FedMint: Intelligent Bilateral Client Selection in Federated Learning With Newcomer IoT Devices

Osama Wehbi, Graduate Student Member, IEEE, Sarhad Arisdakessian, Omar Abdel Wahab, Member, IEEE, Hadi Otrok, Member, IEEE, Azzam Mourad, Senior Member, IEEE, and Mohsen Guizani, Fellow, IEEE

Abstract—Federated learning (FL) is a novel distributed privacy-preserving learning paradigm, which enables the collaboration among several participants (e.g., Internet of Things (IoT) devices) for the training of machine learning models. However, selecting the participants that would contribute to this collaborative training is highly challenging. Adopting a random selection strategy would entail substantial problems due to the heterogeneity in terms of data quality, and computational and communication resources across the participants. Although several approaches have been proposed in the literature to overcome the problem of random selection, most of these approaches follow a unilateral selection strategy. In fact, they base their selection strategy on only the federated server’s side, while overlooking the interests of the client devices in the process. To overcome this problem, we present in this article FedMint, an intelligent client selection approach for FL on IoT devices using game theory and bootstrapping mechanism. Our solution involves the design of: 1) preference functions for the client IoT devices and federated servers to allow them to rank each other according to several factors, such as accuracy and price; 2) intelligent matching algorithms that take into account the preferences of both parties in their design; and 3) bootstrapping technique that capitalizes on the collaboration of multiple federated servers in order to assign initial accuracy value for the newly connected IoT devices. We compare our approach against the VanillaFL selection process as well as other state-of-the-art approach and showcase the superiority of our proposal.

Index Terms—Bootstrapping, client selection, federated learning (FL), game theory, incentive mechanism, Internet of Things (IoT), newcomer client, pricing.

I. INTRODUCTION

The adoption and popularity of Internet of Things (IoT) devices is surging day after day. Based on reports from the international data corporation (IDC), by 2025 the world will contain around 41.6 billion IoT devices. These devices generate large amounts of data. By taking advantage of the heterogeneity and affinity of these data, businesses have the chance to improve their production and business strategies and hence increase their profits. However, most of the times, the common strategy for analyzing IoT data is to gather the data from the devices and to offload these data to a central server for training and pattern extraction [1], [2]. This might not necessarily be scalable in the light of the exponential growth of IoT devices and significant data heterogeneity over the devices [3], [4], [5]. Additionally, considering the vast volume of pervasive IoT data-sets in the big data age [6], coupled with the IoT devices resource constrained nature [7], [8], it is becoming harder to move large amounts of data over the network to cloud data centers for centralized analysis [9], [10]. Another major concern with such an analysis strategy relates to the privacy risks stemming from the sharing of the data with third-party servers [11], [12], [13]. This is problematic, especially if there is sensitive information in the training data.

To address communication and privacy concerns, federated learning (FL) has emerged as a potential solution [14]. FL enables localized and distributed training on individual devices [15], [16], and was initially introduced by Google in 2016 for collaborative learning through Android smartphones [17], [18]. FL holds great promise in transforming data analytics across various critical domains, such as healthcare, transportation, finance, and smart homes, as it can be implemented on any edge device [19]. The typical federated training process involves multiple communication rounds, which conclude when the global model reaches the desired accuracy [20]. The federated server, acting as the edge server in FL, begins by generating a generic machine-learning model. During each communication cycle, the server shares the global model parameters with a selected set of client devices. The clients then train the model using their local data and send back the updated parameters to the federated server. The server aggregates these updates to form a new global model [21], which is subsequently sent back to the clients. This iterative process continues until the desired accuracy level is achieved or a predetermined number of communication rounds is reached.

20884 IEEE INTERNET OF THINGS JOURNAL, VOL. 10, NO. 23, 1 DECEMBER 2023

2327-4662 © 2023 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See https://www.ieee.org/publications/rights/index.html for more information.
In this work, we present FedMint, a novel client selection mechanism for FL. FedMint incorporates game theory and bootstrapping to consider the needs, preferences, and restrictions of both client devices and federated servers. Unlike previous approaches focusing predominantly on server needs, our approach ensures fairness by considering clients’ requirements. This fosters a balanced client selection process that maximizes client and server accuracy incentives. Moreover, our approach utilizes distributed matching, which suits the distributed nature of FL environments. We also address newcomer IoT devices, enhancing the environment’s flexibility and adaptability.

A. Problem Statement

In the default FL process, the clients that participate in the learning rounds are selected randomly [22]. This might be problematic for two main reasons. First, in order for the training to take place and the results to be communicated with the federated server, the IoT devices (hereafter, interchangeably also called clients) need to dedicate appropriate amounts of resources, such as CPU, RAM, and bandwidth to be able to train the model and transmit it efficiently [23], [24]. Nonetheless, this criterion is often not realized due to the resource limitations of certain IoT devices, which have low computing and communication capabilities, causing considerable latencies to the synchronized parameter aggregation process at the server [3]. Second, owing to the data heterogeneity at the level of the clients in FL, different clients might have different types of data sets with varying sizes, qualities, and distributions, a problem that is referred to as nonindependent and identically distributed (non-IID). Thus, selecting clients at random can result in having clients with low resources or clients that hold smaller amounts of data [25]. This might hinder the objective of achieving a certain desired accuracy level, and could result in a considerably high number of communication rounds. In addition, the problem of newcomer client devices is challenging and has not been appropriately addressed in the literature. The existence of newcomer IoT devices in the environment makes the client selection process further complicated. Dealing with devices that do not have any recorded or historical data from previous interactions that can assist in evaluating the status of such a device in order to derive a decision is very challenging. Moreover, most of the existing client selection approaches have a unilateral selection mechanism in which the federated servers take the selection decision based on specific standards, which results in an unfair or win-lose situation wherein the server’s needs are met while the clients’ opinions are completely neglected.

B. Contributions

In this work, we propose FedMint, a novel client selection approach for FL on IoT devices. FedMint addresses the limitations of random client selection by incorporating trust bootstrapping and matching game theory [26], [27], [28], [29]. Initially, we employ a bootstrapping method for new IoT devices to obtain their initial accuracy values. This involves collaboration between active federated servers and a central bootstrapping server, leveraging recorded interactions to train a decision tree (DT) model and predict newcomer accuracy. Next, we utilize a matching game approach, where clients and federated servers create preference lists based on specific criteria. Unlike random selection, matching occurs in each FL communication round. Our approach enables federated servers to consider IoT device data types and accuracy levels for an efficient selection process. Additionally, IoT devices’ preferences are taken into account in terms of monetary rewards.

Thereafter, the main contributions of our work can be summarized as follows.

1) Designing an accuracy bootstrapping model that helps federated servers assign initial accuracy values for the newcomer IoT devices having no past participation. The proposed bootstrapping model ensures fairness between already active, and newcomer IoT devices in terms of the chance to participate in future FL training rounds.

2) Devising a rewarding model for federated servers to encourage them to engage in the bootstrapping process. This is achieved by setting a limit on the bootstrapping requests that a federated server can make and linking the increase of this limit to the number and quality of contributions made by each server to the bootstrapping phase.

3) Proposing a bilateral client selection approach for FL using matching game theory. To the best of our knowledge, this is the first client selection approach in FL that takes into account the preferences of both the federated servers and client devices in the selection process while considering the newcomer IoT devices problem.

4) Elaborating a distinct optimization problem for the federated servers as well as for the client IoT devices while expressing the objectives and constraints of both parties in the selection process.

5) Implementing a set of distributed matching algorithms which takes into account the preferences of both the federated servers and client devices. The proposed algorithms lead to a stable matching point from which neither the servers nor clients have incentive to deviate.

C. Paper Outline

In Section II, we study the literature on client selection and discuss the originality of our solution. In Section III, we create the optimization problems for the federated servers and client IoT devices. In Section IV, we introduce the bootstrapping mechanism and architecture in addition to the bootstrapping motivation function for the federated servers. In Section V, we describe Fed-IoT matching game principles and terminology, interpret preference functions, and suggest algorithms for creating preference lists. In Section VI, we deliver the distributed version of the matching algorithms implementation. In Section VII, we demonstrate the simulation setup we utilized to run our tests and interpret the outcomes. Finally, we conclude our work in Section VIII.
II. RELATED WORK

In this section, we review relevant literature on client selection and trust bootstrapping in FL.

A. Client Selection

Several works in FL focus on client selection. In [30], a probability-based client selection mechanism is proposed. Clients are selected based on a probability calculated using an evaluation function on the client’s device. This approach aims to minimize the required training rounds to achieve the desired accuracy level. In [31], the POWER-OF-CHOICE framework is introduced, which balances solution bias and convergence speed. This approach demonstrates faster convergence (3×) and higher test accuracy (10% improvement) compared to random selection. In [32], the FedCS approach addresses the challenge of limited computational resources on client devices. It efficiently manages clients based on their resource conditions, mitigating the issue of resource variety. Constraints are imposed on the accepted updated models to ensure effective FL.

AbdulRahman et al. [33] introduced FedMCCS, a multicriteria methodology targeting the client selection process in FL to address the heterogeneity of client devices. This approach takes into account a device’s resources. The resources are assessed to predict whether or not the device can accommodate an FL task. Huang et al. [34] proposed an approach, called RBCS-F, which advocates a fairness-guaranteed algorithm. The algorithm seeks to establish a suitable balance among both training efficiency and fairness, while minimizing the average model exchange time. This approach ensures that trainers with low importance are not neglected from participating in the federated training process.

Zhao et al. [35] introduced Newt, a novel client selection approach that investigates a tradeoff between accuracy and system advancement. As a novel feature of client selection technique design, a control on selection frequency is incorporated in the approach. Chen et al. [36] proposed a novel client selection method in FL that considers clients’ data quality, computing power as well as their wireless channel conditions. The proposed method uses a multiobjective optimization algorithm to select a subset of clients that can contribute the most to the model training while minimizing the communication overhead while ensuring fairness among clients. Simulation results demonstrate that the proposed framework outperforms existing methods in terms of convergence speed and communication efficiency.

Zhang et al. [37] proposed a multiagent approach to optimize FL in distributed industrial IoT systems. Specifically, this article focuses on the client selection process and utilizes a reinforcement learning algorithm to dynamically select clients based on their historical performance and network conditions. The multiagent approach enables the FL system to adapt to the dynamic and heterogeneous nature of the distributed industrial IoT systems. Yang et al. [38] proposed a novel approach for client selection in FL for wireless edge AI. The study offers a lead FL algorithm that, based on data and computing capabilities, picks the most appropriate clients to participate in the FL process. The suggested method is based on neuromorphic computing and employs a spiking neural network to assess each client’s eligibility. According to the findings, the lead FL algorithm may be a potential solution to the client selection problem in FL for wireless edge AI.

Chen et al. [39] developed a matching-theoretic method in multiaccess edge computing network with incomplete preference list to handle the low-latency task scheduling problem. The matching occurs between the edge nodes in charge of the FL task and the end devices in a wide environment. Experiments reveal that the complete preference list matching approach performs slightly better than the matching approach by reducing the latency due to the missing information. Finally, to manage FL task allocation and defend against malicious clients, Kang et al. [40] applied a modified one-to-one two-sided matching theory between workers and task publishers, as well as a worker reputation metric.

The works surveyed focus more on the federated server’s needs, neglecting the client devices’ needs and preferences. This approach can lead to biased and potentially unfair scenarios where the federated server has complete control over selecting clients. On the other hand, our approach considers the preferences and restrictions of the federated server as well as the clients during the selection process, ensuring fair and unbiased decisions. Accordingly, our proposed solution prioritizes minimizing specific metrics and maximizing client incentives, alongside federated server’s global model accuracy. Additionally, we employ a distributed version of the matching game theory that is better suited to the distributed nature of FL. Moreover, the articles cited above do not address the problem of newcomer IoT devices in FL, which we do in our work.

B. Trust Bootstrapping

Trust bootstrapping has been used to solve the recommender system cold-start problem in many fields, such as business, networking, and cloud services. Cao et al. [41] tried to mitigate the malicious client aspect in FL by introducing FLTrust. Unlike the normal byzantine-robust methods that rely on statistics analysis for malicious clients detection, the proposed approach counts on a small amount of collected data to bootstrap trust. Dong et al. [42] introduced FLOD. As a new byzantine-resistant FL approach. The proposed method capitalizes on trust bootstrapping and the hamming distance-based aggregation, in addition to additive homomorphic encryption and multiple optimizations to protect privacy and byzantine-robustness in FL. The above-mentioned approaches named as FLTrust and FLOD, showcase that using trust bootstrapping to assign initial trust score for each local model is effective compared to other methods. Wahab et al. [43] tried to solve the recommendation system cold start problem using FL by introducing a double deep Q learning model that counts on the IoT devices trust score, as well as the resources availability in the selection process. However, these solutions, unlike our approach, are aimed at the bootstrapping of the FL process and the model in general rather than newcomer IoT devices, which our work focuses on.
III. PROBLEM FORMULATION

In this section, we express the client IoT devices selection problem in FL as an optimization problem and clarify the relevant constraints. Note that, we describe the various symbols utilized throughout this work in Table I.

A. Client IoT Device Optimization Problem

The primary goal of IoT devices is to increase their profitability. The earnings $E(i)$ of every IoT device are determined based on the amount of resources that the device promises to dedicate for the FL process (i.e., $CPU_{proi}$, $RAM_{proi}$, and $band_{proi}$). The earnings of an IoT device is the summation of two functions, namely: 1) operational and 2) network traffic earnings.

1) Operational Earnings: The operational earnings of an IoT device are made up of two measures, i.e., $CPU_{proi}$ and $RAM_{proi}$. The CPU and RAM utilization cost (in MIPS) measures the amount of CPU and RAM used by a specific IoT device $i$ when performing federated server operations. Formally, the operational earnings $e_i(o)$ of $i \in I$ are defined as stated below

$$e_i(O) = CPU_{proi} \times pw + RAM_{proi} \times pw'.$$ (1)

2) Traffic Earnings: In FL, the devices must send/receive the model parameters to/from the federated servers at certain bandwidth rates. Depending on the underlying demand, the active physical links may be unavailable at different periods. Thus, the traffic earnings of IoT devices are estimated as the bandwidth earnings incurred on the $s \leftrightarrow i$ link multiplied by the link’s scaled undergoing delay as a penalty. In formal terms, the traffic earnings $e_i(t)$ of an IoT device $i \in I$ interacting with a federated server $s \in S$ is defined as follows:

$$e_i(t) = (Band_{proi} \times pw) \times (1 - L_{is}).$$ (2)

As a result, each IoT device $i \in I$ must maximize the objective function in (3), where $std$ represents the standard deviation (SD) of the IoT device $i$’s local accuracy compared to the federated server’s global model accuracy of $s \in S$ at round $r_n$. Such a multiplication reflects the fact that each IoT device will be penalized in terms of the gap between its local accuracy and the overall global model accuracy. Therefore, as the IoT device’s local accuracy $Acc_i$ tends to be closer to the global model’s accuracy $Acc_s$, the IoT device will receive a higher reward

$$E(i) = (e_i(O) + e_i(t)) \times (1 - std_{rn}).$$ (3)

Constraint 1: Each IoT device $i \in I$ can be matched with only one single federated server, per each communication round

$$0 \leq |\gamma(i)| \leq 1.$$ (4)

B. Federated Servers Optimization Problem

Federated servers are interested in maximizing the accuracy $Acc_s$ of the deep learning model, by selecting the most appropriate set of client devices in terms of historical accuracy. The accuracy $Acc_s$ of the FL training process can be derived as per

$$Acc_s = \frac{\sum_{n=1}^{l'} weighted\_accuracy_n}{\sum_{n=1}^{l'} test\_data\_size_n}$$ (5)

where $l'$ represents subset of IoT devices of $I' \subseteq I$ that participated in FL round $r_n$ with federated server $s$. Weighted_accuracy can be calculated for each IoT $i \in I$ as follows:

$$weighted\_accuracy_i = Acc_i \times test\_data\_size_i.$$ (6)

In this way, a federated server $s$ will be more interested in selecting an IoT device $i$ over device $\tilde{i}$ in a certain communication round if and only if

$$Acc_i > Acc_{\tilde{i}}.$$ (7)

Constraint 2: The selected client IoT devices total number should not exceed the amount of requested IoT $C_s$ by $s$

$$N_s \leq C_s.$$ (8)
The central server trains a DT regression model on these data sets to predict the newcomer’s accuracy (Step V). The bootstrapping server sends the initial expected accuracy to the requesting server (Step VI). The DT uses a tree structure to develop regression or classification models, recursively dividing the data set into smaller segments based on attribute values. The resulting tree has leaf nodes representing numerical decisions and decision nodes with branches based on attribute values. The root node represents the most reliable predictor using statistical metrics [e.g., SD reduction (SDR)].

B. Decision Tree Creation

The DT model is a well-known supervised machine learning model that capitalizes on the ID3 technique for creating DTs. This is done by performing a greedy top-down search across the range of possible paths without backtracking [46]. DT uses a tree structure to develop regression or classification models. The major distinction between regression and classification DTs is that the results of classification-based DTs are categorical, whereas the results of regression-based DTs are continuous. The regression DTs accept both ordered and continuous data.

The ID3 approach can be applied to create a regression DT by using SDR instead of Information Gain [47]. The SDR technique is going to be used in order to measure the homogeneity in a feature. A DT is created in a top-down way out-of a root node by splitting the data into segments containing homogeneous samples with comparable values. To determine the uniformity of a numerical data sample, we utilize SD. A numerical sample with a low SD is more likely to be homogeneous, whereas a sample with a high SD is less likely to be homogeneous. The SDR is a measure of the decrease in the SD after splitting a data set based on a certain attribute. The most homogeneous branch is determined by the attributes that result in the greatest SD decrease. The SDR of a specific feature $X$ can be derived by subtracting the SD of the target $Y$, i.e., $SD(Y)$, before the split, from the SD after the split by $X$ $SD(Y, X)$ represented by

$$SDR(Y, X) = SD(Y) - SD(Y, X) \quad (9)$$

where a certain SD can be calculated for one attribute $x$ as follows:

$$SD(x) = \sqrt{\frac{\sum(x - \bar{x})^2}{n}}. \quad (10)$$

The symbol $n$ represents the sample size and $\bar{x}$ represents the sample’s mean that can be derived as follows:

$$\bar{x} = \frac{\sum x}{n}. \quad (11)$$

The SD for two attributes (Target and Predictor) is defined as follows:

$$SD(Y, X) = \sum_{c \in X} P(c)SD(c). \quad (12)$$

Then, the coefficient of variance (CV) can be computed as per (5)

$$CV = \frac{SD}{\bar{x}} \times 100\%. \quad (13)$$
To demonstrate how SDR may be used in practical FL scenarios to generate DTs, we provide an illustrative example using a portion of the data set shown in Table II. This data set was generated in order to be used subsequently in the experimental analysis. The data set stores information on the IoT devices that contribute in the FL rounds in terms of their type, deployment region, provider, and observed accuracy.

First of all, we should calculate the SD for the target which means the Accuracy denoted by SD (Accuracy). By applying the above-explained equations on the Accuracy feature we can find that:

1) \[ n = 14; \]
2) \[ \bar{x} = (917.55/14) = 65.53; \]
3) SD (Accuracy) = 13.96;
4) CV = 21.31%.

The previous calculations are needed in order to evaluate the splitting impact of each feature. Starting with the Provider feature as illustrated in Table III, the data set is grouped by category where the SD for each group is calculated alongside with the frequency of each category.

The Provider feature SDR can be determined by applying the following illustrated methodology.

1) \[ \text{SD} (\text{Accuracy}, \text{Provider}) = P(\text{P1}) \times \text{SD(P1)} + P(\text{P2}) \times \text{SD(P2)} + P(\text{P3}) \times \text{SD(P3)} + P(\text{P4}) \times \text{SD(P4)} = (5/14) \times (4.16) + (2/14) \times (1.74) + (3/14) \times (18.51) + (4/14) \times (8.50) = 8.13. \]

2) \[ \text{SD} (\text{Accuracy}, \text{Provider}) = \text{SD} (\text{Accuracy}) - \text{SD} (\text{Accuracy}, \text{Provider}) = 13.96 - 8.13 = 5.83. \]

By applying the same concept for all the features (i.e., Region and DeviceType) we will obtain the following SD values.

1) \[ \text{SD} (\text{Accuracy}, \text{Provider}) = 5.83. \]
2) \[ \text{SD} (\text{Accuracy}, \text{Region}) = 13.96 - 9.51 = 4.45. \]
3) \[ \text{SD} (\text{Accuracy}, \text{DevType}) = 13.96 - 12.28 = 1.67. \]

Based on the calculated values, the root node should be assigned to the attribute with the greatest SDR which is the Provider in our example. The data set is partitioned depending on the values of the Provider feature as shown in Fig. 2. This process is repeated on the nonleaf branches until all data has been processed, or until a branch’s coefficient of variation (CV) falls below a specific threshold and/or few or no more instances remain in the branch. Finally, if there are more than one occurrence at a leaf node, we use the average as the final value for the target. By applying the above rules on our data sample with \( n = 3 \) as number of instances threshold and CV = 10% as coefficient of deviation threshold, we obtain the final tree structure illustrated in Fig. 3.

C. Motivation Function

Federated servers should be encouraged to engage in the bootstrapping process. Therefore, it is very important to provide a rewarding technique for these servers that have participated in the bootstrapping process as encouragement for them to participate again. We propose (14), that represents a positive relation between the number of bootstrapping calls that a federated server can make \( \text{calls}(s) \), with respect to its cumulative previous number of contributions \( C_{cont} \) and the data rate \( DR \), that the federated server provides based on the
total data set size uploaded at time \( t \) by all the participated servers

\[
Calls(s'_t) = Calls(s'_t-1) + (|Ccont| + |Ccont \cdot DR_t| + 1)
\]

(14)

where \( DR_t \) of a specific federated server \( s' \) can be calculated as follows:

\[
DR_t = \frac{\text{uploaded}_\text{data}_\text{size}_t}{\sum_{n=1}^{S} \text{uploaded}_\text{data}_\text{size}_n}. 
\]

(15)

Establishing win-to-win or rewarding methodology between federated servers and the bootstrapping server is important to guarantee the constant update of the provided data by the federated servers used in the bootstrapping model training. Thus, each federated server is going to be forced to contribute by its data in order to be able to make further bootstrapping requests from the bootstrapping server, to assign accuracy values for new IoT devices that can be potential clients to a certain FL round. Such a relationship is beneficial to both parties. In fact, the federated servers can benefit by inquiring about newcomer IoT devices and get extra number of eligible bootstrapping calls. On the other hand, the bootstrapping server can benefit from the data provided by the federated servers to keep the DT model updated.

V. PROPOSED APPROACH: INTELLIGENT CLIENT SELECTION MECHANISM

In this section, we describe the proposed architecture, explain the preference functions for both clients and federated servers, and provide the intelligent client selection algorithms.

A. Proposed Architecture and Solution Overview

The initial communication round between federated server and client IoT devices is illustrated in Fig. 4. This round is important for the distributed intelligent selection approach, as it allows both parties to exchange all the needed information for the selection process. The communication starts with a demand request to all the active IoT broadcasted by a specific federated server (Step I), after that the interested/available IoT devices reply to the federated server with a message that contains all the needed information including the IoT device accuracy from previous work (Step II). If the IoT device is newly deployed then the federated server is going to use the bootstrapping methodology discussed in Section IV. Finally, the federated server sends his offer to the designated IoT device (Step III).

Our solution takes two inputs, i.e., a set of federated servers that need to select a set of clients to execute an FL task, and a set of active IoT devices that are ready and willing to participate in the FL process.

Hereafter, we highlight the main steps of our solution, mainly, the preference list creation and the matching process based on them.

1) Preference Lists Creation: In this step, each of the clients and federated servers build their preferences lists.

a) Initially each active federated server broadcasts to the IoT devices in the environment the requirements of each FL task. The IoT devices that are active and willing to participate reply by sending to the server their accuracy values obtained from their participation in previous FL tasks or from the bootstrapping server alongside their resource information.

b) Client IoT Device Preference List: Contains the list of federated servers sorted based on the reward values that these servers offer to pay for the IoT device. IoT devices prioritize the federated servers that offer higher rewards.

c) Federated Server Preference List: Contains the list of IoT devices sorted according to the accuracy values of the IoT devices. Federated servers prioritize the clients that have higher accuracy values.

2) Matching: In this step, the matching between the federated servers and client IoT devices takes place. The matching is accomplished based on the matching algorithms which we describe in Section VI. The algorithms rely on the preference lists of the IoT devices and federated servers which we propose in Sections V-C and V-D, respectively. The aim of this step is to reach a stable matching point wherein each IoT device is matched to a federated server and both parties do not have any incentive to deviate from this matching. At the end of the matching game, each federated server will have the requested number of clients \( Cs \) and each client device will be matched to a federated server.

The high-level system architecture of our suggested approach is illustrated in Fig. 5.

B. Matching Fed-IoT Game Formulation

In this section, we formulate the Fed-IoT matching game, and establish preference functions of both the federated servers and client IoT devices. We finally provide the algorithms that allow us to create these preference functions. It should be noted that our matching approach is inspired by the methodology discussing [48].

Definition 1: We define \( \gamma \) as a matching relation produced by the matching game between the IoT devices and federated servers, where \( \gamma \) is a function \( I \cup S \rightarrow 2^{I \cup S} \) that satisfies the following conditions.

1) \( \gamma(i) \subseteq S \), where \( |\gamma(i)| = 0 \) implies that client \( i \) is not assigned to any federated server.

2) \( \gamma(s) \subseteq I \), where that \( N_s < C_i \) implies that the federated server \( s \) did not reach the needed number of clients for its FL task.

3) \( i \in \gamma(s) \) if \( \leftrightarrow \gamma(i) = s, \forall i \in I, s \in S \).
Definition 2: An IoT-Server pair \((i, s)\) is said to block a matching relation \(\gamma\) if \(\exists (i, s)\) where \(i \in \gamma(s)\) and \(s \in \gamma(i)\) we have \(i \succ s\) \(\gamma(s)\) and \(s \succ i\) \(\gamma(i)\).

Definition 3: When a federated server \(s\) reaches the needed number of clients \(C_s\), it is considered as saturated. If a server still needs some clients, any IoT device \(i\) will be accepted as long as it meets the requirements.

Definition 4: A stable matching relation \(\gamma\) exists when: 1) there are no blocking relationships and 2) every federated server is matched with the needed number of client devices.

C. Client IoT Device Preference Function

IoT devices wish to be matched with those federated servers that maximize the former’s reward. There exist a complete, strict, and transitive preference relation \(P_f(S)\) for each IoT device \(i \in I\) with each federated server \(s \in S\). A preference relationship \(s \succ_i s'\) means that federated server \(s\) is preferred over federated server \(s'\) to IoT device \(i\). Furthermore, if an IoT device \(i\) does not have a clear preference between joining \(s\) or staying unpaired, a federated server \(s\) is said to be undesirable to \(i\). Based on this description, an IoT device \(i\)’s preference function can be described as follows:

\[
s_1 \succ_i s_2 \Leftrightarrow P_f(s_1) > P_f(s_2)
\]

where

\[
P_f(s) = \begin{cases} 
  +\infty, & \text{if } s \text{ offers the highest reward} \\
  -\infty, & \text{otherwise.}
\end{cases}
\]

D. Federated Server Preference Function

A federated server prefers to improve the accuracy efficiency by selecting the IoT devices that can train the deep learning model with the best possible local accuracy. There exist a complete, strict, and transitive preference relation \(P_s(I)\) for each IoT device \(s \in S\) with each federated server \(i \in I\). A preference relationship \(i_1 \succ s i_2\) means that IoT device \(i_1\) is preferred by federated server \(s\) over IoT device \(i_2\). Furthermore, if \(s\) does not have a clear preference between selecting \(i\) or staying unpaired, the client device \(i\) is supposed to be undesirable to \(s\). Based on this description, a federated server \(s\)’s preference function can be represented as follows:

\[
i_1 \succ s i_2 \Leftrightarrow P_s(i_1) > P_s(i_2)
\]

where

\[
P_s(i) = \begin{cases} 
  +\infty, & \text{if the selection of } i \text{ maximizes the accuracy of the FL task} \\
  -\infty, & \text{otherwise.}
\end{cases}
\]

E. Client Device Preference List Creation Algorithm

In Algorithm 1, we show how to establish a preference list for each client IoT device.

The algorithm takes a collection of federated servers \(S\) as input and generates the client IoT device \(i\) preference list \(P_f\). The federated servers are ordered in the preference list by their preference order to \(i\). The algorithm starts by visiting each nonvisited federated server (line 1) and validating the data type (e.g., MNIST data set) requested by the federated server (line 3). If the requested data type is available on \(i\), that corresponding federated server \(s\) will be added to \(i\)’s preference list \(P_f\) (line 4). Finally, (17) is used to compute the preference ordering across the collection of maintained federated servers in \(P_f\) (line 7). It is worth nothing mentioning that this algorithm is carried out independently by each IoT device \(i\) in \(I\) separately. The complexity for the first part of algorithm 1 is \(O(n)\), where \(n\) is the number of active federated servers since we loop through all of them. However, once that part is done, to sort the dictionary, the complexity becomes \(O(n\log n)\).

F. Federated Server Preference List Creation Algorithm

Federated servers are primarily concerned with increasing the accuracy of training their global models by trying to select the IoT devices with the highest possible accuracy levels.

Algorithm 2 is executed by each federated server in \(s \in S\) to generate their preference lists. The algorithm takes a collection of eager IoT devices as input, returning the preference list \(P_s\) for each federated server. IoT devices are sorted in the list based on their order of preference for each \(s \in S\). The

---

**Algorithm 1** Establishment of IoT Device Preference List

**Input:** Collection of active federated servers \(S\)

**Output:** Client IoT device \(i\) Preference List \(P_f\)

1: for each \(s \in S\) do
2:   mark \(s\) as visited
3:   if data type \(\phi_s \in \varphi_i\) then
4:     add \(s\) to \(P_f\)
5:   end if
6: end for
7: Sort the federated servers in \(P_f\) Using Eq. (17)
Algorithm 2 Federated Server Preference List Establishment

**Input:** Collection of client IoT devices willing to participate in the federated learning training

**Output:** Preference list $P_s$ of federated server $s$

1: for each $i \in I$ do
2: mark $i$ as visited
3: if data type $\varphi_i \in \varphi$ then
4: if $Acc_i$ is Unknown then
5: // Start Bootstrapping Steps
6: Server $s$ sends an inquiry message $\beta_i^s$ to $\delta$
7: $\delta$ receives the message
8: $\delta$ collects data from $S$
9: $\delta$ rewards each server in $S$ using Eq. (14)
10: $\delta$ updates the DT model & predict $\hat{Acc}_i$
11: $\delta$ Replies to the inquiry message by $Acc_i$
12: $\text{SET } \hat{Acc}_i = \hat{Acc}_i$
13: end if
14: add $i$ to $P_s$
15: end if
16: end for
17: Sort $P_s$ Using Eq. (19) to rank the IoT devices in $P_s$

Algorithm first validates the data type and availability of local accuracy $Acc_i$ for each IoT device in $i \in I$ (lines 1–4). If device $i$ has the requested data type but $Acc_i$ is empty, the federated server initiates a bootstrapping request $\beta_i^s$ to the central bootstrapping server $\delta$ (line 6). The Trust bootstrapping stages described in Section IV-A are represented between lines 5–12. After the bootstrapping process concludes, the federated server $s$ assigns the predicted accuracy $\hat{Acc}_i$ provided by $\delta$ to $i$, and adds it to $P_s$ (line 14). Finally, the algorithm utilizes (19) to determine the appropriate ordering of IoT devices for each federated server (line 17). The algorithm's complexity depends on whether $Acc_i$ is known or unknown. If unknown, the time complexity is $O(nm \log(m))$, reflecting the DT creation complexity, where $n$ is the number of data points, $m$ is the number of features, and $\log(m)$ is the depth of the tree. If $Acc_i$ is known, the time complexity is $O(n \log n)$, where $n$ is the number of participating IoT devices. The algorithm only needs to sort the participating IoT devices based on their preferences.

VI. SELECTION: FEDERATED SERVER AND CLIENT SELECTION ALGORITHMS

Once the preference lists have been created, the next step is to devise the appropriate algorithms that would perform the actual matching based on these lists. The end result of this stage is client devices being matched with the federated servers. Our solution is highly distributed in the sense that the IoT devices and federated servers connect directly to accomplish the matching without the need for any third-party central entity.

A. Matching Algorithm—IoT Devices

In this section, we present Algorithm 3, executed by IoT devices as part of the matching game. The algorithm takes the preference list of federated servers obtained from Algorithm 2. It iterates over the preference list of each IoT device (line 2), selecting the most desired server (line 3). The IoT device sends a work request message to the server (line 4) and waits for a response (line 5). The device and server are paired if the server agrees to be matched (lines 7 and 8). The server is moved to the bottom of the preference list if the reply is negative (line 6). The process is repeated for each FL round (line 12). Algorithm 1 is executed within Algorithm 3 to obtain updated preference lists (line 11), considering changes in server performance and the introduction or removal of servers and clients. The time complexity of Algorithm 3 is $O(n)$ per round, where $n$ is the number of federated servers in the preference list $P_i$, as it checks the availability of each server for FL.

B. Matching Algorithm—Federated Server

Algorithm 4 is employed by each federated server, taking a queue of IoT devices as input. The algorithm checks if the queue is empty (line 2) to determine if the server still accepts requests. If there are pending requests, the algorithm examines whether the requested number of clients has been reached (line 3). If not, the server sends an accept message to the IoT device (line 4), increments the selected client count, and adds the device to the selected list (lines 5 and 6). However, suppose the requested number of clients has been reached but the current IoT device is ranked higher than any previously selected device (line 7). In that case, the server breaks the agreement with the lower-ranked device (lines 8 and 9), sends an accept message to the current device, and adds it to the selection list (lines 10 and 11). If the IoT device is not ranked higher than any previously selected device, it receives a reject message (line 13), and lower-ranked devices are removed by the server (line 14). This process is repeated for each FL round (line 18), with Algorithm 2 executed to update the preference lists reflecting changes in the environment (line 17).

The time complexity of Algorithm 4 for one round is $O(n \log n)$, where $n$ is the number of IoT devices in the queue $Q_i$. The algorithm iterates through each device in the queue. It uses a priority queue or a balanced search tree to
efficiently compare device priorities and select the server's acceptable devices. The time required for sending and receiving messages and waiting for replies is relatively small and constant.

Our approach has four algorithms that can be divided into two groups. The first group comprises the first two algorithms (i.e., Algorithms 1 and 2), which are dedicated to creating the preference lists of the different players of the game (i.e., the federated servers and the clients). Algorithm 1 establishes the preference list of IoT devices by taking a collection of active servers as input and outputting the client IoT device pair. Algorithm 2 accepts messages from the accepted list and establishes the preference list of federated servers by taking a collection of IoT devices as input and outputting the preference list of federated servers. Once Algorithms 1 and 2 run and provide their outputs, those outputs serve as input for the second group of Algorithms 3 and 4. These algorithms are responsible for the game between the federated servers and the client devices. Given the distributed nature of the approach, each entity (i.e., player) has an algorithm dedicated to it. Algorithms 3 and 4 run concurrently to make the matching possible. In particular, Algorithm 3 runs on the IoT device's side, while Algorithm 4 runs on the federated server's side. Like this, the two sets of algorithms run sequentially as the output of the first two and are used as an input for the latter set, which runs concurrently, to make the client selection possible.

Based on the above, the overall time complexity of the proposed approach is dominated by the Federated Server Preference List Establishment algorithm, which is $O(n \log(m))$ in the worst case. However, this complexity can be reduced if the accuracy of the client IoT devices is known beforehand. Furthermore, the proposed approach can be further optimized by using parallel processing and distributed computing techniques.

VII. EXPERIMENTS

In this section, we describe the context in which we ran our simulations and discuss the findings of our experiments.

A. Experiments Setup

In our simulations, we use the MNIST and CIFAR data sets. The MNIST data set is from the national institute of standards and technology (NIST).\(^1\) The training sample comprises manually written digits by 250 distinct persons, 50% of them are secondary school students and 50% are from the Census Bureau. The testing data set, likewise, includes the same distribution of manually written digital information. The MNIST data set comprises 60k pictures as training data set and 10k as testing data set, all having a size of $28 \times 28$ pixels and 256 gray levels [49]. In terms of label and size distributions, the data set was split over the client IoT devices in a non-IID fashion. An initial set of $C = 100$ IoT devices was created and each round ten clients are going to be added, each having a data set size in the interval of $[100, 450]$ images. Each IoT device has at least one label and no more than four labels in its class label distribution. To run the simulations, we build our own platform in which each IoT device has a CPU capacity ranged between 300 and 700 MIPS, RAM capacity ranged between 400 and 900 MB, and bandwidth ranged between 500 and 900 MB/s. The latency across each federated server and IoT device pair varies between 0.1 and 5 s.

Additionally, we use the CIFAR-10 data set, which is a widely used benchmark data set in computer vision tasks [50]. The data set consists of 50,000 training images and 10,000 testing images, each of size $32 \times 32$ pixels and with three color channels (RGB).\(^2\) The images in the CIFAR-10 data set belong to one of ten classes: 1) airplane; 2) automobile; 3) bird; 4) cat; 5) deer; 6) dog; 7) frog; 8) horse; 9) ship; and 10) truck. The data set is split in a non-IID fashion. An initial set of $C = 100$ IoT devices was created, where each client device is randomly assigned a set of classes, with some clients having access to more classes than others. The data set is distributed using the SharedDistributor method [22] with a shared size of 200, and each client has access to three shared images per class.

In a series of various experiments showcased below, we compare our proposed method, FedMint, against the baseline approach, VanillaFL, which was initially introduced by Google [22], as well as against a state-of-the-art approach [51] in which clients are clustered based on their local model weights. We call this approach “hierarchical” in our experiments. In the experiments involving the VanillaFL approach, client selection occurs randomly. We have used the MNIST data set in this series of experiments. To conduct our simulations, we initialize the set of clients, $C$, to 100 and incrementally add ten clients in each round. We set the value

\(^1\)http://yann.lecun.com/exdb/mnist/
\(^2\)https://www.cs.toronto.edu/kriz/cifar.html
Fig. 6. Our solution increases the IoT devices rewards compared to [22] by more than 58%. (a) Federated Server 1 (proposed approach versus VanillaFL). (b) Federated Server 2 (proposed approach versus VanillaFL).

Fig. 7. Our solution achieves high accuracy levels compared to [22] by more than 20%. (a) Federated Server 1 (proposed approach versus VanillaFL). (b) Federated Server 2 (proposed approach versus VanillaFL).

of $K$ to 10 for both the VanillaFL approach and FedMint. Additionally, the number of federated servers in FedMint is set to 2, and the total number of FL rounds, $R$, is set to 15. While for the second comparison with experiments that involve [51], we have used the CIFAR-10 data set and conducted it with an initial set of 100 clients and $K$ set to 10. We also used two active federated servers and conducted a total of 500 rounds.

B. Experimental Results

In this section, we highlight the results of our various experiments. In particular, we compare and contrast our approach in several areas, such as monetary rewards, model accuracy, and effect of bootstrapping.

1) FedMint Versus VanillaFL: In this set of experiments, we compare our FedMint approach versus VanillaFL by investigating two key metrics. First, IoT monetary rewards, and second, the global model accuracy.

1) IoT Devices Monetary Rewards: In Fig. 6, we study the average reward that an IoT device gains after having participated in the FL task, over the FL rounds number. Fig. 6 shows that our solution maximizes the clients rewards considerable compared to VanillaFL. This stems from the fact that our solution takes into consideration each client’s preference in the selection mechanism. On the other hand, the random client selection in VanillaFL is server oriented, thus totally ignoring the client’s preferences in the selection process. We notice from Fig. 6 that (a) the rewards of the IoT devices that participated in FL rounds with Federated Server 1 using our approach are higher than those obtained by the IoT devices in VanillaFL in all the FL rounds by 58.29% with minimum and maximum differences of 37.31% and 67.19%, respectively. Similarly, in the case of Federated Server 2 [Fig. 6(b)], our solution enables the IoT devices to obtain higher rewards than those obtained in VanillaFL in average by 61.14% inline with minimum and maximum differences of 48.67% and 69.11%, respectively.

2) FL Model Accuracy: Each federated server aims to maximize its global model’s accuracy by selecting the IoT devices with the highest local accuracy. In Fig. 7, we measure the global model’s accuracy in both our solution and VanillaFL. The results illustrated in Fig. 7(a) and (b) represent the average global accuracy for Federated Server 1 and Federated Server 2, respectively, with respect to the FL communication rounds. To avoid biased results, we run each method several times and compute the average accuracy. Overall, as can be seen in Fig. 7(a), our solution outperforms VanillaFL in terms of model’s accuracy. Specifically, our solution achieves an accuracy level of 66.12% in the first communication round compared to 43.99% in VanillaFL in the case of Federated Server 1. Similarly, we notice from Fig. 7(b) the accuracy of our solution in the first communication round is 67.10% compared to 48.18% in VanillaFL. In the 15th rounds, both our solution and VanillaFL reach their highest accuracy levels on both servers. In more detail, Federated Server 1 reaches an accuracy level of 77.45% which is considerably higher than that of VanillaFL by 19.70%. Federated Server 2 reaches its
In this work, we proposed FedMint, a novel bilateral matching approach that effectively addresses the cold start problem faced by newcomer IoT devices in FL. FedMint considers the preferences and constraints of both federated servers and client devices and includes a trust bootstrapping system that provides initial accuracy assignments for newcomer IoT devices. Our simulation results demonstrate that FedMint significantly enhances the accuracy of the FL global model from the

highest accuracy level of 79.92% which is higher than that of VanillaFL by 20.56%.

2) FedMint Versus Hierarchical [51]: In this experiment, we compare our approach against the state-of-the-art [51] in terms of model accuracy. Fig. 8 demonstrates the superior accuracy achieved by applying our FedMint method, as indicated by the two upper lines representing Federated Servers 1 and 2, respectively. For Federated Server 1, the proposed approach is represented by the red upper line, while the hierarchical approach is represented by the lower light-purple line. The bootstrapping solution achieves an accuracy level of 16.39% in the first communication round, compared to 8.20% in the hierarchical approach. Similarly, for Federated Server 2, illustrated by the dark-blue line for the federated server that applies the matching approach for client selection versus the light-green line that uses hierarchical clustering selection approach, the bootstrapping solution achieves an accuracy level of 25.10% in the first communication round, compared to 12.60% in the other approach. In the 500th round, both Federated Servers 1 and 2 that adopt our bootstrapping solution achieve their highest accuracy, represented by 45.62% and 44.45%, respectively. This accuracy is higher than the hierarchical clustering accuracy approach by more than 15%.

3) Impact of Bootstrapping: In this series of experiments, investigate the trust bootstrapping mechanism efficiency by showing the accuracy results for the FedMint approach in Fig. 9 by assigning random initial accuracy versus using bootstrapping for getting initial accuracy for the newcomer IoT devices. The MSE plot in Fig. 10 for the Bootstrapping DT model is then interpreted. The MSE measures the amount of error in machine learning models. For a perfect model, the MSE value is 0 and this value increases as the model error increases.

As shown in Fig. 9, the accuracy obtained by applying bootstrapping illustrated by the two upper lines represent Federated Servers 1 and 2, respectively, is quite better. In case of Federated Server 1, the light-blue upper line represents the results using our bootstrapping. On the other hand, the lower light-green line represents the results by assigning random accuracy, we notice that bootstrapping solution achieves an accuracy level of 72.21% in the first communication round compared to 51.56% in the random accuracy approach. Similarly, for Federated Server 2 illustrated by the dark-blue line for the federated server that applies the bootstrapping for initial accuracy versus the red line that use randomly assigned accuracy. We notice that the bootstrapping solution achieves an accuracy level of 64.77% in the first communication round compared to 48.53% in the random accuracy approach. In the 15th round, both Federated Servers 1 and 2 that adopt our bootstrapping solution achieve their highest accuracy represented by 82.38% and 79.32%, respectively, which is higher than the random accuracy approach by more than 24%.

In Fig. 10, we study the performance of the trust bootstrapping DT model by applying K-Fold cross validation to plot the model’s MSE with $K = 10$.

Fig. 10 shows that our model performs very well since the maximum MSE value reached is very low around 0.00118. Also, we observe from the figure that the MSE value decreases as the round number increases which mean that the bootstrapping model performance is improving, where the MSE value at the first round was 0.0115 which is less by 0.0042 than the last round equal to 0.0073.

VIII. CONCLUSION

In this work, we proposed FedMint, a novel bilateral matching approach that effectively addresses the cold start problem faced by newcomer IoT devices in FL. FedMint considers the preferences and constraints of both federated servers and client devices and includes a trust bootstrapping system that provides initial accuracy assignments for newcomer IoT devices. Our simulation results demonstrate that FedMint significantly enhances the accuracy of the FL global model from the
perspective of the federated servers while also maximizing the monetary rewards of the client IoT devices. We also compared our approach against the baseline random client selection FL strategy, as well as other state-of-the-art clustering approach using two distinct data sets, namely, MNIST and CIFAR-10. Our approach improves accuracy and increases the income rewards of participating IoT devices in both comparisons, which confirms the effectiveness of our approach. Therefore, the successful implementation of FedMint can significantly enhance the effectiveness and efficiency of FL. As a future direction, we aim to investigate further the application of game theoretical strategies in this field, as well as explore the security aspect of the FL participants in distributed environments. We would also explore the potential benefits of transfer learning in the bootstrapping technique.

REFERENCES

[1] O. A. Wahab, “Intrusion detection in the IoT under data and concept drifts: Online deep learning approach,” IEEE Internet Things J., vol. 9, no. 20, pp. 19706–19716, Oct. 2022.
[2] X. Wang et al., “Toward accurate anomaly detection in Industrial Internet of Things using hierarchical federated learning,” IEEE Internet Things J., vol. 9, no. 10, pp. 7110–7119, May 2022.
[3] D. C. Nguyen, M. Ding, P. N. Pathirana, A. Seneviratne, J. Li, and H. V. Poor, “Federated learning for Internet of Things: A comprehensive survey,” IEEE Commun. Surveys Tuts., vol. 23, no. 3, pp. 1622–1658, 3rd Quart., 2021.
[4] H. Elayan, M. Aloqaily, and M. Guizani, “Sustainability of healthcare data analysis IoT-based systems using deep federated learning,” IEEE Internet Things J., vol. 9, no. 10, pp. 7338–7346, May 2022.
[5] I. Sorkhoh, M. A. Arfaoui, M. Khabbaz, and C. Assi, “Optimizing information freshness in RIS-assisted coordinated autonomous driving,” in Proc. IEEE Int. Conf. Commun., 2022, pp. 1518–1523.
[6] “Cisco global cloud index: Forecast and methodology, 2015–2020,” Cisco, San Jose, CA, USA, White Paper, 2016.
[7] X. Pei, X. Deng, S. Tian, L. Zhang, and K. Xue, “A knowledge transfer-based semi-supervised federated learning for IoT malware detection,” IEEE Trans. Dependable Secure Comput., vol. 20, no. 3, pp. 2127–2143, May/June, 2023.
[8] R. Li, P. Hong, K. Xue, M. Zhang, and T. Yang, “Resource allocation for uplink NOMA-based D2D communication in energy harvesting scenario: A two-stage game approach,” IEEE Trans. Wireless Commun., vol. 21, no. 2, pp. 976–990, Feb. 2022.
[9] I. Sorkhoh, D. Ebrahimi, C. Assi, S. Sharafeddine, and M. Khabbaz, “An infrastructure-assisted workload scheduling for computational resources exploitation in the fog-enabled vehicular network,” IEEE Internet Things J., vol. 7, no. 6, pp. 5021–5032, Jun. 2020.
[10] G. Rathee, S. Garg, G. Kaddoum, B. J. Choi, M. M. Hassan, and S. A. AlQahtani, “TrustSys: Trusted decision making scheme for collaborative artificial intelligence of things,” IEEE Trans. Ind. Informat., vol. 19, no. 1, pp. 1059–1068, Jan. 2023.
[11] K. Rasheed, A. Qayyum, M. Ghaly, A. Al-Fuqaha, and J. Qadir, “Collaborative federated learning for healthcare: Multi-modal Covid-19 diagnosis at the edge,” IEEE Open J. Comput. Soc., vol. 3, pp. 172–184, 2022.
[12] B. McMahen, E. Moore, D. Ramage, S. Hampson, and B. A. Y. Arcas, “Communication-efficient learning of deep networks from decentralized data,” in Proc. Artif. Intell. Stat., 2017, pp. 1273–1282.
[13] A. Albaseer, M. Abdalhah, A. Al-Faqaha, and A. Erbad, “Balanced energy consumption based on historical participation of resource-constrained devices in federated edge learning,” in Proc. Int. Wireless Commun. Mobile Comput. (IWCMC), 2022, pp. 300–305.
[14] A. Hammoud, H. Otkar, A. Mourad, and Z. Dzijdz, “On demand fog federations for horizontal federated learning in IoV,” IEEE Trans. Netw. Service Manag., vol. 19, no. 3, pp. 3062–3075, Sep. 2022.
[15] A. Hammoud, A. Mourad, H. Otkar, and Z. Dzijdz, “Data-driven federated autonomous driving,” in Proc. Int. Conf. Mobile Web Intel. Inf., 2022, pp. 79–90.
[16] F. Chi, R. Fantacci, and B. Picano, “A matching theory framework for tasks offloading in fog computing for IoT systems,” IEEE Internet Things J., vol. 5, no. 6, pp. 5089–5096, Dec. 2018.
[17] A. E. Roth and M. Sotomayor, “Two-sided matching,” in Handbook of Game Theory With Economic Applications, vol. 1, Amsterdam, The Netherlands: North-Holland, 1992, pp. 485–542.
[18] S. Arisdakessian, O. A. Wahab, A. Mourad, H. Otrok, and M. Guizani, “A survey on IoT intrusion detection: Federated learning, game theory, social psychology, and explainable AI as future directions,” IEEE Internet Things J., vol. 10, no. 5, pp. 4059–4092, Mar. 2023.
[19] F. Chi, R. Fantacci, and B. Picano, “A matching game for tasks offloading in integrated edge-fog computing systems,” Trans. Emerg. Telecommun. Technol., vol. 31, no. 2, 2020, Art. no. e3718.
[20] J. Goetz, K. Malik, D. Bui, S. Moon, H. Liu, and A. Kumar, “Active federated learning,” 2019, arXiv:1909.12641.
[21] Y. J. Cho, J. Wang, and G. Joshi, “Client selection in federated learning: Convergence analysis and power-of-choice selection strategies,” 2020, arXiv:2010.01243.
[22] T. Nishio and R. Yonetani, “Client selection for federated learning with heterogeneous resources in mobile edge,” in Proc. IEEE Int. Conf. Commun. (ICC), 2019, pp. 1–6.
[23] S. AbdulRahman, H. Tout, A. Mourad, and C. Talhi, “FedMCSCs: Multicriteria client selection model for optimal IoT federated learning,” IEEE Internet Things J., vol. 8, no. 6, pp. 4723–4735, Mar. 2021.
[24] T. Huang, W. Lin, W. Wu, L. He, K. Li, and A. Y. Zomaya, “An efficiency-boosting client selection scheme for federated learning with fairness guarantee,” IEEE Trans. Parallel Distrib. Syst., vol. 32, no. 7, pp. 1552–1564, Jul. 2021.
[25] J. Zhao, X. Chang, Y. Feng, C. H. Liu, and N. Liu, “Participant selection for federated learning with heterogeneous data in intelligent transport system,” IEEE Trans. Intell. Transp. Syst., vol. 24, no. 1, pp. 1106–1115, Jan. 2023.
[26] M. Chen, Z. Yang, W. Saad, C. Yin, H. V. Poor, and S. Cui, “A joint learning and communications framework for federated learning over wireless networks,” IEEE Trans. Wireless Commun., vol. 20, no. 1, pp. 269–283, Jan. 2021.
[27] W. Zhang et al., “Optimizing federated learning in distributed industrial IoT: A multi-agent approach,” IEEE J. Sel. Areas Commun., vol. 39, no. 12, pp. 3688–3703, Dec. 2021.
[28] H. Yang et al., “Lead federated neuromorphic learning for wireless edge AI,” Nature, vol. 13 no. 1, 2022.
[29] D. Chen et al., “Matching-theory-based low-latency scheme for multi-task federated learning in MEC networks,” IEEE Internet Things J., vol. 8, no. 14, pp. 11415–11426, Jul. 2021.
and cyber security.

Machine learning, game theory, federated cloud and fog, artificial intelligence,

Teaching Assistant, his research interests encompass areas such as federated

cloud and fog, artificial intelligence, and cyber security.

Osama Wehbi (Graduate Student Member, IEEE) received the master’s degree in business computing from Lebanese University, Beirut, Lebanon, in 2020, and the M.Sc. degree in computer science from Lebanese American University, Beirut, in 2022.

He is currently working as a Research Assistant with the Cyber Security Systems and Applied AI Research Center, Department of CSM, Lebanese American University, and Mohammad Bin Zayed University of Artificial Intelligence, Abu Dhabi, UAE. With previous experience as a Research and Teaching Assistant, his research interests encompass areas such as federated machine learning, game theory, federated cloud and fog, artificial intelligence, and cyber security.

Sarhad Arisdakessian received the master’s degree in computer science from Lebanese American University, Beirut, Lebanon, in 2020. He is currently pursuing the Ph.D. degree in computer science with the Department of Computer Engineering, Polytechnique Montréal, Montreal, QC, Canada.

His main research interests are in the areas of Internet of Things, federated learning, and game theory.

Omar Abdel Wahab (Member, IEEE) received the M.Sc. degree in computer science from Lebanese American University, Beirut, Lebanon, in 2013, and the Ph.D. degree in information and systems engineering from Concordia University, Montreal, QC, Canada, in 2017.

He is an Assistant Professor with the Department of Computer and Software Engineering, Polytechnique Montréal, Montreal. From January 2019 to July 2022, he was an Assistant Professor with the Université du Québec en Outaouais, Gatineau, QC, Canada. In 2017, he did a Postdoctoral Fellowship with the École de Technologie Supérieure, Montreal, where he worked on an industrial research project in collaboration with Rogers and Ericsson. His current research activities are in the areas of cybersecurity, Internet of Things, and artificial intelligence.

Dr. Wahab is a recipient of many prestigious grants from prestigious agencies in Canada, such as the Natural Sciences and Engineering Research Council of Canada and Mitacs. He is a TPC member and a publicity chair of several prestigious conferences and a reviewer of several highly ranked journals.

Hadi Otrok received the Ph.D. degree in ECE from Concordia University, Montreal, QC, Canada, in 2008.

He holds a Full Professor position with the Department of Electrical Engineering and Computer Science, Khalifa University, Abu Dhabi, UAE. Also, he is an Affiliate Associate Professor with the Concordia Institute for Information Systems Engineering, Concordia University, and an Affiliate Associate Professor with the Electrical Department, École de Technologie Supérieure, Montreal. His research interests include the domain of blockchain, reinforcement learning, crowd sensing and sourcing, ad hoc networks, and cloud security.

Dr. Otrok is also an Associate Editor for IEEE TRANSACTIONS ON SERVICES COMPUTING, IEEE TRANSACTIONS ON NETWORK AND SERVICE MANAGEMENT, Ad Hoc Networks (Elsevier), and IEEE NETWORK. He also served from 2015 to 2019 as an Associate Editor at IEEE COMMUNICATIONS LETTERS. He co-chaired several committees at various IEEE conferences.

Safa Otoum (Member, IEEE) received the M.A.Sc. and Ph.D. degrees in computer engineering from the University of Ottawa, Ottawa, ON, Canada, in 2015 and 2019, respectively.

She is an Assistant Professor of Computer Engineering with the College of Technological Innovation, Zayed University, Dubai, UAE. She is a registered Professional Engineer in Ontario. Her research interests include blockchain applications, applications of ML and AI, IoT, and intrusion detection and prevention systems.
Azzam Mourad (Senior Member, IEEE) received the M.Sc. degree in CS from Laval University, Quebec, QC, Canada, in 2003, and the Ph.D. degree in ECE from Concordia University, Montreal, QC, Canada, in 2008.

He is currently a Professor of Computer Science and the Founding Director of the Cyber Security Systems and Applied AI Research Center, Lebanese American University, Beirut, Lebanon, the Visiting Professor of Computer Science with New York University Abu Dhabi, Abu Dhabi, UAE, and the Affiliate Professor with the Software Engineering and IT Department, École de Technologie Superieure, Montreal. His research interests include cyber security, federated machine learning, network and service optimization and management targeting IoT and IoV, cloud/fog/edge computing, and vehicular and mobile networks.

Prof. Mourad has served/serves as an Associate Editor for IEEE TRANSACTIONS ON SERVICES COMPUTING, IEEE TRANSACTIONS 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.

Mohsen Guizani (Fellow, IEEE) received the B.S. (with Distinction), M.S., and Ph.D. degrees in electrical and computer engineering from Syracuse University, Syracuse, NY, USA, in 1985, 1987, and 1990, respectively.

He is currently a Professor of Machine Learning and the Associate Provost with Mohamed Bin Zayed University of Artificial Intelligence, Abu Dhabi, UAE. Previously, he worked in different institutions in the USA. He has authored ten books and more than 800 publications. His research interests include applied machine learning and artificial intelligence, Internet of Things, intelligent autonomous systems, smart city, and cybersecurity.

Dr. Guizani has won several research awards, including the 2015 IEEE Communications Society Best Survey Paper Award, the Best ComSoc Journal Paper Award in 2021 as well five Best Paper Awards from ICC and Globecom Conferences. He is also the recipient of the 2017 IEEE Communications Society Wireless Technical Committee Recognition Award, the 2018 AdHoc Technical Committee Recognition Award, and the 2019 IEEE Communications and Information Security Technical Recognition Award. He was listed as a Clarivate Analytics Highly Cited Researcher in Computer Science in 2019, 2020, and 2021. He served as the Editor-in-Chief of IEEE NETWORK and is currently serving on the editorial boards of many IEEE transactions and magazines. He was the Chair of the IEEE Communications Society Wireless Technical Committee and the Chair of the TAOS Technical Committee. He served as the IEEE Computer Society Distinguished Speaker and is currently the IEEE ComSoc Distinguished Lecturer.