OPTIMIZATION OF ROUTING USING TRAFFIC CLASSIFICATION IN SOFTWARE DEFINED NETWORKING

Vikas Verma[1,*] and Manish Jain[2]

Received: August 27, 2021; Revised: July 28, 2022; Accepted: September 27, 2022

Abstract

Efficient routing is an essential task for any network as it directly impacts the network's performance. In this paper, we have used traffic classification techniques to optimize routing in Software Defined Network (SDN) and provided a cost aware routing framework. Instead of classifying traffic based on a particular parameter, we perform the traffic classification using a hybrid approach that is based on machine learning techniques. Our framework uses supervised machine learning techniques to classify network flow and detect elephant flows which are too heavy in size and require significant bandwidth. As the requirements of each elephant flow vary with the application, we further perform Quality of Service (QoS) based traffic classification, which classifies these elephant flow into QoS classes as per their QoS requirements. For this purpose, we have used semi-supervised machine learning algorithms. Further, we have proposed a routing algorithm that makes use of the Dijkstra algorithm to compute the best possible shortest path based on the QoS requirements of the elephant flow. The proposed method was implemented and tested on Mininet using a POX controller. The simulation results show that our framework successfully classifies elephant flows with an accuracy of 80% and also computes a low-cost path for each elephant flow based on the actual internet data set.

Keywords: Software Defined Networking, Quality of service, POX controller, QoS classes

Introduction

Software Defined Networking (SDN) (ONF *et al*., 2012) has gained much exposure in recent years as it decouples the control plane from the data plane. Machine Learning algorithms can be easily implemented in SDN due to their programming nature. However, the same is not possible in classical networks. In modern networks, there is a tremendous growth of applications that generate a massive amount of data for different QoS requirements. These QoS requirements need to be fulfilled in order to increase the network's performance. Traffic classification plays a vital role while managing network resources (Xie *et al*., 2019). It needs to be considered while applying different mechanisms to meet the QoS and minimize the operational cost of the network. There are several techniques available for traffic classification in SDN. However, machine learning algorithms provide a good result with a low computation cost. Traffic classification can be done based on the

JECRC University, Jagatpura, Jaipur India, 303905, E-mail: vikaverma1988@gmail.com; halomanish@gmail.com
** Corresponding Author*

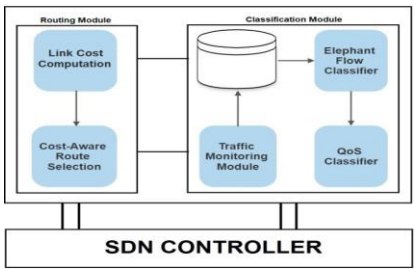

Figure 1. Proposed framework and its modules

size and bandwidth requirements of the flow, QoS requirement of the flow, the application's types of flow, etc.

Machine learning algorithms are used to determine the pattern from network data as these learn from the training dataset. The SDN platform has the ability to include intelligence in the network devices. Optimization, management, and maintenance of the network devices can be quickly addressed using a machine learning algorithm in SDN. Efficient routing is an essential function for any network. In SDN, the controller decides the routing path for each flow by a flow table for every switch. Inefficient routing decisions can increase network link latency and increase bandwidth consumption, which affects the overall performance of SDN (Patrushev and Drozdova, 2017).

To address the various issues for routing in SDN, we have designed a framework that provides a cost-aware routing for each elephant flow in an SDN based on its QoS needs. We use a hybrid approach for traffic classification that is a combination of machine learning algorithms (supervised and semi-supervised). We classify the flow into steps once based on its size and bandwidth requirement, which is done by a supervised machine learning algorithm, and the second time for the QoS requirement of the flow; this classification is done by using a semi-supervised machine learning algorithm. Further, we have proposed a routing algorithm to compute the best possible shortest path on the basis of QoS requirements for each elephant flow. In this framework, we only deal with the elephant flows that need significant bandwidth for transmission and have a large packet size, for which we want to decide on a low-cost QoS path to reduce the cost affecting the overall network's QoS parameters. Each elephant flow is further divided into one of the four QoS classes as per QoS requirements, which are defined in the latter sections. At the same time, mice flow that is short-lived and light in size is directly forwarded to the switch by the controller. By analysis of the results, we observed that using our framework low-cost QoS path can be achieved for elephant flow which decreases the overall network operational cost and the network throughput gets increased.

Wnag *et al*. (2016) designed a framework for QoS-aware traffic classification in SDN using a semisupervised machine learning technique. Wang used deep packet inspection (DPI) for identifying QoS- significant elephant flows. The author labelled each flow into the QoS class based on its QoS requirements. The test accuracy had exceeded up to 90% in his case. However, in current work, flows have been classified on the basis of their size and their bandwidth requirements and further classified into QoS classes.

Pasca *et al*. (2017) used machine learning techniques to classify flow based on its application. The classifier provided 90% accuracy. Pasca used the Yen-K-shortest path algorithm for finding low-cost shortest paths for each flow. Here, we have used QoS classification for only elephant flow and routed these flows by calculating the QoS-based cost for each possible shortest path.

Yahyaoui *et al*. (2020) proposed a novel class-based routing called LUNA that was able to minimize the flow completion time. The author classified the flow into elephant flow and mice flow based on their size. Association rules were used to generate the forwarding rules and route each flow based on its class (i.e., mouse or elephant). While in this work, we decide the route for each flow by calculating QoS based cost of network links.

Azzouni *et al*. (2017) proposed NeuRoute, a machine learning-based dynamic routing framework for SDN. They implement NeuRoute as a routing application on Pox Controller's head. NeuRoute predicts traffic matrix in real-time, uses a neural network to learn traffic characteristics, and generates forwarding rules accordingly. S.

Lin *et al.* (2016) proposed QoS-aware adaptive routing (QAR) for multi-layer hierarchical SDN. Lin used reinforcement learning for their approach. QAR was proposed to enable adaptive, time-efficient and QoS aware packet forwarding upon the proposed architecture. However, in our work we have use QoS as a parameter for each link's cost calculation as per the requirement of flow. Here, we generate a forwarding rule for only heavy traffic flow in the network, which could be done using traffic classification in our approach.

Materials and Methods

In this section, we have described our framework for SDN controllers. As shown in Figure 1, it includes two main modules: the Classification module, which classifies elephant flow into Quality of Service (QoS) classes based on its QoS requirements. The second

module is the Routing module responsible for the best route selection based on link cost.

In the Classification Module, we use machine learning algorithms to classify the traffic flows into elephant and mice flows. Elephant flows are long-lasting and require larger bandwidth for transmission. However, mice flows are in the network for a short period, and they are also delay-intolerant (Xie *et al*., 2019). From previous studies, it has been observed that approx 80% of the traffic flow in any data center are mice flows (Akella and Maltz, 2010). We further classify each elephant flow based on its Quality of Service (QoS) requirements. Each flow needs to be classified as one of the four QoS classes based on its QoS needs like bandwidth, delay, jitters, etc. In the Routing Module of our proposed framework, we provide an algorithm to identify a cost-aware routing for each elephant flow based on its QoS class. The different modules of the framework and their functioning are discussed below:

Traffic Monitoring Module

A large amount of traffic is generated in the SDN controller. We pass this traffic information to the Traffic MonitoringModule that stores the data in terms of features like source destination IP, rate, time a traffic demand arrives, requestor response IP, user IP, size of the response bytes, etc. The traffic monitoring module is meant to collect all information about the traffic available in the network infrastructure. It also keeps a record of the network's historical data like topology, events, incidents, network users, and corresponding application profiles.

Elephant Flow Classifier

A large number of traffic classification schemes have been discussed in the related work section. However, Machine Learning (ML) techniques give better results for recognizing traffic with lower computational costs than other approaches. Therefore, ML-based methods have been extensively studied by various researchers.

Traffic flows are first collected by the Traffic Monitoring module and stored in user statistics. Elephant flow classifier of the Classification Module classifies the elephant flow. For this purpose, we have used a supervised machine learning technique, Logistic Regression, (Hosmer *et al*., 2013) to identify the elephant flow. Elephant flows are bandwidth-hungry and long-lived; while mice flows are short-lived and light in size, they do not need large bandwidth for transmission. Therefore, we find out the routing path only for elephant flow to reduce the cost affecting the overall network's QoS parameters. Elephant flows are further sent to the next classifier. At the same time, mice flow is directly forwarded to the switch by the controller.

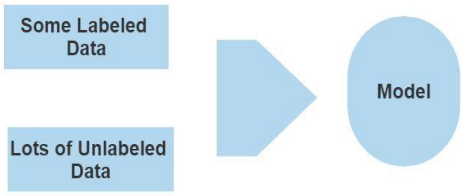

Figure 2. A semi-supervised ML approach

QoS Classifier

Our next Module of classification is QoS Classifier. It is a trained classifier used for QoS-aware traffic classification. This module classifies elephant flow into their QoS category as per their requirements, e.g., bandwidth requirements, acceptable delay, and priority of the flow. We define the QoS class for each elephant flow based on the type of data and its QoS requirements. We can have Text, video, voice, conference, streaming, bulk data, and interactive types of data in flow. However, all kinds of data may not be included here because we are not classifying the mice flows, and it is possible that some specific types of data belong to mice flow only. After filtering the elephant flow using the elephant flow classifier, the corresponding flow is assigned to its respective QoS class. To assemble a continuous classifier, we utilize the first N packets (N=40 in our work) to characterize the QoS classes. Now we have training datasets that contain labelled and unlabelled types of data. We use a semi-supervised machine learning algorithm for this module. Semi-supervised algorithms need only a tiny part of labelled data and give fine-grained results.

It is assumed that the flow that requires the same QoS parameter must have similar statistical features. This is a typical semi-supervised learning assumption called cluster assumption. A semi-supervised machine learning algorithm uses both types of data, labeled and unlabelled, for training, as shown in Figure 2. The dataset first contains n rows for labeled data XL = (X_1;X_2;X_3: : : :X_n) with corresponding labels (Y_1; Y_2; Y_3: : : :Y_n). Apart from the same dataset, we have unlabeled data XU = (X_u+1; X_u+2; X_u+3: : : : X_u+m); therefore, the total number of training instances is n+m in the datasets. For our work, Expectation-Maximization (EM) algorithm is adopted. EM can be utilized for the latent variables (a variable that is not directly recognizable and deduced from the other noticed variable) to predict their values. An Expectation Maximization (EM) algorithm is an iterative strategy to discover (local) maximum likelihood. The EM iteration alternates between performing an expectation (E) step, which

makes a capacity for the assumption for the log-likelihood evaluated using the current estimate for the parameters, and a maximization (M) step, which computes parameters maximizing the expected log-likelihood found on the E step (Dieng and Paisley, 2019) These parameter estimates are then used to decide the distribution of the latent variables in the next E step. The regularization parameter for EM is:

$$\lambda_p^{(t)} \leftarrow \lambda_p^{(t-1)} + \Delta \tag{3.1}$$

Where λ_p^((t)) is the regularization parameter k is the contractive factor k<1, Δ estimated statistical error, and λ_p^((t-1)) is the initial regularization parameter. We first train both ML classifiers offline with labeled traffic instances that we received from user statistics. When we provide the features vector to the classifier, the Machine Learning algorithm's coefficients will be changed depending on the distinction between the temporary outcome and the naturally labeled result.

Link cost computation

The link cost computation (LCC) module is used for computing the cost of each link after each unit of time. Link cost can be calculated by measuring each link's latency. LCC module additionally calculates bandwidth available in each link at a particular time when flow arrives. Therefore decisions that a flow can accompany a route or not can be made.

We use the solution proposed by Phemius *et al.* (2013) for the computation of each link's latency in a software-defined network. It works moderately while calculating the latency of single links. It can also be used for full paths or to determine a round-trip delay. As stated by K. Phemius, the latency of a connection can be calculated by calculating the entire trip time (ETT), including the time of packet traveling from the controller to switch one to switch two and to the controller. Single ETT is the controller to switch one and to controller. ETT can be calculated by giving a STATISTICAL_ REQUEST command to switch. Now the switch will respond with a STATISTICAL_ RESPONSE message in time St1. Similarly, from switch 2 one way ETT St2 can be found.

$$Latency\ (s1, s2) = \text{Ttotal} - \left(\frac{S_{t1}}{2}\right) - \left(\frac{S_{t2}}{2}\right) \tag{3.2}$$

Whenever a route is found, its cost is calculated, and its cost depends upon available bandwidth and the latency of the links. Here, latency is the measure of delay that a particular link will cause, and the available bandwidth observed that why the particular link should be chosen between other links. Sometimes, two links have the same latency; in that case, the available bandwidth part will solve one link's preference over the other. A normalized bandwidth of the link ij can be calculated as:

$$\Upsilon_i = \frac{\Upsilon_i}{\text{Max}\ (\Upsilon)} \forall_i \tag{3.3}$$

The link cost of every link (LCij) is given as the cost of the connection between nodes i and j in the network. The cost of a link depends on the link's latency (li) and available bandwidth i of that link.

$$LC_{ij} = \lambda_a * l_i + \lambda_b * \frac{1}{\Upsilon_i} \tag{3.4}$$

Here, li is the link latency between switch a and switch b. λa and λb are appropriate weights given to emphasize the corresponding variable impact on cost. Dijkstra algorithm calculates the shortest route in the network using source and destination paire. The cost of the shortest path P (CP) is the sum of link cost of all links in the path (TP) calculated as:

$$C_P = \sum_i^{T_P} LC_{ij} \tag{3.5}$$

Cost-Aware Route Selection

This module collects the information from the LCC module and decides to route for each category of elephant flow with its QoS requirements. The module calculates network topology and link statistics in every unit of time n by sending Link Layer Discovery Protocol (LLDP) packet. We will update the cost for every link using the Link Cost computation module. Whenever an elephant flow classifier detects a new elephant flow, the controller collects its features, and it is classified in one of the QoS Categories. Dijkstra's algorithm is used to calculate the shortest route from source to destination. It has computational complexity O(V+ElogV).

Algorithm 1. **Algorithm for path Selection**

1. For each unit of time n, send an LLDP packet to compute each link's cost and check the network's topology.
2. For each new elephant, flow finds the class (f s QoS Class).
3. Compute path P ← Dijkstra (Src, Dst).
4. CP ← Compute_Cost (P, QoS class).
5. For i=T to 0 do
6. Pi ← Neighbor_path (Src, Dst).
7. Ci ← path_cost (P_i, QoS Class)
8. if Ci ≤ Cp then
9. P=Pi and update link cost
10. end if
11. End for

Table 1. Shows Experiment Setup Details

Classification	Details
Operating System	Ubuntu 20.04.1
Simulation tool	Mininet 2.2.0
Controller	POX
Openflow Switch	Openflow 1.0
ML Algorithms	Supervised (Logistic Regression) and Semi-Supervised

Table 2. QoS Classes as per requirements

Application	Acceptable Delay	Minimum Bandwidth Requirements	Application
Class 1	15mSec	10 Mbps	High
Class 2	25 mSec	5 Mbps	High
Class 3	40 mSec	2.5 Mbps	Medium
Class 4	>40 mSec	1 Mbps	Low

The bandwidth, acceptable delay, and priority of the flow are the measured QoS requirements of the flow computed from the feature vectors. An algorithm calculates the shortest path (P) from source to destination using the Dijkstra's algorithm (Dijkstra *et al*., 1959) and estimates its cost Cp. For each iteration I, which decreases to zero, our algorithm calculates the neighbor path (Pi) by ignoring the path computed above and estimating its cost (Ci). For the estimation of a number of possible paths, we set the value of T based on the available number of switches in the network; for our experiment, we consider its value 12. Now the loop variable decreases to zero; we calculate the neighbor path Pi by ignoring the path computed above and estimating its cost Ci. If a neighbor path incorporates a lower price, we have a tendency to replace the present path with the neighbor path. The above algorithm provides the best shortest path for each elephant flow as per its QoS requirements.

Results and Discussions

To evaluate and test our proposed framework, we experimented with a Mininet POX controller. Table 1. Contains experimental set-up details. A traffic classification of all flows was done using real traffic traces. The training datasets includes 40 features like pair of IP address source and destination, couple of port address, the protocol used, size of the packet, duration of flow, the total number of traffic packets in the backward and forward direction, the total volume of flow in forward and backward movement, standard deviation and mean of packet length in both directions, etc.

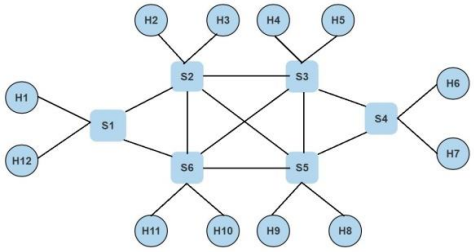

Figure 3. Network Topology

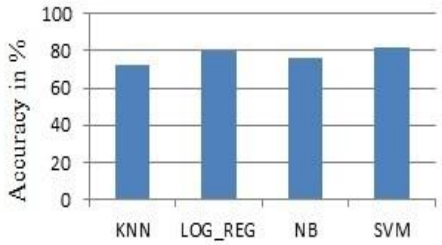

Figure 4. Accuracy of the Classifier

The proposed framework is evaluated on the network topology described in Figure. 3. The entire setup runs on Mininet. Each link is connected to two switches with a total bandwidth of 35 Mbps and the connection between the host and switch has a bandwidth of 256 Mbps. First of all, we trained our elephant flow classifier using the dataset described above. We have a total of 245,000 traffic instances which were divided into a ratio of 80% for training and 20% for testing purposes during our experimentation. Different machine learning algorithms like SVM, Logistic regression, KNN and Naive Bayes were applied to the dataset and accuracy was calculated. In Figure 4. We observed an accuracy of 80% when SVM and logistic regression were used. Further, for our elephant flow classifier, we have used the logistic regression technique because it works well for high dimensional data.

Now, each elephant flow is futher classified into one of the QoS classes by the QoS Classifier. QoS classes shown in Table 2 are decided on the basis of type of data and QoS requirements of different types of flow. Classes are named as class 1 (voice/video conference), class 2 (interactive data), class 3 (streaming) and class 4 (bulk data transfers). We have used a semi-supervised machine learning algorithm Expectation-Maximization (EM) to train our QoS classifier. Semi-supervised learning permits the algorithm to learn from a few labelled examples and classify many unlabeled instances. An expectation-maximization algorithm is an approach for performing maximum likelihood estimation in the presence of latent variables. It does this by first

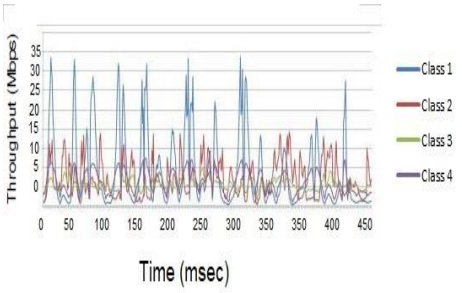

Figure 5. Throughput without using proposed Routing Algorithm

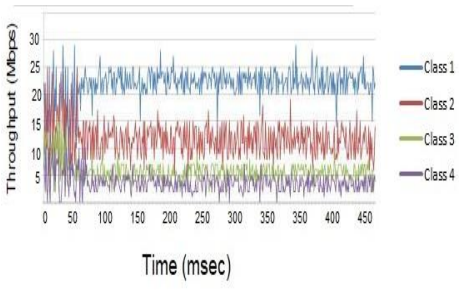

Figure 6. Throughput after using proposed Routing Algorithm

Table 3. Comparaison with existing solutions

Paper	Traffic Classification Basis	Classification Technique	Application
Lin *et al.* (2016)	QoS aware traffic classification	Deep packet Inspection and semi supervised machine learning algorithm	Reroute elephant flow for resource utilization
Yahyaoui *et al.* (2020)	Elephant and mice flow classification	K means clustering algorithm (LUNA)	Used to find forwarding rule
Xiao *et al.* (2015)	Elephant flow Detaction	Cost-sensitive learning method	To train and
Malik *et al.* (2020)	Application based traffic classification	Deep Learning (DEEP SDN)	To identify
Proposed method	QoS aware elephant flow Detaction	Combination of supervised and semi supervised machine learning algorithm	Optimize routing for elephant flow.

estimating the values for the latent variables, then optimizing the model, then repeating these two steps until convergence.

The mapping strategy instances between the QoS classes and the QoS prerequisite have been shown in Table 2. We use a semi-supervised machine learning algorithm, Expectation - Maximization (EM), which provides fine-grained results for classifying the flow into one of the QoS classes. Semi-supervised learning permits the algorithm to learn from a few labelled examples and classify many unlabeled instances

The mapping strategy instances between the QoS classes and the QoS prerequisite have been shown in Table II. We use a semi-supervised machine learning algorithm, Expectation- Maximization (EM), which provides fine-grained results for classifying the flow into one of the QoS classes. Semi-supervised learning permits the algorithm to learn from a few labelled examples and classify many unlabeled instances. Table 3. Shows a comparison of existing traffic classification techniques and their applications with our proposed method.

Analysis of Throughput for TCP Flow

For analysis of the network throughput to evaluate our framework, we use the topology described above. We consider Host H1 and H6 because they are situated at the longest end of the network. Host H1 sent four TCP flows to H6, and H6 sent four TCP flows to H1 by using the Iperf traffic generator (Marsan *et al*., 2003). Each flow belongs to one class from those enlisted in Table 2. Traffic flows were generated one by one at regular intervals of time. We have analyzed that when we classify the flow and send them directly to the controller without using our routing module of the framework, the flows were not routed to the accurate path, and each type of flow got affected by another type of flow as shown in Figure 5. At the same time, when we sent the same flows using our routing algorithm, class 1 flow gets the path having the least traffic, so TCP window size gets higher in the case of class 1 in contrast with other classes. It also has been observed that after the addition of each new flow rule, older flows that already exist in the network do not suffer because of the congestion avoidance mechanism of TCP. There was no fluctuation in TCP throughput, as shown in Figure 6.

Figure 6. shows results in terms of throughput achieved by TCP traffic flow on the above topology when we analyzed QoS requirements of flows and used both modules of our framework. We have seen that class 1 achieved larger throughput as compared to other classes. This is because class 1 flows get routed with the path having the least congestion. At the same time, the flow of other classes is also exhibiting an increase in throughput.

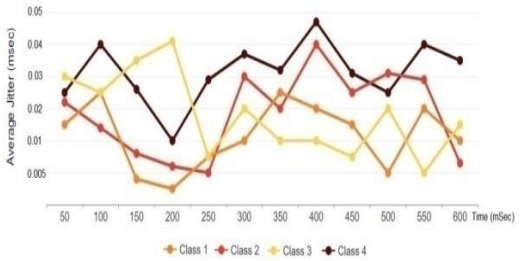

Figure 7. Average Jitter of flows without using proposed Routing Algorithm

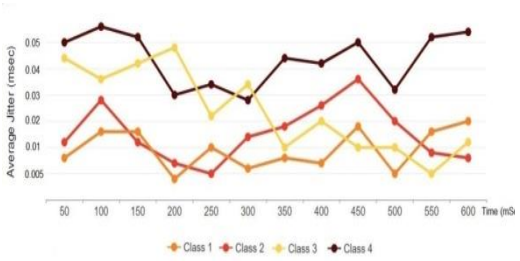

Figure 8. Average Jitter of flows after using proposed Routing Algorithm

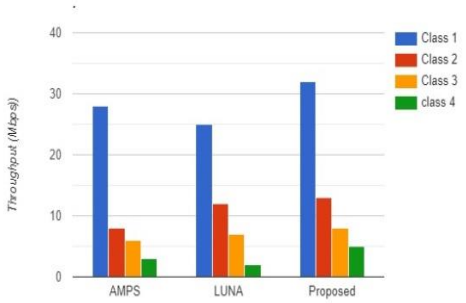

Figure 9. Comparision of TCP throughput with existing solutions

Analysis of Jitter for UDP Flow

This experiment uses a similar setup as the previous experiment. The aim of this experiment is to verify that higher priority flow should experience comparatively less Jitter. Four UDP flows were sent from H1 to H6 and vice versa. Average Jitter was calculated for all UDP flow. The delay requirements of each flow (a time delay in the sending and receiving of the data packets over a network connection) have been given in Table The average rate Jitter for all types of UDP flows has been calculated within the gap of each fifty seconds Figures 7 and 8. Show the Jitter determined during this experiment. In the figures, X-coordinate shows a time gap at which Jitter is calculated, whereas The y-coordinate shows the average Jitter calculated in that time duration. Figure 7. Shows the average Jitter when the routing module has been used; hence the path is allocated to each flow according to its priority. Therefore, we can see that high-priority flow gets less Jitter. In incase of Figure 8. The routing module of the framework was not taken into consideration; hence routing of flow was done on the basis of fewer number hops. This leads to high Jitter for flows.

Figure 9. shows the comparison of TCP throughput with an existing solution. Here, we have considered the four classes as mentioned in Table 2. The graph shows the comparison of our proposed work along with existing techniques namely, AMPS and LUNA.

Conclusions

For optimization of routing in SDN, we introduced a framework that does the cost-aware routing for elephant flows in SDN. The proposed routing framework classifies the flows not only based on their size but also on their QoS requirements using machine learning techniques. We have used a combination of supervised and semi-supervised machine learning algorithms for traffic classification. To the best of our knowledge, we are the first to apply traffic classification for providing a routing framework for SDN. We also proposed a routing algorithm that computes the shortest path as per the need of QoS parameters for each of the elephant flows. The framework has been developed using a POX controller and evaluated over mini net. The framework classified each elephant flow with an accuracy of 80% and led to significant improvements in network throughput without fluctuation for each class. In the future, the framework can be extended for achieving energy efficiency in SDN.

Acknowledgments

I would like to acknowledge the funding support from JECRC University, India. My special thanks are extended to all professors from the Computer Science Department, School of Engineering and Technology, JECRC University, India.

References

Akella, A., and Maltz, D.A. (2010). Network Traffic characteristics of data centers in the wild. Proceedings of

the 10th ACM SIGCOMM conference on Internet measurement, NY, United States., p. 267-280.

Azzouni, A., Boutaba, R., and Pujolle, G. (2017). Neuroute: Predictive dynamic routing for software-defined networks. 13th International Conference on Network and Service Management (CNSM), Tokyo, Japan, p. 1-6.

Dieng, A.B., and Paisley, J. (2019) . Reweighted expectation maximization. arXiv.org > stat > arXiv: 1906. 05850, 2019. Iprf, (2020).

Dijkstra, E.W. (1959). A note on two problems in connexion with graphs. Numerische Mathematik., 1(1):269-271. doi:10.1007/BF01386390.

Hosmer, D. W., Lemeshow, S., and Sturdivant, R. X. (2013). Applied logistic regression (3rd ed.). John Wiley & Sons.

Lin, S., Akyildiz, I.F., Wang, P., and Luo, M. (2016). Qos-aware adaptive routing in multi-layer hierarchical software - defined networks: A reinforcement learning approach. IEEE International Conference on Services Computing (SCC), San Francisco, CA, USA., p. 25-33.

Malik, Ali., de Fréin, Ruairí., al-zeyadi, Mohammed., and Andreu-Perez, Javier. (2020). Intelligent SDN Traffic Classification using Deep Learning.

Marsan, M.A., and Donnet, B. (2003). A self-contained and efficient IP traffic generator. ACM SIGCOMM Computer Communication Review., 33(2):105-112. doi:10.1145/948305.948331

Open Networking Foundation [ONF], (2020). Software-defined Networking: The new norm for networks. Palo Alto, CA 94303 Available from: www.opennetworking.org, Accessed date: April 2020.

Pasca S.T.V., Kodali, S.S.P., and Kataoka, K. (2017). AMPS: Application-aware multipath flow routing using machine learning in sdn. in 2017 Twenty-third National Conference on Communications (NCC), Chennai, India., p. 1-6.

Patrushev, G.G., and Drozdova, V.G. (2017). Routing efficiency evaluation with SDN solutions integration in the data network. // 18th International Conference of Young Specialists on Micro/Nanotechnologies and Electron Devices (EDM), Erlagol Altai., pp. 190-194.

Phemius, K., and Bouet, M. (2013) . Monitoring latency with openflow. 9th International conference on Network and service management (CNSM), Zurich, Switzerland., p. 122-125.

Wnag, P., Lin, S., and Luo, M. (2016). A framework for qosaware traffic classification using semi-supervised machine learning in sdn. in 2016 IEEE International Conference on Services Computing (SCC), San Francisco. CA., p. 760-765.

Xiao, P., Qu, W., Qi, H., Xu, Y., and Li, Z. (2015). An Efficient Elephant Flow Detection with Cost-Sensitive in SDN. In Proceedings of the 6 th International Conference on Internet and Networked Systems (ICINS), doi: 10.4108/icst.iniscom.2015.258274.

Xie, J., Yu, F.R., Huang, T., Xie, R., Liu, J., Wang, C., and Liu, Y. (2019). A Survey of Machine Learning Techniques Applied to Software Defined Networking (SDN): Research Issues and Challenges. in IEEE Communications Surveys and Tutorials., 21(1):393-430.

Yahyaoui, h., Aidi, S., and Zhani, M.F. (2020). On using flow classification to optimize traffic routing in sdn networks. IEEE 17th Annual Consumer Communications Networking Conference (CCNC), Las Vegas, USA., p. 1-6.