v Switch) was considered as OF switch and Floodlight was considered as the controller. Open v Switch is an open source; apache 2.0-licensed virtual switch which is developed under Linux kernel. Floodlight is an open source, apache licensed controller, which developed by using Java platform. A topology as shown in Fig. 3 is set for performance evaluation.

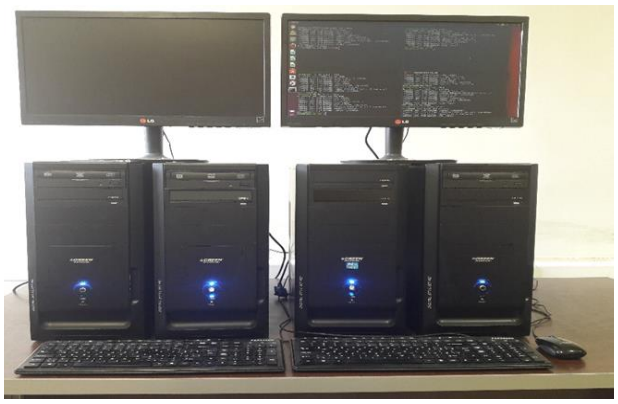
Fig. 3. Scenario topology which evaluated in the laboratory.

TABLE II. SYSTEM DESCRIPTION

Controller and hosts	
CPU	Core i7
RAM	16 GB
OS	Ubuntu 14.10
Controller version	Floodlight v 9.0
Switch	
CPU	Core i5
RAM	8 GB
OS	Ubuntu 14.10
OVS version	2.0.1

There are four systems in this figure, from left to right: the first system is an OF switch (it is converted to OF switch by installing ovs on it). The second and third systems are the first and second clients. The last system is the controller. The systems' configurations are shown in Table 2.

After setting up the topology, the host stablishes FTP and HTTP connections for downloading files and video streams, text messaging, peer to peer connection with BitTorrent client, downloading and uploading file parts to/from other hosts simultaneously and aggregates statistics related to these flows.

So, the case study protocols for this scenario are FTP, HTTP, instant messaging, video streaming and peer to peer protocol. Sampling of the flows is done 20 times in 1 second intervals.

In all scenarios, sampling from switch statistics (extracting required data from switch's flow tables) is done 6000 times to collect about 65000 data records. After data aggregation and preprocessing of flows statistics, a data set with 600 records is obtained.

After that the different flow class variable is manually set in the learning data set, and finally the data is converted to CSV format, for creating traffic classification model. Final Model is converted to java code for using as a traffic classifier module in online phase. Results in details are provided in the Fig. 4 to 7.

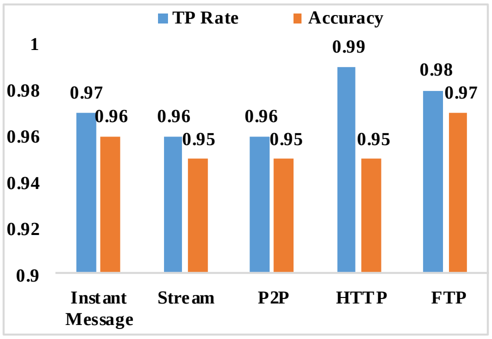

Fig. 4. The result of the Feedforward algorithm.

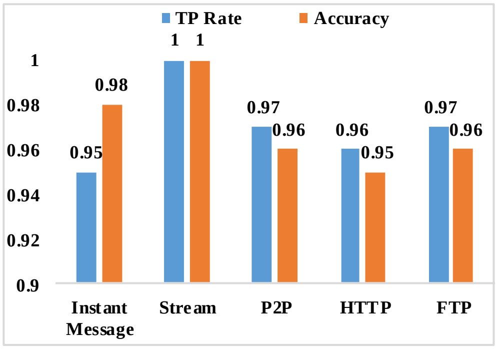

Fig. 5. The result of the MLP algorithm.

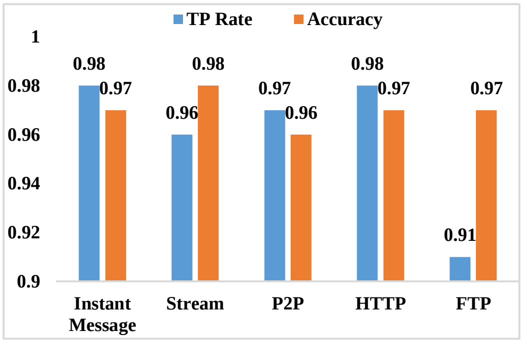

Fig. 6. The result of the NARX (Levenberg-Marquardt) algorithm.

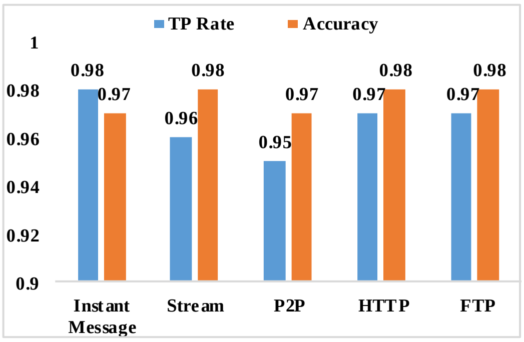

Fig. 7. The result of the NARX (Naïve Bayes) algorithm.

III. CONCLUSION

The proposed method in this paper is used for application recognition of flows resources with the help of SDN and data mining techniques based on machine learning. Applying traffic classification techniques to the OF network makes the network be application-aware, and enables the network to know flow's requirements. Due to maintaining a global view of the network, the controller could dynamically allocate bandwidth to flows on demand and thus improve their QoS and the analysis and prediction of traffic patterns in network make the controller further optimize resource allocation. This method mainly focused on minimizing controllers' processing overhead and network traffic overhead for network traffic classification. The accuracy of tested class variables in our experiments for feedforward, MLP, NARX (Levenberg-Marquardt) and NARX (Naïve Bayes) algorithms are 95.6%, 97%, 97% and 97.6% respectively. The highest accuracy of the previous methods was 94% but the accuracy of the proposed method is 97.6%. Also, the proposed method does not impose any processing overhead to the controller because unlike the base method, packets' contents are not checked. Our on-going and future works include implementations on different device platforms (iOS, Windows, Linux) and detection of flows belonging to a new application which is not part of the trained classifier.

REFERENCES

[1] C. Trois, M. D. Del Fabro, L. C. de Bona, M. Martinello, "A survey on SDN programming languages: toward a taxonomy," IEEE Communications Surveys and Tutorials, vol. 18, no. 4, pp. 2687-2712, 2016.

[2] B. A. A. Nunes, M. Mendonca, X. N. Nguyen, K. Obraczka, T. Turletti, "A survey of software-defined networking: Past, present, and future of programmable networks," IEEE Communications Surveys and Tutorials, vol. 16, no. 3, pp. 1617-1634, 2014.

[3] T. Huang, F. R. Yu, C. Zhang, J. Liu, J. Zhang, J. Liu, "A Survey on Large-scale Software Defined Networking (SDN) Testbeds: Approaches and Challenges," IEEE Communications Surveys & Tutorials, vol. 19, no. 2, pp. 891-917, 2016.

[4] T. Benson, A. Akella, D. A. Maltz, "Unraveling the Complexity of Network Management," In NSDI, pp. 335-348, 2009.

[5] S. Rowshanrad, M. R. Parsaei, M. Keshtgari, "IMPLEMENTING NDN USING SDN: A REVIEW ON METHODS AND APPLICATIONS," IIUM Engineering Journal, vol. 17, no. 2, pp. 11-20, 2016.

[6] M. R. Parsaei, R. Javidan, A. Fatemifar, S. Einavipour, "Providing Multimedia QoS Methods over Software Defined Networks: A Comprehensive Review," International Journal of Computer Applications, vol. 168, no. 9, pp. 55-59, 2017.

[7] A. Mendiola, J. Astorga, E. Jacob, M. Higuero, "A survey on the contributions of Software-Defined Networking to Traffic Engineering," IEEE Communications Surveys & Tutorials, vol. 19, no. 2, pp. 918-953, 2017.

[8] M. Karakus, A. Durresi, "Quality of Service (QoS) in Software Defined Networking (SDN): A survey," Journal of Network and Computer Applications, vol. 80, pp. 200-218, 2016.

[9] M. Dusi, R. Bifulco, F. Gringoli, F. Schneider, "Reactive logic in software-defined networking: Measuring flow-table requirements," IEEE International Conference on Wireless Communications and Mobile Computing (IWCMC), pp. 340-345, 2014.

[10] F. Hu, Q. Hao, K. Bao, "A survey on software-defined network and openflow: From concept to implementation," IEEE Communications Surveys & Tutorials, vol. 16, no. 4, pp. 2181-2206, 2014.

[11] H. Kim, N. Feamster, N, "Improving network management with software defined networking," IEEE Communications Magazine, vol. 51, no. 2, pp. 114-119, 2013.

[12] M. R. Parsaei, S. H. Khalilian, R. Javidan, "A Comparative Study on Fault Tolerance Methods in IP Networks versus Software Defined Networks," International Academic Journal of Science and Engineering. Vol. 3, no. 4, pp. 146-154, 2016.

[13] T. J. Parvat, P. Chandra, "A Novel approach to deep packet inspection for intrusion detection," Procedia Computer Science, vol. 45, pp. 506-513, 2015.

[14] N. Williams, S. Zander, G. Armitage, "A preliminary performance comparison of five machine learning algorithms for practical IP traffic flow classification," ACM SIGCOMM Computer Communication Review, vol. 36, no. 5, pp. 5-16, 2006.

[15] H. Cui, Y. Zhu, Y. Yao, L. Yufeng, Y. Liu, "Design of intelligent capabilities in SDN," IEEE International Conference on Wireless Communications, Vehicular Technology, Information Theory and Aerospace & Electronic Systems (VITAE), pp. 1-5, 2014.

[16] Z. Arslan, A. Alemdaroglu, B. Canberk, "A traffic-aware controller design for next generation software defined networks," IEEE International Conference on Communications and Networking (BlackSeaCom), pp. 167-171, 2013.

[17] J. Zhang, Y. Xiang, W. Zhou, Y. Wang, "Unsupervised traffic classification using flow statistical properties and IP packet payload," Journal of Computer and System Sciences, vol. 79, no. 5, pp. 573-585, 2013.

[18] Z. A. Qazi, J. Lee, T. Jin, G. Bellala, M. Arndt, G. Noubir, "Application-awareness in SDN," ACM SIGCOMM computer communication review, vol. 43, no. 4, pp. 487-488, 2013.

[19] M. R. Parsaei, S. M. Rostami, R. Javidan, "A Hybrid Data Mining Approach for Intrusion Detection on Imbalanced NSL-KDD Dataset," International Journal of Advanced Computer Science and Applications, vol. 7, no. 6, pp. 20-25, 2016.

[20] S. S. Parsa, M. Sourizaei, M. M. Dehshibi, R. E. Shateri, M. R. Parsaei, "Coarse-grained correspondence-based ancient Sasanian coin classification by fusion of local features and sparse representation-based classifier," Multimedia Tools and Applications, vol. 76, no. 14, pp. 15535-15560, 2017.

[21] A. Nabaei, M. Hamian, M. R. Parsaei, R. Safdari, T. Samad-Soltani, H. Zarrabi, A. Ghassemi, "Topologies and performance of intelligent algorithms: a comprehensive review," Artificial Intelligence Review, pp. 1-25, 2016. doi:10.1007/s10462-016-9517-3.

[22] H. Komijani, S. Rezaeihassanabadi, M. R. Parsaei, S. Maleki, "Radial Basis Function Neural Network for Electrochemical Impedance Prediction at Presence of Corrosion Inhibitor," Periodica Polytechnica Chemical Engineering, vol. 61, no. 2, pp. 128-132, 2017.

[23] M. R. Parsaei, R. Taheri, R. Javidan, "Perusing the effect of discretization of data on accuracy of predicting Naïve Bayes algorithm," Journal of Current Research in Science, (1), pp. 457-462, 2016.