Accelerating Asynchronous Federated Learning Convergence via Opportunistic Mobile Relaying

Jieming Bian, *Graduate Student Member, IEEE*, and Jie Xu, *Senior Member, IEEE*

***Abstract*—This paper presents a study on asynchronous Federated Learning (FL) in a mobile network setting. The majority of FL algorithms assume that communication between clients and the server is always available, however, this is not the case in many real-world systems. To address this issue, the paper explores the impact of mobility on the convergence performance of asynchronous FL. By exploiting mobility, the study shows that clients can indirectly communicate with the server through another client serving as a relay, creating additional communication opportunities. This enables clients to upload local model updates sooner or receive fresher global models. We propose a new FL algorithm, called FedMobile, that incorporates opportunistic relaying and addresses key questions such as when and how to relay. We prove that FedMobile achieves a convergence rate $O(\frac{1}{\sqrt{NT}})$, where N is the number of clients and T is the number of communication slots, and show that the optimal design involves an interesting trade-off on the best timing of relaying. The paper also presents an extension that considers data manipulation before relaying to reduce the cost and enhance privacy. Experiment results on a synthetic dataset and two real-world datasets verify our theoretical findings.**

***Index Terms*—Convergence analysis, federated learning, mobile relaying.**

I. Introduction

FEDERATED learning (FL) is a distributed machine learning approach in which numerous clients possessing decentralized data cooperate to develope a shared model, under the guidance of a central server [1]. The majority of FL algorithms [1], [2], [3], [4], [5], [6], [7], [8], [9], [10] explore the synchronous communication setting, where clients can periodically and concurrently synchronize and interact with the server. In this context, communication frequency is a crucial aspect of the FL algorithm design, as it assumes that the communication channel between clients and the server is consistently and universally accessible. However, this assumption is often unrealistic in real-world systems, where clients may only have intermittent opportunities to communicate with the server and display varying communication patterns. For instance, mobile clients (such as mobile devices or vehicles) can only connect with the server (e.g., base stations, sensing hubs, roadside units) when they come within the server's communication range [11]. Consequently, in such mobile systems, FL must be conducted asynchronously, conforming to each client's unique communication pattern with the server.

The asynchronous nature of these mobile systems renders the methods developed for synchronous FL ineffective. As a result, asynchronous FL methods are necessary to tackle such systems. However, the body of literature on asynchronous FL is notably smaller compared to its synchronous counterpart. Although some insights have been gained (for example, [12] demonstrates that the convergence rate of asynchronous FL can match that of synchronous FL given the same communication interval), the performance of asynchronous FL is significantly constrained by the arbitrary communication patterns of individual clients, which are not an algorithm parameter. With only sporadic client-server interactions, asynchronous FL may converge slowly or even fail to achieve convergence.

To tackle the problem arising from sporadic client-server interactions, in this paper, we focus on exploiting the communication opportunities among clients within the mobile system, an aspect largely overlooked in previous works. As devices in the mobile system continually move over time, numerous client-client encounters are created. These meetings allow a client to indirectly communicate with the server by using another client as a relay, thereby generating additional communication opportunities (both uploading and downloading) with the server. Specifically, a client's local model updates can be uploaded to the server sooner if the relay client connects with the server before the sending client does. Similarly, the client can receive a more recent global model from a relay client that has recently connected with the server.

However, the added benefits provided by the relay for either uploading or downloading introduce new challenges. Firstly, determining when to upload (download) via relaying and selecting the appropriate relay is crucial. If a client uploads to a relay too early, only minimal new information can be transmitted to the server, as the client has completed just a few local steps. Conversely, if the upload to the relay is delayed, the benefit of utilizing the relay may be significantly diminished. Secondly, determining how to relay the local model updates in a manner that prevents duplication of updates received by the server and maintains storage efficiency is also essential. To address these challenges, we propose a novel asynchronous FL algorithm with opportunistic relaying, called FedMobile. The main contributions of our work are as follows:

- Our work focuses on harnessing the benefits of utilizing relaying resulting from client-client meetings, an aspect not previously explored in mobile FL. We propose a new

Manuscript received 7 October 2023; revised 1 January 2024 and 22 February 2024; accepted 17 March 2024. Date of publication 1 April 2024; date of current version 16 July 2024. This work was supported by NSF under Grant 2033681, Grant 2006630, Grant 2044991, and Grant 2319780. The review of this article was coordinated by Prof. Haejoon Jung. *(Corresponding author: Jie Xu.)*

The authors are with the Department of Electrical and Computer Engineering, University of Miami, Coral Gables, FL 33146 USA (e-mail: jxb1974@miami.edu; jiexu@miami.edu).

Digital Object Identifier 10.1109/TVT.2024.3384061

asynchronous FL algorithm, FedMobile, which accelerates convergence while maintaining storage efficiency and preventing duplicate update transmissions.
- We offer a theoretical convergence analysis of FedMobile and unveil an intriguing trade-off regarding the optimal timing of relaying, which can, in theory, enhance the convergence speed.
- We introduce an extension of FedMobile where clients and their relays transmit data with quantization/compression, reducing the communication costs caused by additional relaying.
- We perform extensive experiments on a synthetic dataset and two real-world datasets to corroborate our theoretical findings. The results confirm that our proposed method outperforms the state-of-the-art asynchronous FL method by significantly reducing time consumption.

The rest of this paper is structured as follows. Section II reviews related works. In Section III, we present the system model and briefly revisit the state-of-the-art asynchronous FL method [12]. FedMobile and its extension are detailed in Section IV, while Section V provides the theoretical convergence analysis. Section VI presents an extension for a more general case. Experimental results are reported in Section VII. Finally, Section VIII concludes the paper. All technical proofs can be found at the Arxiv (i.e. https://arxiv.org/abs/2206.04742).

II. Related Work

A. Federated Learning

FedAvg [1] uses local stochastic gradient descent (SGD) to reduce the number of communications between the server and clients. Convergence has been analyzed for FedAvg [2], [3], [4], [5], [13] and its variants (e.g., momentum variants [14], [15] and variance-reducing variants [16], [17]) in both iid and non-iid data settings. However, the majority of works on FL study the synchronous setting and treat the communication frequency as a tunable parameter of the algorithm.

In recent years, asynchronous distributed optimization and learning have been extensively studied. Works such as [18], [19], [20], [21], [22] focus on single SGD steps by distributed nodes with iid data distributions, which do not represent typical FL settings. The literature on asynchronous FL is smaller and has varied emphases. For instance, some existing works [23], [24], [25] still assume universal communication at all times, with asynchronicity resulting from an algorithmic decision rather than a constraint. Other works [26], [27], [28], [29] employ asynchronous model aggregation to tackle the "straggler" issue in synchronous FL. The asynchronous setting most similar to ours is examined in [12], [30], which consider arbitrary communication patterns. However, these studies only focus on the interaction between the server and the clients, overlooking the interactions among clients themselves. In this work, we propose FedMobile, an algorithm that leverages the benefits of relaying through client interactions, and demonstrate its superior convergence performance both theoretically and empirically compared to state-of-the-art asynchronous method [12].

TABLE I
Key Notations

Symbol	Semantics
t	time slot which is measured by one local update
N	the number of mobile clients
$\mathcal{S}^t$	the set of clients who meet the server
f_i	non-convex loss function for client i
F_i	estimated loss function based on mini-batch data sample
η	the local learning rate
g	the stochastic gradient
τ_i^{last}	the last time when client i meets the server
τ_i^{next}	the next time when client i will meet the server
Δ	the upper bound of times interval
θ, Θ	the upload search interval parameters
ω, Ω	the download search interval parameters
m_i^t	the CLU stored by client i at time t
$\tilde{m}_i^t$	the manipulated CLU
n_i^t	the CLU only contains client i's own updates
$Q(\cdot)$	the quantization operator
μ_i^t	the perturbation noise
ϵ_i^t	the difference between $\tilde{m}_i^t$ and m_i^t

Our work is remotely related to decentralized FL [31], [32], [33] (and some hybrid FL works [34]) where clients can also communicate among themselves during the training process, typically by using a type of gossip algorithm to exchange local model information. However, these works assume a fixed topology of clients and the communication among the clients is still synchronous and periodic.

B. Opportunistic Relaying

Opportunistic relaying is a wireless communication technique where intermediate nodes in a wireless network temporarily act as relay nodes to forward data packets to their intended destination. The fundamental concept behind this technique is to harness the unused resources in the network, such as energy, idle time, and bandwidth, to enhance the network's overall performance. Although the idea of opportunistic relaying has been well extended and studied in many fields (e.g. wireless networks [35], UAV [36]), to our best knowledge, there is no prior work on opportunistic relaying in FL. Additionally, previous studies concentrate on transmitting the unchanged value in situations where no learning process is performed. Our work offers the first principled investigation of the interplay between opportunistic relaying via client-client communication and the convergence of FL, and how to design the optimal relaying strategy to maximize the convergence speed in the learning process.

III. Model and Preliminaries

To clarify our problem, we begin by presenting essential notations, as outlined in Table I. We consider a mobile FL system with one server and N mobile clients. The mobile clients work together to optimize the model parameters $x \in \mathbb{R}^d$ by minimizing the global objective function $f(x)$:

$$\min_x f(x) = \frac{1}{N}\sum_{i=1}^{N} f_i(x) = \frac{1}{N}\sum_{i=1}^{N} \mathbb{E}_{\zeta_i}[F_i(x,\zeta_i)] \quad (1)$$

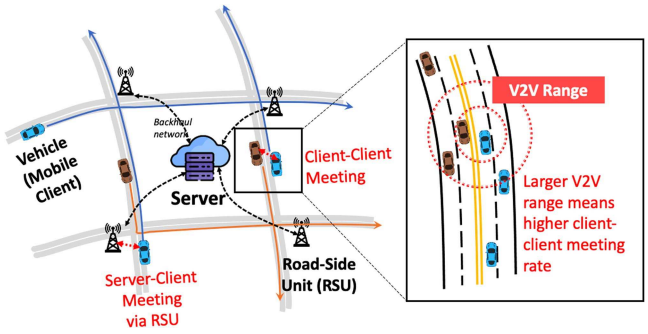

Fig. 1. Illustration of Mobile FL system in VANET.

where $f_i : \mathbb{R}^d \to \mathbb{R}$ is a non-convex loss function for client i and F_i is the estimated loss function based on a mini-batch data sample ζ_i drawn from client i's own dataset.

Mobility Model: We consider a discrete time system where time is divided into slots of equal length. Clients move independently in a network and make contact with the server only at certain time slots. At each time t and for each client i, let $\tau_i^{\text{last}}(t)$ be the last time when client i meets the server by t (including t), and $\tau_i^{\text{next}}(t)$ be the next time when client i will meet the server (excluding t). We assume that at any given time t, client i can **estimate** $\tau_i^{\text{next}}(t)$. For the sake of proving theoretical convergence, we will assume in the theoretical analysis section that at any t, client i knows its exact $\tau_i^{\text{next}}(t)$. However, in the experiment section, we will compare performances under both the scenarios where $\tau_i^{\text{next}}(t)$ is exactly known by the client and where $\tau_i^{\text{next}}(t)$ is estimated by the client. The communication pattern is arbitrary and different for different clients but we assume that the time interval between any two consecutive server meetings for any client is bounded by Δ, i.e., $\tau_i^{\text{next}}(t) - \tau_i^{\text{last}}(t) \leq \Delta, \forall t, \forall i$.

Clients can also meet among themselves due to mobility. When two clients meet, they can communicate with each other via, e.g., device-to-device (D2D) communication protocols [37]. For ease of analysis, we assume that at each time t, a client can meet at most one other client (the extension to multiple clients is straightforward). Let $\rho \in [0, 1]$ be the probability of a client meeting another client at a time slot. When $\rho = 0$, clients do not meet with each other, thus degenerating to the conventional case. We assume that the client-server meetings remain unchanged regardless of the value of ρ. In the following, we provide a realistic example of a Vehicular Ad Hoc Network (VANET).

Fig. 1 is an illustration of Vehicular Ad Hoc Network (VANET). Consider a road network where many vehicles (as mobile clients) are moving and performing FL (e.g., for traffic prediction) using on-board computing power. Some roadside units (RSUs) are deployed in the road network which the vehicles can connect to via wireless. All pertinent RSUs and the server are interconnected through high-speed communication technologies such as fiber optic networks [38]. This ensures that the transmission latency between the server and the RSUs is exceedingly low and can, therefore, be disregarded. In a typical deployment, the RSUs are only deployed in certain parts of the road network (e.g., some road intersections) and do not cover the entire road network [11]. Therefore, the vehicles (clients) can communicate with the server only when they enter the communication range of a RSU. Because vehicles have different speeds and routes, their meeting times with servers (via the RSUs) are naturally asynchronous.

In the VANET use case, vehicles (clients) often have a predetermined route and move at a roughly constant speed. Thus, it is easy to calculate the next server meeting given the next RSU location and the vehicle trajectory. If a vehicle's moving speed is fixed, then the client-server meetings do not change (because distances between RSUs cannot be changed). A client-client meeting occurs if two clients enter the Vehicle-to-Vehicle (V2V) communication (e.g., Wi-Fi Direct or LTE Direct) range of each other. Thus, by increasing the communication range (e.g., by using a larger transmission power), more client-client meetings can occur while the client-server meetings stay the same. Thus, ρ is a parameter that models the client-client meeting rates in this case. A smaller client-client communication range results in a smaller ρ while a larger client-client communication range results in a larger ρ.

State-of-the-art Asynchronous FL method: Before we introduce our proposed method, let's briefly revisit the state-of-the-art asynchronous FL method, ASYNC [12], which operates under arbitrary communication patterns.

Local model update: For any client i, when it meets the server at t, it downloads the current global model x^t. The client then uses $x_i^t = x^t$ as the initial model to train a new local model using its local dataset until it meets the server again. This is done by using a mini-batch SGD method:

$$x_i^{s+1} = x_i^s - \eta g_i^s, \forall s = t, \ldots, \tau_i^{\text{next}}(t) - 1 \tag{2}$$

where $g_i^s = \nabla F_i(x_i^s, \zeta_i^s)$ is the stochastic gradient on a randomly drawn mini-batch ζ_i^s and η is the learning rate. Here, we assume that a client performs one step SGD at each time slot to keep the notations simple. Let $m_i^t \in \mathbb{R}^d$ be the cumulative local updates (CLU) of client i at time t since its last meeting with the server, which is updated recursively as follows:

$$\begin{aligned} m_i^t &= \eta g_i^{t-1}, \text{if } t = \tau_i^{\text{last}}(t) + 1; \\ m_i^t &= m_i^{t-1} + \eta g_i^{t-1}, \forall t = \tau_i^{\text{last}}(t) + 2, \ldots, \tau_i^{\text{next}}(t) \end{aligned} \tag{3}$$

Global model update: At any t, let $\mathcal{S}^t$ denote the set of clients who meet the server ($\mathcal{S}^t$ may be empty). These clients upload their CLUs to the server which then updates the global model as

$$x^t = x^{t-1} - \frac{1}{N} \sum_{i \in \mathcal{S}^t} m_i^t \tag{4}$$

The updated global model x^t is then downloaded to each client i in $\mathcal{S}^t$, and the client starts its local training with a new initial model $x_i^t = x^t$.

IV. FedMobile

In the vanilla asynchronous FL described in the Section III, a client can upload its CLU and download the global model only when it meets the server. When the meeting intervals are long, however, this information cannot be exchanged in a timely manner, thus hindering the global training process. To overcome this issue, we take advantage of *mobility* and propose FedMobile

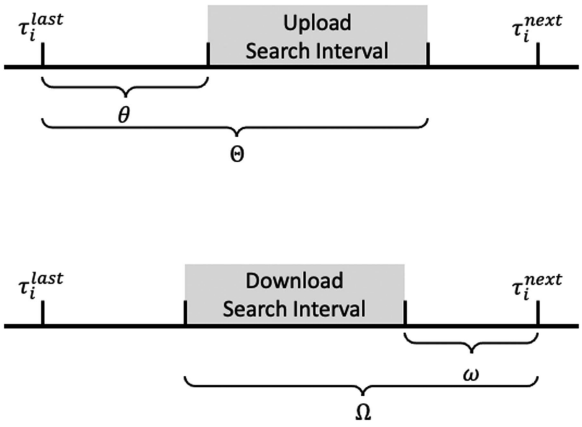

Fig. 2. Upload/download search intervals.

that creates indirect client-server communication opportunities, thereby improving the FL convergence speed. In a nutshell, a client i can use another client j as a relay to upload its CLU to the server (or download the global model from the server) if client j's next (or last) server meeting is earlier (or later) than client i's. Thus, the CLU can be passed to the server sooner (or the client can train based on a fresher global model). We call such client j an upload (or download) *relay* for client i.

For now, we assume that a client can only use *at most one* upload relay and *at most one* download relay between any two consecutive server meetings, considering the extra cost incurred due to the client-client communication. However, FedMobile can be easily extended (we discuss it in The General Case section). Next, we describe separately how FedMobile handles uploading and downloading, which are essentially the dual cases to each other.

A. Uploading CLU via Relaying

Upload Timing and Relay Selection: Since a client can use the upload relay at most once between its two consecutive server meetings, the timing of uploading via relaying is crucial. If client i uploads too early (i.e., close to τ_i^{last}), then the CLU has little new information since the client has run just a few mini-batch SGD steps. If client i uploads too late (i.e., close to τ_i^{next}), then the CLU will be uploaded to the server late even if a relay is used. To make this balance, FedMobile introduces a notion called *upload search interval* defined by two parameters θ and Θ, where $0 \leq \theta \leq \Theta \leq \Delta$ (See Fig. 2). Client i will upload its CLU via a relay only during the interval $[\tau_i^{\text{last}} + \theta, \tau_i^{\text{last}} + \Theta]$. In addition, not every other client that client i meets during the search interval is qualified. We here define *semi-qualified* upload relay and *qualified* upload relay as follows:

Definition 1: Client j is a **semi-qualified** upload relay if $\tau_j^{\text{next}} \leq \tau_i^{\text{last}} + \Theta$, i.e., client j is able to relay client i's CLU to the server before the end of the search interval. Further, Client j is a **qualified** upload relay if in addition $\tau_j^{\text{next}} < \tau_i^{\text{next}}$, i.e., client j can indeed deliver the CLU earlier than client i's own server meeting.[1]

[1]It is possible that $\tau_i^{\text{last}} + \Theta \geq \tau_i^{\text{next}}$ due to the fixed value of Θ, so a semi-qualified upload relay is not necessarily qualified.

FedMobile picks the *first* qualified upload relay during the search interval. Note that it is possible that no qualified upload relay is met, in which case no uploading via relaying is performed. The determination of setting parameters θ and Θ will be addressed in Section V, following the theoretical analysis of their impact on convergence.

Upload Relay Mechanism: FedMobile implements a streamlined and storage-efficient mechanism to guarantee the delivery of a specific piece of information to the server precisely once, all while avoiding the need to retain client ID information. When a CLU exchange event occurs at time t involving a sender client i and a relay client j, the following steps are employed:

RESET (by sender): After sending its current CLU m_i^t, sender client i resets its CLU to $m_i^t := 0$

COMBINE (by relay): After receiving m_i^t from client i, client j updates it stored CLU m_j^t by incorporating m_i^t, i.e., $m_j^t := m_j^t + m_i^t$.

In this way, FedMobile essentially offloads the uploading task of m_i^t from client i to client j, who, by our design, has a sooner server meeting time than client i.

Remark: Upon transmission of the local update from the sender (client i) to relay (client j), the local training processes of both the sender and the relay remain unaffected. Specifically, each continues to train locally based on their local models. This process persists until each client reaches its next scheduled server communication, denoted as $\tau_i^{\text{next}}(t)$ for client i and $\tau_j^{\text{next}}(t)$ for client j.

Remark: In FedMobile, a client has the capability to function as both a sender and a relay between consecutive server meeting times. When a client i transmits message m_i^t to client j, it is possible that client i has already received CLUs (Client Local Updates) from other clients. Consequently, the message m_i^t may contain CLUs associated with those other clients. In our relay mechanism, the relay client is not required to store the ID information of the sender client. Even if the relay client were to gain knowledge of the sender client's ID information, it would still face substantial difficulty in determining which clients' CLUs are included in the CLU sent by the sender client. Due to the presence of CLUs mixed with unknown clients' local updates, the relay client faces significant difficulty in discerning the personal privacy of the sender client.

Remark: For higher communication efficiency and/or better privacy protection, the clients may send an altered CLU to the relay for uploading. We discuss this extension and provide its convergence analysis in the following subsection.

Upload Relay Protocol: In the upload relay process, the sender client i identifies a potential upload relay client j within the communication range. Client i initiates the process by sending a 'Beacon' message to client j to inquire about its willingness to serve as an upload relay. If client j agrees, it responds with a 'Willingness' message, including its estimated next meeting time with the server (τ_j^{next}). Client i then decides whether to select client j as its upload relay based on τ_j^{next} and its own next meeting time (τ_i^{next}). If client j is not chosen, client i sends an 'Acknowledgement' message to conclude the interaction. Otherwise, client i transmits the CLU m_i^t to client j, who

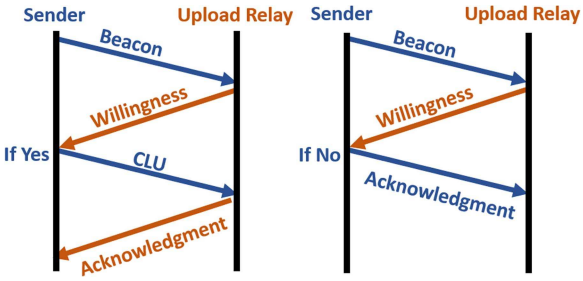

Fig. 3. Upload relay protocol.

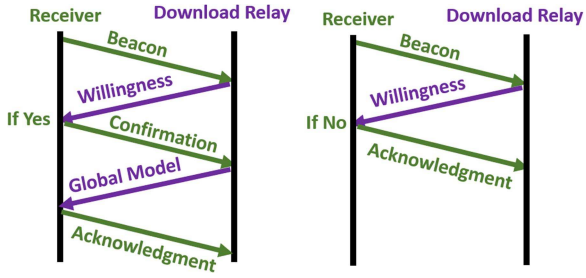

Fig. 4. Download relay protocol.

acknowledges receipt and concludes the process. This protocol is depicted in Fig. 3.

B. Downloading Global Model Via Relaying

Download Timing and Relay Selection: Similar to the uploading CLU case, when downloading a global model via relaying also involves a trade-off. FedMobile introduces a *download search interval* to make this balance, which is defined by two parameters ω and Ω, where $0 \leq \omega \leq \Omega \leq \Delta$ (see Fig. 2). Given client i's next server meeting time τ_i^{next}, client i will only download a new global model via relaying during the search interval $[\tau_i^{\text{next}} - \Omega, \tau_i^{\text{next}} - \omega]$. Similarly, We here define *semi-qualified* download relay and *qualified* download relay as follows:

Definition 2: Client j is a **semi-qualified** download relay if $\tau_j^{\text{last}} \geq \tau_i^{\text{next}} - \Omega$, i.e., client j's global model is less than Ω time slots older than client i's next global model directly from the server. Further client j is a **qualified** download relay if in addition $\tau_j^{\text{last}} > \tau_i^{\text{last}}$, i.e., client j's global model is indeed fresher than client i's current global model that it received directly from the server.

FedMobile picks the *first* qualified download relay during the search interval. Again, it is possible that no qualified download relay is met, in which case no downloading via relaying occurs. The determination of setting parameters ω and Ω will be addressed in Section V, following the theoretical analysis of their impact on convergence.

Download Relay Mechanism: To be able to relay a global model to other clients, every client keeps a copy of the most recent global model that it received (from either the server or another client). We denote this copy for client j by $x^{\psi_j(t)}$, where $\psi_j(t)$ is the time version (or timestamp) of the global model. Upon a global model exchange at time t between a receiver client i and a relay client j:

REPLACE (by receiver): After receiving $x^{\psi_j(t)}$ from client j, client i replaces its local model with $x^{\psi_i(t)}$, i.e., $x_i^t := x^{\psi_j(t)}$, and resumes the local training steps.

Remark: Client i also replaces its global model copy with $x^{\psi_j(t)}$ since it is a fresher version by our design. Thus, $\psi_i(t)$ is updated to $\psi_j(t)$.

Remark: An alternative to the current downloading scheme is that relay client j simply sends its current local model x_j^t to client i, who then replaces its current local model x_i^t with x_j^t, i.e., $x_i^t := x_j^t$. In this way, the clients do not have to keep a copy of the most recent global model, thereby reducing the stored data. The convergence analysis is not affected by this change and the same convergence bound can be proved (see the proof of Theorem 1).

Download Relay Protocol: In the download relay scenario, the receiver (client i) similarly sends a 'Beacon' message to a potential download relay (client j). Upon client j's agreement, indicated by a 'Willingness' message containing its last server meeting time (τ_j^{last}), client i compares this with its own last meeting time (τ_i^{last}). If client j is selected as the relay, client i sends a 'Confirmation' message, prompting client j to transmit its stored global model version $x^{\psi_j(t)}$ to client i, who then acknowledges receipt. If client j is not chosen, client i directly sends an 'Acknowledgement' message to conclude. This protocol is illustrated in Fig. 4.

C. Relaying Manipulated CLU

We present an extension of FedMobile where clients upload manipulated CLUs via relaying. Two kinds of manipulation operations are considered. The first is quantization and compression, which aim to reduce the size of the data being transmitted. The second type involves perturbation, implemented to enhance privacy protection. To avoid confusion, we let n_i^t denote the cumulative gradient update of client i's own, which is not combined with any CLUs received from other clients. We call n_i^t the private-CLU of client i.

Quantization: The quantization process is denoted by $Q(\cdot)$, signifying the quantizer operator. To ensure that each local update undergoes quantization exactly once, a client i, upon encountering a suitable relay client j, will quantize its private-CLU n_i^t instead of the already stored CLU m_i^t. This approach is taken because m_i^t may include CLUs from others who have used client i as a relay, and these have been previously compressed. Consequently, during the upload relay from client i to client j, there is a quantization-related discrepancy represented as $Q(n_i^t) - n_i^t$, in contrast to scenarios without quantization.

Perturbation: When client i uploads CLUs to a relay client j, it may incorporate a noise term, such as Gaussian Noise μ_i^t, to safeguard its privacy.

Although quantization and perturbation effects differ, they can both be conceptualized as introducing a variation term in the original upload transmission. For simplicity and clarity, this difference is denoted as ϵ_i^t. During the upload relay, if quantization is employed to reduce data size, $\epsilon_i^t = Q(n_i^t) - n_i^t$ represents the discrepancy introduced by quantization. In contrast, when perturbation is used to increase privacy, $\epsilon_i^t = \mu_i^t$ signifies the added noise.

Upload Timing and Relay Selection: This is the same as the original FedMobile strategy.

Upload Relay Mechanism: Upon a CLU exchange event at time t between a sender client i and a relay client j:

RESET (by sender): Before sending the CLU to the relay j, the sender first records the noise ϵ_i^t. Then the manipulated CLU, i.e., $\tilde{m}_i^t = m_i^t + \epsilon_i^t$, is sent to the relay. The sender then resets its CLU to $m_i^t := -\epsilon_i^t$ and its private CLU to $n_i^t := 0$.

COMBINE (by relay): After receiving $\tilde{m}_i^t$ from client i, client j updates it stored CLU m_j^t by incorporating $\tilde{m}_i^t$, i.e., $m_j^t := m_j^t + \tilde{m}_i^t$.

Remark: With the help of relay, the server could get the client i's manipulated CLU $\tilde{m}_i^t = m_i^t + \epsilon_i^t$ sooner. Note that in the case of the manipulated CLU, the sender's CLU is reset to $m_i^t := -\epsilon_i^t$ instead of $m_i^t := 0$. This reset ensures that when client i itself reaches the server, its uploaded CLU includes $-\epsilon_i^t$, effectively correcting the previously received manipulated CLU $\tilde{m}_i^t$ back to m_i^t.

V. Convergence Analysis

A. FedMobile With CLU

Our analysis utilizes the following assumptions.

Assumption 1 (Lipschitz Smoothness): There exists a constant $L > 0$ such that $\|\nabla f_i(x) - \nabla f_i(y)\| \leq L\|x - y\|, \forall x, y \in \mathbb{R}^d$ and $\forall i = 1, \ldots, N$.

Assumption 2 (Unbiased Local Gradient Estimate): The local gradient estimate is unbiased, i.e., $\mathbb{E}_\zeta F_i(x, \zeta) = \nabla f_i(x), \forall x$ and $\forall i = 1, \ldots, N$.

Assumption 3 (Bounded Variance): There exists a constant $\sigma > 0$ such that $\mathbb{E}[\|\nabla F_i(x, \zeta_i) - \nabla f_i(x)\|^2] \leq \sigma^2, \ \forall x \in \mathbb{R}^d$ and $\forall i = 1, \ldots, N$.

Assumption 4 (Bounded Second Moment): There exists a constant $G > 0$ such that $\mathbb{E}[\|\nabla F_i(x, \zeta_i)\|^2] \leq G^2, \forall x \in \mathbb{R}^d$ and $\forall i = 1, \ldots, N$.

The *real sequence* of the global model is calculated as

$$x^t = x^0 - \frac{1}{N} \sum_{i=1}^{N} \sum_{s=0}^{\phi_i(t)} \eta g_i^s, \forall t \tag{5}$$

where we define $\phi_i(t)$ to be the time slot up to when all corresponding gradients of client i have been received at time t. In the vanilla asynchronous FL case, $\phi_i(t)$ is simply $\tau_i^{\text{last}}(t) - 1$. In FedMobile, $\phi_i(t) > \tau_i^{\text{last}}(t) - 1$ because more information can be uploaded earlier than t due to relaying.

We also define the *virtual sequence* of the global model, which is achieved in the imaginary ideal case where all local gradients are uploaded to the server instantly at every slot,

$$v^t = x^0 - \frac{1}{N} \sum_{i=1}^{N} \sum_{s=0}^{t-1} \eta g_i^s, \forall t \tag{6}$$

First, we bound the difference $(t-1) - \phi_i(t)$, which characterizes how much CLU information of client i is missing compared to the virtual sequence.

Lemma 1: Assuming at least one semi-qualified upload relay client exists in every upload search interval, then we have $(t-1) - \phi_i(t) \leq \max\{\Delta - \theta, \Theta\} \triangleq C(\theta, \Theta; \Delta), \forall i = 1, \ldots, N, \forall t$.

Proof: It is obvious that if the server meeting time interval $\tau_i^{\text{next}}(t) - \tau_i^{\text{last}}(t) \leq \Theta$, then $(t-1) - \phi_i(t) = t - \tau_i^{\text{last}}(t) \leq \Theta$ already holds. Otherwise, for all $t \leq \tau_i^{\text{last}}(t) + \Theta$, then $(t-1) - \phi_i(t) = t - \tau_i^{\text{last}}(t) \leq \Theta$ also holds. Thus, we only need to consider the case $\tau_i^{\text{next}}(t) - \tau_i^{\text{last}}(t) > \Theta$ and for time slot $t > \tau_i^{\text{last}}(t) + \Theta$. In this case, a semi-qualified relay client is also a qualified relay client because

$$\tau_i^{\text{last}}(t) + \Theta < \tau_i^{\text{next}}(t) \tag{7}$$

By the assumption that at least one semi-qualified relay exists in the search interval, at least one qualified relay must exist. This further implies that the qualified relay client is able to upload a CLU before t. Because this CLU contains gradients of client i for at least θ steps since $\tau_i^{\text{last}}(t)$, we have $\phi_i(t) \geq \tau_i^{\text{last}}(t) + \theta - 1$. Therefore,

$$\begin{aligned}(t-1) - \phi_i(t) &= \big(t - \tau_i^{\text{last}}(t)\big) + \big(\tau_i^{\text{last}}(t) - \phi_i(t) - 1\big) \\ &\leq \Delta - \theta \end{aligned} \tag{8}$$

To summarize the above cases, $(t-1) - \phi_i(t) \leq \max\{\Delta - \theta, \Theta\}$ is established. □

Next, we bound the difference $t - \psi_i(t)$, which characterizes the version difference between the current global model and client i's copy of the global model.

Lemma 2: Assuming at least one semi-qualified download relay exists in every download search interval, we have $t - \psi_i(t) \leq \max\{\Delta - \omega, \Omega\} \triangleq D(\omega, \Omega; \Delta), \forall i = 1, \ldots, N, \forall t$.

Proof: Let t' be the meeting time between receiver client i and relay client j. Clearly, $\tau_i^{\text{next}} - \Omega \leq \tau_j^{\text{last}}(t) \leq t' \leq \tau_i^{\text{next}} - \omega$ by the definition of the search interval.

For $t \leq t'$, client i has not met client j yet, so $\psi_i(t) = \tau_i^{\text{last}}(t)$. Therefore,

$$\begin{aligned} t - \psi_i(t) &= t - \tau_i^{\text{last}}(t) \leq t' - \tau_i^{\text{last}}(t) \\ &\leq \tau_i^{\text{next}}(t) - \omega - \tau_i^{\text{last}}(t) \leq \Delta - \omega \end{aligned} \tag{9}$$

For $t > t'$, client i has met client j, so $\psi_i(t) \geq \tau_j^{\text{last}}(t)$. Therefore,

$$t - \psi_i(t) \leq t - \tau_j^{\text{last}}(t) \leq t - (\tau_i^{\text{next}}(t) - \Omega) \leq \Omega \tag{10}$$

To sum up, $t - \psi_i(t) \leq \max\{\Delta - \omega, \Omega\}$ □

The following lemma then bounds the model differences in the real sequence and the virtual sequence.

Lemma 3: The difference of the real global model and the virtual global model is bounded as follows

$$\mathbb{E}\left[\|v^t - x^t\|^2\right] \leq C^2(\theta, \Theta; \Delta)\eta^2 G^2 \tag{11}$$

For each client i, the difference of its local model and the virtual global model is bounded as follows

$$\mathbb{E}\left[\|v^t - x_i^t\|^2\right] \leq 3(2D^2(\omega, \Omega; \Delta) + C^2(\theta, \Theta; \Delta))\eta^2 G^2 \tag{12}$$

Proof: The proof is shown in the supplementary material. □

Now, we are ready to bound the convergence of the real model sequence achieved in FedMobile.

Theorem 1: Assuming at least one semi-qualified upload (download) relay exists in every upload (download) search interval, by setting $\eta \leq 1/L$, after T time slots, we have

$$\frac{1}{T}\sum_{t=0}^{T-1}\mathbb{E}\left[\|\nabla f(x^t)\|^2\right] \leq \frac{4}{\eta T}\left(f(x^0)-f^*\right)+\frac{2L\eta\sigma^2}{N} + 4(3D^2(\omega,\Omega;\Delta)+2C^2(\theta,\Theta;\Delta))L^2\eta^2G^2 \quad (13)$$

Proof: The proof is shown in the supplementary material. □

Remark: The convergence bound established in Theorem 1 contains three parts. The first part diminishes as T approaches infinity. Both the second and third terms are constants for a constant η, with the second term depending on our algorithm parameters θ, Θ, ω and Ω. For a given final step T, one can set $\eta = \frac{\sqrt{N}}{L\sqrt{T}}$ so that the bound becomes

$$\frac{4L}{\sqrt{NT}}\left(f(x^0)-f^*\right)+\frac{2\sigma^2}{\sqrt{NT}} + \frac{4N}{T}(3D^2(\omega,\Omega;\Delta)+2C^2(\theta,\Theta;\Delta))G^2 \quad (14)$$

Furthermore, if $T \geq N^3$, the above bound recovers the same $O(\frac{1}{\sqrt{NT}})$ convergence rate of the classic synchronous FL [5] .[2] Next, we discuss the above convergence result in more detail and investigate how the mobility affects the convergence. To this end, we consider for now a fixed client-client meeting rate ρ, and investigate how the convergence result depends on our algorithm parameters.

Bound optimization: Since the convergence bound can be improved by lowering $C(\theta,\Theta;\Delta)$ and $D(\omega,\Omega;\Delta)$, we first investigate them as functions of the algorithm parameters.

Proposition 1: (1) $C(\theta,\Theta;\Delta)$ is non-decreasing in Θ and non-increasing in θ. Moreover, $\forall\theta,\Theta$, $C(\theta,\Theta;\Delta) \geq \frac{\Delta}{2}$, and the lower bound is attained by choosing $\theta = \Theta = \frac{\Delta}{2}$. (2) $D(\omega,\Omega;\Delta)$ is non-decreasing in Ω and non-increasing in ω. Moreover, $\forall\omega,\Omega$, $D(\omega,\Omega;\Delta) \geq \frac{\Delta}{2}$, and the lower bound is attained by choosing $\omega = \Omega = \frac{\Delta}{2}$.

Proposition 1 implies that one should use shorter search intervals, namely $\Theta-\theta$ and $\Omega-\omega$, to lower $C(\theta,\Theta;\Delta)$ and $D(\omega,\Omega;\Delta)$. However, the best C and D are no smaller than $\frac{\Delta}{2}$, which are achieved by choosing $\theta = \Theta = \frac{\Delta}{2}$ and $\omega = \Omega = \frac{\Delta}{2}$. In other words, a short upload search interval around the time $\tau_i^{\text{last}} + \frac{\Delta}{2}$ and a short download search interval around the time $\tau_i^{\text{next}} - \frac{\Delta}{2}$ improve the FL convergence. This suggests that the best timing for uploading is at exactly $\tau_i^{\text{last}} + \frac{\Delta}{2}$ and the best timing for downloading is at exactly $\tau_i^{\text{next}} - \frac{\Delta}{2}$, which are neither too early nor too late in both cases.

Probability of meeting a semi-qualified relay: The convergence bound in Theorem 1, however, is obtained under the assumption that a client can meet at least one semi-qualified upload (download) relay client in each upload (download) search interval, which may not always hold. How easily a client can find a semi-qualified relay client depends on the client-client meeting rate. In the VANET example, this rate depends on the D2D communication range.

Let $Q_u(\theta,\Theta)$ (and $Q_d(\omega,\Omega)$) be the probability that at least one semi-qualified uploading (download) relay is met in an uploading (download) search interval with length $\Theta-\theta$ (and $\Omega-\omega$). They are characterized as follows

Proposition 2: Assuming sufficiently many clients in the system and that clients meet each other uniformly randomly. Let $P_{\text{int}}(\cdot)$ be the distribution of server meeting intervals. Then $Q_u(\theta,\Theta) = 1-\prod_{t=0}^{\Theta-\theta}(1-\rho q_u(\Theta-\theta-t))$ and $Q_d(\omega,\Omega) = 1-\prod_{t=0}^{\Omega-\omega}(1-\rho q_d(t))$, where $q_u(\cdot)$ and $q_d(\cdot)$ are distributions computed based on $P_{\text{int}}(\cdot)$.

Proof: The proof is shown in the supplementary material. □

Proposition 2 states an intuitive result that one should use a larger search interval to increase the probability of meeting a semi-qualified relay. This, however, is not desirable for lowering the convergence bound according to Proposition 1. This is exactly where mobility can help improving the FL convergence: by increasing the client-client meeting rate ρ, a shorter search interval $\Theta-\theta$ (or $\Omega-\omega$) can be used to achieve the same Q_u (or Q_d), but a smaller C (or D) is obtained. In fact, by relaxing the constraint that a client can only meet one other client at a time slot, with sufficiently many clients in the system, both Q_u and Q_d can approach 1 even if the search interval is just one slot.

B. FedMobile With Manipulated CLU

Then we validate the convergence of FedMobile with manipulated CLU uploading. We first state an additional assumption on the noise term ϵ_i^t.

Assumption 5 (Bounded Error): The noise term ϵ_i is bounded, i.e., $\mathbb{E}[\|\epsilon_i^t\|^2] \leq q\mathbb{E}[\|n_i^t\|^2]$, $\forall i = 1,\ldots,N$, $\forall t$ for some positive real constant q.

The *corrupted real sequence* can be written as

$$\widetilde{x}^t = x^t + \frac{1}{N}\sum_{i\in U_t}\epsilon_i, \forall t \quad (15)$$

where we define U_t as the set of clients for whom only corrupted CLUs have been received by the server via the relay, and $|U_t|$ is the size of U_t. The real sequence x^t is the imaginary real sequence where the noise is not added.

Lemma 4: The difference of the corrupted real global model and the virtual global model is bounded as follows

$$\mathbb{E}\left[\|v^t-\widetilde{x}^t\|^2\right] \leq 2C^2(\theta,\Theta;\Delta)\eta^2G^2 + 2q\Theta^2\eta^2G^2 \quad (16)$$

For each client i, the difference of its local model and the virtual global model is bounded as follows

$$\mathbb{E}\left[\|v^t-x_i^t\|^2\right] \leq 6(D^2(\omega,\Omega;\Delta)+C^2(\theta,\Theta;\Delta)+q\Theta^2)\eta^2G^2 \quad (17)$$

Proof: The proof is shown in the supplementary material. □

Theorem 2: With manipulated CLU uploading, assuming at least one semi-qualified upload (download) relay client exists in every upload (download) search interval, by setting $\eta \leq 1/L$,

[2]There is a subtle difference because T in the asynchronous setting is the number of time slots while in the synchronous setting T is the number of rounds. However, they differ by at most a factor of Δ.

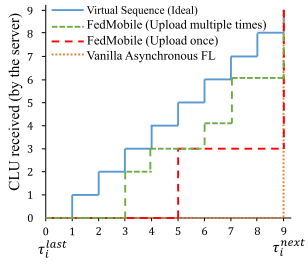

Fig. 5. Uploading.

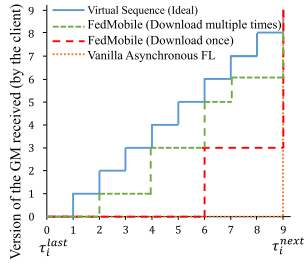

Fig. 6. Downloading.

after T time slots, we have

$$\frac{1}{T}\sum_{t=0}^{T-1}\mathbb{E}\left[\|\nabla f(x^t)\|^2\right] \leq \frac{4}{\eta T}\left(f(x^0)-f^*\right)+\frac{2L\eta\sigma^2}{N} + 4(3D^2(\omega,\Omega;\Delta)+4C^2(\theta,\Theta;\Delta)+4q\Theta^2)L^2\eta^2G^2 \quad (18)$$

Proof: The proof of Theorem 2 follows the proof of Theorem 1 by replacing Lemma 3 with Lemma 4. □

Remark: With setting $\eta=\frac{\sqrt{N}}{L\sqrt{T}}$, the convergence bound established in Theorem 2 becomes

$$\frac{4L}{\sqrt{NT}}\left(f(x^0)-f^*\right)+\frac{2\sigma^2}{\sqrt{NT}} + \frac{4N}{T}(3D^2(\omega,\Omega;\Delta)+4C^2(\theta,\Theta;\Delta)+4q\Theta^2)G^2 \quad (19)$$

Furthermore, if $T\geq N^3$, the above bound recovers the same $O(\frac{1}{\sqrt{NT}})$ convergence rate of the classic synchronous FL [5].

VI. The General Case

FedMobile can be easily extended to allow multiple upload (download) relay communications between any two consecutive server meetings. We will still have the upload (download) search interval, and FedMobile will simply pick the first K qualified upload (download) relays to perform the upload (download) relay communications. The exact mechanisms of how uploading (downloading) is performed will be slightly changed to handle out-of-order information and avoid redundant information. Our convergence analysis still holds correctly for this generalized case, although the bound may become looser compared to the actual performance.

Figs. 5, 6 illustrate the key ideas behind FedMobile. Fig. 5 plots hypothetical sequences of CLUs received by the server from a representative client between two consecutive server meetings. The virtual sequence is the ideal case where each local stochastic gradient g_i^t is uploaded to the server instantly after it is computed. Therefore, the received CLU curve is a steady staircase. In the vanilla asynchronous FL, the client uploads the CLU only when it meets the server. Therefore there is a big jump at the next server meeting time but it is all 0 before. FedMobile with at most one upload relaying allows some local stochastic gradient information to be received by the server earlier. The generalized FedMobile allows more CLUs to be relayed and received by the server at earlier time slots. One can imagine that when ρ is large enough, it becomes much easier for the client to find qualified relays that can quickly upload CLUs to the server at every time slot. Therefore, the received CLU curve approaches that in the ideal case. Similarly, Fig. 6 plots the hypothetical sequence of the global models received by the client.

VII. Experiments

A. Setup

We implement FL simulation on the Pytorch framework and perform the model training on one Geforce RTX 3080 GPU. All experiment results are averaged over 3 repeats. To simulate communications among the clients, at each time slot, we uniformly sample ρN clients over total N clients and randomly construct $\frac{\rho N}{2}$ client pairs. Any two clients in the same pair are simulated to communicate. We conduct experiments on a synthetic dataset and two real-world datasets, i.e., FMNIST [39] and CIFAR10 [40].

Synthetic dataset: The synthetic dataset is generated on a least-squares linear regression problem. Each data sample has a 200-dimensional feature vector and its real-valued target is calculated as the vector product of the feature and an underlying linear weight plus a 0-mean Gaussian noise. The FL system has 50 clients with each client having 40 data samples. The default training hyper-parameters are: learning rate equals 0.01, learning rate decay factor equals 0.99 until learning rate reaches 0.0001, training batch size equals 128, the total number of training time slots equals 150, default upload parameters $\theta=10,\Theta=40$ and default download parameters $\omega=5,\Omega=25$.

FMNIST: The FL system has 50 clients with each client having 400 data samples. We utilize the Dirichlet function ($\alpha=0.3$) which is typically utilized to simulate the level of non-IID in FL. We use LeNet [41] as the backbone model. The default hyper-parameters are: learning rate equal to 0.1, learning rate decay factor equals to 0.99 until learning rate reaches 0.001, training batch size equals 128, the total training time slots equals 250, default upload parameters $\theta=10,\Theta=40$ and default download parameters $\omega=5,\Omega=25$.

CIFAR10: The FL system has 50 clients with each client having 600 data samples. The data allocation method is the same as that used for FMNIST. We use ResNet-9 [42] as the backbone model. The training configuration details are as follows: learning rate equals 0.01, training batch size equals 128, the total number of training time slots equals 500, default upload parameters $\theta=10$, $\Theta=40$ and default download parameters $\omega=5,\Omega=25$.

Benchmarks: The following benchmarks are considered in our experiments. (1) **ASYNC**: This is the state-of-the-art asynchronous FL method proposed in [12] to handle arbitrary communication patterns. (2) **Virtual-U**: This method assumes that

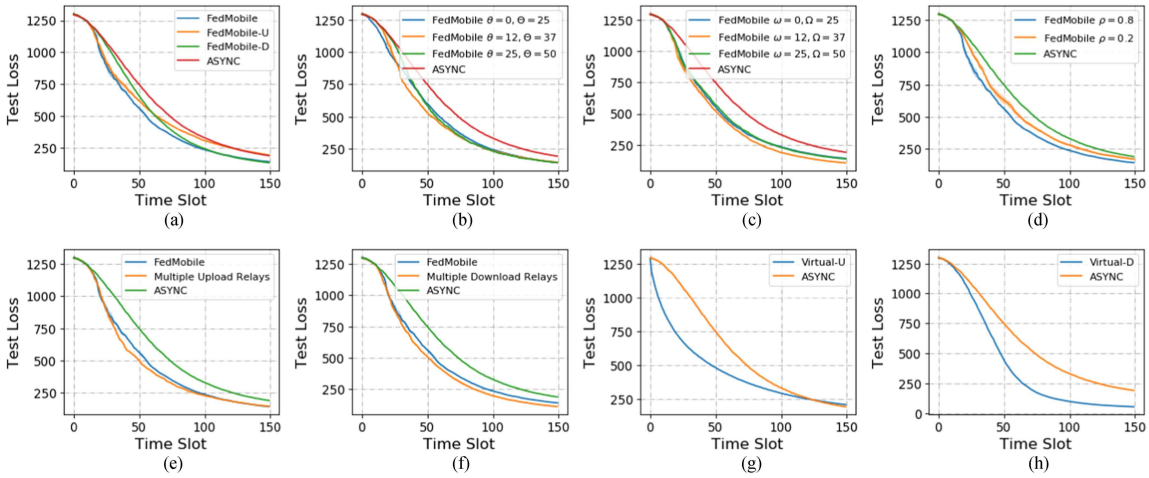

Fig. 7. Results on the synthetic data. (a) Performance. (b) Upload Search Inter. (c) Download Search Inter. (d) Client Meeting rate. (e) Multiple Uploads. (f) Multiple Downloads. (g) Virtual Upload. (h) Virtual Downloads.

each client can upload its local updates immediately to the server at every slot via an imaginary channel but can only download the global model at their actual communication slots. (3) **Virtual-D**: This method assumes that each client can download the global model from the server at every slot via an imaginary channel but can only upload their CLUs at their actual communication slots. We also consider several variations of FedMobile. (1) **FedMobile**: This is the proposed algorithm combining both upload and download relay communications. (2) **FedMobile-U**: This is the proposed algorithm with only upload relaying. (3) **FedMobile-D**: This is the proposed algorithm with only download relaying.

Communication Patterns: We consider four distinct communication patterns in total. Three of these patterns simulate asynchronous communication with the server. The **Fixed-Interval** pattern involves each client $i \in \{1, \ldots, 50\}$ communicating with the server at intervals defined by the slots $i + 50n, \forall n = 0, 1, 2, \ldots$.

For **Random-Interval**, each client i first communicates with the server at slot i, and then continues to communicate at random intervals ranging between 30 and 50 slots.

The **Random-Interval (Exponential)** pattern is similar, with the first communication of each client $i \in \{1, \ldots, 50\}$ occurring at slot i. However, subsequent communications with the server follow an exponentially distributed pattern with a mean of 30-time slots. The distribution is truncated (with a maximum of 80-time slots) to ensure that the intervals between consecutive server meetings remain bounded.

Lastly, we simulate a **Smart Public Transportation** communication pattern, aiming to mimic a real-world public transportation system. This pattern reflects communication structures in four actual bus routes used in Miami, where we envision RSUs being installed.

Our main objective is to enhance the convergence speed in the asynchronous FL setting. It is important to mention that in the experimental results presented below, both FedMobile and the baselines eventually achieve convergence. However, to better highlight the difference in convergence speed between FedMobile and the baselines, we have chosen not to plot the final convergence stage due to the space limitation. We intend to demonstrate that when targeting a specific test accuracy, the utilization of FedMobile leads to a reduction in the required number of time slots.

B. Results on Synthetic Data

Fig. 7 reports the results (averaged over three repeats) on the synthetic data with a fixed-interval communication pattern. Fig. 7(a) compares **FedMobile** and its variations with **ASYNC** in terms of the test loss. As can been seen, incorporating either upload or download relaying into asynchronous FL improves the FL performance, and a further improvement can be achieved when uploading and downloading are combined. Fig. 7(b) and (c) illustrates the impact of upload/download search interval on the FL convergence. In both cases, the best search timing is around the middle point of the two consecutive server meeting times, confirming our theoretical analysis in Theorem 1 and Proposition 1. Fig. 7(d) shows the impact of ρ on the FL convergence. As predicted by our analysis, a higher ρ in the system improves FL convergence since more timely relay communication opportunities are created. In Fig. 7(e) and (f), we allow clients to use multiple relays to create more communication opportunities with the server whenever possible. The results show that further improvement can indeed be achieved. We also conduct experiments on the ideal relaying scenarios, namely **Virtual-U** and **Virtual-D**, to illustrate what can be achieved in the ideal case. The results verify our hypothesis that more communication opportunities with the server benefit convergence. It is also interesting to note that virtual uploading/downloading has a more significant impact on the early/late FL slots, suggesting that an adaptive design may better balance the FL performance and the resource cost.

C. Results on CIFAR10

We now report the results (averaged over three repeats) on CIFAR10. We validate that our proposed method, **FedMobile**

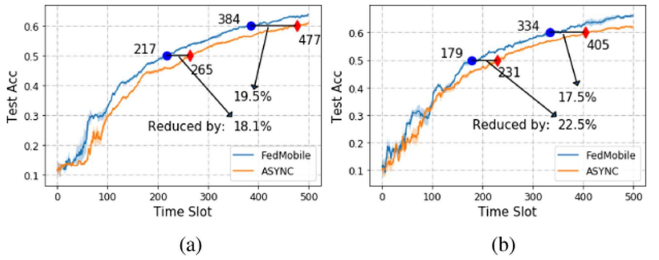

Fig. 8. Performance comparison on CIFAR10. (a) CIFAR10 (fixed). (b) CIFAR10 (random).

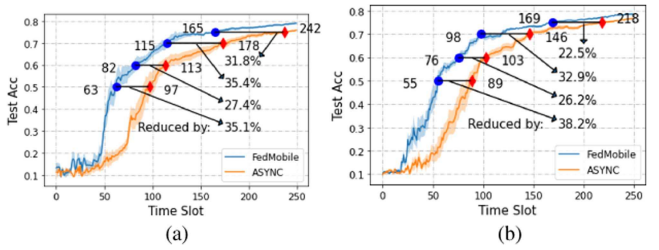

Fig. 9. Performance comparison on FMNIST. (a) FMNIST (fixed). (b) FMNIST (random).

can achieve a better convergence speed than **ASYNC** on CIFAR10 under both Fixed-Interval and Random-Interval settings in Fig. 8. Note that the main point of our paper is utilizing relays to improve the convergence speed. Hence, by setting a target accuracy level and comparing the required number of time slots to reach it, the notable advancements of FedMobile are evident. For a specific example, to achieve 60% accuracy under fixed-interval setting, **ASYNC** needs about 384 slots while **FedMobile** only needs 477 slots. The required time slot decreases by 19.5%.

D. Results on FMNIST

Then we report the results (averaged over three repeats) on FMNIST. Fig. 9 shows that **FedMobile** achieves a better convergence speed than **ASYNC** on FMNIST under both the Fixed-Interval and Random-Interval settings.

Scalability Analysis: To validate the scalability of our proposed FedMobile method, we conduct supplementary experiments under a Fixed-Interval setting. Initially, we set the interval at 50-time slots, aiming to evaluate whether FedMobile maintains its superiority over the baseline method as the interval increases. With a constant number of clients, we extend the interval to 70-time slots and adjust the upload/download search intervals accordingly (i.e., $\theta = 20, \Theta = 60, \omega = 10, \Omega = 30$). The outcomes, depicted in Fig. 10(a), affirm that FedMobile surpasses the baseline method.

Furthermore, we validate FedMobile's performance with an increased client count. Maintaining the fixed interval at 50-time slots, we augment the client number from 50 to 100, with each client possessing 400 data samples exhibiting a non-IID degree of $\alpha = 0.3$. The findings, showcased in Fig. 10(b), reveal that FedMobile consistently exceeds the baseline method. Notably, the inclusion of more clients accelerates the training process for both FedMobile and the baseline method, with FedMobile

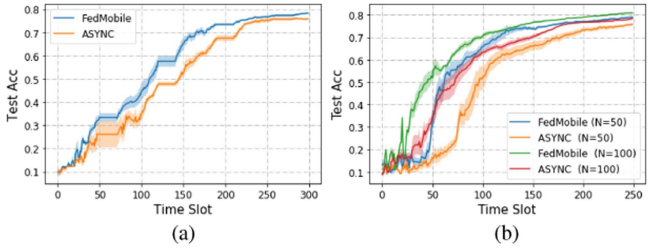

Fig. 10. Performance comparisons under various configs. (a) Larger communication intervals. (b) Larger amount of clients.

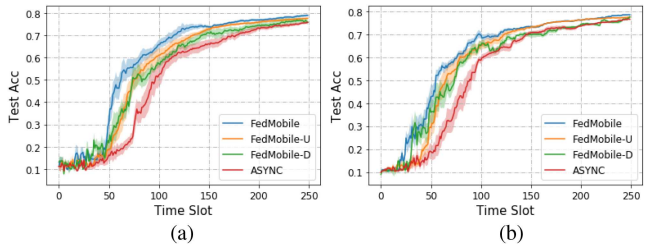

Fig. 11. Effect of upload/download relay. (a) Upload/download (fixed). (b) Upload/download (random).

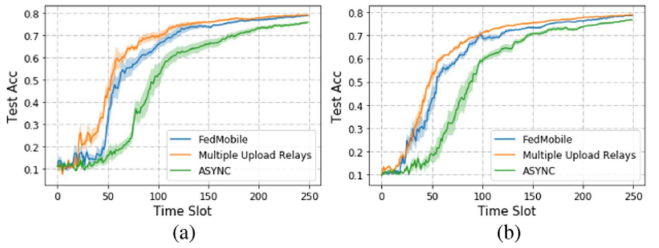

Fig. 12. Effect of multiple upload relays. (a) Multiple upload (fixed). (b) Multiple upload (random).

continually demonstrating superior performance. These experiments collectively attest to the scalability of FedMobile across diverse configurations.

Effect of Upload/Download: Under both communication patterns, Fig. 11(a) and (b) demonstrate that while upload and download relay communications individually can improve the FL convergence performance, combining them results in an additional benefit.

Multiple Relays: In the previous result of the synthetic dataset, we find that both multiple uploads relays and multiple downloads relays can improve FedMobile's convergence speed. In the real-world dataset experiments, Fig. 12(a) and (b) show that using multiple upload relay communications further improves the FL convergence performance. However, we observed in our experiments that this is not the case with using multiple download relay communications. This is likely due to the complexity of real-world datasets, which causes the assumptions for our theoretical analysis to be violated. This represents a limitation of our current analysis and requires further investigation.

Virtual-U and Virtual-D: To further present the different effects between multiple uploads and multiple downloads, we consider two ideal cases. **Virtual-U** and **Virtual-D** work as the ideal cases for upload relaying and download relaying and hence we conduct experiments to investigate their performance on

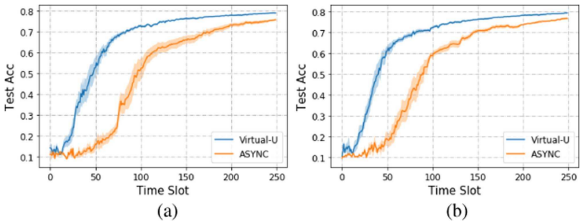

Fig. 13. Virtual-U on FMNIST. (a) Virtual-U (fixed). (b) Virtual-U (random).

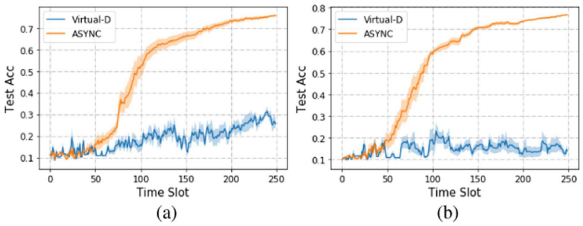

Fig. 14. Virtual-D on FMNIST. (a) Virtual-D (fixed). (b) Virtual-D (random).

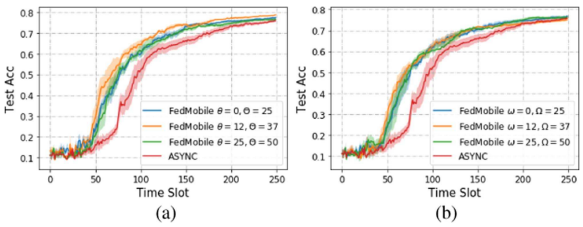

Fig. 15. Effect of upload/download Interval. (a) Upload Search Inter. (b) Download Search Inter.

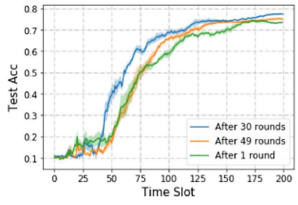

Fig. 16. Download time.

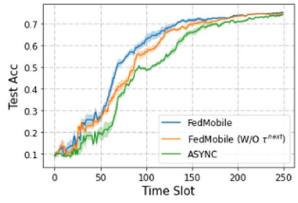

Fig. 17. Unknown next meeting time under the *Random-Interval (Exponential)*.

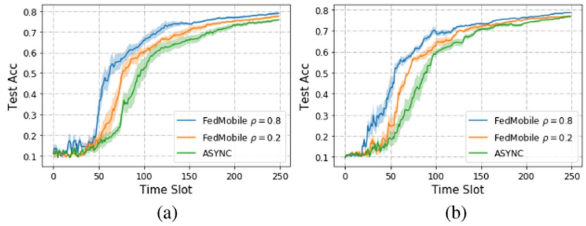

Fig. 18. Client meeting rate effect. (a) Client meeting rate (fixed). (b) Client meeting rate (random).

the real-world dataset. Figs. 13 and 14 report the performance of **Virtual-U** and **Virtual-D** on FMNIST with both fixed and random interval communication patterns. Similar to Fig. 7(g) for the synthetic data, Fig. 13(a) and (b) show that **Virtual-U** can greatly improve the convergence performance. However, different from Fig. 7(h) for the synthetic data, Fig. 14(a) and (b) show that **Virtual-D** fail to converge on the real-world dataset. We conjecture that this is likely due to the complexity of the real-world dataset where some of the assumptions needed for our theoretical analysis do not strictly hold. However, as Fig. 11 shows, using one-time download relaying still has benefits on the convergence even in the real-world dataset.

Setting of upload/download interval: Our theoretical analysis emphasizes the importance of precise timing in both the upload and download relay mechanisms. To achieve optimal performance and avoid relay operations that occur too early or too late, it is crucial to carefully set the parameters for the upload/download search interval. These parameters should be around the median value of the maximum time interval (denoted as Δ) between consecutive server communications. This concept is demonstrated in Fig. 15, where adjusting the upload/download parameters around the midpoint of Δ yields relay timings that result in the best convergence speed.

However, it's worth noting the marginal variability among the three download search intervals depicted in Fig. 15(b). To better comprehend the impact of download relay timing, we devised a hypothetical scenario wherein each client could download the global model 1/30/49 time slot(s) after their last server interaction. The outcome, as illustrated in Fig. 16, unambiguously demonstrates that the optimal download relay timing should steer clear of being too early or too late.

Unknown Next meeting time: We further evaluate our proposed method using the **Random-Interval (Exponential)** pattern. We here analyze two distinct scenarios: one in which clients are informed of their next meeting time with the server, and another where this time is not known. In the latter case, the expected value of the exponential distribution is used to predict the timing of the next server interaction. As illustrated in Fig. 17, our findings reveal that FedMobile consistently outperforms the baseline, regardless of whether it employs precise next meeting time data or estimates based on the expected value of the exponential distribution.

Client-Client Meeting Rate: We further investigate the effect of client-client meeting rate under the fixed and random-interval communication pattern. Fig. 18(a) and (b) shows that the higher client-client meeting rate improves convergence speed.

Relaying Manipulated CLU: We test two types of manipulation. In the first type, we directly add Gaussian noises to the relayed CLU. Fig. 19 shows the convergence curves under different amount of noises. (e.g. Gaussian Noise $\mathcal{N}_1(0, 0.01)$ and Gaussian Noise $\mathcal{N}_2(0, 0.001)$). In the second type, we utilize the low precision quantizer in [43] to quantize the CLU before relaying. Here the quantization level is defined as s. Fig. 20 shows the convergence curves under different quantization levels. In both

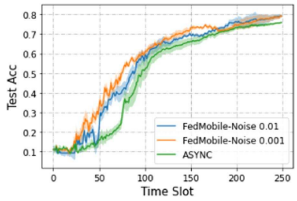

Fig. 19. Noise level.

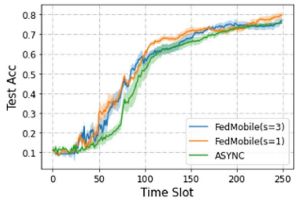

Fig. 20. Quantization.

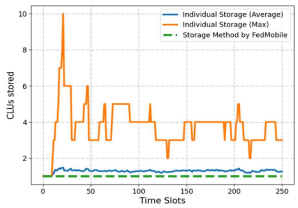

Fig. 21. Storage efficient.

cases, FedMobile still outperforms the baseline method provided that the added noise is small or the adopted quantization level is low.

Storage Efficient In FedMobile, we employ the RESET and COMBINE actions to establish a streamlined and storage-efficient method that ensures precise, one-time delivery of specific information to the server, without retaining client ID information. Compared to the approach of recording each received client ID and storing each local update separately at the relay client, FedMobile's storage mechanism exhibits significantly higher efficiency, as illustrated in Fig. 21.

FedMobile in Smart Publication Transportation: We here contemplate implementing our suggested method, FedMobile, in a more real-world communication pattern scenario, such as smart public transportation. This pattern embodies communication structures seen in four actual bus routes operating in Miami between West Kendall (WK) and Dadeland South (DS), assuming the installation of RSUs. For simplicity, and without losing generality, we posit that two RSUs are positioned at the two destinations (West Kendall and Dadeland South). We hypothesize that on each bus line, there are 12/13 buses (totaling 50 buses for all 4 lines) that begin from different starting points at the commencement of the training process. On each line, 5 buses initiate their route from West Kendall to Dadeland South, while the remaining 5 start from Dadeland South heading towards West Kendall. Each bus will switch its direction upon reaching the terminal (either Dadeland South or West Kendall) and will continue operating on its route throughout the entire training process. This arrangement is visually represented in Fig. 23. Aligning with the real-world settings, we consider the buses to travel at a speed between 0.4 to 0.6 miles per minute. The

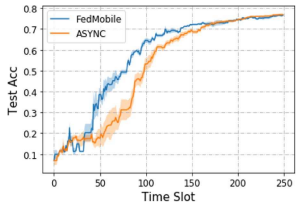

Fig. 22. Performances (smart publication transportation).

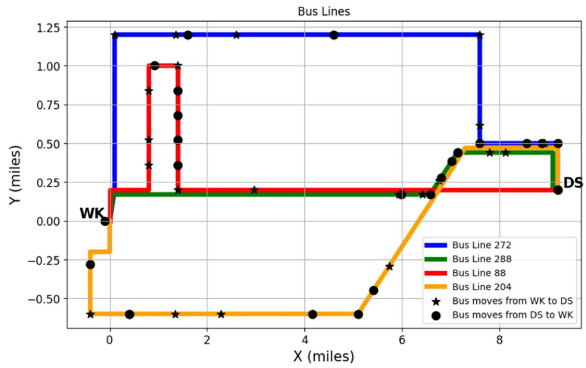

Fig. 23. An illustration of smart public transportation is provided, using four real bus lines in Miami.

time slot, defined as the duration required for one round of local training, is set to 30 seconds. The V2V communication range is fixed at 0.6 miles. To simulate the training, we employ the local client data distribution as described by FMNIST in the setup section. The outcomes presented in Fig. 22 confirm that our recommended method, FedMobile, surpasses the performance of the state-of-the-art Asynchronous FL method, ASYNC.

VIII. Conclusion

This paper focuses on the design of asynchronous Federated Learning (FL) algorithms for practical systems where continuous client-server communication is not always available. We emphasize the importance of client mobility and the resulting random client-client communication opportunities in facilitating timely information exchange for model training in asynchronous FL. To harness the advantages of additional client-client communication, we propose a new FL algorithm called FedMobile. FedMobile not only accelerates convergence but also maintains storage efficiency and prevents duplicate update transmission. We provide a detailed convergence analysis and conduct extensive experiments to validate our proposed method. The results demonstrate that FedMobile significantly improves convergence speed. We believe that FedMobile has the potential to advance distributed machine learning, especially FL, in various real-world systems such as mobile gaming, mobile sensing, and smart vehicular networks.

References

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

[2] X. Li, K. Huang, W. Yang, S. Wang, and Z. Zhang, "On the convergence of FedAvg on non-IID data," in *Proc. Int. Conf. Learn. Representations*, 2019.
[3] A. Khaled, K. Mishchenko, and P. Richtárik, "Tighter theory for local SGD on identical and heterogeneous data," in *Proc. Int. Conf. Artif. Intell. Statist.*, 2020, pp. 4519–4529.
[4] H. Yang, M. Fang, and J. Liu, "Achieving linear speedup with partial worker participation in non-IID federated learning," in *Proc. Int. Conf. Learn. Representations*, 2020.
[5] H. Yu, S. Yang, and S. Zhu, "Parallel restarted SGD with faster convergence and less communication: Demystifying why model averaging works for deep learning," in *Proc. AAAI Conf. Artif. Intell.*, 2019, pp. 5693–5700.
[6] M. Chen, H. V. Poor, W. Saad, and S. Cui, "Wireless communications for collaborative federated learning," *IEEE Commun. Mag.*, vol. 58, no. 12, pp. 48–54, Dec. 2020.
[7] Y. Li, D. Yuan, A. S. Sani, and W. Bao, "Enhancing federated learning robustness in adversarial environment through clustering non-IID features," *Comput. Secur.*, vol. 132, 2023, Art. no. 103319.
[8] Y. Li, L. Zhong, D. Yuan, H. Chen, and W. Bao, "ICB FL: Implicit class balancing towards fairness in federated learning," in *Proc. Australas. Comput. Sci. Week*, 2023, pp. 135–142.
[9] Y. Liu, J. James, J. Kang, D. Niyato, and S. Zhang, "Privacy-preserving traffic flow prediction: A federated learning approach," *IEEE Internet Things J.*, vol. 7, no. 8, pp. 7751–7763, Aug. 2020.
[10] Y. Zheng, S. Lai, Y. Liu, X. Yuan, X. Yi, and C. Wang, "Aggregation service for federated learning: An efficient, secure, and more resilient realization," *IEEE Trans. Dependable Secure Comput.*, vol. 20, no. 2, pp. 988–1001, Mar./Apr. 2023.
[11] A. Guerna, S. Bitam, and C. T. Calafate, "Roadside unit deployment in internet of vehicles systems: A survey," *Sensors*, vol. 22, no. 9, 2022, Art. no. 3190.
[12] D. Avdiukhin and S. Kasiviswanathan, "Federated learning under arbitrary communication patterns," in *Proc. Int. Conf. Mach. Learn.*, 2021, pp. 425–435.
[13] Y. Liu, L. Xu, X. Yuan, C. Wang, and B. Li, "The right to be forgotten in federated learning: An efficient realization with rapid retraining," in *Proc. IEEE Conf. Comput. Commun.*, 2022, pp. 1749–1758.
[14] J. Wang, V. Tantia, N. Ballas, and M. Rabbat, "SlowMo: Improving communication-efficient distributed SGD with slow momentum," in *Proc. Int. Conf. Learn. Representations*, 2019.
[15] W. Liu, L. Chen, Y. Chen, and W. Zhang, "Accelerating federated learning via momentum gradient descent," *IEEE Trans. Parallel Distrib. Syst.*, vol. 31, no. 8, pp. 1754–1766, Aug. 2020.
[16] J. Konečny, H. B. McMahan, D. Ramage, and P. Richtárik, "Federated optimization: Distributed machine learning for on-device intelligence," 2016, *arXiv:1610.02527*.
[17] S. P. Karimireddy, S. Kale, M. Mohri, S. Reddi, S. Stich, and A. T. Suresh, "SCAFFOLD: Stochastic controlled averaging for federated learning," in *Proc. Int. Conf. Mach. Learn.*, 2020, pp. 5132–5143.
[18] I. Mitliagkas, C. Zhang, S. Hadjis, and C. Ré, "Asynchrony begets momentum, with an application to deep learning," in *Proc. IEEE 54th Annu. Allerton Conf. Commun. Control Comput.*, 2016, pp. 997–1004.
[19] S. Hadjis, C. Zhang, I. Mitliagkas, D. Iter, and C. Ré, "Omnivore: An optimizer for multi-device deep learning on CPUs and GPUs," 2016, *arXiv:1606.04487*.
[20] J. Chen, X. Pan, R. Monga, S. Bengio, and R. Jozefowicz, "Revisiting distributed synchronous SGD," 2016, *arXiv:1604.00981*.
[21] S. Zheng et al., "Asynchronous stochastic gradient descent with delay compensation," in *Proc. Int. Conf. Mach. Learn.*, 2017, pp. 4120–4129.
[22] W. Dai, Y. Zhou, N. Dong, H. Zhang, and E. Xing, "Toward understanding the impact of staleness in distributed machine learning," in *Proc. Int. Conf. Learn. Representations*, 2018.
[23] T. Chen, G. Giannakis, T. Sun, and W. Yin, "LAG: Lazily aggregated gradient for communication-efficient distributed learning," in *Proc. Adv. Neural Inf. Process. Syst.*, 2018, pp. 5055–5065.
[24] V. Smith, C.-K. Chiang, M. Sanjabi, and A. S. Talwalkar, "Federated multi-task learning," in *Proc. Adv. Neural Inf. Process. Syst.*, 2017, pp. 4427–4437.
[25] T. Nishio and R. Yonetani, "Client selection for federated learning with heterogeneous resources in mobile edge," in *Proc. IEEE Int. Conf. Commun.*, 2019, pp. 1–7.
[26] M. van Dijk, N. V. Nguyen, T. N. Nguyen, L. M. Nguyen, Q. Tran-Dinh, and P. H. Nguyen, "Asynchronous federated learning with reduced number of rounds and with differential privacy from less aggregated Gaussian noise," 2020, *arXiv:2007.09208*.
[27] Z. Chai, Y. Chen, A. Anwar, L. Zhao, Y. Cheng, and H. Rangwala, "FedAT: A high-performance and communication-efficient federated learning system with asynchronous tiers," in *Proc. Int. Conf. High Perform. Comput. Netw. Storage Anal.*, 2021, pp. 1–16.
[28] Y. Chen, Y. Ning, M. Slawski, and H. Rangwala, "Asynchronous online federated learning for edge devices with non-IID data," in *Proc. IEEE Int. Conf. Big Data*, 2020, pp. 15–24.
[29] X. Li, Z. Qu, B. Tang, and Z. Lu, "Stragglers are not disaster: A hybrid federated learning algorithm with delayed gradients," 2021, *arXiv:2102.06329*.
[30] D. Basu, D. Data, C. Karakus, and S. Diggavi, "Qsparse-local-SGD: Distributed SGD with quantization, sparsification and local computations," in *Proc. Adv. Neural Inf. Process. Syst.*, 2019, Art. no. 1316.
[31] A. Lalitha, O. C. Kilinc, T. Javidi, and F. Koushanfar, "Peer-to-peer federated learning on graphs," 2019, *arXiv:1901.11173*.
[32] X. Zhang, Y. Liu, J. Liu, A. Argyriou, and Y. Han, "D2D-assisted federated learning in mobile edge computing networks," in *Proc. IEEE Wireless Commun. Netw. Conf.*, 2021, pp. 1–7.
[33] H. Xing, O. Simeone, and S. Bi, "Decentralized federated learning via SGD over wireless D2D networks," in *Proc. IEEE 21st Int. Workshop Signal Process. Adv. Wireless Commun.*, 2020, pp. 1–5.
[34] Y. Guo, Y. Sun, R. Hu, and Y. Gong, "Hybrid local SGD for federated learning with heterogeneous communications," in *Proc. Int. Conf. Learn. Representations*, 2021.
[35] S. Cui, A. M. Haimovich, O. Somekh, and H. V. Poor, "Opportunistic relaying in wireless networks," *IEEE Trans. Inf. Theory*, vol. 55, no. 11, pp. 5121–5137, Nov. 2009.
[36] D. Liu et al., "Opportunistic UAV utilization in wireless networks: Motivations, applications, and challenges," *IEEE Commun. Mag.*, vol. 58, no. 5, pp. 62–68, May 2020.
[37] A. Asadi, Q. Wang, and V. Mancuso, "A survey on device-to-device communication in cellular networks," *IEEE Commun. Surveys Tuts.*, vol. 16, no. 4, pp. 1801–1819, Fourth Quarter 2014.
[38] A. B. Reis, S. Sargento, F. Neves, and O. K. Tonguz, "Deploying roadside units in sparse vehicular networks: What really works and what does not," *IEEE Trans. Veh. Technol.*, vol. 63, no. 6, pp. 2794–2806, Jul. 2014.
[39] H. Xiao, K. Rasul, and R. Vollgraf, "Fashion-MNIST: A novel image dataset for benchmarking machine learning algorithms," 2017, *arXiv:1708.07747*.
[40] A. Krizhevsky et al., "Learning multiple layers of features from tiny images," 2009. [Online]. Available: https://api.semanticscholar.org/CorpusID:18268744
[41] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proc. IEEE*, vol. 86, no. 11, pp. 2278–2324, Nov. 1998.
[42] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2016, pp. 770–778.
[43] D. Alistarh, D. Grubic, J. Li, R. Tomioka, and M. Vojnovic, "QSGD: Communication-efficient SGD via gradient quantization and encoding," in *Proc. Adv. Neural Inf. Process. Syst.*, 2017, pp. 1707–1718.

Jieming Bian (Graduate Student Member, IEEE) received the B.A. degree in economics from the University of Colorado Denver, Denver, CO, USA, in 2019, and the M.S. degree in operations research from Columbia University, New York, NY, USA, in 2021. He is currently working toward the Ph.D. degree with the Electrical and Computer Engineering Department, University of Miami, Coral Gables, FL, USA. His research interests include communication efficiency and client scheduling federated learning problems.

Jie Xu (Senior Member, IEEE) received the B.S. and M.S. degrees in electronic engineering from Tsinghua University, Beijing, China, in 2008 and 2010, respectively, and the Ph.D. degree in electrical engineering from University of California, Los Angeles, Los Angeles, CA, USA, in 2015. He is currently an Associate Professor with the Department of Electrical and Computer Engineering, University of Miami, Coral Gables, FL, USA. His research interests include mobile edge computing/intelligence, machine learning for networks, and network security. He was the recipient of the NSF CAREER Award in 2021.