A Novel Federated Fog Architecture Embedding Intelligent Formation

Hala Shamseddine, Jana Nizam, Ahmad Hammoud, Azzam Mourad, Hadi Otrok, Haidar Harmanani, and Zbigniew Dziong

Abstract

Network delays cause disturbance and reduction in the Quality-of-Service (QoS) for Internet-of-Things (IoT) while end-users are running critical real-time services. In parallel, federated fogs are not effective when formed without considering the performance perceived by the end-users. This article presents a novel architecture for the federated fog concept and proposes an adaptive and intelligent federation formation approach using Genetic Algorithm and Machine Learning models. Fog federations serve as a solution for fog providers to offer the required QoS they serve. Such a concept allows efficient distribution of load among multiple fog providers that share their resources. Throughout this process, the issue of QoS deterioration, due to local overloads, is relatively solved. Hence, the end users can enjoy a delay-free experience when using real-time applications. Real data is used to evaluate the proposed architecture and formation mechanism. The results show a notable improvement in the throughput as well as a decrease in the response time for the services requested.

Introduction

Network-Based Distributed Computing emerged in the 1960s, and four decades later, this paradigm turned into what is known nowadays by Cloud Computing, offering a variety of services at the personal and organizational levels. Similarly, Internet-of-Things (IoT) devices have gained vast popularity lately due to their low deployment cost as they upload collected data to powerful devices (i.e., cloud servers) for fast processing [1]. A drawback for such a popularity consists of exhausting the network resources when uploading the sensed IoT data to the clouds [2]. The fallout from this exhaustion will result in jamming the network and requiring additional time for processing IoT requests. With this in mind, and with the fact that some IoT applications require low response time as well as high throughput, as users might be running demanding real-time applications and services, the Fog Computing paradigm has emerged. In general, fog computing enhances the Quality-of-Service (QoS) of the IoT applications by offering computing servers closer to the IoT devices than the cloud's. Such a technology can reduce the time of uploading the data and getting responses from the servers [3]. With fog computing in-hand, Application Content Providers are able to develop real-time applications without considering the network delays as a barrier when utilizing such services due to the short distance between the fog nodes and end devices.

Nevertheless, fog nodes are limited in terms of resources compared to cloud servers because of their high deployment cost. As a consequence, congested areas may suffer from a diminished QoS when users request services from fog nodes. Typically, idle fog nodes wait for an incoming set of requests to process them immediately. However, if the node is already occupied and is performing a certain set of tasks, newly arrived tasks will be queued and wait for their turn to get scheduled and processed.

In parallel, the concept of federations for cloud computing was presented in 2009 [4]. Similar to clustering, such a concept consists of uniting computing servers from several cloud providers to increase their processing capabilities in order to respond to a certain high demand for resources [5]. In effect, the members of the cloud federations will benefit by increasing their profit through allocating their idle servers and expanding their geographical footprints without the need of new points of presence. Many research efforts consist of building models on top of the federation architecture, whether to optimize the latter's formation process, reinforce its security, or enhance the service provided [6]. A few scholars adopted this ideology for the fog paradigm in an attempt to achieve a fog federation formation model. Their aim was to surpass the processing boundaries caused by relying on a single fog provider when deploying services. For instance, some efforts were tailored toward enhancing the formation of the federated fog through increasing the payoff of the fog providers [7, 8, 9]. Others tackled specific application improvements such as providing better video streaming services for the end-users [10]. To the best of our knowledge, no work has investigated a comprehensive federated fog architecture. In addition, in a real-time processing environment, there is a need for forming these fog federations according to how the service will affect the performance for the end devices in order to grant a satisfying QoS. Inspired by the introduction of a broker entity for managing Cloud-related auctions, we propose in this article a novel federated fog architecture where a broker is responsible for forming and maintaining fog federations. In the proposed approach, the broker relies on an adaptive and intelligent approach for forming federations using *Genetic* and *Machine Learning* techniques, combined. The genetic

Digital Object Identifier:
10.1109/MNET.011.2000505

*Hala Shamseddine, Jana Nizam, Azzam Mourad, and Haidar Harmananis are with Lebanese American University;
Ahmad Hammoud and Zbigniew Dziong are with Ecole de Technologie Superieure (ETS); Hadi Otrok is with Khalifa University.*

model is a meta-heuristic applied to explore a diverse set of possible federated fog formations in the wide search space, and then select the best one among them in terms of offered service quality, whereas the machine learning mechanism is invoked to predict the network performance and load between the fog node running the requested service and the user requesting that service. It is worthwhile to mention that both of the techniques applied in our approach were used as separate in the literature. For instance, some scholars have considered enforcing a genetic model for seeking the best possible cloud federation formation in short time frames [11]. Others have considered it for solving the resource scheduling dilemma [12]. The machine learning technique was mainly adopted for resource management and security problems in fog computing [13]. Experimental evaluation indicates that the proposed scheme is offering a reliable set of fog federations with high QoS compared to Evolutionary and Random approaches. Several metrics assess the network performance of the offered services. We design two performance metrics (i.e., response time and throughput) to predict the service quality of the federation. The main contributions of this work are summarized as follows:

- Devising a novel federated fog architecture that embeds all the participating entities and takes into account real-life parameters and constraints within a fog environment. To the best of our knowledge, this is the first attempt at addressing those aspects within a comprehensive fog federation architecture.
- Elaborating an adaptive federation formation process that is based on Genetic and Intelligent models. The proposed scheme achieves efficient results in terms of the number of satisfied users.
- Proposing a Machine Learning model for evaluating the fitness of the federations and dynamically adapting the Genetic Model for improving its results from one generation to another.

The rest of the article is organized as follows. In the following section, we illustrate the proposed federated fog architecture and discuss its components. Following that, we propose the fog federation formation mechanism through utilizing a genetic algorithm technique. Then we discuss the machine learning models used and their effectiveness. Following that, we evaluate the performance of the proposed architecture. Finally, we conclude the article.

Scheme Overview

In this section, we present the fog federation architecture, depicted in Fig. 1, and discuss each component/entity. As illustrated in this architecture, there exists a set of fog providers with their fog nodes distributed geographically within a certain city. Application content providers want to provide their services for the users in exchange for profit. The users, on the other hand, are expecting these services to run smoothly in order to satisfy their needs. Users cannot deploy all of the services they need on their devices due to resource limitations, thus, these services are placed on available fog nodes. This way, users request services hosted on the fog node running the required service in order to get processed and return the desired outputs. At the same time, fog providers also have limited resources, as discussed earlier; when a lot of requests need to be processed, the system performance could degrade. The strength of the proposed architecture consists of the broker advancing a location-aware adaptive and intelligent fog federation formation in order to deliver the best performance. The fog federation formation process consists of executing genetic and learning models to explore the best way to form these federations in a short time while maximizing the network performance. To further elaborate, we interpret these components in an environment where an autonomous driving application is enabled for the vehicles. Accordingly, below is a detailed description of each entity.

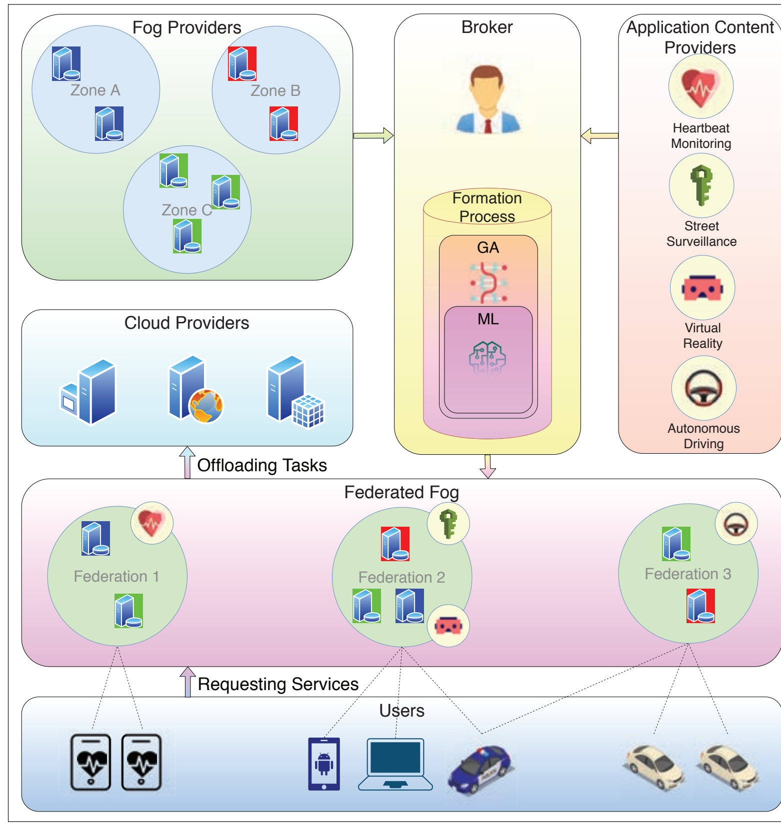

FIGURE 1. Federated Fog Architecture.

Application Content Providers: The parties that want to offer their services in order to gain profit. For instance, a company that developed an Autonomous Driving System (ADS) would want to offer it for its subscribers with low latency.

Users: The main entities for whom the whole architecture is designed. A user may need to request a specific service deployed somewhere else. Typically, the applications are offered on the network so that users could request them from particular locations. For example, a user could be considered as a vehicle in need of using the ADS in order to enable auto-pilot mode.

Fog Providers: The fog resource owners. Generally, they are located in certain areas near the end-users in order to provide them with a smooth service experience. Fog providers may own several fog nodes, and each fog node has the ability to deploy one or more services from the Application Content Providers. A fog node running an

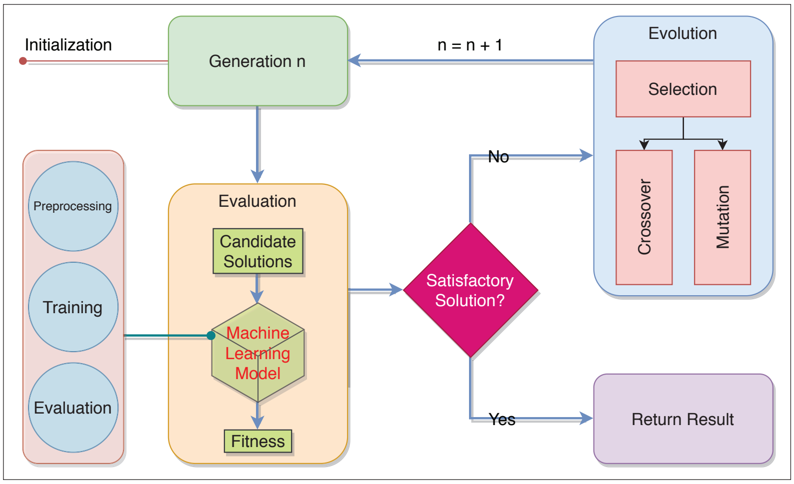

FIGURE 2. Adaptive and Intelligent Formation Process.

ADS service may receive requests from vehicles in need of processing a chunk of data.

Cloud Providers: The entities with powerful servers. Commonly, the fog providers may offload some of the heavy received requests to the clouds to avoid overwhelming their costly fog nodes. Many works in the literature focused on this specific Cloud-Fog offloading mechanism. Some scholars tried to resolve it by optimizing the task scheduling process [14].

Broker: Responsible for the performance optimization of the proposed architecture. As a central authority, the broker's responsibility is to devise a federated solution that can satisfy all entities by reaching a satisfactory service quality of the designated applications. Specifically, the broker gets contacted by the Application Content Providers in order to deploy their services. The broker, in return, checks down the idle fog nodes, and then advances a fog federation formation technique that can maintain a good QoS. It is worth mentioning that this entity does more jobs than just forming the fog federation such as managing members, monitoring federations, and so on. However, due to space constraints of this article, we will only focus on the formation process, leaving the remaining roles for future work.

The broker may determine that the ADS service needs to be placed on n number of fog nodes belonging to n different fog providers, distributed geographically. This way, the application continues to run smoothly for all subscribing vehicles. The techniques used in the formation are discussed in detail in the next sections.

Adaptive Fog Federation Formation

One of the main advantages of the proposed architecture is the broker's acknowledgment of the supply and demand of resources by both the fog providers and application content providers, respectively. The broker then needs to proceed with placing services on the fog nodes through advancing a fog federation formation technique. The main objective is to provide a satisfying QoS for the largest possible number of end-users. Hence, we devise an adaptive and intelligent federated fog formation mechanism using an enhanced Genetic algorithm model which relies on a machine learning technique for evaluating the fitness of the evolved solutions. The mechanism is depicted in Fig. 2.

The Genetic Algorithm, known as GA, mimics the occurrence of natural populations that breed and produce offspring. It imitates the natural selection process where the fittest individuals are selected for reproduction in order to produce offspring of the next generation. This technique is being applied by many researchers to solve real-world complex problems and to obtain sub-optimal results in a short time. For instance, some scholars used such a technique for scheduling fog tasks [12]. Others applied it in the field of ameliorating Cloud Federations for maximizing the monetary profit of the cloud providers [11].

In a nutshell, GA is a time-aware algorithm that explores the search space in order to find a satisfying solution for a specific real-life problem. We apply GA as a methodology for finding the best fog federation formation in a short time. The three main components of this algorithm are the "Initialization," "Evaluation," and "Evolution."

Initialization: The Genetic model starts from a generated set of candidate solutions, called Initial Population. Each one of them has eligibility for becoming a valid solution. In our problem, a candidate solution represents a unique set of formed federations. To digitize a solution, we apply a permutation-based encoding procedure in which we portray each set of formed federations by relying on the unique providers' identifiers for grouping providers.

Evaluation: To state whether a certain formation is considered as a good or bad solution, there is a need for devising a function that takes the solution as input, and outputs a value that can be used to select the best solution. To demonstrate, we suppose that a user u_i requests a service s_j hosted on a fog node deploying that service. If the response time rt_{u_i,s_j} and throughput tp_{u_i,s_j} of that particular invocation meet the minimum required set by the application owner $\hat{rt}_{s_j}$ and $\hat{tp}_{s_j}$, respectively, then the invocation is considered satisfying. Otherwise, the solution will be penalized by 1 on behalf of each dissatisfactory request. To obtain the penalty rate, we divide the total dissatisfied invocations by the total number of invocations. Intuitively, the objective is to minimize that penalty rate as much as possible. To avoid the penalties in real scenarios, we employ learning models for predicting its throughput and response time. These models are presented in the next section.

Evolution: This component is the transitory function that shifts the solution from one generation to another. It is split into three sub-components: Selection, Crossover, and Mutation.

Selection: The selection sub-component is inspired by the natural selection process in which not all candidate solutions are worth keeping for the next generation. It consists of selecting only a few candidate solutions that remain in the system in order to pass their genes to the new solutions. The selected ones are modified into new candidate solutions by applying genetic operators on them (i.e., crossover and mutation).

Crossover: The crossover function is applied to a pair of candidate solutions. Using a binary mask (i.e., a set of 0s and 1s), the pair can exchange genes accordingly in order to obtain two new offsprings inheriting a formation that is similar to the formation of their ancestors. In other words, the two newly created candidate solutions represent two fog federation formations that are similar to both formations of the selected pair of parents at the same time.

Mutation: The mutation is the genetic entropy which slightly alters a solution in a random manner. More precisely, a solution can become more effective by swapping two fog providers from two different federations. Hence, the mutation sub-component applies small changes to the formed federations in order to explore more solutions in the search space.

The newly evolved population again goes through the Evaluation phase until reaching a satisfying solution, as illustrated in Fig. 2.

Prediction of QoS Metrics through Machine Learning Models

Before forming the federations and deploying the services on the fog nodes, a study on how the IoT applications are going to perform should be established in order to maintain the minimum required QoS for these applications. To elaborate, IoT applications require low response time and high throughput, as users might be running demanding real-time applications and services. Hence, when these metrics (i.e., throughput and response time) are maintained as required, users' satisfaction will occur.

Intuitively, a user-service invocation can only be accurately evaluated via its status information after it gets processed. However, in a complex environment with billions of possible invocations, there is a need to evaluate these requests using other methods due to time constraints and resources' exhaustion resulting when invoking all possible requests. Therefore, due to the fact that ignoring the missing invocations might lead to decreasing the quality of the formed federations, we employ machine learning to predict the throughput and the response time of user-service invocations.

Machine Learning is a subset of Artificial Intelligence (AI) in which a computing device can discover statistically significant patterns in the available data, then it would be able to predict new data [15]. Through feeding a learning model with training data, the model generates a prediction function that takes a service request as an input. This way, missing information about new invocations can be estimated in order to fulfill the new users' needs without having any previous history about them. In our approach, the generated machine learning models should be able to estimate the QoS values of the requests. The values predicted by the model are accurate to a certain extent, depending on the learning model used and the quality of the learning data. The genetic model, presented in the previous section, relies on the prediction model in order to estimate the quality of allocating a certain provider to a certain federation.

Below, we briefly explain the steps for advancing a machine learning model to predict the QoS metrics for the new invocations, and discuss the most practical and appropriate learning models yielding high accuracy.

Preprocessing: The primary step for running machine learning algorithms is to preprocess the data by combining features. Features chosen are those that define a good relationship in order to obtain high-quality learning data. Filtering techniques are also applied in this step, where noisy and incomplete rows get dropped from the data in order to avoid deteriorating the accuracy of the model. The data can be fetched from the server logs offered by the participating fog providers.

> One of the main advantages of the proposed architecture is the broker's Acknowledgment of the supply and demand of resources by both the fog providers and application content providers, respectively. The broker then needs to proceed with placing services on the fog nodes through advancing a fog federation formation technique.

In a location-aware fog environment, some metrics' values strongly depend on the locations of the users and providers. Thus, the features we are interested in are: *User-Id*, *User-Longitude*, *User-Latitude*, *Fog-Node-Id*, *Fog-Node-Longitude*, *Fog-Node-Latitude*, *Fog-Node-Status*, *Invocation-Throughput*, and *Invocation-Response-Time*. The selected features define a good relationship among others for reaching our desired predicted throughput and response time. The throughput and response time values depend on the location of the user as well as the location of the fog node and its status upon requesting. The locations are represented through the features: User-Longitude, User-Latitude, Fog-Node-Longitude and Fog-Node-Latitude.

The features User-Id and Fog-Node-Id were included to differentiate multiple users and fog nodes that may exist at the same location. In addition, the Fog-Node-Status feature gives an idea about the status of the fog node when the request was invoked.

Training: Once the data is preprocessed, the training phase is invoked. Each machine learning model has its own training mechanism and way of learning on the data for predicting accurate values. Most learning models that search through training data for empirical relationships tend to identify and exploit apparent relationships in the training data. However, it is worth mentioning that not all machine learning algorithms would fit properly on the data in hand due to the different levels of learning complexity each of them has in comparison with the data.

Evaluation: To evaluate each of the tested machine learning algorithms, three evaluation metrics need to be analyzed:
- R-squared, or R^2, to determine how close the data is fitted to the model.
- Mean Squared Error, or MSE, which measures the difference between the estimated and actual values.
- Median Absolute Error, or MAE, which shows the difference between the actual value of a data point and its predicted value.

In the next subsections, we discuss the implementation and results of several machine learning models training from the WSDream dataset provided.

Implementation of Machine Learning Algorithms

Five machine learning algorithms were used for the prediction of missing throughput and response time values in the user-service invocations present

	Response Time						Throughput					
	R^2		MSE		MAE		R^2		MSE		MAE	
	+ CV	- CV	+ CV	- CV	+ CV	- CV	+ CV	- CV	+ CV	- CV	+ CV	- CV
Decision Trees	0.428	0.373	2.28	2.5	0.038	0.044	0.8182	0.75	2156.694	2443.37	1.59	1.79
MLR	0.0101	0.0208	3.96	3.93	0.58	0.6	0.0121	0.01306	11719.36	11633.832	35.212	35.093
PLR	0.021	0.048	3.916	3.759	0.6	0.516	0.0129	0.0367	11709.946	11443.54	35.0658	30.915
KNN	0.289	0.269	2.842	2.878	0.176	0.1716	0.45	0.521	7434.20	6852.018	10.23	7.9145
Bagging	0.937	0.575	0.2489	1.695	0.0203	0.07	0.9796	0.859	241.474	1657.707	0.659	2.0634

TABLE 1. Results of machine learning algorithms applied with and without cross-validation (CV).

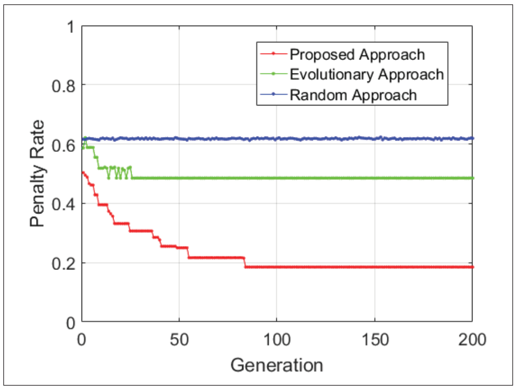

FIGURE 3. Fitness of the Formed Federations.

in the data set, including Decision trees, Multiple Linear Regression (MLR), Polynomial Linear Regression (PLR), K-Nearest Neighbors (KNN), and Bagging. All the mentioned algorithms were implemented for obtaining the most accurate throughput and response time of invocations, which is a pure regression problem. On the other hand, what differentiates these algorithms from each other is the way each of them learns on the existing data in order to predict these values. To start with, the first machine learning algorithm tackled is Decision Trees, which is a non-parametric supervised learning algorithm. The main goal of this learning method is the prediction of values of the target variable, which is throughput or response time in our case, through learning a set of decision rules, or if-then rules, then forming a tree having several paths according to the extracted rules set. Thus, the generated learning model will analyze possible solutions' paths that lead to the final predicted decision. Second, MLR is also implemented. This is another essential data analysis technique used to view whether there is a linear relationship between the dependent variable (throughput or response time) and the other features. MLR learns on the data by modelling the relationship among the multiple independent variables (denoted by X) and the dependent variable (throughput or response time, denoted by Y) through fitting a linear equation on the data. Therefore, every X value is linked to a value of Y and the data is separated by a straight line characterized by this equation. Third, we extend the MLR to a PLR algorithm in order to increase the accuracy of the former algorithm by generating a curve for most of the data points in the data set instead of a straight line while increasing the degree of the equation leads to getting better results. Fourth, we employ a KNN supervised machine learning algorithm that is non-parametric and relies on continuous data. As its name implies, KNN predicts a certain value by finding the *k* nearest neighbors of the query point in the data set, then computing the target value (predicted value) based on these *k* nearest points. Lastly, we apply the Bagging algorithm to improve the accuracy of certain machine learning algorithms. The Bagging regressor works by dividing the original dataset into random subsets called bags, fitting base estimators on each of these bags, and finally averaging the individual prediction resulting from each bag separately to form a final prediction.

Evaluation of the Machine Learning Models

By using the provided collective WSDream (https://github.com/wsdream/wsdream-dataset) dataset, a huge record of data concerning users requesting services, each from different locations, were analyzed. It is worth mentioning that the WSDream dataset contains real metrics collected from 339 users requesting services from 5,825 nodes to design and evaluate the machine learning models. The targeted output was based on the relation between these users and the web services through matrices of throughput, defined by the average rate of the successful message size delivery over a communication channel per second, and response time, defined as the time elapsed between the requesting of the service by the user and actually receiving it. The obtained values of the R^2 scores, MSE, and MAE are summarized in Table 1.

As can be seen from the table, the best scores were recorded for the Bagging algorithm for both throughput and response time predictions. In particular, for the three evaluation measures chosen, Bagging with *k*-fold (*k*=10) cross-validation gave the lowest error scores (MSE = 0.2489 and MAE = 0.0203 for response time, and MSE = 241.474 and MAE = 0.659 for throughput) and the highest R^2 scores (0.937 for response time and 0.9796 for throughput). Therefore, the Bagging was chosen as a model for predicting the throughput and response time values of the invocations when forming the federations.

Experiments: Evaluation of the Fog Federation Formation

Throughout this section, we discuss the results obtained by implementing the adaptive formation algorithm presented earlier, and compare them with the results obtained from two benchmark models. The first benchmark model is the Evolutionary game-theoretical model presented in [11]. The second benchmark model is based on a Random Formation technique. The metrics that we used to compare these models are the fitness of the solution per generation, the percentage of users achieving a satisfying throughput, and the percentage of users achieving a satisfying response time. The models are time-aware, meaning they evolve/vary with respect to time. Thus, the x-axis always represents the generation number, whereas the y-axis represents the designated series.

In Fig. 3, we study the fitness of the obtained solutions. As mentioned before, the fitness is considered as the penalty rate calculated by the number of users receiving a dissatisfactory service when making a request, with respect to the total number of invocations. Note that the proposed and the Evolutionary model solutions are improving with the increase in the generation number, while the Random model lacks the logic needed for enhancing the formation. In addition, our proposed mechanism outperforms the evolutionary one in terms of minimizing the penalty rate. For instance, the three solutions started from a very close rate at the first generation. However, a wide gap starts to appear at the 83rd Generation, where the proposed approach stabilizes at 0.185. This is mainly due to our advanced mechanism taking into consideration the QoS delivered to the users. To elaborate further, the advanced approach chooses the best candidate solution as an actual solution for the current generation, and nominates it as a candidate solution for the next generation. Hence, the penalty rate is decreasing over generations since better solutions, having lower penalty rates, are obtained after through the evolution process. When the penalty rate remains constant after a certain generation, it means that the model has converged and no better solutions could be found.

We also evaluated the percentages of satisfied users in terms of throughput and response time in Figs. 4 and 5, respectively. Clearly, the proposed approach increases the percentage of satisfaction and converges to 85 percent in terms of satisfied users in terms of throughput, whereas the Evolutionary approach gets stuck at 63 percent after convergence. The Random approach could not reach a better percentage than 54 percent. Also, the proposed model decreases the response time of the invoked request compared to the other two models. For instance, the percentage of users receiving a satisfying service in terms of response time, using our approach, is 78 percent after convergence and it surpassed the Evolutionary and the Random approaches by 37 percent and 44 percent, respectively. These results show that the percentages of users satisfied in terms of throughput and response time increase over generations since these percentages are inversely related to the penalty rates shown in Fig. 3. Therefore, when the penalty rate decreases over generations, which is the case as shown in Fig. 3, the number of dissatisfied users will also decrease, and hence the percentage of satisfied users in terms of throughput and response time will increase, as can be seen clearly in Figs. 4 and 5. The constant percentages of satisfied users are obtained in parallel with the model convergence observed in Fig 3. Thus, the proposed model and the related architecture improve significantly the users' QoS.

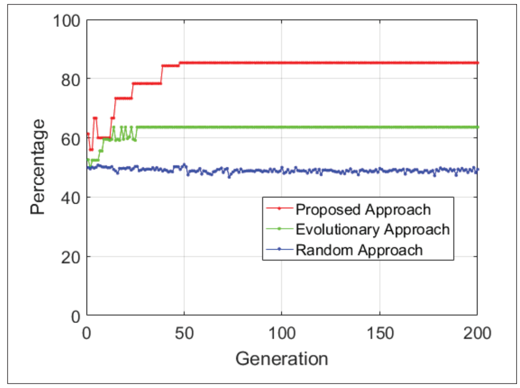

FIGURE 4. Percentage of users satisfied in terms of throughput.

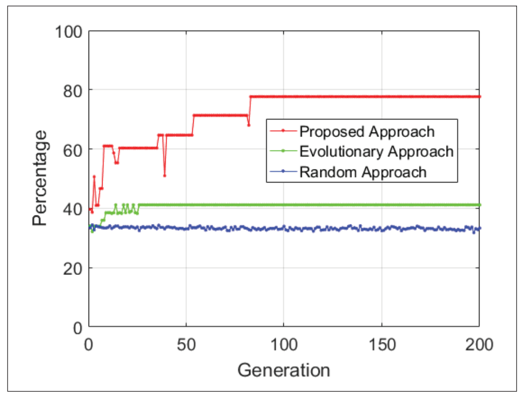

FIGURE 5. Percentage of users satisfied in terms of response time.

Conclusion

A fog federation can be defined as a group of fog providers merging their resources in order to better serve the users and thus improve the QoS and user satisfaction while decreasing the penalty on fog providers. This article presented a novel architecture for the federated fog and advanced a fog federation formation mechanism that increases the throughput and decreases response time by adopting a Genetic algorithm and utilizing Machine Learning. The latter was used to predict missing values of throughput and response time of user-service invocations, then these invocations were fed to the genetic algorithm that in turn was able to analyze them and provide us with the best combination of fogs that can federate with each other to better serve the users. Through the presented results, we proved the effectiveness of the proposed approach in increasing the number of users receiving a satisfying throughput and response time and therefore acquiring better QoS compared to a random formation mechanism and to an evolutionary game theoretical model from the literature. Future enhancements to this research study would include studying the privacy and security issues of the broker and its relationships with the clients and server while requesting services and exchanging data. In addition, more complex machine learning techniques could be advanced, such as reinforcement learning and deep learning, to be compared with the models used in our approach.

References

[1] I. Jabri *et al.*, "Vehicular Fog Gateways Selection on the Internet of Vehicles: A Fuzzy Logic with Ant Colony Optimization Based Approach," *Ad Hoc Networks*, vol. 91, 2019, p. 101879.

[2] K. Kaur *et al.*, "A Big Data-Enabled Consolidated Framework for Energy Efficient Software Defined Data Centers in IoT Setups," *IEEE Trans. Industrial Informatics*, vol. 16, no. 4, 2019, pp. 2687–97.

[3] A. Al-Fuqaha *et al.*, "Internet of Things: A Survey on Enabling Technologies, Protocols, and Applications," *IEEE Commun. Surveys & Tutorials*, vol. 17, no. 4, 2015, pp. 2347–76.

[4] B. Rochwerger *et al.*, "The Reservoir Model and Architecture for Open Federated Cloud Computing," *IBM J. Research and Development*, vol. 53, no. 4, 2009, pp. 4–1.

[5] H. Otrok *et al.*, "A Cluster-Based Model for QoS-OLSR Protocol," *Proc. 2011 7th Int'l Wireless Commun. Mobile Computing Conf.*, IEEE, 2011, pp. 1099–1104.

[6] A. Hammoud *et al.*, "On the Detection of Passive Malicious Providers in Cloud Federations," *IEEE Commun. Lett.*, vol. 23, no. 1, 2018, pp. 64–67.

[7] M. Mukherjee *et al.*, "Revenue Maximization in Delay-Aware Computation Offloading Among Service Providers with Fog Federation," *IEEE Commun. Lett.*, 2020.

[8] N. C. Luong *et al.*, "A Machine-Learning-Based Auction for Resource Trading in Fog Computing," *IEEE Commun. Mag.*, vol. 58, no. 3, 2020, pp. 82–88.

[9] C. Anglano *et al.*, "A Game-Theoretic Approach to Coalition Formation in Fog Provider Federations," *Proc. 3rd Int'l. Conf. Fog and Mobile Edge Computing (FMEC)*, IEEE, 2018, pp. 123–30.

[10] V. Veillon, C. Denninnart, and M. A. Salehi, "F-FDN: Federation of Fog Computing Systems for Low Latency Video Streaming," *Proc. 3rd Int'l. Conf. Fog and Edge Computing* (ICFEC), IEEE, 2019, pp. 1–9.

[11] A. Hammoud *et al.*, "Cloud Federation Using Genetic and Evolutionary Game Theoretical Models," *Future Generation Computer Systems*, 2019, pp. 92–104.

[12] K. Han, Z. Xie, and X. Lv, "Fog Computing Task Scheduling Strategy Based on Improved Genetic Algorithm," *Computer Science*, vol. 4, p. 22, 2018.

[13] K. H. Abdulkareem *et al.*, "A Review of Fog Computing and Machine Learning: Concepts, Applications, Challenges, and Open Issues," *IEEE Access*, vol. 7, 2019, pp. 153 123–40.

[14] X.-Q. Pham and E.-N. Huh, "Towards Task Scheduling in a Cloud-Fog Computing System," *Proc. 18th Asia-Pacific Network Operations and Management Symposium (APNOMS*, IEEE, 2016, pp. 1–4.

[15] R. F. Atallah *et al.*, "Scheduling the Operation of a Connected Vehicular Network Using Deep Reinforcement Learning," *IEEE Trans. Intelligent Transportation Systems*, vol. 20, no. 5, 2018, pp. 1669–82.

Biographies

Hala Shamseddine received her Bachelor of Science degree in computer science from the Lebanese American University (LAU) based in Beirut, Lebanon. She is currently pursuing her Master of Science degree in computer science at LAU. Her research interests include: fog and cloud federations, fog and cloud computing, machine learning, and federated learning.

Jana Nizam graduated from the Lebanese American University in 2020 with a Bachelor of Science degree in computer science. Her research interests include fog computing, fog federation, and artificial intelligence (AI). She currently works in the AI field to further expand her knowledge and skills. Her passions go further into the world of business intelligence in the tech world.

Ahmad Hammoud is currently a Ph.D. candidate in the electrical engineering Department at Ecole de Technologie Superieure (ETS). He received his B.S. in business computing and M.S. in computer science from the Lebanese University in 2016 and the Lebanese American University in 2019, respectively. His current research interests include cloud and fog computing, federated cloud and fog, game theory, blockchain, artificial intelligence, and security. He is a reviewer for several prestigious conferences and journals.

Azzam Mourad is currently an associate professor of computer science with the Lebanese American University and an affiliate associate professor with the Software Engineering and IT Department, Ecole de Technologie Superieure (ETS), Montreal, Canada. He received his M.Sc. in CS from Laval University, Canada (2003) and Ph.D. in ECE from Concordia University, Canada (2008). He has published more than 100 papers in international journals and conferences on security, network and service optimization and management targeting IoT, cloud/fog/edge computing, vehicular and mobile networks, and federated learning. He has served/serves as an associate editor for *IEEE Transaction on Network and Service Management*, *IEEE Network*, *IEEE Open Journal of the Communications Society*, *IET Quantum Communication*, and *IEEE Communications Letters*, the General Chair of IWCMC2020, the General Co-Chair of WiMob2016, and the Track Chair, a TPC member, and a reviewer for several prestigious journals and conferences. He is an IEEE Senior Member.

Hadi Otrok holds an associate professor position in the department of EECS at Khalifa University of Science and Technology, an affiliate associate professor in the Concordia Institute for Information Systems Engineering at Concordia University, Montreal, Canada, and an affiliate associate professor in the Electrical Engineering Department at Ecole de Technologie Superieure (ETS), Montreal, Canada. He received his Ph.D. in ECE from Concordia University. He is a senior member of the IEEE, and an associate editor for *Ad-Hoc Networks* (Elsevier) and *IEEE Network*. He served in the editorial board of IEEE communication letters. His research interests include the domain of computer and network security, crowd sensing and sourcing, ad hoc networks, and Blockchain.

Haidar Harmanani received the B.S., M.S. and Ph.D. degrees in computer engineering from Case Western Reserve University, Cleveland, Ohio, in 1989, 1991 and 1994, respectively. He is currently a professor of computer science at the Lebanese American University. He has served on the program committees of various IEEE conferences (ISCAS 2016, NEWCAS (2006-2016), MWSCAS 2007, ICECS (2000-2016), and DATE 98), and was the General Chair of ICECS 2011 and the Technical Program Chair of IDT 2011. He has been an ABET CAC Commissioner since 2018. His research interests include electronic design automation, high-level synthesis, and design for testability. He is a senior member of IEEE and ACM.

Zbigniew Dziong is an expert in the domain of performance, control and resource management for data, wireless, and fog-cloud networks. He participated in research projects for many leading companies such as Bell Labs, Nortel, Ericsson, and France Telecom. He received his Ph.D. degree from Warsaw University of Technology, Poland. From 1987 to 1997, he was a professor at INRS-Telecommunications, Montreal, Canada. From 1997 to 2003, he worked for the Performance Analysis Department at Bell Labs, Lucent Technologies, Holmdel, New Jersey, USA. Since 2003 he has been with École de technologie supérieure (University of Quebec), Montreal, Canada, as a full professor.