Predicting Response Time in SDN-Fog Environments for IIoT Applications

Guilherme Werneck de Oliveira*, Rodrigo Toscano Ney*, Juan Luis Herrera†,
Daniel Macêdo Batista*, R. Hirata Jr.*, Jaime Galán-Jiménez†, Javier Berrocal†, Juan Manuel Murillo†,
Aldri Luiz dos Santos‡, Michele Nogueira‡
*Department of Computer Science
University of São Paulo (USP), Brazil
{werneck, rodrigo.ney, batista, hirata}@ime.usp.br
†Department of Computer Science and Communications Engineering
University of Extremadura (UEx), Spain
{jlherrerag, jaime, jberolm, juanmamu}@unex.es
‡Department of Computer Science
Federal University of Minas Gerais (UFMG), Brazil
{aldri, michele}@dcc.ufmg.br

2021 IEEE Latin-American Conference on Communications (LATINCOM) | 978-1-6654-4035-6/21/$31.00 ©2021 IEEE | DOI: 10.1109/LATINCOM53176.2021.9647803

***Abstract*—In IoT application scenarios, the response time is one of the attributes that most require attention and, for this reason, the paradigm of decentralized (or fog) computation has gained ground. Moreover, to help reduce the response time of decentralized IoT networks, routing optimization approaches can be employed using software-defined networking (SDN). When both contexts are combined, a new one called SDN-Fog Environments appears. This work presents a solution to predict the response time of Industrial Internet of Things (IIoT) applications using supervised and unsupervised learning for SDN-Fog Environments. Results show that the prediction of the response time of IIoT scenarios was close to the times obtained by solving the problem in the literature. Furthermore, according to the best-performing models, the prediction framework had less than 50 milliseconds of variation, executed in less than one second.**

***Index Terms*—prediction, response time, IoT, Fog, SDN, machine learning**

I. Introduction

The Internet of Things (IoT) has achieved rapid popularization, reaching a wide range of applications in healthcare, environmental monitoring, home automation, smart mobility, and Industry 4.0 [1], [2]. Thus, more and more IoT devices are deployed in various public and private environments, progressively becoming common objects of everyday life. The IoT has been seen by many as the next stage of the Internet, and its implementation enables the evolution not only of industrial infrastructures but also of smart cities because, with IoT, the development of transport systems, waste control, energy, among others, becomes more efficient and improves the quality of life of citizens. On the other hand, the physical infrastructure of heterogeneous systems is complex and requires efficient and dynamic solutions for deploying, configuring, and managing IoT networks [3], [4].

In whatever IoT application scenario, the response time for a given activity is one of the attributes that most require attention [5] and, for this reason, the paradigm of decentralized computation processing has gained ground in this context. *Fog computing* is an example that supports applications that require strict-response time, separating the processing of a given task into nodes of distinct processing capacity interconnected by network and closer to end-devices [6]. The latency is another factor to be seen in the composition of the response time and, to make an optimal decision about the location of processing devices, the decentralized computation distribution problem (DCDP) [7] should be considered. To reduce the latency of fog computing, routing optimization approaches can be employed [8] by using software-defined networking (SDN) [9]. In other words, SDN controller placement strategies are used to minimize the latency among devices. However, the latency between SDN controllers and switches must also be considered when computing network response time. The problem of placing SDN controllers on top of switches is known as Controller Placement Problem (CPP) [10].

The combination of both problems, DCDP and CPP, results in another one, the SDN-Fog deployment problem, posed initially by Herrera *et al.* [11] to describe a single effort approach to solve them together. The proposed approach consists of a Distributed Application Deployment Optimization (DADO) framework, which contains Mixed Integer Linear Program (MILP) into its core for identifying the deployment of both services and SDN controller in order to meet specific requirements. The results presented by the authors showed that the framework is suitable for small network topologies, infrastructures with under 300 nodes, in which the optimization problem is solved in a few seconds. However, the solution takes between 4664 and 20982 seconds and needs approximately 18 exabytes of memory in larger networks because the SDN-Fog deployment problem is NP-hard. One way to optimize MILP models is to use heuristics like branch-and-bound (BB) and branch-and-cut (BC) [12]. However, the resolution complexity tends not to reduce face to dynamic environment, in other words, when there is an excessive number of variables and

restrictions, such as complex IoT network environments and real-world traffic flows [13].

Prediction of network behavior can bring advantages for many applications that have strict Quality of Service (QoS) requirements [14], such as placement of devices, resource allocation, besides allowing service designers to predict the performance of their applications for different load conditions [15]. For example, for the optimization problem presented above, predicting variables that make up the network can be helpful instead of just considering the problem's solution modeled in MILP, which requires a long execution time and high computational power. Furthermore, predicting the network response time can be very useful when planning network capacity, as it can elucidate the behavior of a given configuration without necessarily implementing it. Thus, infrastructure managers can see if their strategy, network components, and devices are enough to ensure a good QoS to end-users.

Therefore, this work presents a solution to predict the response time of Industrial Internet of Things (IIoT) applications considering specific scenarios in SDN-Fog Environments. The main contributions of this work are: *i)* an analysis of the relationship among Fog IIoT Factory Scenario, DCDP, and CPP; *ii)* a new network behavior prediction approach considering the context of SDN-Fog for IIoT; *iii)* the sharing of a framework with pre-trained machine learning models to predict the response time of IIoT networks in the SDN-Fog context.

The remaining of the paper is structured as follows: Section II presents related work. Section III details the work proposal. Section IV presents the experimental design with the description of the Fog IIoT Factory Scenario, the presentation of the machine learning framework, and the details of the data set and the experiments. Section V shows the results obtained and discussions about them. Finally, Section VI concludes the paper and highlights future work.

II. Related Work

In order to predict network behavior, Hardegen *et al.* [13] implemented a model based on Deep Neural Networks (DNNs) to perform 240 flow characteristics prediction experiments. In addition, they presented a case-based study where throughput, flow, and topology were the target predictor variables for specific paths. For predicting network traffic, Aldhyani *et al.* [16] proposed a method using sequence mining, combining non-crisp Fuzzy-C-Means (FCM) clustering and the weight exponential method to improve deep learning Long Short-Time Memory (LSTM) and Adaptive Neuro-Fuzzy Inference System (ANFIS) time series models. When the scenario is specific to cloud-based networks, Nourikhah *et al.* [17] analyzed the response time of 10 real-world services and proposed a prediction model using time series analysis, such as, naïve, mean, auto regressive integrated moving average (ARIMA), and auto regressive fractionally integrated moving average (ARFIMA) methods.

Considering networks for IoT devices, White *et al.* [18] presented the IoTPredict algorithm, which consists of a neighborhood-based prediction approach taking into account the response time and the throughput of a network released by Zheng *et al.* [19]. In addition, Yang and Zhang [20] considered time similarity to predict the response time of IoT service. The method proposed by the authors combines three main steps: (1) a calculation method of service response time similarity between different users; (2) the initial similarity values are adjusted, and a similarity threshold selects similar neighbors to improve prediction accuracy; (3) using a densified user-item matrix, service response time is predicted by collaborative filtering for currently active users. Table I summarizes the main differences between this paper and related work on the nature of the network prediction features, on the scenario, cloud-based, or not, on the context, IoT, or not, and on the scenario's solution, Fog-SDN, or not.

TABLE I
Comparison of Related Works

Work	Network prediction feature	Cloud based	IoT context	SDN-Fog problem
[13]	Flow	No	No	No
[16]	Traffic	No	No	No
[18]	Response time	No	Yes	No
[20]	Response time	No	Yes	No
[17]	Response time	Yes	No	No
This work	Response time	Yes	Yes	Yes

III. Work Proposal

Based on supervised and unsupervised learning, we created a Machine Learning (ML) framework to predict the response time of SDN-Fog environments before performing all possible network setups. Fig. 1 illustrates how the ML approach can compose the MILP-based solution. In this context, predicting response times of possible network scenarios (The ML Framework) helps select the most suitable one concerning the expected response time for IIoT applications. Thus, the optimization execution in a later stage (MILP Allocation) becomes less costly and in less time, as the number of scenarios is reduced in the first stage, unlike the previous approach.

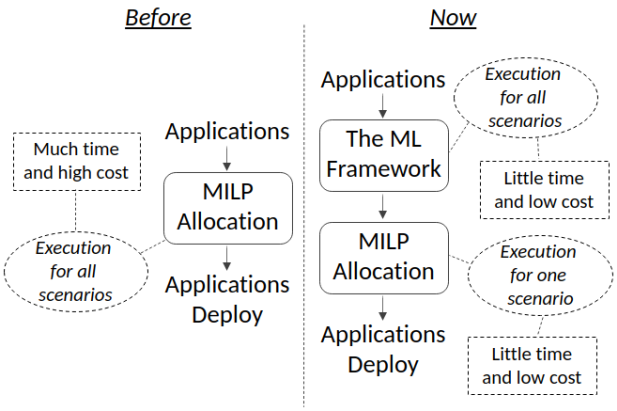

Fig. 1. The machine learning approach.

To ensure that the machine learning framework considers all network characteristics, it uses the data generated by Herrera *et al.* [11], [21] to train a pre-trained model that can predict with high accuracy the expected response time of a particular IIoT network setup, known as a scenario. This pre-trained model will be available as a default functionality of the framework. The framework evaluates how different ML approaches can predict continuous features in this context, such as linear regression, decision trees, and neural networks. In addition, the framework also aims to assess how the unsupervised machine learning technique based on dimensionality reduction impacts prediction.

IV. Experimental Design

In this section, we present the design and some details of the experiments.

A. Fog IIoT Factory Scenario

The Fog IIoT Factory Scenario is similar to the environment proposed by Herrera *et al.* [11]. It is based on fog computing to support the QoS requirements of IIoT devices and describes the details needed to predict applications' response time in this context. Fig. 2 illustrates this scenario. It consists of IIoT devices installed in robots and ten fog servers that provide computing services to the devices. Each fog server is directly connected to an SDN switch and has an 800 MHz CPU and 1 GB of RAM. Regarding SDN controllers, the environment includes at least one controller for each switch, being a classic SDN control model.

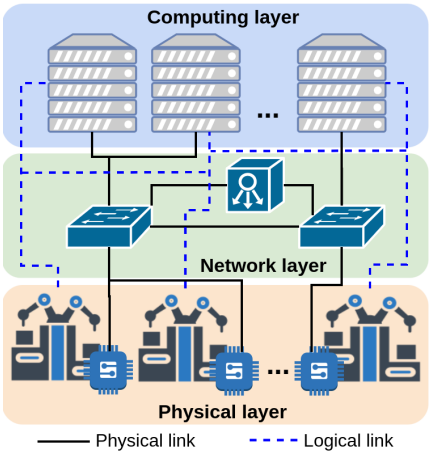

Fig. 2. Fog IIoT factory scenario.

An application that manages the information of robots was designed as a composition of a series of microservices. This application collects the robots' status through sensors connected to IIoT devices and processes the data in fog servers to provide commands accordingly. The commands processed by the microservices can be ordered separately or combined through workflows. Each workflow executes a specific functionality, by linking the deployed microservices depending on the desired behavior. Thus, each microservice has a computational complexity to perform the processing of a given activity. This processing is done by MCycles, being among 100, 500, or 1000.

The combination of the execution time of a microservice plus the communication latency of the network nodes results in the response time of specific functionality. Furthermore, this combination is directly influenced by the location of the microservices on different fog servers and by the SDN controllers available in the environment. All of these factors can cause response time delays. Thus, the reduction of latency between the various network nodes must be sought through the combination of DCDP and CPP to ensure that the response time of each specific functionality is consistent with the strict requirements of IIoT applications.

B. The Data Set

The IIoT factory data set consists of 17 features per scenario: computing nodes features such as ID, clock speed (MHz) and RAM (MB); switches IDs; links features such as the node ID that is the source of the link, ID of the node that is the destination of the link, latency (s) and capacity (MB); microservices features such as ID of the microservice, execution cycles (MCycles), RAM required (MB), input and output size of the microservice (MB); and workflows features such as ID, the IDs list of the microservices requested in the workflow, ID of the computing node requesting the workflow and whether the workflow should have a response (always true).

C. Machine Learning Framework

The modeling and execution of the SDN-Fog deployment problem in MILP can jointly solve DCDP and CPP to optimize the response time for IIoT applications. However, models' execution requires high computational power, in addition to consuming a relatively large amount of time for networks with more than 300 nodes [11]. Thus, to avoid these issues, the developed framework uses network environment features to predict the response time of IIoT scenarios. Fig. 3 presents the two main parts of the framework. The first part is responsible for processing the input, and output data of the DADO [11], [21], respectively, to compose the scenarios that are the primary inputs to the ML models. The second part is composed of the implementation and execution of supervised and unsupervised learning approaches to evaluate the response time prediction behavior of scenarios composed by the first part.

To compose the scenarios, the data processing step uses the Fog IIoT factory features along with a response time estimation for different workflows to compose the scenarios. Each scenario had features extended statistically according to their average, minimum, 25%, 50%, 75%, and maximum values, accounting for 113 features in total.

The framework composes Linear Regression, eXtreme Gradient Boosting (XGBoost) Regressor, and LSTM network on supervised ML models. The Linear Regression objective is to find the best estimates for its coefficients, which minimize errors in predicting target features. It fits a linear model to minimize the residual sum of squares between the observed

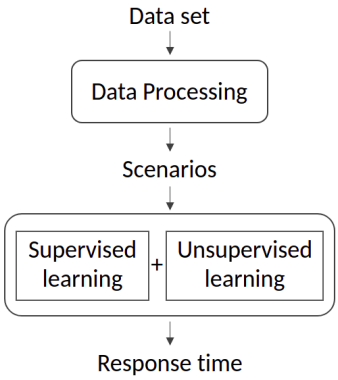

Fig. 3. Machine learning framework.

targets in the data set, in this context, response time, and the targets predicted by the linear approximation. XGBoost is an implementation of Gradient Boosted Decision Trees designed for speed and performance proposed by Chen [22]. This algorithm aims to minimize the loss function, choosing distinct optimal adjustment values for each region of the tree instead of a single one for the entire tree. Based on this concept, XGBoost calculates the fit of its sequential trees according to the weighted errors of the predecessor trees to predict the feature target of the model. Another model presented in the framework is the LSTM network, a type of Recurrent Neural Network (RNN) developed by Hochreiter and Schmidhuber [23] to avoid the long-term addiction problem or vanishing gradients problem, that causes stop learning from further training. LSTM network has a chain structure that, instead of having a single repetition module neural network layer, has four related. This approach allows data-dependent controls into the RNN cell approach to be implemented, ensuring that the gradient of the objective function does not vanish. An LSTM has three such gates to protect and control the state of the cell. In this way, long-term information that was lost during the execution of traditional models based on RNN is now maintained and used in a controlled manner in multi-layer networks [23], [24].

We apply Principal Component Analysis (PCA) to assess how unsupervised ML impacts response time prediction. PCA allows us to reduce the size of a data set while maintaining the features that have the most significant variance contribution [25], [26].

D. Experiments

We present an experimental comparison of ML models implemented using Scikit-learn and XGBoost for all scenarios composed in the data set processing step. The proposed framework and the Jupyter Notebooks used in the experiments are available on Github[1]. We observed three topology categories: *small* (10 IIoT and 10 fog nodes), *medium* (25 IIoT and 15 fog nodes) and *large* (50 IIoT and 25 fog nodes). The universe of data used for training the models contains 84 network scenarios, accounting 880 nodes for small, 1000 nodes for medium and 1125 nodes for large topology. To test the proposed models, the data universe includes 47 scenarios that total 880 nodes for small and 240 nodes for large topology with 30 fog nodes instead of 25. The difference in the amount of fog nodes is original from the source data set and was kept to also evaluate the models performance in distinct structures than those used for training.

We carried out a study on the main components using PCA, and we verified four components to compose with supervised learning models. Fig. 4 shows the number of components needed to explain variability of the data. The first component corresponds to 56.87%, the second to 21.44%, the third to 13.01% and the fourth to 4.90% of variability. Therefore, selecting a number of components above four to perform the dimensionality reduction will not drastically influence the model, since other components correspond to only 3.78% of variability. According to the calculated coefficient matrix, the features that most contributed to the first component formation are related to the computational power of IoT devices, such as processing and memory. For the second component, the features of network nodes number, IoT, Fog and Switch, and related to links, such as capacity and latency, were the ones that contributed the most. For the features related to the flow steps, they formed the third component. Finally, for the fourth and last component, the most important features are the workflows sizes and the number of devices per workflow.

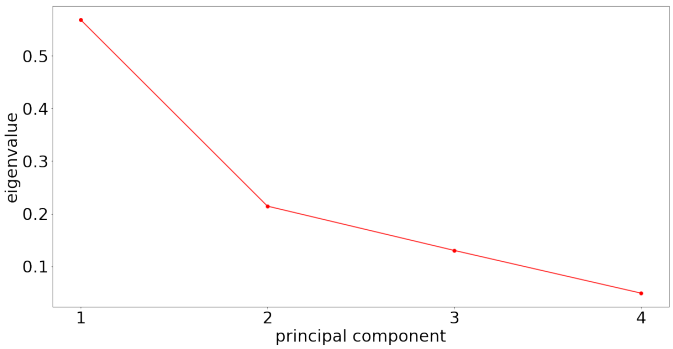

Fig. 4. The impact of the first four principal components.

Model performance validation was implemented using 5-fold cross-validation, i.e., five different measurements validate the implemented models, assessing their generalization ability of predictive and preventing overfitting. In addition, all data used in testing step were totally distinct from those used for training the models.

V. Results and Discussion

Fig. 5 and Fig. 7 illustrate a graphical representation of the Linear Regression and XGBoost performance on the training step, and Fig. 6 and Fig. 8 on the testing step. When comparing the training and testing steps, a factor that impacted the performance drop was the low number of test scenarios, one of the limitations found. Despite this, it is possible to verify by the approximation of the blue line, which represents the

[1]https://github.com/rodrigoney/ml-sdn-dado

predicted response times, that the XGBoost model had the best performance when compared to the Linear Regression.

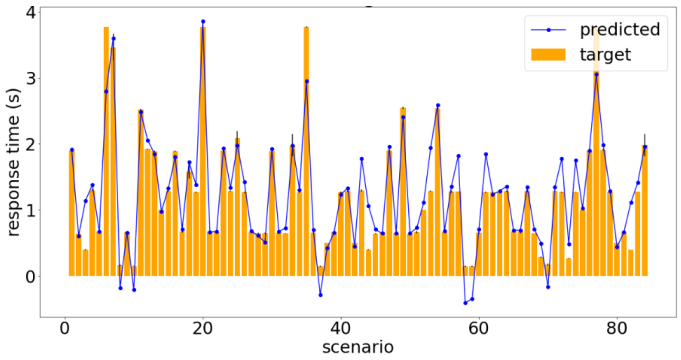

Fig. 5. Linear regression training performance.

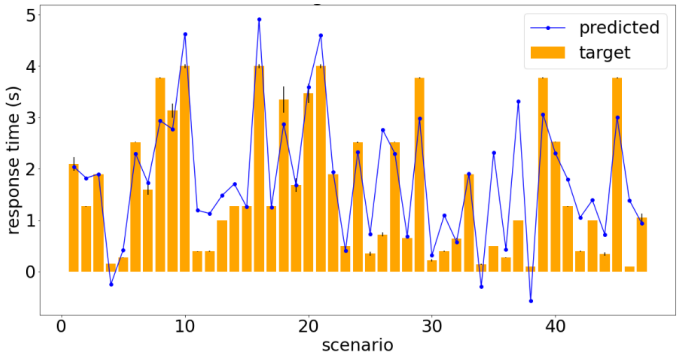

Fig. 6. Linear regression testing performance.

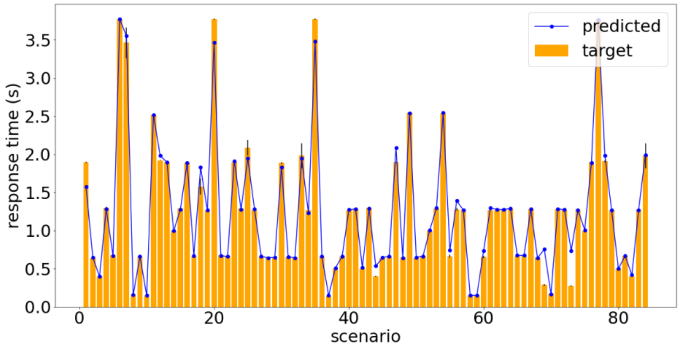

Fig. 7. XGBoost training performance.

Mean Squared Error (MSE) and R^2, or R-squared, score methods were used to evaluate the models accuracy besides the graphical representation. While R2 Score represents the proportion of the variance for a dependent variable that is explained by an independent variable or variables in a regression model, MSE measures the average squared difference between estimated values and the actual value. The resulting accuracy of the implemented models is described in Table II.

Although the four main components represent 96.22% of all data, it is possible to observe that the PCA strategy does not improve both models' performance, neither for the MSE

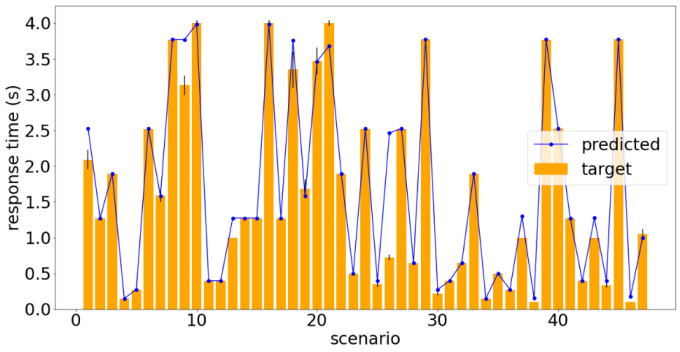

Fig. 8. XGBoost testing performance.

reduction nor the R2 Score increase. Also, when comparing the Linear Regression with PCA and XGBoost with PCA models, it is possible to see a greater negative impact on XGBoost. This phenomenon occurs due to the ineffective reduction of the error calculated through the low composition of features in successive gradient boosted trees. As XGBoost uses the various features to calculate the error and compose it in the learning function, reducing the features of the IIoT environment shown is not a good strategy to predict the network response time. Furthermore, the XGBoost model achieves the best performance at an average rate of 49.60 milliseconds for MSE and 96.63% for R2 Scored.

TABLE II
MODELS ACCURACY WITH RAW DATA

Model	MSE		R2 Score	
	Training	Test	Training	Test
Linear Regression	0.0822	0.4999	0.8892	0.7025
Linear Regression with PCA	0.1547	0.8215	0.7917	0.5111
XGBoost	0.0107	0.0885	0.9854	0.9473
XGBoost with PCA	0.1977	0.9754	0.7338	0.4196

In order to deepen the study of the impacts of data in relation to the implemented models, two approaches were carried out, normalization and z-score technique application. Table III shows that applying data normalization to the Linear Regression model improves its performance, unlike the XGBoost model, which is improved by applying the z-scored technique. However, both applied techniques do not improve the performance of the models, Table II, when run using raw data from the IIoT network.

TABLE III
MODELS ACCURACY WITH DATA NORMALIZED AND Z-SCORED

Model	MSE		R2 Score	
	Training	Test	Training	Test
Linear Regression Normalized	0.0857	0.8065	0.8845	0.7392
Linear Regression Z-Scored	0.0924	1.3510	0.8755	0.5632
XGBoost Normalized	0.1796	1.064	0.7582	0.6558
XGBoost Z-Scored	0.0108	0.8904	0.9854	0.7121

VI. Conclusion

This work presented an approach to predict IIoT network response time in an SDN-Fog context. The proposed framework consists of four approaches: Linear Regression, Linear Regression with PCA, XGBoost, and XGBoost with PCA. The experiments showed that the XGBoost is the most suitable for predicting response time in an SDN-Fog environment in terms of accuracy. Overall, the results showed that the prediction of the response time of scenarios was close to the times obtained by solving the problem given by Herrera *et al.* [11], in other words, when compared to the Herrera *et al.* results [21], the framework had an average of fewer than 50 milliseconds of variation according to the best model. Moreover, the approach based on ML models was executed in less than one second for Linear Regression and XGBoost models. Another factor to be considered is the composition of supervised and unsupervised learning models to predict the network response time. In this case, the unsupervised PCA technique did not prove to be an advantageous alternative for the IIoT context presented when composed with the supervised developed models. Therefore, response time prediction in IIoT networks based on ML models shows to be a viable and advantageous option for the SDN-Fog context in network setup planning time. As future work, we seek to evolve and adjust the framework to consider strict network response times and evaluate others machine learning approaches based on neural networks.

Acknowledgments

This work was developed with support of the Conselho Nacional de Desenvolvimento Científico e Tecnológico (CNPq). We thank the research agency for the financial proc. 130762/2021-0. The research is part of the INCT of the Future Internet for Smart Cities funded by CNPq proc. 465446/2014-0, Coordenação de Aperfeiçoamento de Pessoal de Nível Superior – Brasil (CAPES) – Finance Code 001, FAPESP proc. 14/50937-1 and proc. 15/24485-9. It is also part of the FAPESP proc. 18/22979-2, proc. 18/23098-0 and the project RTI2018-094591-B-I00 (MCI/AEI/FEDER,UE).

References

[1] M. d. V. D. da Silva, A. Rocha, R. L. Gomes, and M. Nogueira, "Lightweight Data Compression for Low Energy Consumption in Industrial Internet of Things," in *Proceedings of the IEEE 18th Annual Consumer Communications Networking Conference (CCNC)*, 2021, pp. 1–2.

[2] D. M. Batista, A. Goldman, R. Hirata, F. Kon, F. M. Costa, and M. Endler, "InterSCity: Addressing Future Internet research challenges for Smart Cities," in *Proceedings of the 7th International Conference on the Network of the Future (NOF)*, 2016, pp. 1–6.

[3] I. Alam, K. Sharif, F. Li, Z. Latif, M. M. Karim, S. Biswas, B. Nour, and Y. Wang, "A Survey of Network Virtualization Techniques for Internet of Things Using SDN and NFV," *ACM Computing Surveys*, vol. 53, no. 2, pp. 1–40, 2020.

[4] Z. Chi, Y. Li, H. Sun, Y. Yao, and T. Zhu, "Concurrent Cross-Technology Communication among Heterogeneous IoT Devices," *IEEE/ACM TON*, vol. 27, no. 3, pp. 932–947, 2019.

[5] H. Xu, W. Yu, D. Griffith, and N. Golmie, "A Survey on Industrial Internet of Things: A Cyber-Physical Systems Perspective," *IEEE Access*, vol. 6, pp. 78 238–78 259, 2018.

[6] A. Yousefpour, C. Fung, T. Nguyen, K. Kadiyala, F. Jalali, A. Niakanlahiji, J. Kong, and J. P. Jue, "All One Needs to Know about Fog Computing and Related Edge Computing Paradigms: A Complete Survey," *Journal of Systems Architecture*, vol. 98, pp. 289–330, 2019.

[7] B. Choudhury, S. Choudhury, and A. Dutta, "A Proactive Context-Aware Service Replication Scheme for Adhoc IoT Scenarios," *IEEE Transactions on Network and Service Management*, vol. 16, no. 4, pp. 1797–1811, 2019.

[8] N. Wang, K. H. Ho, G. Pavlou, and M. Howarth, "An Overview of Routing Optimization for Internet Traffic Engineering," *IEEE Communications Surveys & Tutorials*, vol. 10, no. 1, pp. 36–56, 2008.

[9] F. Y. Okay and S. Ozdemir, "Routing in Fog-Enabled IoT Platforms: A Survey and an SDN-Based Solution," *IEEE Internet of Things Journal*, vol. 5, no. 6, pp. 4871–4889, 2018.

[10] T. Das, V. Sridharan, and M. Gurusamy, "A Survey on Controller Placement in SDN," *IEEE Communications Surveys & Tutorials*, vol. 22, no. 1, pp. 472–503, 2020.

[11] J. L. Herrera, J. Galán-Jiménez, J. Berrocal, and J. M. Murillo, "Optimizing the Response Time in SDN-Fog Environments for Time-Strict IoT Applications," *IEEE Internet of Things Journal*, pp. 1–1, 2021.

[12] M. Pióro and D. Medhi, "CHAPTER 5 - General Optimization Methods for Network Design," in *Routing, Flow, and Capacity Design in Communication and Computer Networks*, ser. The Morgan Kaufmann Series in Networking, M. Pióro and D. Medhi, Eds. Morgan Kaufmann, 2004, pp. 151–210.

[13] C. Hardegen, B. Pfülb, S. Rieger, and A. Gepperth, "Predicting Network Flow Characteristics Using Deep Learning and Real-World Network Traffic," *IEEE Transactions on Network and Service Management*, vol. 17, no. 4, pp. 2662–2676, 2020.

[14] D. Mirkovic, G. Armitage, and P. Branch, "A Survey of Round Trip Time Prediction Systems," *IEEE Communications Surveys & Tutorials*, vol. 20, no. 3, pp. 1758–1776, 2018.

[15] S. Bhulai, S. Sivasubramanian, R. van der Mei, and M. van Steen, "Modeling and Predicting End-to-End Response Times in Multi-tier Internet Applications," in *Managing Traffic Performance in Converged Networks*, L. Mason, T. Drwiega, and J. Yan, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2007, pp. 519–532.

[16] T. H. H. Aldhyani, M. Alrasheedi, A. A. Alqarni, M. Y. Alzahrani, and A. M. Bamhdi, "Intelligent Hybrid Model to Enhance Time Series Models for Predicting Network Traffic," *IEEE Access*, vol. 8, pp. 130 431–130 451, 2020.

[17] H. Nourikhah, M. K. Akbari, and M. Kalantari, "Modeling and Predicting Measured Response Time of Cloud-Based Web Services using Long-Memory Time Series," *The Journal of Supercomputing*, vol. 71, pp. 673–696, 2015.

[18] G. White, A. Palade, C. Cabrera, and S. Clarke, "IoTPredict: Collaborative QoS Prediction in IoT," in *Proceedings of the IEEE International Conference on Pervasive Computing and Communications (PerCom)*, 2018, pp. 1–10.

[19] Z. Zheng, Y. Zhang, and M. R. Lyu, "Investigating QoS of Real-World Web Services," *IEEE Transactions on Services Computing*, vol. 7, no. 1, pp. 32–39, 2014.

[20] H. Yang and L. Zhang, "Response Time Prediction of IoT Service Based on Time Similarity," *Journal of Computing Science and Engineering*, vol. 11, no. 3, pp. 100–108, 2017.

[21] J. L. Herrera, J. G. Jiménez, J. Berrocal, and J. M. Murillo, "Optimizing Response Time in SDN-Edge Environments for Time-Strict IoT Applications," 2020, https://dx.doi.org/10.21227/m9s7-1q10. Accessed at October 29, 2021.

[22] T. Chen and C. Guestrin, "XGBoost: A Scalable Tree Boosting System," in *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2016, p. 785–794.

[23] S. Hochreiter and J. Schmidhuber, "Long Short-Term Memory," *Neural Computation*, vol. 9, no. 8, pp. 1735–1780, 11 1997.

[24] A. Sherstinsky, "Fundamentals of Recurrent Neural Network (RNN) and Long Short-Term Memory (LSTM) Network," *Physica D: Nonlinear Phenomena*, vol. 404, p. 132306, 2020.

[25] M. Alsharif, A. Hilary, K. Yahya, and S. Chaudhry, "Machine Learning Algorithms for Smart Data Analysis in Internet of Things Environment: Taxonomies and Research Trends," *Symmetry*, vol. 12, p. 88, 01 2020.

[26] D. Rafique and L. Velasco, "Machine Learning for Network Automation: Overview, Architecture, and Applications [Invited Tutorial]," *J. Opt. Commun. Netw.*, vol. 10, no. 10, pp. D126–D143, Oct 2018.