Delay Estimation in Fogs based on Software-Defined Networking

Daniela M. Casas-Velasco*
Email: danielac@lrc.ic.unicamp.br

William F. Villota-Jacome*
Email: wfernando@lrc.ic.unicamp.br

Nelson L. S. da Fonseca*
Email: nfonseca@ic.unicamp.br

Oscar M. Caicedo Rendon†
Email: omcaicedo@unicauca.edu.co

*Institute of Computing, University of Campinas, Brazil
†Telematics Engineering Group, University of Cauca, Colombia

***Abstract*—Fog computing brings the advantages and power of cloud computing to the edge of the network. Software-Defined Networking (SDN) has been considered as a feasible solution to cope with the complexity of the orchestration of fog devices. Nevertheless, the use of an SDN controller introduces delays into the transport of packet flows in the fog layer, which may impact on the Quality of Service (QoS) of applications in the continuum IoT-Fog-Cloud. In this paper, we propose a regression model for predicting delay values in an SDN-based fog layer. To build up the regression model, we constructed a dataset, performed data cleaning, carried out feature selection, and applied different Machine Learning (ML) techniques. Our evaluation results reveal that the Random Forest (RF) technique overperforms Decision Tree (DT) and Neural Network (NN) techniques on predicting the delay in an SDN-based fog layer. Furthermore, the predicted delay values reinforce that a fog layer based on SDN can support different latency-sensitive applications.**

***Index Terms*—Fog Computing, Software-Defined Networking, Machine Learning, Internet of Things, Knowledge-Defined Networking, Quality of Service, Delay**

I. Introduction

The ever-increasing number of mobile devices and the exponential-raise of the number of Internet of Things (IoT) devices have led to a significant growth in network traffic. Handling such increase and yet meeting the Quality of Service (QoS) requirements of delay-sensitive applications is a challenging task. Fog computing helps to meet delay requirements by bringing the cloud to the network edge. Notwithstanding, the design and operation of a fog is an inherently complex problem due to [1]: the necessary interaction between fog resources and cloud servers (*i.e.*, service synchronization and orchestration); the intermittent connectivity between mobile devices and the fog; and the service-centric approach of fog which is designed to focus on services rather than on the location of devices.

Software-Defined Networking (SDN) is a candidate to deal with such a complex design problem due to its features of programmability and centralized control. Indeed, these features make fog layers more flexible, scalable, efficient, and adaptable [1]. However, an SDN-based fog layer can add delay to end-to-end communications since, by default, the first packets of a flow are forwarded to and processed by an SDN controller. Such an increase in delay (communication and processing) may compromise the meeting of QoS requirements of delay-sensitive applications (*e.g.,* Industries 4.0 and Autonomous Vehicles) [2].

Some papers [3]–[7] have considered the delay in service provisioning to enhance the QoS provisioning in fog computing, but these papers have not considered fog layers based on SDN. Other papers discuss the introduction of SDN in fog computing. In [8], the authors manage a wireless fog infrastructure by using a hybrid SDN controller. The work in [9] proposes an SDN-based scheme for supporting multi-domain fog/cloud services. The work in [10] defines a framework that improves the resilience of networks by including fog nodes integrated to SDN. The work in [11] explores how a control topology impact on real-time services in a fog-cloud environment. Despite these contributions, to the best of our knowledge, up to now, no other work has investigated the impact of the delay introduced by the use of SDN in the fog on the QoS provisioning to fog applications. We argue that an accurate delay estimation is fundamental to meet the QoS requirements of delay-sensitive applications in the IoT-Fog-Cloud continuum. In this regard, recent investigations [12], [13] indicate that the delay estimation can be carried out by Machine Learning (ML) techniques. In fact, these investigations introduce accurate models that contribute to the QoS provisioning.

In this paper, we present a predictive model that allows estimating the delay introduced by SDN in the fog layer. Our solution follows the Knowledge-Defined Networking (KDN) paradigm [12], which is an emerging paradigm that applies ML to the management of SDNs. Also, we define a regression model to predict the delay in an SDN-based fog layer. Our prediction model can be used to determine the impact of the delay on the delay-sensitive applications, which is fundamental to solutions intended to improve the performance of applications in the IoT-Fog-Cloud continuum. Our solution includes three steps from the Cross-Industry Standard Process for Data Mining (CRISP-DM) methodology [14]: building up a dataset from an emulated SDN-based fog layer, data cleaning and feature selection to the created dataset, and foreseeing the

978-1-7281-0962-6/19/$31.00 ©2019 IEEE

delay. To sum up, the contributions of this paper are: *i)* a dataset to model the delay of a fog layer based on SDN, *ii)* a regression model that allows predicting the delay in the fog layer; and *iii)* the evaluation of different regression techniques disclosing that Random Forest (RF) is the algorithm that most accurately predicts the delay in an SDN-based fog layer. Our evaluation results corroborate that different delay requirements can be supported in an IoT-Fog-Cloud continuum.

The remainder of this paper is organized as follows. In Section II, we present the related work. In Section III, we introduce the model to predict the delay in an SDN-based fog layer. In Section IV, we present the delay predictions. In Section V, we present some conclusions and implications for future work.

II. Related Work

Several investigations have addressed QoS provisioning in fog computing. The work in [3] introduces a mathematical model to study the processing and communication delays in the fog. In [4], the authors investigated the performance of IoT on fogs, using metrics such as power consumption and service latency. The work in [5] proposed a workload allocation model for evaluating the trade-off between power consumption and delay in a cloud-fog computing system. The approach in [6] studies the network components that introduce delay and develops a delay estimation framework for fog-based IoT applications by using predictive algorithms. The mentioned investigations [3]–[6] propose mathematical models that are based on assumptions about the operation of systems. Such assumptions may not capture the dynamics of the fog layer.

Some investigations deal with the relationship between SDN and fog computing. The proposal in [9] introduces an SDN-based network slicing scheme to support multi-domain services in a fog-cloud environment. The work in [8] proposes a solution for managing wireless fog infrastructure by employing an SDN meta-controller layer that allows adding new controllers at runtime in failure or overhead cases. In [10], the authors introduce a framework to improve the resilience of networks by employing SDN. Fog nodes are used as local decision-making elements interacting with SDN to push updates about the flow table throughout the network. The work in [11] explores how a control topology impact on real-time services in a fog-cloud environment by assessing the tradeoff between the number of control layers and the management capacity of controllers.

To the best of our knowledge, so far, no other work has investigated the impact of using SDN on the delays in a fog. In this paper, we argue that an accurate delay estimation is crucial to meet the QoS requirements of delay-sensitive applications located on the IoT-Fog-Cloud continuum. Actually, recent investigations [12], [13] have suggested the estimation of network delay by the employment of ML techniques.

Furthermore, potential QoS degradation problems can be addressed by analyzing the delay patterns in the fog layer [15] and a precise estimation of the delay is a fundamental step towards a solution to these problems.

III. Regression model to predict the delay in an SDN-based fog layer

In this section, we introduce our regression model for predicting the delay in an SDN-based fog layer by presenting: the motivation, the methodology, the data understanding, the data preparation, and the construction of the model.

A. Motivation

Let us consider an IoT scenario that requires deploying services by guaranteeing QoS agreements that attend delay-sensitive applications. In particular, an SDN controller has been deployed in the the fog layer to deal with the requirements of such IoT applications. The SDN controller is in charge of distributing the network traffic in the IoT-Fog-Cloud continuum according to the QoS requirements of each application. In this scenario, two additional delay components are introduced. First, a delay is introduced by the SDN controller when there is no rule established to forward/route a particular flow. This new flow is forwarded to be handled by the controller. Second, a processing delay caused by the application (running on the top of the SDN controller) that deals with the first flow coming from the data plane. These two delay components (communication and processing) can strongly affect the QoS provisioning.

In this paper, we argue that modelling and characterizing the delay in an SDN-based fog layer can facilitate the QoS provisioning. The knowledge discovered can be used to automatically improve the network via the the SDN controller. In this way, the controller can take advantage of the delay estimation to make decisions about whether the processing of the applications in the fog layer is feasible or not.

Furthermore, the delay model can be used as a baseline for tuning the parameters of routing and flow classification (mice/elephants) algorithms that also impact on QoS provisioning.

B. Methodology

We followed the CRISP-DM methodology [14] to build up a regression model intended to estimate delay values in SDN-based fogs. In particular, we carried out three CRISP-DM steps: Data Understanding, Data Preparation, and Modelling. Data Understanding generates, gathers, and defines the set of data and the set of classes related to delay forecasting. This step is essential since datasets vary not only from one problem to another but also from one period of time to another. Data Preparation aims at building the final dataset by cleaning data and selecting appropriate attributes that reduce the computational overhead and yet increase accuracy to estimate the delay. Modelling refers to the application of different ML techniques to predict the delay. In this step, we assess several candidate algorithms to choose the one that best foresees the delay in a fog layer based on SDN.

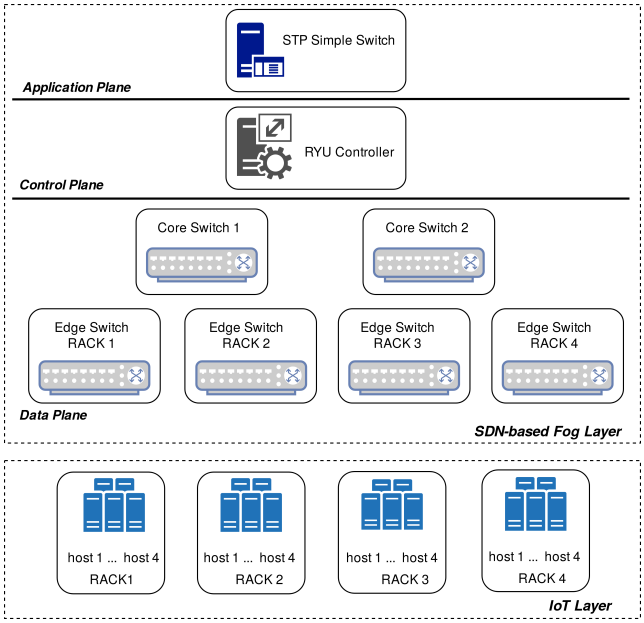

Fig. 1: Test environment for dataset construction

By following this methodology, first, we do not need to make constraining assumptions (*e.g.,* link speeds and traffic model) that are usually required in traditional network modelling. Moreover, SDN provides a global control to manage large (physical/virtual) networks which is a facilitating environment to the methodology employed. Second, we consider the delay estimation as a regression problem since our goal is to map a set of new input data that characterizes an SDN-based fog layer onto a set of continuous-valued output [16].

C. Data understanding

This step constructs an initial dataset that represents flow/packet features in an SDN-based fog layer. To construct this initial dataset, we performed the following activities: *i)* designed an experimental scenario to emulate a fog layer based on SDN in an IoT-based context, *ii)* deployed the experimental scenario; and iii) generated network traffic to fill the dataset with data.

As depicted in Fig. 1, we designed the experimental scenario by defining a fog layer and an IoT layer. To emulate the fog layer based on SDN, we relied on the three architectural SDN planes and defined the IoT layer as hosts that represent IoT-based devices generating traffic. The fog layer followed a fat-tree topology [17] with an SDN controller, two core switches and four edge switches. In this scenario, the fog layer was enabled to receive and send traffic from and to the IoT layer that consisted of 16 hosts (*i.e.*, four per edge switch).

To deploy the experimental scenario, for the SDN Control Plane, we run a Ryu SDN controller [18] in a Virtual Machine (VM) with Linux Ubuntu Server 18.04. For the SDN Application Plane, we deployed a simple switch SDN application (*i.e.,* Spanning Tree Protocol) in order to handle the core and edge switches. For the SDN Data Plane, we deployed the fat-tree topology composed by two core switches and four edge switches with link speed and delay of $100Mb$ and $10ms$, respectively. We emulated the data plane in a VM with Ubuntu Server 14.04 by using Mininet 2.2.2 [19]. We used the D-ITG generator [20] to generate the traffic from the IoT layer.

Once we deployed the experimental scenario, we generated and collected data from the SDN-based fog layer by sending traffic to the hosts located in the IoT layer. In particular, to generate and collect such data, we used the *ITGSend* and *ITGRecv* commands of D-ITG. These commands allow emulating different traffic (*e.g.,* UDP, Telnet, and DNS). The result was an initial dataset (or raw dataset) with two general traffic features (*i.e.,* Source IP and Destination IP) and four QoS performance features (*i.e.,* bitrate, jitter, packet loss, and time), and a target which is the delay. The initial dataset included a total of 190429 instances. An instance is a pair of input variables (*i.e.*, the six features) and output variables (*i.e.*, the delay).

D. Data Preparation

This step builds up the final dataset. We performed the following activities: *i)* explored and cleaned the data (*i.e.,* remove or add instances); and *ii)* selected the relevant data for delay prediction (*i.e.,* identify and discard irrelevant and redundant features). We carried out the above activities by using the Weka data mining tool since it offers a comprehensive collection of ML algorithms and data pre-processing procedures [21].

1) Data cleaning: This activity involves the selection of clean subsets of data, and the use of procedures to improve data quality concerning well-known issues such as outliers, noise, inconsistency, and incompleteness [22]. For this cleaning, first, we got rid of 162 instances in the final dataset because of inconsistent delay information such as negative values. Second, we took out 2061 instances since the bitrate or time values were 0, such instances presented lack of response in the data collection or faulty in their measurement. Finally, we removed 23489 instances because they were outliers, instances that deviate so much from the other instances. We used the Weka *InterquartileRange* filter to identify such outliers. Summarizing, after performing the cleaning step, we obtained a dataset composed of a total of 166930 instances.

2) Feature selection: This activity targets to reduce the dimension of dense dataset by identifying irrelevant or redundant features. We performed this feature selection to mainly avoid two issues. First, the increase of computational cost with extremely low accuracy gain, introduced by irrelevant features. Second, the over-fitting (*i.e.,* a ML model that fits the training data but does not generalize well to predict new data) introduced by redundant features. For the feature selection, we ignored header information (*i.e.,* Source IP and Destination IP) since it does not describe the delay of packets. We only considered performance information (*i.e.,* time, jitter, packet loss, bitrate, and delay features) to let the learned model to fit unknown packets suitably. Ignoring header information, that has a typical structure and inconsistent values, allows

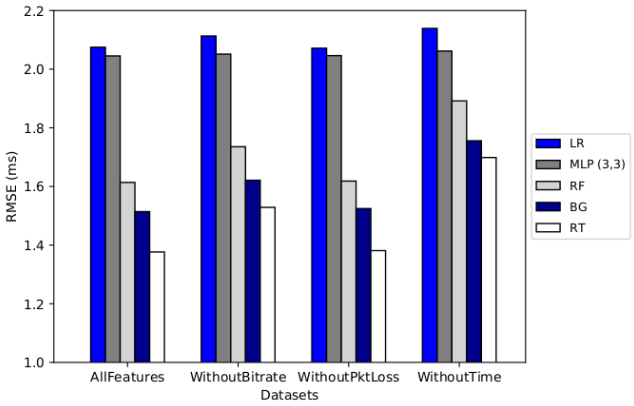

Fig. 2: Feature selection test

obtaining a model that is independent of IP addressing and network topologies and with a high modelling generalization.

For the feature selection, we evaluated just the four performance features by the correlation attributes ranking. This ranking was computed using the *CorrelationAttributeEval* and *ReliefFAttributeEval* functions offered by Weka. These functions rank the features according to the worthiness of an attribute by measuring the correlation between it and the target class and they also consider the value of the given attribute for the nearest instance and the target class. As a result, the ranking output disclosed that jitter is the most correlated feature to our target class (*i.e.,* delay). To select a proper feature combination in the dataset that allows building a more accurate ML-based model, we always maintained the jitter as a feature. Thus, we defined four possible datasets: *(i)* dataset with all features (*i.e.,* time, jitter, packet loss, bitrate, and delay), *(ii)* dataset without bitrate, *(iii)* dataset without packet loss; and *(iv)* dataset without time.

To select the features of the final dataset and measure its suitability for creating the delay prediction model based on ML, we employed the convention error metrics used in regression problems namely, Mean Absolute Error (MAE) and Mean Squared Error (MSE) [16]. Most of the investigations related to network traffic prediction evaluate their regression models with these metrics [23]–[25]. MAE is the average of the absolute error between the actual and predicted values. MSE is the average of the squares of the error between the actual and predicted values. MAE is simpler and easier to interpret than MSE. However, MSE is more useful for heavily penalizing large errors. In this work, we used Root Mean Squared Error (RMSE) that represents the standard deviation of the error between the actual and the predicted values. It is important to highlight that we considered a 95% confidence interval in the RMSE measurement. However, since we intend to show the evolution of RMSE as the number of instances increases in the dataset, such confidence interval is not depicted in the figures for the sake of visual interpretation.

Ten ML regression algorithms were applied to the four datasets to measure the corresponding RMSE. These algorithms are: Linear Regression (LR) [23], three DTs [26], [27], and six variations of Multilayer Perceptron (MLP) [28]. The LR is a basic regression that calculates coefficients for a line or hyperplane that best fits the training data; we set it to use a ridge regularization technique in order to reduce the complexity of the learned model. The DTs algorithms operate by breaking a dataset into smaller subsets from which they build a decision tree cumulatively. The first decision node in a tree is the best prediction called root node. In particular, the three DTs algorithms that we evaluated are: the RF [26], REPTree (RT), and Bagging (BG) [27]. The MLP is a neural network with processing neurons organized into layers that approximate a function that best fits the real value output. For this algorithm, we considered six variations in its configuration (*i.e.,* one, two and three hidden layers, each case with two and three neurons).

Fig. 2 shows the RMSE obtained by each ML algorithm in the four datasets. The dataset with the smallest RMSE value is the one with all the performance features. Notably, the mean RMSE value for such dataset is a 0.04%, 2.48%, and 4.96% smaller than the mean RMSE values obtained for the datasets without packet loss, bitrate and time, respectively. In Fig. 2, we only include the results from the MLP algorithm configured with three hidden layers and three neurons (MLP (3,3)), since among the considered configurations it is the one which produces the smallest RMSE value.

E. Modelling

This subsection presents the selection, assessment, and modelling performed for finding the ML algorithm to be used to construct the regression model. From the ten algorithms used in the Feature selection activity, we chose the top five algorithms with the smaller RMSE values when applied to the dataset composed by all the features. These algorithms are: LR, RT, RF, BG and MLP (3,3).

Before assessing the algorithms for training the dataset, it is not possible to know which algorithm provides the most accurate prediction of the delay. In this regard, in the assessing activities, we evaluated the algorithms for the dataset with all features by using cross-validation. Cross-validation allowed us to analyze the RMSE behavior when the number of instances increases from 1250, 12500, 25000, 50000, 75000, 100000, 125000, 150000, to 166930. The cross-validation is typically set with k-fold = 10 to avoid under-fitting [29]. We executed ten times each test by choosing random instances for obtaining a total of 100 RMSE values.

Fig. 3 depicts the mean of RMSE obtained from the executions of each algorithm with several datasets including 25000, 75000, 150000, and 166930 instances. Results revealed that the RF DT algorithm presents the smallest value of RMSE as the number of instances increases. In particular, the RMSE produced by RF was 20.17%, 18.94%, 5.85%, and 0.54% smaller than those produced by MLP, LR, RT, and BG, respectively. We highlight that DT algorithms produce the lowest RMSE values since these algorithms can model complex relationships between variables without needing to

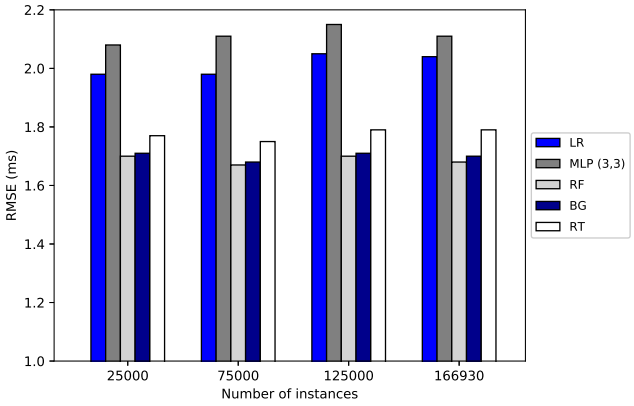

Fig. 3: RMSE Evaluation

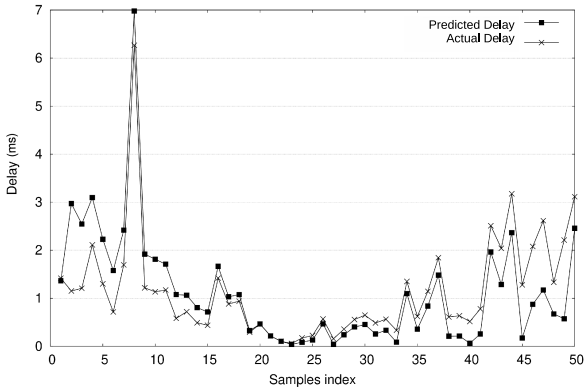

Fig. 4: Actual and predicted delay

make strong modelling assumptions. Furthermore, unlike MLP and LR, DTs can identify relevant independent variables with the built tree and operate well when considering many potential variables to construct a proper prediction model with a low error [28].

The above results indicate that for our final dataset, RF is the algorithm that best predicts the delay produced by an SDN-based fog layer. This is explained by the fact that the RF takes a set of decision trees, with high-variance, to build a model with low variance by finding the average output given by most of the individual trees. As a consequence, in the modelling step we construct the delay regression model by using the training dataset jointly with RF.

IV. Delay prediction

In this section, we present the results of using our model to predict the delay in an SDN-based fog layer by randomly taking instances from the built dataset. In such instances, the delay values are removed. After predicting the delay, we can compare both actual and predicted values from the model.

For the sake of readability, Fig. 4 depicts only the predicting results employing 50 instances. We represent the actual delay values with cross points and the predicted delay values obtained by our model with square points. The difference between both predicted and actual delay values is low. Specifically, we obtained an RMSE value of $0.8227ms$ which confirms that, from the algorithms assessed in the modelling step, the RF is the most appropriate for predicting delays in a fog layer based on SDN.

In addition to the delay prediction and in order to illustrate the importance of our data-driven model, we characterized the delay in the SDN-based fog layer by identifying some groups of the most recurrent values. We processed the predicted delay values with a ML clustering algorithm that, first, divides the data into groups of elements with common characteristics [16]. Second, it allows identifying the most recurrent predicted delay values. In particular, we used the K-means algorithm [30] that takes a statistical vector (*i.e.,* the delay predicted values) as an input to distribute them into clusters, where each value belongs to the nearest cluster.

Cluster	Mean Delay (ms)	Number of instances	Percentage
1	0.48	117297	70.26
2	2.09	41429	24.81
3	5.26	7358	4.4
4	20.09	754	0.45
5	49.67	92	0.05

TABLE I: K-means clustering for the delay

To correctly choose the number of clusters and determine their consistency within our data-driven delay prediction model, we used the *silhouette* validation method. This method provides a measure value (*i.e.,* ranging from -1 to 1) indicating how similar an instance is to its cluster compared to the other ones. In this sense, values ranging in: 0.71 to 1 mean that a strong structure has been found; 0.51 to 0.7 a reasonable structure has been found; 0.26 to 0.5 the structure is weak and could be artificial; $<$ 0.25 no substantial structure has been found [31]. We validated different numbers of clusters (*i.e.,* from 2 to 10) with *silhouette*. The higher value obtained by *silhouette* was 0.7243 for the 2-clusters structure. However, as the main purpose is to obtain some well-defined value ranges for identifying supported delay-sensitive services, we selected the 5-cluster structure which still has an acceptable *silhouette* value of 0.5622 meaning that it is a reasonable structure.

Table I shows the K-means results when clustering the total instances (166930) of the predicted delay values in 5 clusters. Such values demonstrate that the delay in an SDN-based fog layer mainly varies from $0.48ms$ to $2.09ms$ (*i.e.,* clusters 1 and 2) which represents the 95.07% of the total predicted values. The remaining 4.93% belongs to cluster 3, 4, and 5 reaching a maximum mean delay of $49.67ms$. These results indicate that the increase in delay due to the introduction of SDN in fog layers is negligible for real-time applications. Specifically, the SDN-based fog layer can correctly deploy IoT-based applications in areas such as factory automation, smart grids, and Intelligent Transport Systems (ITS) which their required delay ranges from $0.25ms$ to $10ms$, $3ms$ to $20ms$, and $10ms$ to $100ms$, respectively [32].

V. Conclusions

In this paper, we introduced a procedure for estimating the delay in an SDN-based fog layer. We built a dataset that represents a fog layer based on SDN, performed data cleaning and feature selection, and carried out predictions using different ML regression algorithms. Our evaluation results corroborated that for our dataset the used DT algorithms (*i.e.,* RT, BG, and RF) outperform the LR and MLP (3,3) considering the RMSE as a metric for comparison. This happens since the DT algorithms have the capacity of identifying relevant independent variables during the construction of the tree for the prediction model with a low error. As the RF produced the smallest RMSE, we used RF to train and create the data-driven delay regression model. Such a model predicts delay values from $0.48ms$ to $2.09ms$ corroborating that the SDN-based fog layer can support delay-sensitive applications. In particular, such a fog layer can support IoT-based applications in areas such as factory automation, ITS, and smart grids. The dataset and the model serve as a basis for future models of SDN-based fog layers, since they provide information about the impact of using SDN in the fog on QoS provisioning. The datasets used and the delay model are available on github [33].

As future work, we intend to consider variation of the RF configuration parameters (*e.g.,* number of trees) for enhancing performance. Also, we plan to develop an end-to-end scheduling model for addressing the QoS problem in fog computing.

Acknowledgment

The authors would like to thank CAPES, CNPQ, Sao Paulo Research Foundation (FAPESP) under grant #15/24494-8, and the University of Cauca.

References

[1] A. C. Baktir, A. Ozgovde, and C. Ersoy, "How can edge computing benefit from software-defined networking: A survey, use cases, and future directions," *IEEE Communications Surveys & Tutorials*, vol. 19, no. 4, pp. 2359–2391, October 2017.

[2] M. Y. Arslan, K. Sundaresan, and S. Rangarajan, "Software-defined networking in cellular radio access networks: potential and challenges," *IEEE Communications Magazine*, vol. 53, no. 1, pp. 150–156, 2015.

[3] K. Intharawijitr, K. Iida, and H. Koga, "Analysis of fog model considering computing and communication latency in 5g cellular networks," in *PerCom Workshops*. IEEE, March 2016, pp. 1–4.

[4] S. Sarkar, S. Chatterjee, and S. Misra, "Assessment of the suitability of fog computing in the context of internet of things," *IEEE Transactions on Cloud Computing*, vol. 6, no. 1, pp. 46–59, January 2018.

[5] R. Deng, R. Lu, C. Lai, and T. H. Luan, "Towards power consumption-delay tradeoff by workload allocation in cloud-fog computing," in *ICC*. IEEE, June 2015, pp. 3909–3914.

[6] J. Li, T. Zhang, J. Jin, Y. Yang, D. Yuan, and L. Gao, "Latency estimation for fog-based internet of things," in *ITNAC*, November 2017, pp. 1–6.

[7] A. Brogi and S. Forti, "Qos-aware deployment of iot applications through the fog," *IEEE Internet of Things Journal*, vol. 4, no. 5, pp. 1185–1192, October 2017.

[8] A. Hakiri, B. Sellami, P. Patil, P. Berthou, and A. Gokhale, "Managing wireless fog networks using software-defined networking," in *AICCSA*, October 2017, pp. 1149–1156.

[9] R. Bruschi, F. Davoli, P. Lago, and J. F. Pajo, "A scalable sdn slicing scheme for multi-domain fog/cloud services," in *2017 IEEE Conference on Network Softwarization (NetSoft)*, July 2017, pp. 1–6.

[10] A. Modarresi, S. Gangadhar, and J. P. Sterbenz, "A framework for improving network resilience using sdn and fog nodes," in *RNDM*. IEEE, September 2017, pp. 1–7.

[11] V. B. Souza, A. Gmez, X. Masip-Bruin, E. Marn-Tordera, and J. Garcia, "Towards a fog-to-cloud control topology for qos-aware end-to-end communication," in *IWQoS*. IEEE/ACM, June 2017, pp. 1–5.

[12] A. Mestres, A. Rodriguez-Natal, J. Carner, P. Barlet-Ros, E. Alarcón, M. Solé, V. Muntés-Mulero, D. Meyer, S. Barkai, M. J. Hibbett *et al.*, "Knowledge-defined networking," *ACM SIGCOMM Computer Communication Review*, vol. 47, no. 3, pp. 2–10, September 2017.

[13] A. Mestres, E. Alarcón, Y. Ji, and A. Cabellos-Aparicio, "Understanding the modeling of computer network delays using neural networks," in *proceedings Big-DAMA*. AC12M, August 2018, pp. 46–52.

[14] P. Chapman, J. Clinton, R. Kerber, T. Khabaza, T. Reinartz, C. Shearer, and R. Wirth, "Crisp-dm 1.0 step-by-step data mining guide," *SPSS inc*, vol. 16, 2000.

[15] S. Ayoubi, N. Limam, M. A. Salahuddin, N. Shahriar, R. Boutaba, F. Estrada-Solano, and O. M. Caicedo, "Machine learning for cognitive network management," *IEEE Communications Magazine*, vol. 56, no. 1, pp. 158–165, January 2018.

[16] R. Boutaba, M. A. Salahuddin, N. Limam, S. Ayoubi, N. Shahriar, F. Estrada-Solano, and O. M. C. Rendon, "A comprehensive survey on machine learning for networking: evolution, applications and research opportunities," *Journal of Internet Services and Applications*, vol. 9, pp. 1–99, June 2018.

[17] Y. Li and D. Pan, "Openflow based load balancing for fat-tree networks with multipath support," in *ICC*, 2013, pp. 1–5.

[18] A. L. Stancu, S. Halunga, A. Vulpe, G. Suciu, O. Fratu, and E. C. Popovici, "A comparison between several software defined networking controllers," in *TELSIKS*. IEEE, October 2015, pp. 223–226.

[19] R. L. S. de Oliveira, A. A. Shinoda, C. M. Schweitzer, and L. R. Prete, "Using mininet for emulation and prototyping software-defined networks," in *IEEE*, June 2014, pp. 1–6.

[20] A. Botta, A. Dainotti, and A. Pescapé, "A tool for the generation of realistic network workload for emerging networking scenarios," *Computer Networks*, vol. 56, no. 15, pp. 3531–3547, October 2012.

[21] M. Hall, E. Frank, G. Holmes, B. Pfahringer, P. Reutemann, and I. H. Witten, "The weka data mining software: an update," *SIGKDD explorations newsletter*, vol. 11, no. 1, pp. 10–18, November 2009.

[22] D. C. Corrales, A. Ledezma, and J. C. Corrales, "A conceptual framework for data quality in knowledge discovery tasks (fdq-kdt): A proposal," *Journal of computers*, vol. 10, no. 6, pp. 396–405, November 2015.

[23] Y. Zhu, G. Zhang, and J. Qiu, "Network traffic prediction based on particle swarm bp neural network," *JNW*, vol. 8, no. 11, pp. 2685–2691, 2013.

[24] S. Chabaa, A. Zeroual, and J. Antari, "Identification and prediction of internet traffic using artificial neural networks," *Journal of Intelligent Learning Systems and Applications*, vol. 2, no. 03, p. 147, October 2010.

[25] P. Bermolen and D. Rossi, "Support vector regression for link load prediction," *Computer Networks*, vol. 53, no. 2, pp. 191–201, February 2009.

[26] L. Breiman, "Random forests," *Machine Learning*, vol. 45, no. 1, pp. 5–32, October 2001.

[27] Y. Zhao and Y. Zhang, "Comparison of decision tree methods for finding active objects," *Advances in Space Research*, vol. 41, no. 12, pp. 1955–1959, 2008.

[28] R. Arora, "Comparative analysis of classification algorithms on different datasets using weka," *International Journal of Computer Applications*, vol. 54, no. 13, 2012.

[29] G. James, D. Witten, T. Hastie, and R. Tibshirani, *An introduction to statistical learning*. Springer, June 2013, vol. 112.

[30] Z. Fan and R. Liu, "Investigation of machine learning based network traffic classification," in *ISWCS*. IEEE, August 2017, pp. 1–6.

[31] P. J. Rousseeuw, "Silhouettes: a graphical aid to the interpretation and validation of cluster analysis," *Journal of computational and applied mathematics*, vol. 20, pp. 53–65, November 1987.

[32] P. Schulz, M. Matthe, H. Klessig, M. Simsek, G. Fettweis, J. Ansari, S. A. Ashraf, B. Almeroth, J. Voigt, I. Riedel, A. Puschmann, A. Mitschele-Thiel, M. Muller, T. Elste, and M. Windisch, "Latency critical iot applications in 5g: Perspective on the design of radio interface and network architecture," *IEEE Communications Magazine*, vol. 55, pp. 70–78, January 2017.

[33] D. Casas-Velasco, W. Villota-Jacome, N. Fonseca, and O. Caicedo. (2018) Regression model for the estimation of the delay in sdn-based fogs. [Online]. Available: https://github.com/Fernandovj/delayRegressionModel