IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received 26 May 2022, accepted 6 August 2022, date of publication 17 August 2022, date of current version 25 August 2022.

Digital Object Identifier 10.1109/ACCESS.2022.3199369

用blockchain发布FL训练任务，任务发布依赖于一个中心服务器，是一种基于bloackchain stack的contract-FL实现

RESEARCH ARTICLE

FedMarket: A Cryptocurrency Driven Marketplace for Mobile Federated Learning Services

ABDULLAH YOUSAFZAI[1,2], LATIF U. KHAN[1], UMER MAJEED[1], OWAIS HAKEEM[2], AND CHOONG SEON HONG[1], (Senior Member, IEEE)

[1]School of Electronics and Information, Kyung Hee University, Yongin-si 17104, Republic of Korea
[2]School of Systems and Technology, University of Management and Technology, Lahore 54770, Pakistan

Corresponding author: Choong Seon Hong (cshong@khu.ac.kr)

This work was supported in part by the Institute of Information and Communications Technology Planning and Evaluation (IITP) Grant funded by the Korea Government (MSIT) (Artificial Intelligence Innovation Hub) under Grant 2021-0-02068 and the MSIT(Ministry of Science and ICT), Korea, under the Grand Information Technology Research Center support program (IITP-2020-2015-0-00742) supervised by the IITP.

ABSTRACT Federated learning (FL) enables the training of a shared collaborative machine learning model while keeping all the confidential training data on distributed devices. The FL state-of-the-art considers a monopolist FL task publisher. However, we present a FL marketplace where multiple FL task publishers and mobile devices co-exist for a set of diverse and varying learning tasks. Mobile devices participating in the training of FL models provides pay-as-you-go (i.e. using blockchain-based cryptocurrencies) FL training services to the FL task publishers. In the proposed framework, multiple FL task publishers may compete with each other and the participating workers (i.e. mobile devices) can choose one FL task publisher over another for participation in the training of a global model. We utilize code offloading for enabling customized FL pipelines in mobile devices and mitigating the model heterogeneity inherent in varying and changing FL tasks published by the task publishers. Experimental results indicate the efficacy of the proposed framework.

INDEX TERMS Federated learning, blockchain, smart contract, marketplace, computational offloading.

I. INTRODUCTION

The pervasive and ubiquitous penetration of smart devices, e.g., smart mobile devices, smart wrist bands, body sensors, home appliances, and autonomous cars into day-to-day life generates a massive volume of data. These smart devices take leverage of machine learning technologies for improving the generalized user experience. On the other hand, corporations providing the application ecosystems for these smart devices are eager to collect and analyze the massive data distributed across the user devices both for making service improvements and making a business value out of the data. However, this data collection and analysis is not trivial in terms of storage and processing at the central servers. Moreover, enforcement of administrative and regulatory laws such as the European General Data Protection Regulation (GDPR), China's Cyber Security Law, and the United States California Consumer Privacy Act, restricts sharing and collecting user data. This scarcity of training data volume can hinder the performance of machine learning tasks carried out in corporations. Furthermore, the commercial competition between data aggregators and application service providers also restricts data sharing. Since, with these restriction in place the number of training samples would not be large enough to generalize the model which affect the performance of the trained model.

Federated learning (FL) is a distributed privacy-preserving machine learning technique that enables mobile devices to collaboratively train a shared global model without the need of uploading private local data to a central server to overcome the privacy concerns caused by centralized machine learning over confidential user data. Figure 1(a) presents the vanilla FL approach [1]. The pinnacle of success behind FL is mainly due to the decentralized training process of a collaborative machine learning model. This decentralization relaxes and distributes the computation and storage cost of the central server along with preserving the privacy of the private confidential data distributed across a range of user devices and application platforms (hereinafter referred to as workers). In FL, the workers individually and independently train their

The associate editor coordinating the review of this manuscript and approving it for publication was Meng-Lin Ku.

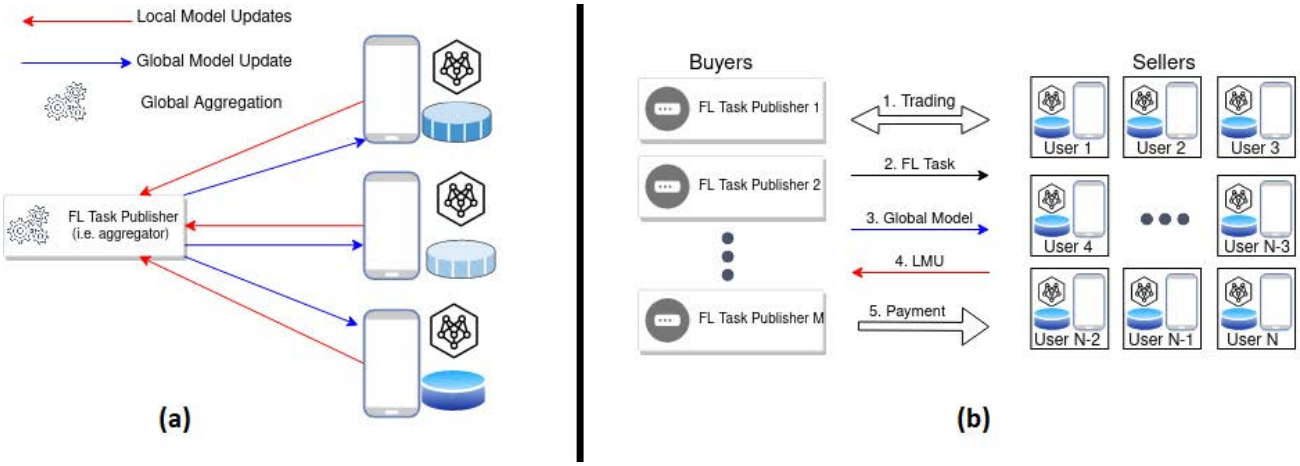

FIGURE 1. (a) Vanilla FL. (b) Illustration of a FL market.

local machine learning models on their confidential data. The local model weights also referred to as local model updates (LMUs) are then delegated to a central aggregation server, while no training data is sent and it remains on the user devices. FL also minimizes the network traffic cost as only LMUs are shared while no data is being shared. After collecting local model weights from a qualified number of worker nodes, the central server aggregates the received weights generally by averaging to form a new global collaborative model, which will be delivered to the workers for the next round of model training. This distributed training iteration repeats until the global model converges to satisfying test accuracy. Through FL, the individual privacy of user data could be effectively preserved as no private data is shared among the user device and the central server. However, due to the lack or in-existence of monetary incentives user devices are reluctant to become workers for FL tasks. Furthermore, the user devices may be competing and reluctant to share their model parameters since the competitors also benefit from a trained global model [2].

Providing monetary incentives for monetizing the on-device FL can increase the expectation of user devices for participating in training the global predictive model. In this context, the notion of a market similar to crowdsourcing platforms such as amazon mechanical turk[1] seems a fitting substitute where the worker monetize their model parameters trained on their private data. In this manuscript, we perceive an FL market (presented in Figure 1(b)) where individuals can monetize execution of the FL process requested by an FL task publisher on their mobile device and provide focused LMUs to the FL task publisher server in compliance with the GDPR to safeguard the user's privacy. FL task publishers that are willing to pay the price to geographically dispersed different user devices for training the FL model on user's local data will publish their task specifications for interested worker mobile devices. In the perceived FL market the FL task publisher establishes on-demand ad-hoc FL networks of distributed mobile devices to complete a learning task. Therefore, unlike the conventional FL environments that consider a monopolist FL task publisher primarily with a single task, the FL market enables the competition between multiple FL task publishers (possibly with multiple numbers of tasks) with each other and enables the participants to choose one FL task publisher over another for participation in the training of a global model. FL task publishers can collect LMUs from a large crowd of users, which makes the market a useful channel for corporations and in the formation of private FL overlays. Individuals providing LMUs can control their FL process by managing how much wall clock time they register on the market including the price of each FL session. On the other hand, the FL task publishers can publish multiple tasks and provide monetary incentives for the LMUs to the users who subscribed to their tasks. Therefore, perceiving a double-sided FL market model where task publishers and individuals jointly operate the market for their rational benefits. Hereinafter, FL task publisher, task publisher, aggregator, FL aggregator, and buyer are used interchangeably. Similarly, worker, client, participant, mobile/user device, and seller are used interchangeably.

The FL marketplace (where mobile users sell LMUs and FL task publishers go to buy LMUs for an FL process) based on guidelines of electronic marketplaces in [3] perform three essential market functions: (a) matching of model sellers and model buyers; (b) facilitating the exchange of information regarding FL process, and the payments associated with market transactions; and (c) providing a legal and regulatory framework, that enables the efficient functioning of the market. Generally, the first two marketplace functions are provided by centralized intermediaries (i.e. business entities referred to as brokers/auctioneers), while the legal and regulatory framework falls under the umbrella of statutory rules and regulations set by governments. However, the centralized intermediaries might be dishonest and result in payment theft, collude, and contract manipulations and pose trust issues to

[1] https://www.mturk.com/

the market participants (i.e. sellers and buyers) for executing the market functions. Furthermore, with the dynamic birth and death of agents in the marketplace, the conventional mode of payments through banking or credit transfers from the network operators (provided the FL task publisher have prior agreements with network operators) is not suitable for the FL marketplace to minimize the churning and free-riding of the worker nodes. To address the problem of trust associated with the payments, we consider blockchain technology [4], due to the consensus, provenance, ownership, immutability, finality, and access control attributes for executing the payment functions of the marketplace. More specifically we utilized the Ethereum blockchain protocol [5], [6] that supports a Turing-complete language and has its native cryptocurrency i.e. Ether. Ethereum virtual machine (EVM) in the ethereum blockchain network executes the code referred to as ''smart contract'' behind DApps and publishes the resulting output to the respective blockchain following a consensus method similar to Bitcoin [4]. The distributed nature of EVM makes smart contracts impossible to censor and eliminates trust issues posed by a centralized administrator in distributed applications such as the marketplace.

The main contributions of this article are summarized as follows:

- We explored an FL environment where multiple FL servers may compete with each other and the participants can choose one FL server over another.
- We utilized blockchain as a means to trade and incentivize the mobile device workers for providing FL services to the FL task publishers.
- We devise a novel method to incentivize reputed workers with high-quality training data that is more useful for the collaborative global model of the task publisher.
- We utilize reverse code offloading to enable customized FL pipelines in mobile devices and to mitigate the model heterogeneity inherent in varying and changing FL tasks published by the task publishers.

II. RELATED WORK

The concept of FL under the definition of distributed artificial intelligence can be traced back to 1980 [7]. Google introduced the privacy-preserving FL concept for context predication in the virtual keyboard for their flagship android operating system [8]. Following this, the authors in [9] discuss the architectural concepts and potential applications of federated machine learning. The majority of the FL state-of-the-art literature can be classified into the performance of learning algorithms in terms of learning time/accuracy, client selection [10], training security [11]–[13], and incentive mechanism design [14]–[16].

Most of the existing FL studies generally assume a monopolist environment consisting of K workers collaborating with a single FL server. There are few articles such as [17], [18] which consider an FL system model comprising of multiple FL task publishers. Authors in [17] considered bandwidth allocation in wireless networks hosting multiple co-existing FL services. On the other hand, authors in [18] proposed a mechanism design oriented FL protocol on a public blockchain network to reward only well-contributing workers. However, the proposal presented in [18] is impractical in terms of the gas cost associated with the smart contract functions since the intensive workload is being delegated in the smart contracts. Furthermore, the overall time latency involved in the FL process will be very high due to the evaluation and voting process of all local models submitted for aggregation. Lastly, their proposed system cannot be generalized since it is bounded to a single model type (i.e. DNN). However, in a practical setting, every task publisher will be interested in training a different machine learning model in different time slots. In this response, we consider an alternate setting to this where multiple FL servers may compete with each other and the participants can choose one FL server over another to train the global model while provisioning monetary incentives to participating clients and handling the model heterogeneity and training data preparation using computational offloading.

III. FedMarket: MARKETPLACE FOR MOBILE FEDERATED LEARNING SERVICES

The system architecture in figure 2 presents the participants of the system, buyers (i.e., FL task publishers), sellers (i.e., mobile devices) interacting via a broker (i.e. a central cloud service), and the ethereum distributed ledger. In FedMarket, mobile devices offer FL services in terms of executing the FL client processes for the FL task publishers with varying QoS attributes. In theory, there could be a large number of FL task publishers providing differential incentives to execute their FL tasks over mobile devices with minimum prices and better QoS. Mobile devices are interested to maximize their profit or revenue by selling their resources at a high price while meeting the requirement of the FL task publishers. To achieve their respective objectives, both the mobile devices and the FL task publishers compete strategically in the FL market. Furthermore, monetary incentives can enhance the motivation of mobile devices to subscribe to the learning task of the publishers [19]. To provision monetary incentive we consider decentralized escrow (similar to one proposed in [20]) based on Ethereum blockchain protocol [5], [6] for executing escrow functions of a marketplace.

A. FL TASK PUBLISHERS: BUYERS

FL task publisher (i.e. buyer) is interested in buying the willingness of mobile devices (i.e. sellers) for the execution of FL worker tasks to train a collaborative global predictive model. For each task to publish the task publishers are required to submit the intermediate level bytecode of the ''`ClientUpdate`'' function of the vanilla FL algorithm [1] to the central broker. Depending upon the target device heterogeneity the publisher can register multiple flavors of the FL worker `ClientUpdate` function for a range of different device types.

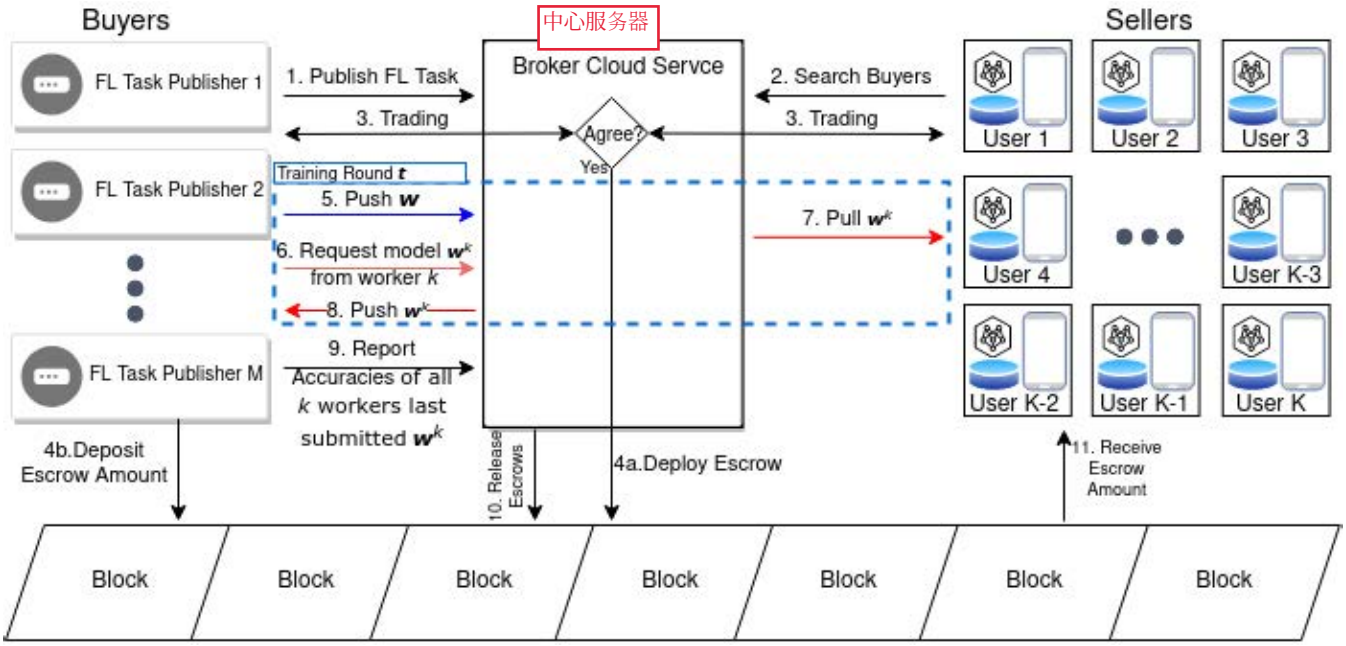

FIGURE 2. FedMarket - a blockchain-based marketplace for multiple FL services.

In vanilla FL participating mobile devices should agree on a particular model w so that the global machine learning model can be trained effectively by aggregating the model weights obtained from mobile applications. However, practically mobile devices/applications, craft their model architecture adaptive to their learning objective and application environments. Mobile applications due to privacy concerns not willing to share the model architecture details. Hence, the model architectures from different mobile applications exhibit heterogeneous model shapes and structures, making it impossible to perform aggregation by conventional FL. Model heterogeneity inherent in mobile environments has attracted considerable research attention due to its practical significance for intelligent mobile applications [21], [22].

On the other hand, in a marketplace where there are multiple FL task publishers, each of the publishers as illustrated in Figure 3 will be using a different type of machine learning algorithms in federated settings. Furthermore, each publisher for each task after a passage of time might be interested in training its predictive model using a different machine learning algorithm in federated settings.

To address the curse of model heterogeneity and changing dynamics of the required model by a task publisher in the FL marketplace we consider the notion of publishing the intermediate level bytecode of the "`ClientUpdate`" which is then dynamically loaded by selling agents using code offloading techniques. Dynamic code loading enables FL task publishers to inject their task-specific FL worker code into the selling agents. This will allow the seller for performing different types of tasks and will need not to

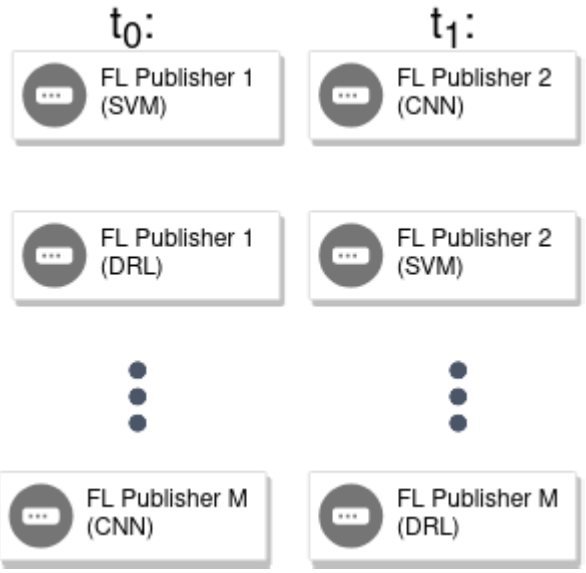

FIGURE 3. Model heterogeneity in FL markets.

maintain and inline itself with all possible buyers and the type of FL mechanism they are implementing. The buyer will implement the `ClientUpdate` procedure which result in transparency in term of heterogeneity of model architecture and machine learning mechanism from the seller's point of view. Furthermore, the task publisher implementation of the FL client also enables the task publisher to implement mechanism that ensure the freshness of the training data, along with capabilities of pre-processing and interactive data acquisition from the seller.

In addition to the `ClientUpdate` function, for each task, the task publisher also needs to submit a test data set and a Model Accuracy Testing (MAT) application to the central broker server. The test data set posted by the FL task publisher is the data set used by the publisher to test the accuracy of the converged global collaborative model. This dataset and the MAT application will be used by the broker for verifying and cross-checking the accuracy reported by the task publisher for fair incentive delivery. Besides publishing the software packages (i.e., `ClientUpdate` function and the MAT application) and the test data set the buyer while posting an FL task in the marketplace needs to publish the following FL task parameters:

- Date and time at which the buyers need LMUs from the mobile devices.
- Details of the FL task preferably using ontological representation such as described in [23].
- Desired price, the price buyer wants to pay a single seller for a complete session of FL client execution.
- Highest acceptable price, the highest price buyer wants to pay a single seller for a complete session of FL client execution.
- Number of workers, the number of sellers required by a buyer for executing the FL client algorithm.
- Top-κ percent workers to which the publisher will release the incentives.

The crude behavior of the FL task publishing agent is presented as Algorithm 1. Once, the buying agent posts a task, and the broker publishes the task after approving through manual and automated profiling of the task for security and vulnerability analysis. The task publisher agent starts to listen to request from the selling agents, it autonomously (i.e. without user intervention) negotiates with selling agents trying to make the best possible deal. Buyer agents are discovered by interested FL sellers through the ontological representation and these buying agents can autonomously negotiate without user intervention and make decisions on its own once it is released into the marketplace. Autonomous negotiation is important for buying agents due to the possibility of involvement of a large number of selling agents interested in registering as a worker for the published task. Once a buying and selling agent has reached an agreement on a price and gotten their respective user's approval. At this stage, the buyer requests the broker for initializing an escrow between the buyer and seller using a blockchain smart contract. Next, the buyer agents deposit the agreed price into the escrow and start consuming the seller FL service by requesting seller model updates via the broker. Once the FL task is finished the buyer will request the final models from all the workers subscribed for the task via the broker to identify the reputed top-κ workers who will receive incentives.

Algorithm 1 TaskPublishingAgent

procedure PublishTask
  Post FL task parameters i.e. ontology, required no. of workers, schedule, & pricing to broker.
  Post a test data set and MAT application to broker.
  Post *ClientUpdate* bytecode to broker.
  if broker approves everything **then**
    Publish the FL task as $\mathcal{T}$
    while Receive worker request via broker until the number of required workers is achieved **do**
      Negotiate for the best deal based upon desired price and highest acceptable price.
      If the worker agrees with the proposed deal request broker to initialize the escrow smart contract.
      Deposit negotiated amount into the smart contract.
      Notify worker via the broker.
    end while
  end if
  if Schedule Approaches **then**
    $FLServer(\mathcal{T})$
  end if
end procedure
procedure FLServer($\mathcal{T}$)
  intialize w_0
  for each round $t = 1, 2 \ldots$ **do**
    $S_t \leftarrow$ (random set of m clients) ▷ m is the client participating rate ranging from (0, 1]
    for each client $k \in S_t$ **in parallel do**
      $w_{t+1}^k \leftarrow RequestUpdate(k, w_t, \mathcal{T})$ **from the broker**
      $w_{t+1} \leftarrow \sum_{k=1}^{K} \frac{n_k}{n} w_{t+1}^k$
      $X_k \leftarrow t$
    end for
  end for
  for each client $k = 1, 2 \ldots K$ **do**
    $w_c^k \leftarrow RequestModel(k, \mathcal{T})$ **from the broker**
    $A_k \leftarrow TestAccuracy(w_c)$
  end for
  $topWorkers \leftarrow FilterWorkers(X, A, k, K)$
  Send *topWorkers* list to broker.
end procedure

1) FINDING TOP-κ REPUTED WORKER

The Top-κ reputed worker (i.e. seller) for a single FL task is identified by the FL task publisher after the FL task is completed to which the workers subscribed. Since all the mobile devices report their final model to the FL task publisher via the broker. The task publisher then evaluate the accuracy of all the models and selects the top-k worker based upon the test accuracy to which the incentives should be provisioned. The task publisher computes the accuracy using the same test data set on which the accuracy of the global model is computed. Furthermore, this is the same data set that is being submitted to the broker while posting the FL task. The broker randomly selects three workers from the chosen top-k workers and recomputes the test accuracy to ensure the truthfulness of the reputation score reported by the task publisher. However, it is imperative that the workers might not be included in

Algorithm 2 Top-κ Worker

procedure FilterWorkers(X, A, κ, K)
  $workers \leftarrow \{1, \ldots, K\}$ ▷ List representing the set of all workers
  $A' \leftarrow$ Create dictionary from A where the key is worker id and value is accuracy of the worker k
  Sort(A')
  while $|workers| > K \times \kappa$ **do** ▷ Iterate until desired threshold of κ-top percentage of worker are filtered.
    $\alpha \leftarrow \arg\min_k X_k^{workers}$ ▷ Using equation to select a cut off worker from the set of workers.
    for each key k in A' **do** ▷ Iterating the dictionary
      if $A_\alpha \geq A_k$ **then** ▷ Filtering rule
        $workers \leftarrow workers - \{kth \text{ worker }\}$ ▷ Remove worker from the filtered list
        remove kth key from A'
        if $|workers| == K \times \kappa$ **then**
          break
        end if
      end if
    end for
  end while
  return $workers$ ▷ Final list of worker who will receive incentives
end procedure

the final training rounds of the global model and their final submitted model accuracy will not be fairly comparable to each other. Since in each round of global model training the publisher randomly chooses the workers and requests their LMUs via the broker. To ensure that monetary incentives are provisioned to only workers with high-quality data suited for the global collaborative model which is of interest to the task publisher we devise the following strategy.

Let us consider $X_k = R$, where $1 < R \leq T$ represents the final global training round a worker k is included for providing LMU to train the global collaborative model, T represents the total number of global training rounds. A_k represent the accuracy of the final local model submitted to the task publisher via the broker by each worker $1 < k \leq K$. A_k is calculated by the task publisher using the test data set used for computing the accuracy of the global predictive model. Now the following equation finds the worker α with minimum final round iteration attendance from the list of K workers.

$$\alpha = \arg\min_k X_k \tag{1}$$

The accuracy of α worker is used as cut out point i.e. for all workers k if the $A_\alpha > A_k$ remove worker k from the list of the top-κ reputed workers. This process is repeated until the cardinality of the top-κ reputed workers' list shrinks to the percentage published by the task publisher. The top-α percentage of workers eligible to receive the monetary incentives are shortlisted using the iterative procedure "FilterWorkers" presented in Algorithm 2.

Algorithm 3 SellingAgent

procedure Find&RequestTask
  Search and select a buyer based on task ontological parameters, schedule, and pricing.
  Send request and negotiate with the selected buyer.
  if buyer accept the request AND broker confirms the escrows **then**
    Dynamically loads the *ClientUpdate*() procedure for the task from the broker.
  end if
  if Schedule Approaches **then**
    Start FLWorker
  end if
end procedure
procedure FLWorker ▷ Run on client k
  while task finishing message not received from broker **do**
    Received a request from the buyer with global model w to submit the LMU.
    $w \leftarrow ClientUpdate(w)$
    Send w to buyer via the broker.
  end while
  Send w to the broker.
end procedure
procedure ClientUpdate(w) ▷ This update procedure is dynamically loaded from the broker, it was originally published by the buyer with broker
  $\mathcal{B} \leftarrow$ (split private data into batches of size B) ▷ Here the buyer can implement any pre-processing or data acquisition logic
  for each local epoch i from 1 to E **do**
    for batch $b \in \mathcal{B}$ **do**
      $w \leftarrow w - \eta \nabla \ell(w; b)$
    end for
  end for
  return w
end procedure

B. MOBILE DEVICES: SELLER

Mobile users are presented by selling agents which sells the user willingness for executing a client FL process published by an FL task publisher to perform local training over the mobile device private data and transmit LMUs to the respective FL task publisher. Algorithm 3 presents the generalized behavior of the FL task agent residing on the client mobile device, forthcoming paragraphs explain the working behavior of the selling agents.

A user can launch multiple selling agents. However, a single selling agent is associated at a time with a single FL task by a task publisher, on completion of the agreement or the FL task, the agent can sell its willingness for executing another FL task. The end-user has total control over the behavior of the selling agent. When the user first initializes the selling agent, they set various parameters from the ML-Schema ontology to guide the matchmaking behavior. The seller's

TABLE 1. Datatype heterogeneity cause by manual curating of training data at each users.

User 1	Label	Feature 1	Feature 2	Feature 3
User 2	Feature 1	Feature 2	Feature 3	Label
User 3	Feature 3	Feature 1	Feature 2	Label

goal is to receive the highest possible price for its resources. Exactly how to achieve this goal is left to the end-user controlling the seller agent. While the users are free in terms of how to achieve their objective. The crude heuristic is: begins searching willing buyers based on the ontological parameters such as model and data set characteristics, date and time at which the selling agent can execute the client FL process, and price ranges if there is any. If the search results do not show up any buyer, the user can change the search parameters accordingly. There may be no buyers (perhaps the search parameter of user interest is not listing any). In this case, the agent fails to achieve its goal.

In an FL marketplace where mobile device workers (especially when workers dynamically load the client FL process) serving an FL task publisher for a common learning task needs structural homogeneity of the data types in all workers. This type of heterogeneity is mainly caused due to the manual curating of the training data available at mobile devices illustrated in Table 1.

To address the structural homogeneity of the data types considering a large number of mobile devices in the FL marketplace the data acquisition for on-device FL is recorded by the onboard electro-mechanical sensors of the smart device. This data acquisition of automatically produced data is naturally supported by the dynamic code loading by the selling agent. Another type of automatically produced data is of system/application log files of the smart device. The user is not required to curate automatically produced data since it will accumulate as the individual uses the service. Trading of FL updates from the automatically produced data is important in the marketplace because of the scale of the users who can subscribe for the FL task relatively at a very low cost. Furthermore, the dynamic code loading of the FL task client in the selling agents allows the task publisher to ensure the freshness of the training data and also can be programmed to label the captured data manually by the end-user operating the selling agents, before the local machine learning execution.

The user can check on its selling agents, see who they have talked to. Once the user finds a buyer of interest, the selling agent will be configured by the user to send a request with the desired price for subscription as a worker to the FL task published by the buyer. If the buyer accepts the proposal and initializes an escrow contract with depositing the desired price into the smart contract marking the selling agent as a subscribed FL worker. At this point, the selling agents autonomously request the broker to migrate/offload the bytecode or binary such as `ClientUpdate` that implements the local model update mechanism. This reverse offloading mechanism is explained in the upcoming subsection. The selling agents should not offload the LMU mechanism directly from the buyers to minimize the security risk as the buyer might inject malicious code in the FL client to compromise the client device security. Instead, the code offloading should be from the broker to the mobile device, where the broker has the responsibility to analyze the LMU mechanism posted by the task publisher for any malicious intents.

1) REVERSE CODE OFFLOADING

Generally, computational offloading is a software-level solution which to some extent augments the capabilities of resource constraint mobile devices by remote execution of the complete or partial compute-intensive application at a remote computing infrastructure. Typically computational offloading is considered for enhancing the energy consumption and execution time for the programs executing on resource constraint mobile devices [24].

The most popular and common technique to leverage the computational power of the cloud in mobile devices is code migration. This technique involves the delegation of code execution by migrating a platform-independent intermediate code (i.e. byte code) to remote cloud servers [25] presented in Figure 4(a).

Many attempts (such as [26]–[28]) have been done in past to enable remote execution using code migration for improving the performance and battery consumption of mobile devices. Most of these code migration-based offloading attempts rely on programmers to specify program partitions using code annotation and skeletons [25].

In our proposed marketplace framework we consider the same concept of code migration for reverse code offloading presented in Figure 4(b), in which the broker offloads the FL intermediate code implementing the LMU (i.e. the client update) mechanism to the mobile device to mitigate the model heterogeneity issues due to the variety of ML tasks posted in the marketplace for collaborative processing. Reverse code offloading brings transparency to the distributed mobile devices concerning the implementation of the model update mechanism and enables the mobile devices to register as a worker for any FL task they are interested/willing to participate. Furthermore, the task publishers curated code offloading to the mobile device enables the task publisher to modify the FL pipeline on the distributed crowd of mobile devices subscribed as workers for the FL task. The client update mechanism published by the task publisher can contain procedures for customized data pre-processing, labeling of automatically produced data, and interactive data acquisition, along with semantics for executing the client update if and only if the freshness of local data meet certain criterion.

C. BROKER

The broker being the central manager of the system is responsible for providing a platform for connecting buyers, and

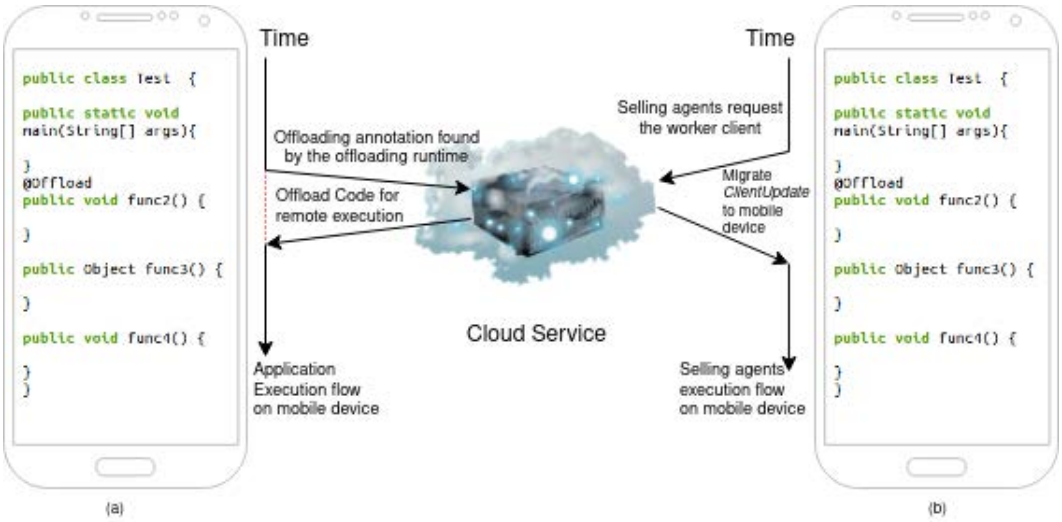

FIGURE 4. (a) Remote execution using code migration (b) Local execution using from remotely migrated code.

sellers. Broker act as a third-party intermediary operating between sellers and buyers. The intermediary business model recognizes that there is value in providing platform services for connecting FL task publishers and participants. This realization of business value motivates the broker to be the custodian for the escrow and avoid scenarios of unwanted troubles, theft, collude, and contract manipulations. The intermediary seeks to act as a trusted agent, providing the opportunity and means for participants (i.e. workers) to monetize and profit from their profiles following the incentive mechanism of decentralized blockchain-based escrows [20]. The general services and behavior exposed by a broker are presented as Algorithm 4 and explained in forthcoming paragraphs. 基于区块链的任务托管

In the proposed networked economy model for FL, the broker collects all buy-bids and sell-bids and provides access for the blockchain smart contract to execute the incentive functions. The broker allows buyers and sellers to find a matching themselves through the process of bidding. However, the matching that results is not necessarily stable. For example, seller A might end up selling its FL service to buyer B, but would have preferred to sell its willingness to execute FL services to buyer C instead had A known about C (perhaps A just didn't see C's listing in her search results).

On the other hand, the broker should provide a software development library (SDL) for the publisher to write standardized FL aggregator, `ClientUpdate` and MAT agent applications. This SDL will provide baseline classes, interfaces, and hooks that need to be implemented in respective software packages to allow the broker to communicate with agents in a homogenous manner. SDL will enable the broker to maintain a uniform application programmable interface (API) services for pushing the global model into the FL worker and can pulling the respective LMU from the FL worker at each iteration of global training triggered by the FL task publishers. Furthermore, the utilization of SDL will enable the broker for a systematic online static analysis and offline security analysis to detect any malicious intents of the buyer on the intermediate code packages registered by the FL task publishers. The publisher can be allowed to connect with sellers if and only if the posted FL client intermediate packages do not have any malicious code. Through this loose coupling in the development of agents, the broker can implement state-of-the-art security features in the SDL for agent communication transparent to the market participant.

Once an FL process is started by the task publisher, the broker act as a relay node between the worker and task publisher as illustrated in figure 5, where as $\sum_{k=1}^{K} \frac{n_k}{n} w_{(t+1)}^{k}$ represents the naive aggregation presented in the seminal FL proposal [1]. The buyer calls the `RequestUpdate` broker service to pull LMUs from a seller k for FL task $\mathcal{T}$. The broker pushes the received global model from buyer to seller and in response receives LMUs from the seller. Broker as a relay node instead of direct communication between the task publisher and the mobile devices in the FL process is incorporated due to the following three reasons.

- The first reason to avoid direct communication between the task publisher and the mobile devices is due to the security/privacy concerns of the worker mobile device especially in an environment where the workers need to load the task publisher curated/developed client update mechanism.
- The second reason of broker as relay node correspond to the network access issues, where the FL task publisher aggregator is not reachable by a public IP address i.e. the publisher is operational behind a NAT, practically allowing any internet-connected device to act as a global model aggregator.
- The last reason is to maintain a record of the final local models of each worker mobile to ensure fair incentive provisioning.

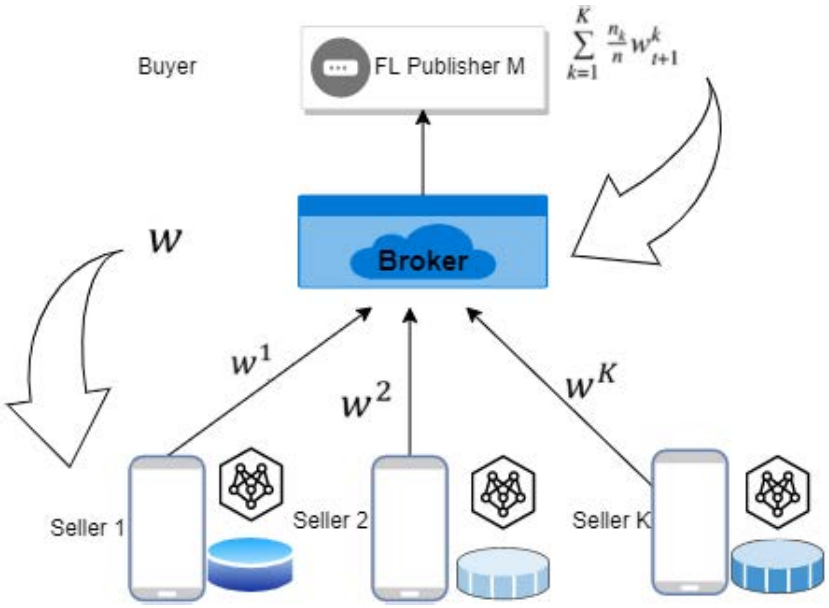

FIGURE 5. FL process between matched seller and buyer using broker as the relay node.

Upon completion of the FL process, the buyer calls `RequestModel` broker service for all the worker to request their respective final local models. This will also serve as a termination signal for the selling agents to stop the FL worker process. Upon delegating the final local models of all the workers to the buyer the broker will wait to receive the list of top-κ sellers from the buyer. Next, the broker will verify the list of top-κ workers and if the broker found the buyer truthful. Then, only escrows to the top-κ sellers are released while keeping some percentage as the service fee. Otherwise, if the buyer is found untruthful, the broker will penalize the buyer by releasing the escrows to all the subscribed workers for the respective task, while keeping the service fee. This penalization of the FL task publisher motivates the task publisher to be truthful while reporting the top-κ workers.

IV. PERFORMANCE EVALUATION AND EXPERIMENTAL RESULTS

In this section, we evaluate the FL process imposed by the proposed framework along with ensuring the provisioning of monetary incentives in terms of cryptocurrencies. However, features such as the optimal matching and price determination between buyers and sellers are the scope of our future research. Furthermore, in the proposed framework the matching is primarily initiated by the user actions behind the mobile device to find a suitable task of interest. All the agents are developed using python and the agents use Web3.py to interact with the ethereum network. The ethereum test-net is implemented using Go Ethereum blockchain implementation. For simplicity of the incentive analysis in the marketplace, the ethereum test-net is configured with zero gas for running a transaction or contract, and all the agents call the smart contract function with zero gas limit. Furthermore, each of the platform agents is configured with an ethereum network address that is used to interact with the underlying ethereum test-net for depositing and receiving cryptocurrency from the escrow smart contract.

TABLE 2. Task parameters.

Task Parameter	Task 1	Task 2
Model	Logistic	LSTM
Dataset	MINIST	Fashion-MNIST
Learning Rate	0.04	0.1
Data Distribution	non-IID	
No. of global iteration	100	1000
No. of local iteration	1	
K (No. of workers)	200	
C (Worker participation rate)	0.5	
Each worker trading rate (in ethers)	10 ether	
Top-κ % of workers	50% of K	
No. of times experiment is repeated	5 times	
Task publisher deposited	2000 ether	
Broker escrow fee	2%	

In our simulation experiments, we introduce two FL task publishers each with a single task and 400 worker mobile devices, whereas we hard-wired 200 worker mobile devices to each task. In practice, the number of tasks an FL task publisher can post depends on the policy of the broker which can allow as many tasks as required. On the other hand, for the seller considering the computational capacity of the mobile devices their participation should be restricted, in our

Algorithm 4 BrokerService

procedure ListenBuyers ▷ Run in cyclic fashion listening buyer requests for task publishing
  Receive FL task details from buyers.
  Receive MAT application and test dataset for the task.
  Receive `ClientUpdate` byte code for the FL task. Security analysis of software packages.
  if software packages pass the security analysis **then**
    Publish FL task $\mathcal{T}$ for clients.
  end if
end procedure
procedure SearchBuyers ▷ Called by the sellers
  Receive search parameters from sellers.
  Return list of all matching active FL tasks to the seller.
end procedure
procedure BidonTask ▷ Called by the sellers
  Receive a bid proposal from seller for a particular task.
  Notify the FL task publisher about the proposal.
  while the task publisher either rejects or accepts **do**
    negotiating messages between seller and buyer
  end while
  if task publisher accepts the bid **then**
    init escrow over blockchain b/w publisher and seller.
    Inform the task publisher about the contract address.
    Wait for publisher escrow deposit to smart contract.
    if task publisher deposit escrow amount **then**
      pair the task publisher and the mobile device.
      inform publisher and seller about the pairing.
    else
      Discard session and remove escrow.
    end if
  end if
end procedure
procedure LoadClient($buyer$, $\mathcal{T}$) ▷ Callable by seller only
  if buyer and seller is paired for the given task **then**
    offload the buyer registered bytecode for task $\mathcal{T}$.
  end if
end procedure
procedure RequestUpdate(k, w, $\mathcal{T}$) ▷ Callable by the buyer only
  if requesting buyer and seller k is paired for the given task $\mathcal{T}$ **then**
    Push current global model w to worker k for task $\mathcal{T}$.
    Pull LMU w^k from worker k for task $\mathcal{T}$.
    Push w^k to the buyer.
  end if
end procedure
procedure RequestModel(k, $\mathcal{T}$) ▷ Callable by the buyer only
  if requesting buyer and seller k is paired for the given task $\mathcal{T}$ **then**
    Send task finishing signal to the seller k.
    Pull and store final local model from seller k for the given task $\mathcal{T}$ and delegate it to the requesting buyer.
  end if
  if local model of all sellers is delegated to the buyer the given task **then**
    Wait until the List of Top-κ workers is received from the buyer.
    Choose Random 3 workers from the top workers' list.
    Confirm the accuracy of the randomly chosen workers using MAT and the data set submitted by the buyer.
    if buyer reported truthfully **then**
      Release escrow amount to the Top-κ workers.
      Release the remaining deposit to the buyer.
    else
      Release escrow amount to all workers.
    end if
  end if
end procedure

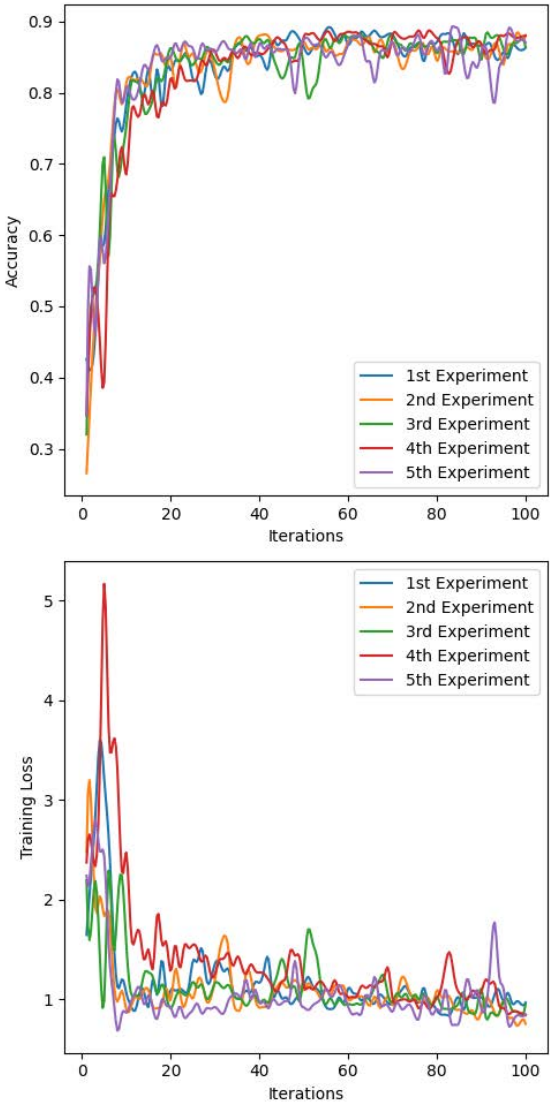

FIGURE 6. Task 1 learning performance per global iteration.

simulation we consider a seller to participate in a single task at a time. The details of the task published by the task publishers with the broker are presented in Table 2. The first task publisher publishes Task 1 to train a collaborative logistic regression global model on the MNIST dataset [29]. While the second task publisher published Task 2 for training a collaborative model using LSTM on Fashion-MNIST dataset [30]. In a crowdsource scenario like the one considered in this manuscript, the data distribution would generally be classified as non-IID. In our experiments, we considered non-IID data distribution in the clients, where the individual clients mostly hold 5 different classes. According to Table 2 each of the task publishers has acquired the services of worker mobile devices on a fixed rate of 10 ethers for the execution of the FL process, hence the task publishers deposited 10 ether to

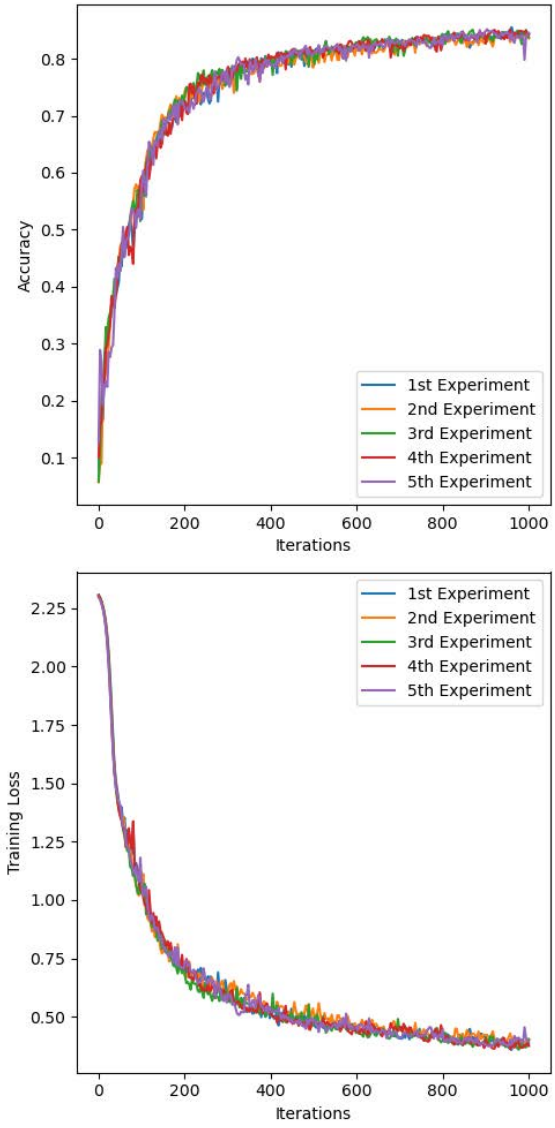

FIGURE 7. Task 2 learning performance per global iteration.

the escrow smart contract of each worker with a total deposit of 2000 ethers for the complete learning task. The broker fee on each escrow is 2%. The task publishers are willing to incentivize up to 50% of the total 200 workers hired, these workers are filtered through the Top-κ procedure. The task publisher in practice can set the number of global iteration to any other stopping condition such as time, or convergence to a particular level of accuracy, herein our simulation is set to 100 global iterations for Task 1 and 1000 global iterations for Task 2, and one local iteration at the client for both the tasks.

A. LEARNING PERFORMANCE

The learning performance of Task 1 is presented in Figure 6, while Figure 7 presents for Task 2. In our simulation setting the mean final global model accuracy of five experiments on Task 1 is 87.18%, while for Task 2 it is 84.074%. Note that, the number of global and local rounds can also play a significant role in improving the accuracy of the global round. However, in our experiments, it is fixed. Compared to the state-of-the-art the reported accuracy can be low since our objective here in this manuscript is not to investigate and improve the learning performance of FL algorithms. Rather the objective is to outline a mechanism for trading mobile device FL services whereby monetizing mobile device workers with high-quality data learning performance. The learning performance plotted in Figure 6 and Figure 7 indicates the acceptable and consistent performance of the FL process in each run of the experiment, wherein each experiment the data distributions hosted in clients varies.

B. FILTERING HIGH-QUALITY WORKERS

High-quality workers' clients in the FL process are identified using the proposed Top-κ filtering procedure presented in section III-A1. Table 3 presents the Top-κ clients filtered in each of the experiments for both the tasks. In the first part of Table 3, each row presents the Top-κ clients by listing the client device ids which are selected in the Top-κ filtering procedure while executing the experiment. In the second part of the Table 3, the Test Accuracy (GM) represents the testing accuracy of the final global model, while the Min Accuracy (LM) represent the minimum accuracy of the final local model submitted by a client (from the Top-κ chosen clients). Similarly, Max Accuracy (LM) represents the maximum accuracy of the final local model submitted by a client from the Top-κ chosen clients. For understanding, in the first experiment of Task 1 86.31% is the test accuracy of the final global model, where 52.88% is the test accuracy of the final local model of client id 172 (i.e. the first client in the Top-κ client column), whereas 82.93% is the test accuracy of the final local model of client id 188 (i.e. the last client in the Top-κ client column). In each of the experiments, the reported accuracy statistics both for the global model and the local clients model are computed using the same test dataset (as explained in the Top-κ filtering procedure). It is noted, that in all of the experiments of Task 1 and Task 2, the best local model submitted by a client outperforms the global model accuracy, except the 5th experiment of Task 2, where the global model accuracy is 84.49% while the accuracy of the best local model submitted by a client is 84.38%. The extra processing steps of testing accuracy of the final local model of all clients proposed in this manuscript enable the task publisher to choose a more suitable model from the local model as a replacement for the global model. Since the accuracy of the final local models and global model is computed using the same dataset, the task publisher can replace the global model with a representative final local model of best-performing clients as a new global model. However, doing so in each global iteration rather than at the end of the FL process will be computationally very expensive for the FL task publisher, and also the resulting global model, in that case, would not be generalized rather than personalized in a case in each global iteration the same client provides the best model. Lastly, the

TABLE 3. High quality workers.

Experiment	Task 1	Task 2
1	[172, 170, 149, 166, 95, 171, 117, 121, 67, 24, 187, 82, 5, 199, 196, 128, 180, 41, 74, 54, 167, 147, 42, 93, 36, 175, 72, 98, 75, 104, 119, 69, 133, 138, 7, 124, 122, 62, 108, 68, 139, 71, 63, 159, 78, 137, 106, 57, 94, 154, 18, 8, 168, 186, 23, 131, 158, 100, 114, 73, 185, 161, 157, 134, 43, 12, 123, 26, 29, 193, 156, 16, 79, 189, 184, 13, 70, 190, 87, 83, 9, 144, 194, 50, 64, 173, 174, 38, 195, 192, 49, 81, 10, 25, 84, 109, 31, 183, 141, 188]	[2, 196, 87, 155, 66, 102, 163, 162, 67, 74, 181, 141, 95, 77, 166, 39, 93, 177, 126, 143, 22, 140, 88, 86, 101, 57, 154, 116, 18, 46, 144, 179, 129, 58, 25, 76, 50, 37, 90, 156, 0, 71, 3, 47, 80, 59, 109, 92, 96, 78, 7, 125, 127, 81, 123, 75, 72, 113, 152, 55, 33, 151, 65, 103, 132, 36, 178, 62, 4, 167, 21, 68, 13, 44, 145, 137, 26, 112, 64, 146, 106, 27, 53, 30, 17, 69, 35, 40, 28, 51, 20, 48, 120, 142, 9, 111, 73, 94, 149, 19]
2	[176, 140, 155, 6, 22, 30, 38, 96, 69, 139, 182, 192, 49, 71, 187, 106, 118, 33, 193, 137, 4, 19, 161, 44, 183, 11, 14, 99, 21, 56, 154, 101, 20, 133, 86, 112, 70, 72, 111, 110, 45, 197, 52, 172, 115, 13, 35, 23, 170, 94, 60, 48, 87, 127, 130, 159, 185, 15, 67, 82, 156, 104, 47, 93, 68, 98, 3, 119, 85, 28, 5, 188, 181, 31, 50, 121, 66, 198, 162, 39, 84, 34, 120, 54, 191, 64, 55, 97, 57, 179, 190, 122, 169, 58, 107, 91, 78, 114, 73, 24]	[78, 25, 106, 74, 90, 54, 174, 63, 53, 149, 42, 143, 59, 124, 10, 73, 187, 51, 23, 1, 62, 98, 152, 96, 136, 82, 160, 182, 112, 87, 92, 115, 199, 44, 109, 146, 86, 39, 156, 7, 110, 13, 0, 75, 173, 129, 158, 134, 4, 100, 71, 36, 116, 168, 151, 177, 57, 45, 77, 148, 43, 118, 183, 147, 121, 155, 28, 97, 159, 15, 65, 135, 166, 99, 181, 6, 18, 85, 145, 127, 132, 12, 11, 138, 89, 93, 179, 190, 103, 58, 69, 117, 154, 14, 48, 17, 184, 142, 144, 191]
3	[87, 117, 116, 3, 175, 197, 8, 78, 120, 169, 64, 160, 90, 24, 133, 143, 96, 52, 166, 76, 121, 186, 126, 110, 138, 49, 89, 132, 5, 107, 61, 108, 194, 167, 146, 55, 112, 34, 62, 0, 113, 163, 105, 88, 13, 22, 82, 94, 21, 30, 77, 115, 106, 125, 71, 191, 29, 95, 198, 74, 174, 79, 57, 168, 27, 40, 150, 60, 176, 11, 86, 100, 97, 145, 75, 137, 155, 92, 67, 93, 99, 102, 83, 84, 151, 119, 181, 10, 19, 59, 129, 199, 165, 164, 192, 193, 85, 111, 189, 188]	[35, 183, 163, 107, 120, 88, 136, 85, 102, 124, 54, 86, 106, 67, 29, 118, 101, 22, 114, 111, 43, 58, 8, 177, 76, 166, 56, 151, 160, 73, 41, 158, 38, 117, 178, 137, 147, 40, 74, 61, 80, 19, 91, 36, 17, 49, 115, 24, 28, 179, 134, 92, 77, 50, 144, 72, 15, 68, 25, 126, 127, 51, 16, 180, 46, 89, 199, 125, 100, 157, 32, 4, 96, 182, 18, 9, 139, 70, 42, 142, 129, 7, 190, 13, 71, 47, 6, 31, 152, 62, 156, 196, 104, 143, 128, 112, 83, 176, 187, 195]
4	[21, 180, 85, 45, 158, 29, 98, 10, 73, 157, 25, 122, 6, 44, 134, 173, 113, 36, 136, 165, 131, 65, 33, 127, 27, 125, 7, 46, 50, 43, 0, 32, 177, 64, 82, 188, 181, 84, 151, 182, 66, 174, 160, 107, 196, 31, 135, 5, 30, 100, 162, 156, 164, 105, 89, 38, 78, 9, 121, 153, 15, 74, 191, 176, 3, 67, 161, 34, 47, 23, 114, 190, 169, 95, 49, 108, 37, 152, 172, 145, 118, 198, 111, 72, 81, 186, 93, 199, 179, 14, 86, 109, 123, 168, 128, 147, 119, 187, 58, 116]	[188, 49, 58, 111, 178, 47, 42, 75, 168, 173, 54, 135, 46, 118, 51, 198, 9, 124, 157, 89, 129, 1, 174, 68, 4, 56, 90, 138, 109, 19, 106, 132, 127, 128, 183, 33, 64, 35, 133, 115, 112, 96, 134, 79, 40, 107, 18, 99, 10, 158, 191, 119, 121, 50, 148, 100, 34, 189, 123, 181, 126, 37, 94, 154, 45, 171, 102, 101, 184, 3, 98, 6, 108, 25, 66, 192, 103, 182, 159, 193, 156, 83, 160, 80, 190, 131, 194, 163, 28, 169, 145, 2, 179, 92, 63, 187, 166, 197, 11, 70]
5	[11, 120, 131, 139, 63, 161, 194, 17, 114, 12, 123, 167, 13, 136, 109, 95, 149, 16, 18, 137, 179, 85, 42, 171, 32, 169, 125, 35, 134, 118, 70, 52, 69, 9, 71, 154, 37, 24, 117, 173, 190, 61, 66, 155, 115, 158, 55, 64, 174, 92, 191, 56, 5, 38, 176, 103, 54, 188, 39, 57, 43, 21, 186, 110, 165, 91, 23, 90, 94, 105, 101, 99, 122, 26, 84, 112, 83, 157, 197, 2, 181, 116, 175, 98, 133, 80, 4, 193, 50, 97, 192, 19, 6, 67, 141, 147, 7, 198, 195, 44]	[29, 130, 170, 146, 37, 63, 166, 67, 163, 109, 157, 165, 110, 53, 128, 155, 127, 54, 47, 182, 115, 10, 70, 64, 107, 180, 111, 33, 31, 181, 72, 50, 42, 117, 84, 9, 164, 23, 27, 19, 100, 136, 76, 120, 46, 15, 122, 88, 175, 174, 167, 35, 58, 116, 0, 41, 85, 108, 134, 74, 62, 22, 30, 14, 75, 71, 144, 93, 49, 87, 45, 160, 121, 83, 161, 98, 137, 92, 65, 132, 21, 96, 61, 60, 143, 142, 2, 169, 147, 34, 48, 112, 176, 158, 77, 151, 177, 57, 69, 113]

Experiment	Test Accuracy (GM)	Min Accuracy (LM)	Max Accuracy (LM)	Test Accuracy (GM)	Min Accuracy (LM)	Max Accuracy (LM)
1	86.31%	52.88%	86.93%	84.31%	74.44%	83.87%
2	88.07%	46.61%	89.1%	83.55%	75.84%	84%
3	86.41%	49.69%	8827%	83.73%	74.91%	83.79%
4	88.03%	60.5%	88.51%	84.29%	75.08%	84.9%
5	87.07%	51.2%	88.64%	84.49%	73.68%	84.38%

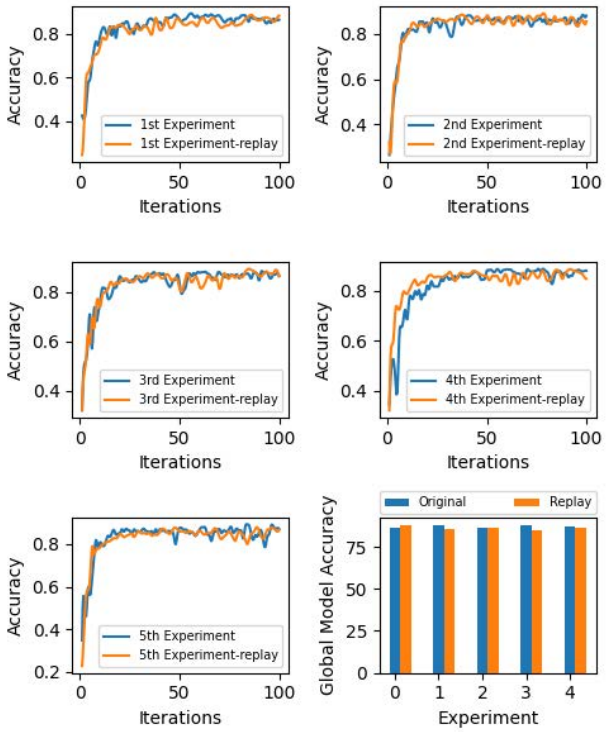

FIGURE 8. FL process replay of Task 1 with the Top-κ clients.

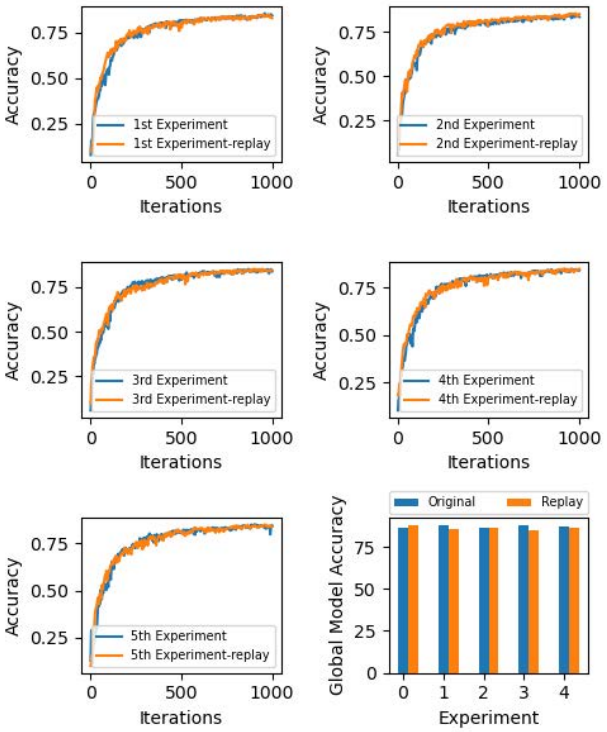

FIGURE 9. FL process replay of Task 2 with the Top-κ clients.

final local model submission can be avoided for communication efficacy, in that case, the task publisher has to keep a record of the last model submitted by each of the clients.

C. REPLAYING FL PROCESS

A replay of the FL process with only Top-κ clients is performed to check whether the final global model achieved by aggregating the local models of only Top-κ participants will result in an acceptable model accuracy or not. This replay process ensures the efficacy of Top-κ filtering procedure. In the replay process, in each global round, all the Top-κ clients have been polled for aggregation i.e. all 100 clients participated in each round no random selection of clients. The random selection of client participation is ignored to ensure a fair comparison with the original experiments. Since in the original experiments in each round almost 100 random clients participate in each global round. Figure 8 presents the replay of each experiment of Task 1, while Figure 9 presents the replay of each experiment of Task 2. The results in Figure 8 and 9 indicate that the fairness of incentives that will be provisioned to the Top-κ participant. In particular, the last sub-plot of Figure 8 and 9 present the comparison between the global test accuracy of the original experiment, and the replay experiment. These subplots indicate for some instances replaying the FL process with only Top-κ participant outperforms the accuracy achieved in the original experiment, this ensure that Top-κ participant have high-quality and useful training data for the said training task. Note that while replaying in each experiment the data hosted by the client in the original experiment and the replay experiment is kept same for fair comparison.

D. MONETARY INCENTIVES

Before the task starts the task publishers of both Task 1 and Task 2 has deposited 10 ether as an escrow for each client totaling 2000 ethers for each of the tasks. On the completion of the task, the broker releases the escrow amount to the clients' ids listed in Table 3. Since the client determined in Table 3 are the high-quality worker in the FL process, these workers are shortlisted by Algorithm 2. To support the intermediary business model of the broker to provide the FL market platform, the broker deducts 2% on each successive release of the escrow to the client devices. The clients' id listed in Table 3 received 9.8 ethers after finishing the task since 2% is deducted by the broker as a fee. The broker fee is a policy matter of the broker and can change from broker to broker, in the case of multi-brokerage systems. On the other hand, 1000 ethers are returned to each of the task publisher's initial deposits of 2000 ethers. As the task publisher is only willing to provide incentives to 50% of the subscribed workers that possess high-quality training data.

V. CONCLUSION

Providing monetary incentives for monetizing the on-device FL can increase the expectation of user devices for participating in training the global predictive model. This paper proposed FedMarket, a brokerage market framework that enables multiple FL task publishers and mobile devices to

co-exist for a set of diverse and varying learning tasks. The proposed framework utilize cryptocurrencies as means to trade FL services and employed a decentralized escrow approach over the blockchain smart contract to provide monetary incentives to FL worker with high-quality training data for participating in an FL task. FedMarket employs code offloading in mobile devices to enable customized FL pipelines in mobile devices and mitigate the model heterogeneity inherent in varying and changing FL tasks published by the task publishers. In the future, we will investigate the proposed framework as a design of double auction-based mechanisms for FL markets where task publishers and worker devices co-exist for a set of diverse and varying learning tasks.

REFERENCES

[1] H. B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. Y. Arcas, "Communication-efficient learning of deep networks from decentralized data," in *Proc. Artif. Intell. Statist.*, 2017, pp. 1273–1282.

[2] W. Y. B. Lim, N. C. Luong, D. T. Hoang, Y. Jiao, Y.-C. Liang, Q. Yang, D. Niyato, and C. Miao, "Federated learning in mobile edge networks: A comprehensive survey," *IEEE Commun. Surveys Tuts.*, vol. 22, no. 3, pp. 2031–2063, 3rd Quart., 2020.

[3] Y. Bakos, "The emerging role of electronic marketplaces on the internet," *Commun. ACM*, vol. 41, no. 8, pp. 35–42, Aug. 1998, doi: 10.1145/280324.280330.

[4] S. Nakamoto, "Bitcoin: A peer-to-peer electronic cash system," Decentralized Bus. Rev., Bitcoin.org, Miami, FL, USA, White Paper, Oct. 2008, Art. no. 21260.

[5] V. Buterin *et al.*, "A next-generation smart contract and decentralized application platform," Ethereum Found., Zug, Switzerland, White Paper, 2014, pp. 1–2, vol. 3, no. 37.

[6] G. Wood, "Ethereum: A secure decentralised generalised transaction ledger," *Ethereum Project Yellow Paper*, vol. 151, pp. 1–32, Apr. 2014.

[7] R. Davis, "Report on the workshop on distributed AI," MIT Artif. Intell. Lab., Cambridge, MA, USA, Working Paper WP-204, 1980.

[8] A. Hard, K. Rao, R. Mathews, S. Ramaswamy, F. Beaufays, S. Augenstein, H. Eichner, C. Kiddon, and D. Ramage, "Federated learning for mobile keyboard prediction," 2018, *arXiv:1811.03604*.

[9] Q. Yang, Y. Liu, T. Chen, and Y. Tong, "Federated machine learning: Concept and applications," *ACM Trans. Intell. Syst. Technol.*, vol. 10, no. 2, pp. 1–19, 2019.

[10] T. Nishio and R. Yonetani, "Client selection for federated learning with heterogeneous resources in mobile edge," in *Proc. IEEE Int. Conf. Commun. (ICC)*, May 2019, pp. 1–7, doi: 10.1109/ICC.2019.8761315.

[11] M. Shayan, C. Fung, C. J. M. Yoon, and I. Beschastnikh, "Biscotti: A ledger for private and secure peer-to-peer machine learning," 2018, *arXiv:1811.09904*.

[12] G. Li, M. Dong, L. T. Yang, K. Ota, J. Wu, and J. Li, "Preserving edge knowledge sharing among IoT services: A blockchain-based approach," *IEEE Trans. Emerg. Topics Comput. Intell.*, vol. 4, no. 5, pp. 653–665, Oct. 2020.

[13] A. P. Kalapaaking, I. Khalil, M. S. Rahman, M. Atiquzzaman, X. Yi, and M. Almashor, "Blockchain-based federated learning with secure aggregation in trusted execution environment for Internet-of-Things," *IEEE Trans. Ind. Informat.*, early access, Apr. 26, 2022, doi: 10.1109/TII.2022.3170348.

[14] J. Kang, Z. Xiong, D. Niyato, S. Xie, and J. Zhang, "Incentive mechanism for reliable federated learning: A joint optimization approach to combining reputation and contract theory," *IEEE Internet Things J.*, vol. 6, no. 6, pp. 10700–10714, Dec. 2019.

[15] A. Yousafzai, P. M. Kumar, and C. S. Hong, "CROWD-CDN: A cryptocurrency incentivized crowdsourced peer-to-peer content delivery framework," *Comput. Commun.*, vol. 179, pp. 260–271, Nov. 2021.

[16] A. Yousafzai, P. M. Kumar, and C. S. Hong, "Blockchain-based incentive management framework for desktop clouds," *Cluster Comput.*, pp. 1–20, Feb. 2022.

[17] J. Xu, H. Wang, and L. Chen, "Bandwidth allocation for multiple federated learning services in wireless edge networks," *IEEE Trans. Wireless Commun.*, vol. 21, no. 4, pp. 2534–2546, 2021.

[18] K. Toyoda, J. Zhao, A. N. S. Zhang, and P. T. Mathiopoulos, "Blockchain-enabled federated learning with mechanism design," *IEEE Access*, vol. 8, pp. 219744–219756, 2020.

[19] J. Edinger, L. M. Edinger-Schons, D. Schäfer, A. Stelmaszczyk, and C. Becker, "Of money and morals—The contingent effect of monetary incentives in peer-to-peer volunteer computing," in *Proc. Annu. Hawaii Int. Conf. Syst. Sci.*, 2019, pp. 1–10.

[20] A. Yousafzai and C. S. Hong, "SmartSON: A smart contract driven incentive management framework for self-organizing networks," 2020, *arXiv:2008.11803*.

[21] J. Pang, Y. Huang, Z. Xie, Q. Han, and Z. Cai, "Realizing the heterogeneity: A self-organized federated learning framework for IoT," *IEEE Internet Things J.*, vol. 8, no. 5, pp. 3088–3098, Mar. 2020.

[22] Z. Chai, H. Fayyaz, Z. Fayyaz, A. Anwar, Y. Zhou, N. Baracaldo, H. Ludwig, and Y. Cheng, "Towards taming the resource and data heterogeneity in federated learning," in *Proc. USENIX Conf. Oper. Mach. Learn. (OpML)*, 2019, pp. 19–21.

[23] G. C. Publio, D. Esteves, A. Ławrynowicz, P. Panov, L. Soldatova, T. Soru, J. Vanschoren, and H. Zafar, "ML-Schema: Exposing the semantics of machine learning with schemas and ontologies," 2018, *arXiv:1807.05351*.

[24] A. Yousafzai, I. Yaqoob, M. Imran, A. Gani, and R. Md Noor, "Process migration-based computational offloading framework for IoT-supported mobile edge/cloud computing," *IEEE Internet Things J.*, vol. 7, no. 5, pp. 4171–4182, May 2020.

[25] A. Yousafzai, A. Gani, R. M. Noor, A. Naveed, R. W. Ahmad, and V. Chang, "Computational offloading mechanism for native and Android runtime based mobile applications," *J. Syst. Softw.*, vol. 121, pp. 28–39, Nov. 2016.

[26] S. Kosta, A. Aucinas, P. Hui, R. Mortier, and X. Zhang, "ThinkAir: Dynamic resource allocation and parallel execution in the cloud for mobile code offloading," in *Proc. IEEE INFOCOM*, Mar. 2012, pp. 945–953.

[27] R. Kemp, N. Palmer, T. Kielmann, and H. Bal, "Cuckoo: A computation offloading framework for smartphones," in *Proc. Int. Conf. Mobile Comput., Appl., Services*, vol. 76, M. Gris and G. Yang, Eds. Berlin, Germany: Springer, 2010, pp. 59–79, doi: 10.1007/978-3-642-29336-8_4.

[28] H. Flores, S. N. Srirama, and R. Buyya, "Computational offloading or data binding? Bridging the cloud infrastructure to the proximity of the mobile user," in *Proc. 2nd IEEE Int. Conf. Mobile Cloud Comput., Services, Eng.*, Apr. 2014, pp. 10–18.

[29] Y. LeCun. (1998). *The MNIST Database of Handwritten Digits*. [Online]. Available: http://yann.lecun.com/exdb/mnist/

[30] H. Xiao, K. Rasul, and R. Vollgraf, "Fashion-MNIST: A novel image dataset for benchmarking machine learning algorithms," 2017, *arXiv:1708.07747*.

ABDULLAH YOUSAFZAI received the B.C.S. degree (Hons.) from Hazara University, Mansehra, Pakistan, in 2009, the M.S. degree in computer science from the COMSATS Institute of Information Technology, Abbottabad, in 2013, and the Ph.D. degree from the University of Malaya, in 2017. He worked as a Postdoctoral Research Fellow under the grant of Brain Korea 21st Century Plus at the Department of Computer Science and Engineering, Kyung Hee University, Republic of Korea. Besides, he worked as an Assistant Professor with the Department of Computer Science and Engineering, HITEC University, Taxila, Pakistan. Before that, he worked as a Brightspark's Research Assistant at C4MCCR, University of Malaysia, and as a Backend Web Developer in Pakistan. He is currently a Postdoctoral Research Fellow under the prestigious National Research Foundation of Korea at the Department of Computer Science and Engineering, Kyung Hee University. His work mainly focuses on distributed computing environments comprising cloud computing systems, edge computing, mobile cloud computing, blockchain systems, and the Internet of Things.

LATIF U. KHAN received the M.S. degree (Hons.) in electrical engineering from the University of Engineering and Technology (UET), Peshawar, Pakistan, in 2017, and the Ph.D. degree in computer science and engineering from Kyung Hee University (KHU), South Korea. He is currently working as a Postdoctoral Research Fellow at KHU. He is also working as a leading Researcher with the Intelligent Networking Laboratory under a project jointly funded by the prestigious Brain Korea 21st Century Plus and Ministry of Science and ICT, South Korea. Prior to joining the KHU, he has worked as a Faculty Member and a Research Associate at UET. He has published his works in highly reputable conferences and journals. His research interests include analytical techniques of optimization and game theory to edge computing, end-to-end network slicing, and federated learning for wireless networks.

UMER MAJEED received the B.S. degree in electrical engineering from the National University of Science and Technology (NUST), Pakistan, in 2015. He is currently pursuing the Ph.D. degree in computer engineering with Kyung Hee University (KHU), South Korea. He is also working as a Researcher with the Intelligent Networking Laboratory under a project jointly funded by the prestigious Brain Korea 21st Century Plus and Ministry of Science and ICT, South Korea. He received the Best Paper Award in 35th IEEE International Conference on Information Networking (ICOIN), Jeju Island, South Korea, in 2021. His research interests include blockchain, mobile edge computing, the Internet of Things, machine learning, and wireless networks.

OWAIS HAKEEM received the bachelor's degree in computer science from the University of Peshawar, in 2012, and the master's degree in computer science from COMSATS University Islamabad (CUI), in 2017, where he is currently pursuing the Ph.D. degree. He is currently working as a Lecturer with the Department of Computer Science, School of Systems and Technology, University of Management and Technology, Lahore, Pakistan. He has published various research articles in prestigious international conferences and journals. His research interests include intelligent autonomous vehicles, edge computing, the Internet of Things, and wireless-based cyber-physical systems.

CHOONG SEON HONG (Senior Member, IEEE) is currently working as a Professor with the Department of Computer Science and Engineering, Kyung Hee University. His research interests include future internet, *ad-hoc* networks, network management, and network security. He is a member of ACM, IEICE, IPSJ, KIISE, KICS, KIPS, and OSIA. He has served as the General Chair, a TPC Chair/a member, or an Organizing Committee Member for international conferences, such as NOMS, IM, APNOMS, E2EMON, CCNC, ADSN, ICPP, DIM, WISA, BcN, TINA, SAINT, and ICOIN. In addition, he is currently an Associate Editor of the IEEE TRANSACTIONS ON NETWORK AND SERVICE MANAGEMENT, the *International Journal of Network Management*, and the *Journal of Communications and Networks* and an Associate Technical Editor of *IEEE Communications Magazine*.