Computing and Informatics, Vol. 44, 2025, 612–634, doi: 10.31577/cai_2025_3_612

SDN-BASED MULTI-OBJECTIVE OPTIMIZATION FOR TASK OFFLOADING WITH ALGORITHM FEDERATED LEARNING IN FOG COMPUTING ENVIRONMENT

Mohammadreza Sharafi Hoveyda,
Mohammadreza Mollahoseini Ardakani*

Department of Computer
Maybod Branch, Islamic Azad University
Maybod, Iran
e-mail: m.sharafihoveyda@iau.ir, mr.mollahoseini@iau.ac.ir

Vahid Ayatollahitafti

Department of Computer
Taft Branch, Islamic Azad University
Taft, Iran
e-mail: vahid.ayat@gmail.com

Abstract. Due to the substantial volume of data associated with the IoT, processing and storing such large amounts of data is not easily feasible. Nevertheless, many of its applications face challenges in cloud computing, such as latency, location awareness, and real-time mobility support. Edge computing helps provide solutions to these challenges. In this article, the MINLP path optimization problem is initially addressed using SDN, SA+GA, OLB-LBMM, and Round-Robin methods. Subsequently, based on the obtained results, the SDN method, which has achieved the best outcomes among the approaches, is selected. This article involves a simulation of IoT for optimal allocation of shared resources in edge computing. The network architecture comprises five distinct layers, including cloud services, the SDN controller, edge computing nodes, edge computation and users. The algorithm employed in this problem is the federated learning and stochastic gradient descent

* Corresponding author

algorithm. It selects the optimal edge node for user service provision through two learning and training phases, aiming to allocate shared resources to three parameters: cloud service providers' revenue, average latency, and user satisfaction. This algorithm is compared with several other methods. The selected model and algorithm, in comparison with other algorithms used in solving similar models, lead to a centralized management system, the implementation of effective network management, and the utilization of various communication media. This approach ensures timely access to services, contributing to increased profits for providers and user satisfaction.

Keywords: Software-defined network, federated learning, edge computing, Internet of Things

1 INTRODUCTION

The rapid advancement of the IoT[1] and social network applications has led to an exponential growth in data generated at network edge. It is anticipated that data generation rate will surpass the current capacity of the Internet in the near future [1]. Given the network bandwidth constraints and privacy concerns, sending all data to a remote cloud is impractical and often unnecessary. As a result, research organizations estimate that over 90 % of data will be processed locally [2]. This data requires fundamental computational resources for real-time processing, leading to high energy consumption on IoT devices. To address this issue, offloading tasks using edge computing has emerged as a promising solution [3, 4]. Edge computing is a common term that aims to meet the application needs by being present everywhere and at all times. Edge computing can be defined as a programming and communication paradigm that brings cloud resources physically or computationally closer to IoT devices. In other words, edge computing acts as an interface between the cloud and the IoT, assisting them in establishing communication. Therefore, with the expansion of the cloud computing application domain and increased availability of resources in the IoT, it leverages the strengths of both technologies [5]. The use of collaborative learning in conjunction with software-defined networking can serve as an effective solution for enhancing secure data offloading in cloud computing environments [6]. Offloading schemes in existing articles have primarily focused on computational power and energy consumption without considering the network load on the path from the device to the edge server. Therefore, dynamic network load should be taken into account during offloading decision-making. This can be achieved through the use of a SDN[2] architecture, which provides centralized logical control, an overview of network conditions, and the enforcement of rules by the controller [7]. Furthermore, network management by SDN enables

[1] Internet of Things

[2] Software-Defined Network

the gathering of network information from heterogeneous wireless devices across various wireless technologies. Therefore, with the network overview provided by SDN, the SDN controller is capable of making optimal decisions in task offloading.

Many optimization problems arising in engineering and sciences involve combinatorial and nonlinear relationships. Such optimization problems are modeled using MINLP[3], which combines the capabilities of MILP[4] and NLP[5]. The precise modeling capability for real-world problems has turned MINLP into an active research area with numerous industrial applications [8]. Collaborative learning is a technology that has attracted researchers' attention for exploring its potential and applications [9, 10] and [11]. FL[6] seeks to address the fundamental question [12] of whether we can train models without the need to transfer data to a central location. In the FL framework, the focus is on collaboration, which is not achieved through standard machine learning algorithms [13]. Additionally, FL allows the algorithm to gain experience, which is not guaranteed through traditional machine learning methods [14, 15]. Given these challenges, we propose a distributed training scheme based on collaborative learning to reduce the training load on each device. In the FL framework, the focus is on collaboration, which is not achieved through standard machine learning algorithms [13]. It allows the algorithm to gain experience. Considering these issues, we propose a collaborative learning-based distributed training scheme to reduce the training load on each device. This paper discusses how to effectively use limited computational and communication resources at the edge to achieve optimal learning performance. We consider a typical edge computing architecture in which edge nodes communicate with the remote cloud through network elements, such as gateways and routers, which, through SDN network management, provide the capability to access network information such as topology, network automation, and infrastructure abstraction, operational cost reduction, etc., from heterogeneous devices. The use of an SDN controller leads to optimal discharge decisions, as well as network flexibility and the implementation of network management [11].

In fact, the main sections of our paper are as follows: In Section 2, a review of the work carried out in this field is presented. In Section 3, the proposed system model is introduced and examined. In Section 4, the problem-solving process with the proposed algorithm is discussed. In Section 5, the approach and evaluation results of the proposed method are presented through simulation. The obtained results are analyzed and compared with previous methods. Finally, the topics discussed in the paper are summarized, and conclusions are presented in Setion 6.

[3] Mixed-Integers Nonlinear Programming
[4] Mixed-Integers Linear Programming
[5] Nonlinear Programming
[6] Federated Learning

2 RELATED WORKS

Computation in fog computing, conducted at network edges, can be viewed as a type of decentralized distributed front-end computation, in contrast to centralized cloud computations. Fog Mobile exemplifies this type of computation, providing latency-aware services for mobile devices. Offloading computations to mobile phones poses a significant challenge due to the need for temporary resources and heterogeneous devices. Based on this, a reinforcement learning-based computation offloading mechanism has been proposed to ensure the provision of services to mobile service customers [16]. In this approach, a distributed reinforcement learning algorithm is used for offloading basic blocks in a decentralized manner, distributing mobile phone codes across geographically distributed spaces. The proposed method significantly leads to a reduction in execution time and access delay for mobile services, along with lower energy consumption for mobile devices. This scheme employs a distributed multi-agent approach for computation offloading in diverse fog computing environments tailored for mobile devices.

One of the practical approaches presented in 2016 [17] introduces a distributed ADMM model for data offloading in fog computing. The model optimizes the performance of the offloading algorithm and guarantees QoS[7] for the end-user. It also provides cloud computations, storage, and computational memory for the end-user (CUE[8]). As data traffic in the network significantly increases, leading to elevated computational density, it results in increased delays. This approach, known as Fog Computing, utilizes distributed servers to tackle these challenges.

Fog computing can provide computational storage, memory, and networking capabilities. However, it cannot replace CS[9]. Fog, with the extension of the cloud, aims to get closer to the CUE to improve efficiency and reduce the amount of data that needs to be transferred to the cloud for analysis, processing, and storage. In fact, several fog nodes exist, each with limited storage capacity, and some users wish to transfer data to the CS. On the other hand, the CS is distant from the CUE, and data needs to be transferred through the network core. And this means that data traffic will increase. Our assumption is that the CUE uploads only some files for storage in CS, and then data can be temporarily stored in some fog nodes during the day and transferred to CS at night. This can help reduce traffic and computational density in the core.

In another framework proposed for offloading load in cloud computing data centers [18], the assumption is that data centers are deployed at the network edge. If a request is transferred to a data center, it directs it to a neighboring data center by considering a probability. These data centers have a large number of servers, and traffic in some of them becomes saturated. In this case, data centers may help alleviate the issue by accepting some of the rejected requests. The goal is to achieve

[7] Quality of Service
[8] Computing for End-User
[9] Cloud Server

quality through collaboration among neighboring data centers. In fog computing, data centers require less capacity compared to cloud computing, hence, there is a higher likelihood of congestion. Therefore, to reduce the likelihood of blocking requests, data centers in fog computing need to collaborate [19].

A key challenge in mobile services is minimizing data transfer time and efficiently executing end-user tasks, considering a policy that can make decisions for computation offloading [20]. This model consists of four sections: User, agent, cloud. Mobile users interact with the cloud through mobile devices that have access to the cloud via WiFi or the internet. When a highly complex program is running, mobile devices face limitations, so they need to rely on cloud systems that are directly accessible to users. This can help prevent additional communication delays. The cloud can provide abundant computational resources, but in a physical sense, it is distant from users, leading to significant transfer delays. An agent is a control center that can gather information about resources. Based on this information, the agent decides whether tasks should be executed on the mobile device or offloaded to the cloud. Based on this, when a new task is created by a user, it can be executed on the mobile device. If the mobile device is incapable of performing this task, the agent decides to offload it to the cloud via a wireless link. If the cloud cannot complete the task or if it takes too long, the task is then performed through the internet and the cloud.

Another study in the research [21] indicates that with the increase in social relationships, the energy consumption of mobile devices rises. This can be considered in the design of computation offloading in mobile cloud computing with the aim of minimizing execution costs in social groups [22]. The model supports the use of game theory in this context and proposes a dynamic computation offloading scheme for the offloading process in mobile cloud computing systems.

In research [23], a task offloading scheme is proposed for SDN in which IoT devices connect to computational nodes in the fog via multiple IoT access points (considering fixed access points and fog nodes). The study also addresses the task offloading problem in an SDN network where VoIP devices connect to fog computing nodes through (APs[10]). Since the non-linear task offloading problem faces challenges, a linear technique is employed to provide an ILP[11] formulation for problem-solving. The greedy solution considers delay, energy consumption, multi-path routes, and dynamic network conditions such as link utilization and SDN programmability. In this research, offloading is performed based on the shortest path to the fog node, considering hop count, with the aim of minimizing average delay while adhering to energy constraints [24].

To facilitate distributed management and scalability, fog nodes utilize SDN techniques. In this approach, the control plane is capable of decision-making, while the data plane straightforwardly handles tasks of transmission and processing [25]. In this study, reinforcement learning is employed to address scalability issues, using

[10] Access Points

[11] Integer Linear Programming

a neural network called DRQN[12], which enhances learning speed and performance. The research designs a homogeneous cooperative task offloading and resource allocation algorithm, aiming to maximize the completion of processing tasks at a minimal overflow rate from the proposed algorithm. The integration of temporal observations from a recurrent network to nodes allows them to synchronize their decisions without explicit knowledge of each other's states and operational sets, making the proposed DRQN-based algorithm resilient against a relatively observable dynamic environment. The results also indicate that intelligently distributed resources, tailored to various latency constraints, significantly impact the overall system performance [26].

Kim et al. [27] proposed a FL approach based on blockchain, where local updates are aggregated in the blockchain. On the other hand, [28, 29] suggested a hierarchical client-edge FL system to reduce communication costs compared to traditional FL.

3 THE PROBLEM MODEL

This study proposes a dynamic computation offloading scheme in SDN networks using the FL algorithm, where IoT devices connect to fog computing nodes through various APs. In this architecture, we specifically consider the following aspects:

1. Local computation decision: Determining whether computational decisions should be made locally on IoT devices or remotely.
2. Optimal node selection for offloading: Choosing the optimal node for computation offloading.
3. Execution of FL Algorithm.

In summary, the architecture is described as follows: We employ a five-layer network architecture, as illustrated in Figure 1, which is briefly explained below:

IoT Layer: In this layer, diverse APs and devices such as sensors, smartphones, tablets, and similar items are placed, which are heterogeneous in terms of storage space, processing capabilities, and communication relationships. Access from this layer to fog servers is usually possible through APs and wireless communication lines. This layer is the first decision point for task offloading to nearby or distant servers.

Edge Layer: In this layer, devices utilize radio access technologies such as WiFi to communicate with each other. Edge nodes in a specific area establish communication with a fog node in the same regional range using a wireless channel interface.

Fog Layer: In this layer, fog nodes are positioned to interact directly with end devices such as smartphones, wireless cameras, and sensor devices through a wireless interface. Each fog node executes one or more computational services and operates a fog agent, striving to fulfill computational and storage requests from

[12] Deep Recurrent Neural Network

IoT nodes to the greatest extent possible. To manage distributed operations on a large geographical scale, fog nodes are divided into colonies. Each colony has a more powerful node called the Fog Orchestration Control Node, which maintains the overall status of the colony, such as the available processing capacity of the current colony nodes. If a request received by one of the leading nodes in the IoT layer cannot be executed at that layer,it is transferred to the Fog Orchestration Control Node and placed in an M/M/1/K queue. Through the Fog Orchestration Control Node, we can collect information about each fog node, specifically including installed services, hardware specifications, and current computational resources (such as CPU, RAM, and storage space). This information is used by the offloading service to make an optimal offloading decision.

SDN Control Layer: The SDN controller is placed in this layer, providing a programming interface for network management. The decision-making regarding the offloading method is taken by this controller.

Cloud Layer: In this layer, powerful data centers with unlimited computing resources are placed to provide appropriate services. Since this research aims to focus on the characteristics of the SDN and fog layers, details of the cloud layer are not elaborated upon.

Definition of variables is provided in Table 1. Users consist of a set of devices such as smartphones, surveillance cameras, electronic devices, etc. denoted by $U = \{u_1, u_2, \ldots, u_m\}$. These users may offload a specific computational or storage workload to cloud service providers, represented by $CS = \{cs_1, cs_2, \ldots, cs_m\}$. Service providers may cater to different users with varied computational requirements in terms of data size and service delay. For users insensitive to delay, computations are sent to the cloud, while for delay-sensitive users, service providers allocate one of the nearby fog nodes for computational tasks. Fog nodes closer to users will encounter lower delays, but geographical location is not the sole factor affecting service delay. Initially, the transmission/reception delay and processing delay are calculated. Each user carries information with them, allowing users to express their needs such as delay requirements, data size, and processing time (processing time and data size have a linear relationship). This information is sent to CS. In the next stage, the cloud, utilizing SDN), identifies a suitable fog node based on the user's requirements to allocate computational radio resources appropriately.

In fact, service delay consists of three time periods: transmission time, CPU processing time, and reception time. The transmission and reception time periods are defined as the time required to send data to the fog node for processing and to receive the processing results, respectively. On the other hand, CPU processing time will be determined by the CPU rate of each fog node; therefore, for each CS_j, when an appropriate fog node is selected from the set $FN^j = \left\{fn_1^j, fn_2^j, \ldots, fn_l^j\right\}$ for each user, it will allocate its shared resources $W^j = \left\{w_1^j, w_2^j, \ldots, w_m^j\right\}$ and computational resources $C^j = \left\{c_1^j, c_2^j, \ldots, c_m^j\right\}$.

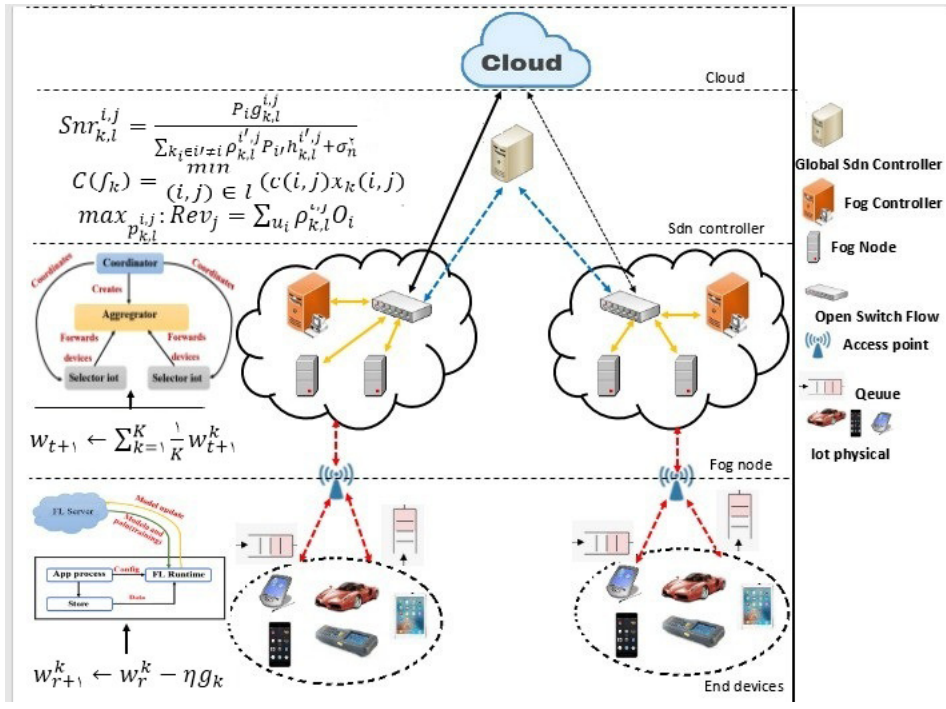


Figure 1. Proposed framework for collaborative learning-based task offloading in fog computing environment

The SDN controller communicates with switches using the OpenFlow protocol, and the communication between the controller and the application layer is achieved through the SDN Northbound API[13].

For the given data network, QoS criteria are associated with each link $(i, j) \in l$, including delay $d(i, j)$, packet loss probability $l(i, j)$, and available bandwidth $c(i, j)$. The combination of rules for packet loss probability is more complex, and therefore, the logarithm of the success probability (which is itself multiplied several times) is considered.

Consequently, the value of $l'(i, j)$ is calculated according to relation (1).

$$l'(i, j) = \log(1 - l(i, j)). \tag{1}$$

As a result, for each path $p = \{i, j, k, \ldots, s, t\}$, the relations (2), (3), (4), (5) calculate

[13] Application Programming Interface

the values of delay, bandwidth, and packet loss probability [5].

$$D_p = d(i,j) + d(j,k) + \cdots + d(s,t), \tag{2}$$

$$C_p = \min\{c(i,j), d(j,k), \ldots, d(s,t)\}, \tag{3}$$

$$L_p = l'(i,j) + l'(j,k) + \cdots + l'(s,t), \tag{4}$$

$$L_p = 1 - \exp(L'_p). \tag{5}$$

Additionally, the variable $x_k(i,j)$ is a binary variable, and it is a bistate function as described in relation (6).

$$x_k(i,j) = \begin{cases} 1, \\ 0. \end{cases} \tag{6}$$

The value of relation (6) becomes equal to one when the routing process between nodes i and j has been performed; otherwise, its value will be zero. Furthermore, to calculate the delay and lost packets, relations (7) and (8) are utilized.

$$D(f_k) = \sum_{(i,j)\in l} l'(i,j) x_k(i,j), \tag{7}$$

$$L'(f_k) = \sum_{(i,j)\in l} l'(i,j) x_k(i,j). \tag{8}$$

According to the law of bandwidth combination, the capacity of a path through which a stream passes is mathematically defined by relation (9).

$$C(f_k) = \min_{(i,j)\in l} (c(i,j) x_k(i,j)). \tag{9}$$

Ultimately, relation (10) is used to calculate the bandwidth.

$$C_{res}(i,j) = c(i,j) - \sum_{f_k \in f} q_k^w x_k(i,j). \tag{10}$$

In order to optimize the allocation of shared resources, the satisfaction of both users and service providers is taken into consideration. Initially, an assessment of user satisfaction requires the calculation of the Signal-to-Noise Interference Ratio. Additionally, service delay plays a crucial role in evaluating user satisfaction. These two factors (Signal-to-Noise Interference Ratio[14] and service delay) together define user satisfaction.

In order to ensure accurate and complete data delivery, the SINR must exceed a minimum threshold value, denoted as Snr_{min}. The SINR is defined as follows (11):

$$Snr_{k,l}^{i,j} = \frac{P_i g_{k,l}^{i,j}}{\sum_{u_i \in u, i' \neq i} \rho_{k,l}^{i',j} \rho'_i h_{k,l}^{i',j} + \sigma_n^2}. \tag{11}$$

[14] SINR

Definition Variable	Variable
Cloud Service Provider	CS_j
Edge Nodes	FN^j
Users	u
Bandwidth	W^j
Computational Resources	C^j
Network Latency	D_p
Probability Packet Loss	L_p
Signal-to-Noise Interference Ratio	Snr_{min}
Received Snr Ratio for user in $u_i$channel fn_j^i	$Snr_{k,l}^{i,j}$
Transmission Power	P_i
User $u_i$and Edge Nodefn_i^j Channel Reward using Channel BW_k^j	$g_{k,l}^{i,j}$
Binary Resource Allocation Variable	$\rho_{k,l}^{i',j}$
Set of Bandwidth-CPU Cycle Pairs	RP^j
Interference Reward from Other Mobile Users' Channels	$h_{k.l}^{i'.j}$
Channel Noise	σ_n^2
Transmission Rate	$r_{k,l}^{i,j}$
Service Delay	$t_{k,l}^{i,j}$
Transmission Time Value	t_t
Processing Time Value	t_p
Reception Time Value	t_r
Data Size	D_i
Number of CPU Cycles	DC_i
Processing Rate	$c_{k,l}^{i,j}$
Random Variable Between 0 and 1	δ_t
Proposed Price	O_i
A parameter in megabits per second	a
Delay Requirement	T_i
Total Revenue for each service provision	Rev_j
Channel Capacity	q_R
CPU Capacity	q_C
Maximum number of users that can be serviced	q_{CP}

Table 1. Variables

The average reward received for confirming each transaction($g_{k,l}^{i,j}$) is expressed as the channel profit and the transmission rate from u_i to fn_l^j using channel B_k^j, subject to the fulfillment of SINR requirements, as follows (12):

$$r_{k,l}^{i,j} = w_k^j \log(1 + Snr_{k,l}^{i,j}). \tag{12}$$

Furthermore, the service delay u_i when the resource pair (w_k^j, c_l^j) is utilized will be defined as follows, where a lower service delay corresponds to greater user satisfaction (13):

$$t_{k,l}^{i,j} = t_t + t_p + t_r = \frac{D_i}{r_{k,l}^{i,j}} + \frac{DC_i}{c_{k,l}^{i,j}} + \delta_t. \tag{13}$$

In the context of service providers' profits, one can consider a linear relationship between profit and data size without introducing a fundamental flaw into the overall framework. Therefore, the proposed profit for each user can be defined as follows (14):

$$O_i = f(D_i, T_i), \tag{14}$$

where the function is chosen as (15):

$$O_i = a\frac{D_i}{T_i}. \tag{15}$$

The total revenue for each service provider is then obtained as (16):

$$Rev_j = \sum_{u_i \in u} \rho_{k,l}^{i,j} O_i. \tag{16}$$

Profits essentially come from the revenue generated by CS offering services to users through edge nodes.

The objective of this article is to increase the revenue of service providers, leading to enhanced user satisfaction and reduced average delay. Therefore, the optimization

problem and its constraints can be formulated as follows:

$$\max(\rho_{k,l}^{i,j}) : Rev_j = \sum_{u_i \in u} \rho_{k,l}^{i,j} O_i, \tag{17}$$

$$s.t \therefore \rho_{k,l}^{i,j} t_{k,l}^{i,j} \leq T_i \forall u_i \in u, rp_{k,l}^{j} \in RP^j, cs_j \in CS, \tag{18}$$

$$\rho_{k,l}^{i,j} Snr_{k,l}^{i,j} \geq Snr_{min}, \forall u_i \in u, rp_{k,l}^{j} \in RP^j, cs_j \in CS, \tag{19}$$

$$\sum_{u_i \in u, fn_l^j \in FN^j} \rho_{k,l}^{i,j} \leq q_r, \forall w_j^i \in W^j, cs_j \in CS, \tag{20}$$

$$\sum_{u_i \in u, w_l^j \in w^j} \rho_{k,l}^{i,j} \leq q_c, \forall fn_j^i \in FN^j, cs_j \in CS, \tag{21}$$

$$\sum_{u_i \in u, rp_{k,l}^j \in RP^j} \rho_{k,l}^{i,j} \leq q_{cp}, \forall cs_j \in CS, \tag{22}$$

$$\rho_{k,l}^{i,j} \in \{1, 0\}. \tag{23}$$

Formula (17) represents the maximization of total revenue for each service provider. (18) indicates the delay requirement for each user, and (19) defines the minimum SINR requirement for each user. (20), (21) and (22) respectively satisfy the constraints of channel capacity, edge node, and service provider capacity.

4 PROBLEM SOLVING

Since each user carries information, we enable them to express their needs, such as latency, data size, and processing time (noting that data size and processing time are linearly related). This information is then sent to CS. In the next stage, a suitable edge node is selected based on the user's needs to appropriately allocate processing and computational resources. The best edge node is selected through the SDN network. Each round of FL consists of six stages:

1. Selection,
2. Configuration,
3. Local Update,
4. Local Aggregation,
5. Global Aggregation,
6. Reporting.

After selecting the optimal edge node, a general base model is stored in the central edge node, and copies of this model are distributed to users' devices. Then, the models are trained based on locally generated data. In the next stage, the updated

parameters from locally trained models are shared with the base general model on the central server. When the central model is re-trained with new parameters, it is shared again with users' devices for the next iteration. With each cycle, the models gather diverse information and improve while maintaining user privacy. Edge nodes communicate with clients using a wireless channel interface, where the number of clients is exceeds than the number of edge nodes ($f < u$).

Problem-solving involves the execution of two algorithms implemented in the Python language.

In Algorithm 1: after obtaining the parameters of the initial model, clients begin by dividing their local data into several batches of constant size. They then perform SGD[15] according to Equation (24) and calculate the average gradient (g_k).

$$w_{t+1} = w_t + \alpha \frac{\partial L}{\partial w_t}. \tag{24}$$

At time step t, we want to update the weight/parameter w. Here, α represents the learning rate (which is not constant and adjusts in accordance with the gradient magnitude). $\frac{\partial L}{\partial w}$ is the gradient of the loss function L (that needs to be minimized with respect to w).

In each batch, the output from the lower layers (clients) and the corresponding labels of the data will be sent to the edge server for data aggregation. To facilitate local updates, mobile devices need to communicate with the edge server. Therefore, the value and size of the batch during time steps (t_{batch}) will affect the communication between clients and the edge server in the learning process on local data, as per Equation (25).

$$t_{batch} = \frac{D_k}{V_{batch}}. \tag{25}$$

Here, D_k represents the amount of local data on clients, and V_{batch} indicates the batch size. When t becomes excessively large, it implies significant communication between clients and the edge server, and the process of local updates is influenced due to network instability, even though there is high bandwidth and low latency. A small V_{batch} increases the number of required local communications, while a large V_{batch} may affect the convergence of the model training.

After calculating the stochastic gradient (w_t), the next step is to update the model and send it towards the server, as shown in Equation (26).

$$w_{r+1}^k \leftarrow w_r^k - \eta g_k. \tag{26}$$

Algorithm 1: Federated Learning (SGD)
Input: Initial model (w)
Output: Updated initial model (w_{k+1})

1. Loop until clients exist

[15] Stochastic Gradient Descent

2. Select a set of clients (u) in each round for collaborative learning.
3. Clients receive model w from the server.
4. Calculate and average the stochastic gradient (using (24)).
5. Update the local model (using (26)).
6. Clients send the updated model to the server.
7. End of loop.

In Algorithm 2: After receiving the trained model from clients, the coordinator server must determine whether FL should continue. This decision is made by testing the trained model on the central server's validation dataset. If the result is satisfactory, there is no need for further iterations that involve additional computations. Otherwise, the global coordinator server can repeat the entire process and distribute the model parameters for the next rounds.

Algorithm 2: Global Aggregator
Input: Trained parameters from clients
Output: Final trained model

1. Loop until fog exists
2. Update client models: $w_{t+1}^k \leftarrow w_t$
3. End of loop
4. Calculate the global model after training

$$w_{t+1} \leftarrow \sum_{k=1}^{K} \frac{1}{k} w_{t+1}^k$$

5. Send the final trained model to clients: w_{t+1}
6. End.

5 EVALUATION

To establish a consistent experimental environment, we first introduce the simulation environment and hardware used in the experiment. Python and MATLAB programming languages are utilized for simulating and implementing the proposed algorithm. The main sections of the simulation are as follows:

- Network architecture design,
- Definition of research variables,
- Performing the process of connecting research variables to the network architecture.

The hardware and defined conditions for simulating the experimental environment have the following specifications:

- The simulation's executing system CPU is a Core i7 processor with a frequency up to 2.50 gigahertz,
- 8 gigabytes of RAM,
- Linux operating system.

We consider a network with CS = 1, serving as service providers, and FN = 5 edge nodes randomly distributed throughout the network. Assume there are U = 100 users randomly distributed across the network.

In Figure 2, the distribution of the 4-layer network is illustrated; the figure is divided into four equal parts (CS at the top, then the SDN network, followed by edge nodes and users in the bottom layer).

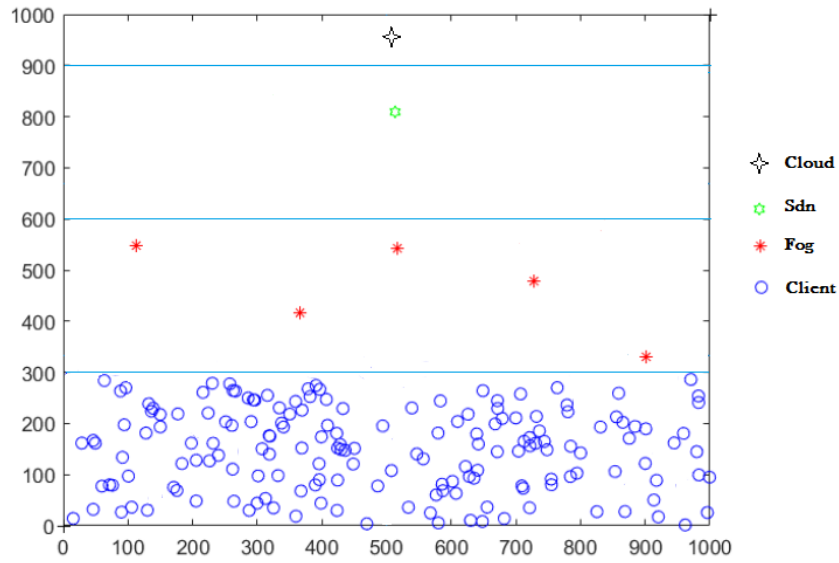


Figure 2. Four-layer network

To compare the objective function, SDN is evaluated against three algorithms: SA + GA, OLB-LBMM, and Round-Robin. In Figure 3, the profit of CS increases with the increase in the number of IoT devices.

To evaluate the performance of the proposed method presented in this article, we compare it with recent relevant works, specifically articles [29, 30] and [31]. In the following, we introduce the evaluation parameters.

1. Impact of Client Mobility on Training Time: When a client is moving in the edge server, factors such as the training phase and dataset can influence the

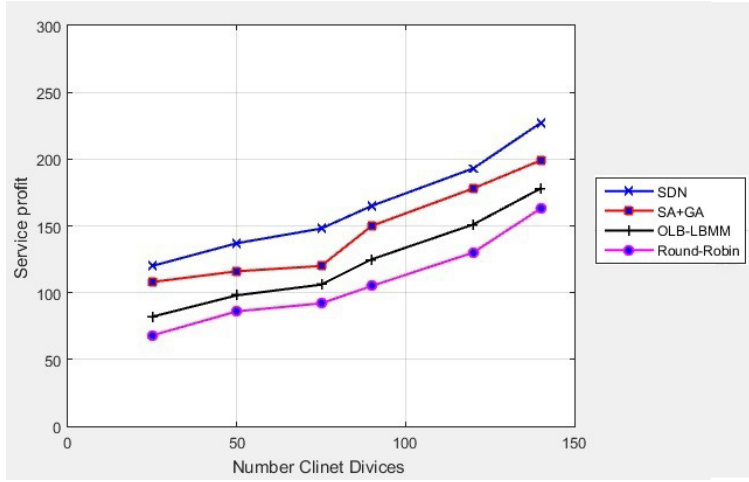
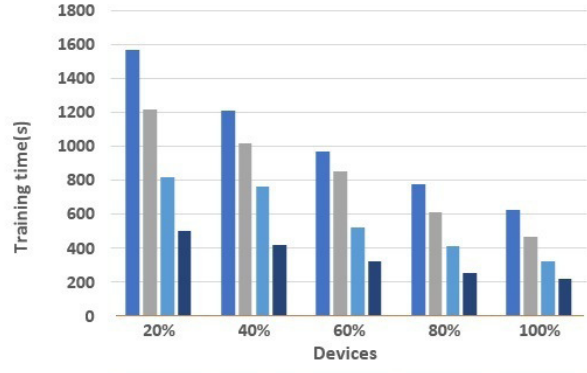


Figure 3. Service profit

training time. In this experiment, the impact of generating 25 % and 50 % of the required data (dataset) for training on a single device is illustrated in Figure 4 and Figure 5, depicting the training phases.

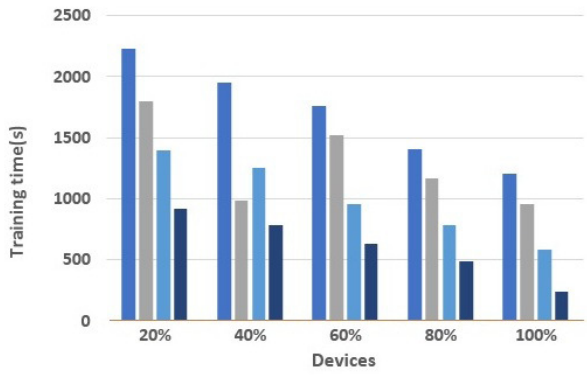


	20%	40%	60%	80%	100%
DRL	1571	1211	971	780	625
EdgeFed	1221	1020	851	610	470
FogFL	821	765	520	416	320
SDN-FL	501	418	320	253	221

Figure 4. Training with 20 % of the dataset

Figure 4 illustrates the effects of device mobility on the training time when 20 % of the dataset is used for training on a single device. The device movement is

completed at 40 %, 60 %, 80 %, and 100 % of these training phases. Figure 4 clearly shows that SDN outperforms the other algorithms. As we move towards the 50 % training time interval, the training time reduces by up to 37 % per round. However, when moving with the datasets, at the 80 % training time where the training is approaching completion, the training time decreases by up to 46 % per round.



	20%	40%	60%	80%	100%
DRL	2230	1955	1756	1402	1201
EdgeFed	1800	982	1520	1170	958
FogFL	1400	1258	956	780	580
SDN-FL	921	782	635	488	239

Figure 5. Training with 50 % of the dataset

Figure 5 also illustrates the effects of device mobility on the training time when 50 % of the dataset is used for training on a single device. The device movement is completed at 40 %, 60 %, 80 %, and 100 % of these training phases. Figure 5 indicates that the training time for devices is longer than that shown in Figure 4. It is evident that using 50 % of the dataset for training mobile user devices, which is more stable and comparable, leads to more efficient results.

2. Impact of mobility on global accuracy: in this experiment, we examine the accuracy of the global model when a system runs periodically between servers: Figure 6 shows the benefit of service providers and the proposed algorithm.

Figure 7 depicts the training accuracy using entropy for a total of 100 different rounds with the availability of 20 % and 50 % of the dataset for mobile users.

In Figure 7, it is evident that all these algorithms maintain accuracy, and the reason for the high accuracy is obtaining the modeled parameters from the central server and training in the edge agent. SDN-FL transfers data to the edge agent,

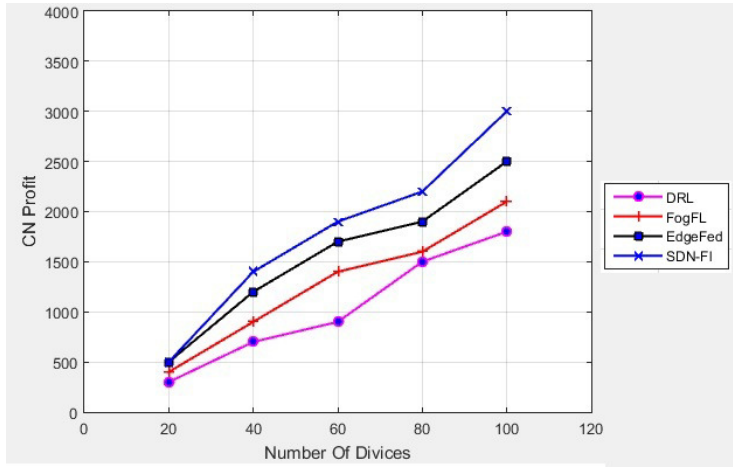


Figure 6. Service providers' profit

where the training takes place, and it maintains the same level of accuracy compared to other methods. However, the training is not repeated at the edge agent. Therefore, the training time is reduced for SDN-FL.

The performance of SDN-FL is influenced by the following factors:

1. Balanced and unbalanced data volumes,
2. Movement frequency of devices,

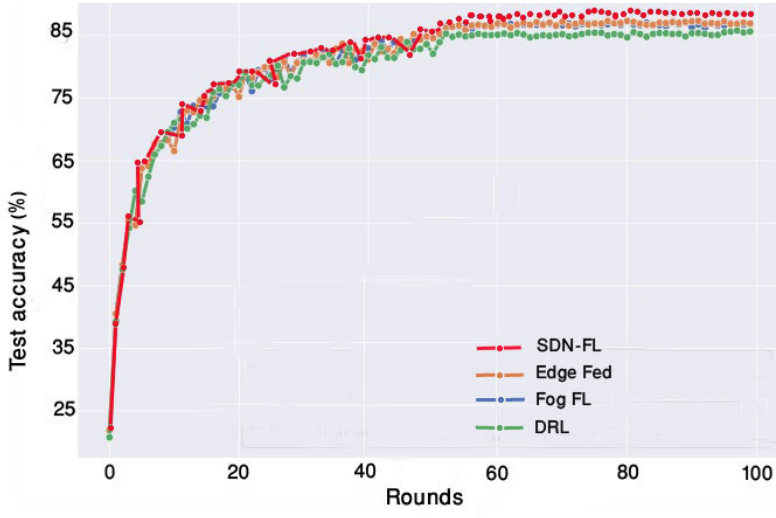


Figure 7. Entropy of training accuracy

3. SDN selection of the best edge node,
4. Model training stages by FL.

The summarized results are as follows:

- Compared to the mentioned methods, the proposed SDN-FL method performs better by reducing the training time for both balanced and unbalanced datasets. In SDN-FL, when moving towards the 50 % training interval, the training time is reduced by up to 37 % per round. However, when the datasets are nearing completion within the 80 % training interval, the training time decreases by 46 % per round.
- According to the entropy chart of SDN-FL, there are no losses in accuracy function.
- In SDN-FL, data transfer between edge servers results in a 3-second overhead, which is negligible compared to the device training time during restart in the destination broker.

6 CONCLUSION

In light of the diverse and complex nature of IoT environments, traditional networks often face significant challenges due to data overload, which can severely impact network performance and service quality. The Software-Defined Networking (SDN) approach, which enables the independent analysis of each flow and the formulation of tailored proportional rules, has garnered considerable attention from researchers. This capability allows for the effective utilization of SDN in heterogeneous IoT settings, facilitating improved management and optimization of network resources.

From a holistic perspective, SDN provides a robust framework for load balancing across the network. In conventional computer networks, the presence of uneven, non-uniform, and randomly distributed input flows can result in load imbalance, leading to various negative outcomes such as diminished Quality of Service (QoS), loss of critical input flows, network overload, inefficient capacity utilization, instability, and ultimately, a decline in overall network efficiency. Therefore, this article emphasizes the importance of fair traffic distribution across different segments of the network and the effective allocation of available resources, leveraging the strengths of SDN architectures.

The integration of Federated Learning (FL) into this context introduces additional complexities, primarily characterized by two significant challenges: training time and accuracy. These issues become increasingly critical when devices exhibit mobility during the FL training process. The proposed SDN-FL framework specifically addresses these challenges, particularly in edge-based scenarios, marking a pioneering effort to mitigate the impact of device mobility on FL training.

Evaluation of Benefits, Training Time, and Accuracy

Benefits: The SDN-FL framework offers several advantages, including enhanced resource allocation, improved load balancing, and more efficient traffic management. By allowing for the dynamic adjustment of network flows based on real-time conditions, the framework ensures that resources are utilized optimally, which is particularly crucial in heterogeneous IoT environments. The ability to tailor network responses to specific conditions enhances the overall user experience and maintains service quality.

Training Time: The results obtained from the implementation of the SDN-FL algorithm indicate a notable improvement in training time compared to existing literature. This enhancement is largely due to the algorithm's capability to navigate a vast solution space more effectively, leveraging the principles of training and learning without the need for extensive parameter tuning. The reduction in training time not only accelerates the deployment of FL models but also makes the system more responsive to real-time changes in the network environment.

Accuracy: In addition to improvements in training time, the SDN-FL framework has demonstrated competitive accuracy levels in model performance. By effectively managing the distribution of training data and adapting to the mobility of devices, the framework ensures that the learning process remains robust and reliable. This is crucial for maintaining high accuracy in predictions and classifications, which are essential for the successful application of FL in IoT scenarios.

In conclusion, the SDN-FL framework not only enhances training efficiency and accuracy but also represents a significant step forward in addressing the unique challenges posed by mobility in FL scenarios within IoT environments. Future research should continue to explore the synergies between SDN and FL, aiming to further optimize network performance and service delivery in increasingly dynamic and heterogeneous settings.

REFERENCES

[1] Chiang, M.—Zhang, T.: Fog and IoT: An Overview of Research Opportunities. IEEE Internet of Things Journal, Vol. 3, 2016, No. 6, pp. 854–864, doi: 10.1109/JIOT.2016.2584538.

[2] Kelly, R.: Internet of Things Data to Top 1.6 Zettabytes by 2020. 2015, https://campustechnology.com/articles/2015/04/15/internet-of-things-data-to-top-1-6-zettabytes-by-2020.aspx.

[3] Mao, Y.—You, C.—Zhang, J.—Huang, K.—Letaief, K. B.: A Survey on Mobile Edge Computing: The Communication Perspective. IEEE Communications Surveys & Tutorials, Vol. 19, 2017, No. 4, pp. 2322–2358, doi: 10.1109/COMST.2017.2745201.

[4] Mach, P.—Becvar, Z.: Mobile Edge Computing: A Survey on Architecture and Computation Offloading. IEEE Communications Surveys & Tutorials, Vol. 19, 2017, No. 3, pp. 1628–1656, doi: 10.1109/COMST.2017.2682318.

[5] Bellavista, P.—Berrocal, J.—Corradi, A.—Das, S. K.—Foschini, L.—Zanni, A.: A Survey on Fog Computing for the Internet of Things. Pervasive and Mobile Computing, Vol. 52, 2019, pp. 71–79, doi: 10.1016/j.pmcj.2018.12.007.

[6] Singh, J.—Singh, P.—Hedabou, M.—Kumar, N.: An Efficient Machine Learning-Based Resource Allocation Scheme for SDN-Enabled Fog Computing Environment. IEEE Transactions on Vehicular Technology, Vol. 72, 2023, No. 6, pp. 8004–8017, doi: 10.1109/TVT.2023.3242585.

[7] Sood, K.—Yu, S.—Xiang, Y.: Software-Defined Wireless Networking Opportunities and Challenges for Internet-of-Things: A Review. IEEE Internet of Things Journal, Vol. 3, 2016, No. 4, pp. 453–463, doi: 10.1109/JIOT.2015.2480421.

[8] Muts, P.—Nowak, I.—Hendrix, E. M. T.: The Decomposition-Based Outer Approximation Algorithm for Convex Mixed-Integer Nonlinear Programming. Journal of Global Optimization, Vol. 77, 2020, No. 1, pp. 75–96, doi: 10.1007/s10898-020-00888-x.

[9] Liang, P. P.—Liu, T.—Ziyin, L.—Allen, N. B.—Auerbach, R. P.—Brent, D.—Salakhutdinov, R.—Morency, L. P.: Think Locally, Act Globally: Federated Learning with Local and Global Representations. CoRR, 2020, doi: 10.48550/arXiv.2001.01523.

[10] Zhuo, H. H.—Feng, W.—Lin, Y.—Xu, Q.—Yang, Q.: Federated Deep Reinforcement Learning. CoRR, 2019, doi: 10.48550/arXiv.1901.08277.

[11] Yu, H.—Liu, Z.—Liu, Y.—Chen, T.—Cong, M.—Weng, X.—Niyato, D.—Yang, Q.: A Fairness-Aware Incentive Scheme for Federated Learning. Proceedings of the AAAI/ACM Conference on AI, Ethics, and Society (AIES '20), 2020, pp. 393–399, doi: 10.1145/3375627.3375840.

[12] Truex, S.—Baracaldo, N.—Anwar, A.—Steinke, T.—Ludwig, H.—Zhang, R.—Zhou, Y.: A Hybrid Approach to Privacy-Preserving Federated Learning. Proceedings of the 12th ACM Workshop on Artificial Intelligence and Security (AISec '19), 2019, pp. 1–11, doi: 10.1145/3338501.3357370.

[13] Süzen, A. A.—Şimşek, M. A.: A Novel Approach to Machine Learning Application to Protect Privacy Data in Healthcare: Federated Learning. NamIk Kemal Medical Journal, Vol. 8, 2020, No. 1, pp. 22–30, doi: 10.37696/nkmj.660762.

[14] Lin, S.—Yang, G.—Zhang, J.: Real-Time Edge Intelligence in the Making: A Collaborative Learning Framework via Federated Meta-Learning. CoRR, 2020, doi: 10.48550/arXiv.2001.03229.

[15] Pandey, S. R.—Tran, N. H.—Bennis, M.—Tun, Y. K.—Manzoor, A.—Hong, C. S.: A Crowdsourcing Framework for On-Device Federated Learning. IEEE Transactions on Wireless Communications, Vol. 19, 2020, No. 5, pp. 3241–3256, doi: 10.1109/TWC.2020.2971981.

[16] Allaoui, T.—Gasmi, K.—Ezzedine, T.: Reinforcement Learning Based Task Offloading of IoT Applications in Fog Computing: Algorithms and Optimization Techniques. Cluster Computing, Vol. 27, 2024, No. 8, pp. 10299–10324, doi: 10.1007/s10586-024-04518-z.

[17] Dang, T. N.—Hong, C. S.: A Distributed ADMM Approach for Data Offloading in Fog Computing. Korean Information Science Society 2016 Winter Conference Pro-

ceedings, 2016, pp. 1057–1059.

[18] Fricker, C.—Guillemin, F.—Robert, P.—Thompson, G.: Analysis of an Offloading Scheme for Data Centers in the Framework of Fog Computing. ACM Transactions on Modeling and Performance Evaluation of Computing Systems (TOMPECS), Vol. 1, 2016, No. 4, Art. No. 16, doi: 10.1145/2950047.

[19] Ding, Y.—Li, K.—Liu, C.—Li, K.: A Potential Game Theoretic Approach to Computation Offloading Strategy Optimization in End-Edge-Cloud Computing. IEEE Transactions on Parallel and Distributed Systems, Vol. 33, 2022, No. 6, pp. 1503–1519, doi: 10.1109/TPDS.2021.3112604.

[20] Zhu, Q.—Si, B.—Yang, F.—Ma, Y.: Task Offloading Decision in Fog Computing System. China Communications, Vol. 14, 2017, No. 11, pp. 59–68, doi: 10.1109/CC.2017.8233651.

[21] Liu, L.—Chang, Z.—Guo, X.: Socially-Aware Dynamic Computation Offloading Scheme for Fog Computing System with Energy Harvesting Devices. IEEE Internet of Things Journal, Vol. 5, 2018, No. 3, pp. 1869–1879, doi: 10.1109/JIOT.2018.2816682.

[22] Sulistyo, M. A.—Setiawan, D.: Deep Reinforcement Learning-Based Algorithm for Dynamic Resource Allocation in Edge Computing. ALCOM: Journal of Algorithm and Computing, Vol. 1, 2025, No. 1, pp. 13–22, doi: 10.63846/fb7zns45.

[23] Phan, L. A.—Nguyen, D. T.—Lee, M.—Park, D. H.—Kim, T.: Dynamic Fog-to-Fog Offloading in SDN-Based Fog Computing Systems. Future Generation Computer Systems, Vol. 117, 2021, pp. 486–497, doi: 10.1016/j.future.2020.12.021.

[24] Luo, J.—Qian, Q.—Yin, L.—Qiao, Y.: A Game-Theoretical Approach for Task Offloading in Edge Computing. 2020 16th International Conference on Mobility, Sensing and Networking (MSN), 2020, pp. 756–761, doi: 10.1109/MSN50589.2020.00129.

[25] Baek, J.—Kaddoum, G.: Heterogeneous Task Offloading and Resource Allocations via Deep Recurrent Reinforcement Learning in Partial Observable Multi-Fog Networks. IEEE Internet of Things Journal, Vol. 8, 2021, No. 2, pp. 1041–1056, doi: 10.1109/JIOT.2020.3009540.

[26] Zhang, T.—Gao, L.—He, C.—Zhang, M.—Krishnamachari, B.—Avestimehr, A. S.: Federated Learning for the Internet of Things: Applications, Challenges, and Opportunities. IEEE Internet of Things Magazine, Vol. 5, 2022, No. 1, pp. 24–29, doi: 10.1109/IOTM.004.2100182.

[27] Kim, H.—Park, J.—Bennis, M.—Kim, S. L.: Blockchained on-Device Federated Learning. IEEE Communications Letters, Vol. 24, 2020, No. 6, pp. 1279–1283, doi: 10.1109/LCOMM.2019.2921755.

[28] Abad, M. S. H.—Ozfatura, E.—Gunduz, D.—Ercetin, O.: Hierarchical Federated Learning ACROSS Heterogeneous Cellular Networks. ICASSP 2020 – 2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), 2020, pp. 8866–8870, doi: 10.1109/ICASSP40776.2020.9054634.

[29] Ren, J.—Wang, H.—Hou, T.—Zheng, S.—Tang, C.: Federated Learning-Based Computation Offloading Optimization in Edge Computing-Supported Internet of Things. IEEE Access, Vol. 7, 2019, pp. 69194–69201, doi: 10.1109/ACCESS.2019.2919736.

[30] Ye, Y.—Li, S.—Liu, F.—Tang, Y.—Hu, W.: EdgeFed: Optimized Federated

Learning Based on Edge Computing. IEEE Access, Vol. 8, 2020, pp. 209191–209198, doi: 10.1109/ACCESS.2020.3038287.

[31] Saha, R.—Misra, S.—Deb, P. K.: FogFL: Fog Assisted Federated Learning for Resource-Constrained IoT Devices. IEEE Internet of Things Journal, Vol. 8, 2021, No. 10, pp. 8456–8463, doi: 10.1109/JIOT.2020.3046509.

Mohammadreza Sharafi Hoveyda is a Ph.D. student in computer software at the Science and Research Branch of Islamic Azad University in Maybod, where he began his studies in 2016. He is currently a student in the Department of Computer Engineering at the Maybod Branch of Islamic Azad University, Maybod, Iran. His research interests include interoperability, cloud computing, fog computing, programming C#.

Mohammadreza Mollahoseini Ardakani received his Ph.D. degree in computer software from the Science and Research Branch, Islamic Azad University, Tehran in 2018. He is Assistant Professor at the Department of Computer Engineering, Maybod Branch, Islamic Azad University, Maybod, Iran. His research interests are interoperability, databases, cloud computing, fog computing, and service oriented architecture.

Vahid Ayatollahitafti received his Ph.D. in computer science from the Universiti Teknologi Malaysia and his M.Sc. in computer engineering from the Islamic Azad University, Research and Science branch, Iran. He is an Assistant Professor in the Faculty of Computing, Taft Branch, Islamic Azad University, Iran. His research interests include computer networks and Internet of Things.