Federated Learning via Indirect Server-Client Communications

Jieming Bian
Department of ECE
University of Miami
Coral Gables, FL 33146
jxb1974@miami.edu

Cong Shen
Department of ECE
University of Virginia
Charlottesville, VA 22904
cong@virginia.edu

Jie Xu
Department of ECE
University of Miami
Coral Gables, FL 33146
jiexu@miami.edu

2023 57th Annual Conference on Information Sciences and Systems (CISS) | 978-1-6654-5181-9/23/$31.00 ©2023 IEEE | DOI: 10.1109/CISS56502.2023.10089783

***Abstract*—Federated Learning (FL) is a communication-efficient and privacy-preserving distributed machine learning framework that has gained a significant amount of research attention recently. Despite the different forms of FL algorithms (e.g., synchronous FL, asynchronous FL) and the underlying optimization methods, nearly all existing works implicitly assumed the existence of a communication infrastructure that facilitates the direct communication between the server and the clients for the model data exchange. This assumption, however, does not hold in many real-world applications that can benefit from distributed learning but lack a proper communication infrastructure (e.g., smart sensing in remote areas). In this paper, we propose a novel FL framework, named FedEx (short for FL via Model Express Delivery), that utilizes mobile transporters (e.g., Unmanned Aerial Vehicles) to establish indirect communication channels between the server and the clients. Two algorithms, called FedEx-Sync and FedEx-Async, are developed depending on whether the transporters adopt a synchronized or an asynchronized schedule. Even though the indirect communications introduce heterogeneous delays to clients for both the global model dissemination and the local model collection, we prove the convergence of both versions of FedEx. The convergence analysis subsequently sheds lights on how to assign clients to different transporters and design the routes among the clients. The performance of FedEx is evaluated through experiments in a simulated network on two public datasets.**

I. Introduction

In recent years, Federated Learning (FL) has emerged as a popular distributed machine learning framework where a number of distributed clients can collaboratively train a common machine learning model under the coordination of a parameter server without exposing their own data to another party. With an unprecedented amount of data being generated on edge devices such as smart phones and Internet-of-Things (IoT) devices as well as the rising privacy concerns associated with uploading this data to the cloud, FL is now widely considered as the next-generation machine learning paradigm to power a broad variety of applications, ranging from healthcare to agriculture, transportation, industrial IoT and mobile applications due to its distributed nature and privacy-preserving advantage.

A main aspect that makes FL stand out from other distributed learning frameworks is its specific consideration on the communication efficiency between the clients and the server. Particularly, the local stochastic gradient descent (SGD) algorithm [1] and the FedAvg algorithm [2] let clients perform multiple local SGD iterations on their own datasets before uploading the results to the parameter server for aggregation. Compared with earlier distributed learning algorithms such as distributed SGD [3] where local computation results must be uploaded to the server after every iteration, local SGD (or FedAvg) has a clear advantage in terms of the communication efficiency while still enjoying guaranteed convergence. This idea inspired many follow-up FL algorithms developed based on different optimization methods and makes FL the favorite choice in communication-constrained learning settings where either the bandwidth between the clients and the server is limited or the communication pattern is random and sporadic.

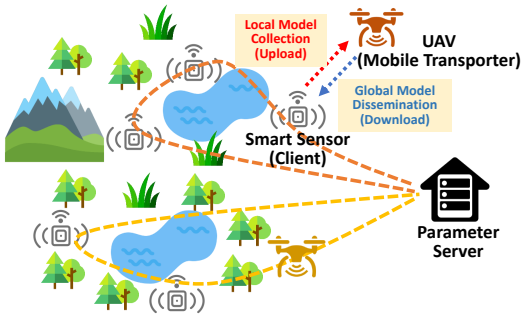

Fig. 1. Illustration of FedEx applied to smart sensing in remote areas with no communication infrastructure.

Despite the differences in the adopted optimization methods and the focused settings, nearly all existing works implicitly assumed that the clients can *directly* communicate with the server. In the asynchronous FL category, the communication assumption is relaxed in some works [4], [5] so that clients may have random and sporadic communication patterns with the server, but clients can still directly communicate with the server, albeit less regularly. However, in many real-world systems, clients may not be able to directly communicate with the server at all due to the lack of a proper communication infrastructure. It is thus imperative to understand whether FL can still work without direct server-client communications and

J. Bian and J. Xu are supported in part by NSF under grants 2006630, 2033681, 2029858 and 2044991. C. Shen is supported in part by NSF under grants 2033671 and 2143559.

how to optimize the FL algorithms in these settings.

In this paper, we propose a new FL framework, called FedEx (Federated Learning via Model Express Delivery), for the considered FL system with no direct server-client communications. To address the no direct communication challenge, FedEx devises an active mobility mechanism, which utilizes mobile transporters (e.g., Unmanned Aerial Vehicles / UAVs) to establish *indirect* communication channels between the server and the clients to facilitate the model information exchange. In other words, the mobile transporters serve as an intermediary between the server and the clients to disseminate global models and collect local model updates, analogous to delivery trucks in a traditional parcel express delivery system. Fig. 1 illustrates the application of FedEx to smart sensing in remote areas with no communication infrastructure.

However, the indirect communication also brings significant new challenges to the convergence analysis and optimization of FedEx. First, in both the global model dissemination phase and the local model collection phase, delay is inevitably introduced by the indirect communication as it takes time for the mobile transporters to move from one location to another. It is unclear whether FedEx can still converge under this delay and if so, how fast. Second, depending on the transporter scheduling policy, learning can be either synchronized or asynchronized at the transporter level, thereby further complicating the convergence analysis of FedEx. Third, clients experience heterogeneous delays depending on their locations and the routes chosen by the transporters. Thus, the performance of FedEx is also contingent on how clients are assigned to the transporters and how the transporters design their routes. We summarize the main contributions of this paper below.

- To our best knowledge, we propose the first FL framework via indirect server-client communications. Two algorithms, coined FedEx-Sync and FedEx-Async, are proposed depending on whether the transporters synchronize their tours among the assigned clients.
- We prove the convergence of both FedEx-Sync and FedEx-Async using a virtual sequence technique.
- Based on the specific forms of the convergence bounds, a bi-level optimization algorithm is proposed to solve the joint client assignment and route design problem.
- The experiments results using two public datasets validate the efficacy of FedEx and are consistent with our theory.

II. Problem Formulation

We consider an FL system with one parameter server and N clients, which are distributed over a large area *without* direct communication capabilities. In other words, no client has a direct communication channel with the server and any pair of clients do not communicate with each other. For notation simplicity, we index the server as 0 and the clients by the set $\mathcal{N} = \{1, 2, \cdots, N\}$. The server and the clients are deployed in fixed locations and do not move. Given the location coordinates of the server and the clients, one can easily calculate the (symmetric) distance matrix $D \in \mathbb{R}^{(N+1)\times(N+1)}$ that describes the distance between any two devices. Specifically, $D_{0i} = D_{i0}$ is the distance between the server and client i and $D_{ij} = D_{ji}$ is the distance between clients i and j.

Each client i has a dataset and the clients must together train a machine learning model under the coordination of the server by solving the following distributed optimization problem:

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

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

To train such a machine learning model, conventional FL frameworks require periodic/non-periodic communications between the clients and the server. However, in our considered setting, all direct communication channels between the clients and the server are absent and hence, these FL frameworks all fail to work. In the next section, we propose a novel FL framework to enable FL in such extreme communication scenarios (i.e., no direct communications).

III. FedEx: FL via Model Express Delivery

To address the lack of direct communications between the clients and the server, our idea is to leverage mobile transporters (e.g., UAVs) to build indirect communication channels between the server and the clients. These mobile transporters transport global/local models between the server and the clients, just like delivery trucks transport parcels between the warehouse and the customers. We call this new FL framework FedEx, short for FL via Model Express Delivery.

Consider that K mobile transporters can be used in FedEx. The specific value of K depends on the available system resources and we consider it as given in this paper. Without loss of generality, we assume that all clients have the same computing speed and all transporters have the same moving speed. We discretize time into slots, indexed by $t = 0, 1, 2, \cdots$, where each time slot corresponds to the duration of completing one local training step by a client. Furthermore, with a slight abuse of notation, we let the matrix D represent the time (in terms of time slots) needed for the mobile transporter to travel from one location to another instead of the distance to simplify our notations.

FedEx works by first partitioning the clients into K non-overlapping subsets and assigning each subset to one mobile transporter. Let $\mathcal{R}_k \subset \mathcal{N}$ represent the subset of clients that are covered by transporter k. We have $\mathcal{R}_k \cap \mathcal{R}_{k'} = \emptyset, \forall k \neq k'$ and $\cup_{k=1}^{K} \mathcal{R}_k = \mathcal{N}$. Moreover, let $R_k = |\mathcal{R}_k|$ be the number of clients in subset $\mathcal{R}_k$. For each transporter k, it determines a round-trip tour among the server and the clients in $\mathcal{R}_k$. Let $z_{ij}, \forall i, j \in \mathcal{R}_k \cup \{0\}$ be a binary variable indicating whether a

path from device i to device j is included in the tour, then the round-trip time (RTT) can be easily calculated as

$$\Delta_k = \sum_{i=0}^{N} \sum_{j\neq i, j=0}^{N} D_{ij} z_{ij}. \tag{2}$$

To make sure that the tour covers all devices exactly once, we have the constraints that each device has exactly one incoming path and one outgoing path, which can be expressed as $2(N+1)$ linear equations:

$$\sum_{i=0, i\neq j}^{N} z_{ij} = 1, \sum_{j=0, j\neq i}^{n} z_{ij} = 1, \forall j = 0, \cdots, N. \tag{3}$$

Finding the shortest tour for a given set of clients $\mathcal{R}_k$ is essentially a *travelling salesman problem* and we defer the optimization of client assignment among the transporters to Section V after understanding the convergence behavior of FedEx. For now, we treat $\mathcal{R}_k, \forall k = 1, \cdots, K$ as already decided along with the corresponding tour RTT Δ_k.

A. FedEx-Sync

In the synchronized version of FedEx, namely FedEx-Sync, the transporters depart from the server *at the same time* every time they start a new tour among their assigned clients. Because the transporters have different tour RTTs, the ones with shorter RTT need to wait for the others with longer RTT to come back to start the next tour. Therefore, FedEx-Sync is naturally composed of synchornized learning rounds, with each round having $\Delta \triangleq \max_k \Delta_k$ time slots. In each round (denote the first slot of this round as t_0), the following events occur.

- At the beginning of each round, each transporter downloads the current global model x^{t_0} from the server. The transporters then start a tour among their assigned clients according to the pre-determined client visiting order.
- When a transporter (say transporter k) meets a client (say client i) at time $t > t_0$, client i downloads the global model, i.e., x^{t_0}, that transporter k currently carries. Then the transporter leaves and client i uses $x_k^t = x^{t_0}$ as the initial model to train a new local model using its own local dataset until the next time it meets the transporter. Because the transporter takes Δ time slots to revisit client i, the local training will last Δ time slots. The local training uses a mini-batch SGD method:

  $$x_i^{s+1} = x_i^s - \eta g_i^s, \forall s = t, \cdots, t+\Delta-1, \tag{4}$$

  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. Let $m_i^t \in \mathbb{R}^d$ be the cumulative local updates (CLU) of client i at time s since its last meeting with the transporter, which is updated recursively as follows

  $$m_i^s = \sum_{s'=t}^{s-1} \eta g_i^s, \forall s = t, \cdots, t+\Delta. \tag{5}$$

- When a transporter (say transporter k) meets a client (say client i) at time $t > t_0$, client i also uploads its current CLU to transporter k. Note, however, that this CLU is obtained based on the global model from the *previous* round, i.e., $x^{t_0-\Delta}$. Transporter k maintains an aggregated CLU u_k^t during the current tour to save storage space and updates it whenever a new client CLU is received according to

  $$u_k^t = u_k^{t-1} + m_i^t. \tag{6}$$

- When the transporter returns to the server, the aggregated CLU is used to update the global model. In FedEx-Sync, the global model is updated synchronously at the end of each round as follows

  $$x^{t_0+\Delta} = x^{t_0+\Delta-1} - \frac{1}{N} \sum_{k=1}^{K} u_k^{t_0+\Delta-1}. \tag{7}$$

B. FedEx-Async

The synchronization in FedEx-Sync is achieved by asking faster transporters to wait for slower transporters. This, however, introduces extra delays for faster transporters. In the case where the slowest transporter takes a tour with a very large RTT, then all the other transporters will have to wait for a long time before starting their next tour. In FedEx-Async, we remove such waiting time by letting the transporter start a new tour immediately after finishing the previous tour. In this way, more clients will be able to perform more frequent global/local model exchanges with the server. FedEx-Async share many similarities with FedEx-Sync and the biggest difference is that each transporter will have *individualized* learning rounds not necessarily synchronized with others. For transporter k, its learning round lasts Δ_k time slots and the following events occur in each round (denote the first slot as t_0).

- At the beginning of each round, transporter k downloads the current model x^{t_0} from the server. Then it starts its tour among the assigned clients.
- When transporter k meets client i at time $t > t_0$, client i downloads x^{t_0} from transporter k and uses $x_k^t = x^{t_0}$ as the initial model to train a new local model until the next time it meets the transporter. Different from FedEx-Syncs, the local training will last Δ_k time slots, which are different across transporters.
- When transporter k meets client i at time $t > t_0$, client i also uploads its current CLU, which is obtained based on the previous round global model $x^{t_0-\Delta_k}$, to transporter k. Transporter k then updates its aggregated CLU u_k^t.
- When the transporter returns to the server, the global model is updated as follows

  $$x^{t_0+\Delta_k} = x^{t_0+\Delta_k-1} - \frac{1}{N} u_k^{t_0+\Delta_k-1}. \tag{8}$$

  Again, this is different from FedEx-Sync since the server does not have to wait for all transporters to update the global model. Note that it is possible that multiple transporters can return to the server in the same time slot (say t). In this case, the global model update rule is changed to

  $$x^{t_0+\Delta_k} = x^{t_0+\Delta_k-1} - \frac{1}{N} \sum_{k'\in\mathcal{S}^{t_0+\Delta_k}} u_{k'}^{t_0+\Delta_k-1}, \tag{9}$$

where $\mathcal{S}^{t_0+\Delta_k-1}$ is the set of clients that return to the server at time slot $t_0 + \Delta_k - 1$.

IV. Convergence Analysis

In this section, we analyze the convergence of FedEx. Because FedEx-Sync can be considered as a special case of FedEx-Async where all $\Delta_k, \forall k$ take the same value, we will focus on the convergence analysis of FedEx-Async.

A. Aligning Client Training

Before analyzing the convergence of FedEx, we first describe an equivalent view of FedEx that aligns the local training of clients in the same subset. Consider a learning round of transporter k that carries the global model x^{t_0}. Because of the different locations of clients in $\mathcal{R}_k$, the clients receive x^{t_0} and start their new round of local training at different time slots. Once they finish the current round of local training, their CLUs based on x^{t_0} will be uploaded via the transporter to the server at time slot $t_0 + 2\Delta_k$. At that moment, the global model gets an update using these CLUs.

The unaligned local training of clients, even if covered by the same transporter, would create a major challenge for the convergence analysis of FedEx. Fortunately, there is an equivalent (but imaginary) client training procedure that produces exactly the same global model sequence. Specifically, imagine that clients in $\mathcal{R}_k$ receive the global model x^{t_0} immediately at time slot t_0 and perform their local training for Δ_k time slots. Then their CLUs are delayed one round to be uploaded to the server. That is, at time slot $t_0 + 2\Delta_k$, the global model gets an update. It is clear that the global model update is not affected at all by this change but the local training among clients in the same subset $\mathcal{R}_k$ is now perfectly aligned. Since we are interested in the convergence of the global model, we consider the equivalent aligned client training procedure in our convergence proof. Where the local training of the clients is aligned while the global model evolution is unaffected. Essentially, the alignment moves the download delay to the upload phase, but the total delay remains the same. With this change, FedEx-Sync becomes a familiar synchronous FL algorithm but with one round CLU upload delay. In addition to the CLU upload delay, Fed-Async still features asynchronized learning across the client subsets.

B. Assumptions

Our convergence analysis will utilize the following standard 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 $\forall i = 1, \cdots, N$.

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 $\forall i = 1, \cdots, N$.

Assumption 3 (Bounded Gradient)**.** There exists a constant $G > 0$ such that $\mathbb{E}\|\nabla F_i(x, \zeta)\|^2 \leq G^2$, $\forall x \in \mathbb{R}^d$ and $\forall i = 1, \cdots, N$.

Assumption 4 (Bounded Variance)**.** There exists a constant $\sigma > 0$ such that $\mathbb{E}_\zeta\|\nabla F_i(x, \zeta) - \nabla f_i(x)\|^2 \leq \sigma^2$, $\forall x \in \mathbb{R}^d$ and $\forall i = 1, \cdots, N$.

C. Convergence Bound

Our convergence analysis relies on understanding the relationship between and the evolution of two sequences of the global model. The *real sequence* of the global model is the actual global models maintained at the server over time, which can be calculated as follows according to FedEx:

$$x^t = x^0 - \frac{1}{N}\sum_{i=1}^{N}\sum_{s=0}^{\phi_i(t)} \eta g_i^s, \forall t, \tag{10}$$

where we define $\phi_i(t)$ as the time slot up to when all corresponding gradients of client i have been received at time t. In other words, at time slot t, the server has received gradients $g_i^0, \cdots, g_i^{\phi_i(t)}$ from client i (via the transporter). In FedEx-Sync, all clients have the same indirect communication patterns with the server and hence $\phi_i(t) = \phi_j(t), \forall i, j \in \mathcal{N}$. In FedEx-Async, clients belonging to the same transporter have the same indirect communication patterns with the server and hence, $\phi_i(t) = \phi_j(t), \forall i, j \in \mathcal{R}_k, \forall k$.

The *virtual sequence* of the global model is defined in the imaginary case where all client gradients are uploaded to the server immediately after they have been calculated. Similar virtual sequences have been utilized in [6]. However, our convergence proof is tailored to the specific problems in our paper and different than all prior works. Specifically, the virtual sequence is defined as

$$v^t = x^0 - \frac{1}{N}\sum_{i=1}^{N}\sum_{s=0}^{t-1} \eta g_i^s, \forall t. \tag{11}$$

Clearly, there is a discrepancy between the real sequence and the virtual sequence due to the delayed upload of the client gradients. For FedEx-Sync, this delay is bounded by

$$(t-1) - \phi_i(t) \leq 2\Delta, \forall i \in \mathcal{N}. \tag{12}$$

For FedEx-Async, the delay is bounded by

$$(t-1) - \phi_i(t) \leq 2\Delta_k, \forall i \in \mathcal{R}_k, \forall k. \tag{13}$$

Lemma 1. The difference between the real global model and the virtual global model can be bounded as follows

$$\mathbb{E}\|v^t - x^t\|^2 \leq \frac{4\eta^2 G^2}{N}\sum_{k=1}^{K} R_k \Delta_k^2. \tag{14}$$

The average difference between all clients' local models and the virtual global model is bounded as follows

$$\frac{1}{N}\sum_{i=1}^{N}\mathbb{E}\|v^t - x_i^t\|^2 \leq \frac{18\eta^2 G^2}{N}\sum_{k=1}^{K} R_k \Delta_k^2. \tag{15}$$

Theorem 1. By setting the learning rate $0 < \eta \leq 1/L$, we have

$$\frac{1}{T}\sum_{t=0}^{T-1}\mathbb{E}\|\nabla f(x^t)\|^2$$

$$\leq \frac{4}{\eta T}(f(x^0) - f^*) + \frac{44\eta^2 G^2 L^2}{N}\sum_{k=1}^{K} R_k \Delta_k^2 + \frac{2L\eta\sigma^2}{N}. \quad (16)$$

Remark 1. For $T \geq N^3$, by setting the learning rate as $\eta = \frac{\sqrt{N}}{L\sqrt{T}}$, the convergence bound recovers the same $O(\frac{1}{\sqrt{NT}})$ convergence rate of the classical synchronous FL [7].

Remark 2. For FedEx-Sync, the convergence bound can be tightened a little bit because the real sequence and the virtual sequence periodically coincide with each other. In addition, because the effective RTT of all transporters is $\max_k \Delta_k$, the convergence bound reduces to

$$\frac{2}{\eta T}(f(x^0) - f^*) + 18\eta^2 G^2 L^2 (\max_k \Delta_k)^2 + \frac{L\eta\sigma^2}{N}. \quad (17)$$

V. Client Assignment and Route Design

In this section, we study the joint client assignment and route design problem to optimize the convergence bound of FedEx.

A. Problem Formulation

We consider a typical setting where the number of clients is much larger than the number of transporters, i.e., $N \gg K$, due to the limited transporter availability and potentially massive deployment of IoT devices. Let $a_i \in \{1, \cdots, K\}$ be the assignment variable of client i, indicating which transporter it is assigned to. We also collect the assignment variables of all clients in $\boldsymbol{a} = (a_1, \cdots, a_N)$. Clearly, $\mathcal{R}_k = \{i : a_i = k\}$.

Given the assigned clients $\mathcal{R}_k$ for each transporter k, we can design a route to minimize the RTT. Let $\Delta_k(\mathcal{R}_k)$ be the minimum RTT given a set of client $\mathcal{R}_k$. Alternatively, $\Delta_k(\mathcal{R}_k)$ can also be written as $\Delta_k(\boldsymbol{a})$ since $\mathcal{R}_k$ is determined by $\boldsymbol{a}$. Client assignment is to solve the following optimization problems

$$\text{FedEx-Sync}: \min_{\boldsymbol{a}} \max_k \Delta_k(\boldsymbol{a}), \quad (18)$$

$$\text{FedEx-Async}: \min_{\boldsymbol{a}} \sum_k R_k(\boldsymbol{a})\Delta_k^2(\boldsymbol{a}). \quad (19)$$

The above problem is a difficult combinatorial optimization problem. Next, we propose a new algorithm to solve this problem.

B. Bi-level Optimization

We develop a bi-level optimization algorithm, called CARD (short for Client Assignment and Route Design).

1) Inner-level optimization: The inner-level optimization is to solve the minimum RTT given a set of client $\mathcal{R}_k$, for each transporter k, namely computing $\Delta_k(\boldsymbol{a})$ for a given $\boldsymbol{a}$. This is a classical traveling salesman problem (TSP) [8], [9], [10]. Considering the high complexity of dynamic programming (i.e., $O(2^n n^2)$ where n is the number of nodes), we use a heuristic algorithm 2-OPT [11], which has a time complexity of $O(n^2)$, to compute $\Delta_k(\boldsymbol{a})$ for a given $\boldsymbol{a}$ in our implementation.

2) Outer-level optimization: To solve the outer-level optimization problem to determine the optimal client assignment, we resort to Gibbs Sampling. For notation simplicity, we use a unified cost function $C(\boldsymbol{a})$, which equals $\max_k \Delta_k(\boldsymbol{a})$ for FedEx-Sync and $\sum_k R_k(\boldsymbol{a})\Delta_k^2(\boldsymbol{a})$ for FedEx-Async. The CARD algorithm visits each client according to a pre-defined sequence, generates a probability distribution of its assignment decision while holding other clients' assignment decision unchanged, and samples a new assignment decision according to this distribution. Repeating this process for sufficiently many iterations ensures that the assignment converges to the optimal solution with high probability.

VI. Experiments

In this section, we evaluate the performance of FedEx using a simulated network environment and standard public datasets.

A. Experiment Setup

Network. We simulate a network with no direct communications where one parameter server and 40 clients are distributed over an area as shown in Fig. 2. The whole area is first divided into 10 blocks of equal size (i.e., 4 units $\times$ 10 units), and 4 clients are randomly distributed in each block. We simulate $K = 4$ mobile transporters in most of our experiments, which have the same moving speed of 4 units per time slot.

DNN Model and Dataset. We conduct the FL experiments on two public datasets, i.e., FMNIST [12] and SVHN [13]. For both datasets, we utilize LeNet [14] as the backbone model.

Dataset Splitting. For the FMNIST experiments, each client possesses 60 training data samples. For the SVHN experiments, each client possesses 800 training data samples. Two data distributions are simulated.

- IID: The clients' data distributions are i.i.d.
- Non-IID: We use the Dirichlet method to create non-i.i.d. datasets, which is widely applied in FL research, e.g., [15]. We use a Dirichelet distribution with parameter 0.3 in the FMNIST experiments and 0.5 in the SVHN experiments.

Transporter Routes. We evaluate different transporter route designs that uses different objective functions to solve the client assignment problem:

- Min-Max: $\max_k \Delta_k(\boldsymbol{a})$.
- Sum-of-Weighted-Squared (SWS): $\sum_k R_k(\boldsymbol{a})\Delta_k^2(\boldsymbol{a})$.
- Shortest-Total: $\sum_k \Delta_k(\boldsymbol{a})$.

B. Results

FedEx-Sync v.s. FedEx-Async under i.i.d. data. We first compare the performance of FedEx-Sync (with Min-Max routes) and FedEx-Async (with SWS routes) under i.i.d. data. Fig. 3 and Fig. 4 demonstrate their convergence curves on FMNIST and SVHN, respectively. As can be seen, under i.i.d. data, both algorithms converge but FedEx-Async outperforms FedEx-Sync. This makes sense since FedEx-Sync spends extra time in waiting for the slowest mobile transporters.

FedEx-Sync v.s. FedEx-Async under non-i.i.d. data. Next, we compare FedEx-Sync and FedEx-Async non-i.i.d. data. Fig.

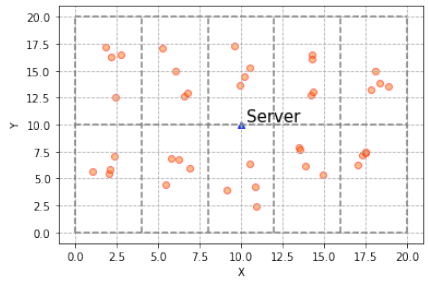

Fig. 2. Network Illustration

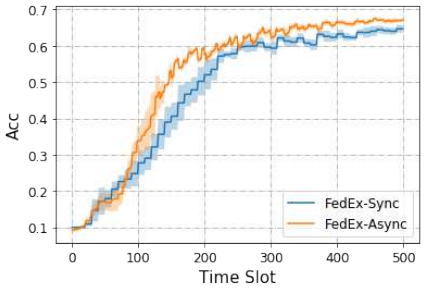

Fig. 3. FedEx under i.i.d data on FMNIST

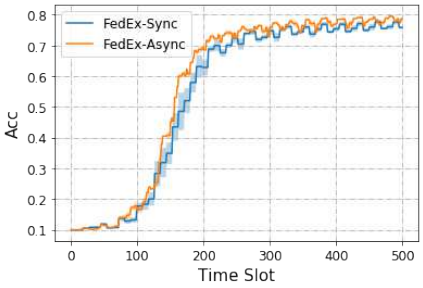

Fig. 4. FedEx under i.i.d data on SVHN

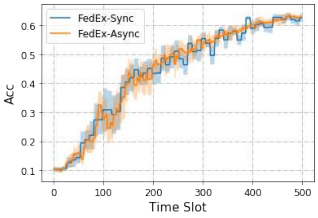

Fig. 5. non-i.i.d on FMNIST

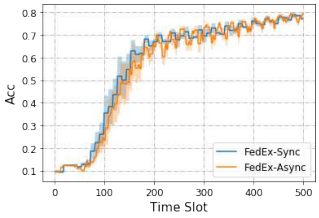

Fig. 6. non-i.i.d on SVHN

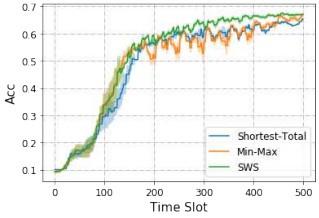

Fig. 7. Impact of Routes (Async)

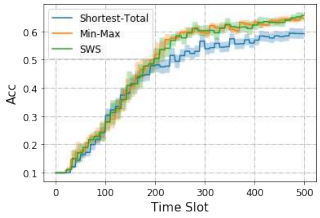

Fig. 8. Impact of Routes (Sync)

5 and Fig. 6 show that FedEx-Sync performs similarly to FedEx-Async. Although individual clients' data can be very non-i.i.d., the overall client data of a transporter can still be similar to the entire data distribution when there are sufficiently many clients on the transporter's route.

Impact of Routes. We empirically investigate the impact of different transporter route designs. Fig. 7 compares the convergence of FedEx-Async under different route designs. The result is consistent with our theoretical analysis in Theorem 1 that the SWS design achieves the best convergence performance. Fig. 8 shows the results for FedEx-Sync. In this case, the longest individual RTT obtained in SWS is the same as that obtained in Min-Max, and hence SWS is also another solution of Min-Max. Therefore, Min-Max routes and SWS routes achieve similar convergence performance and outperform Shortest-Total, in accordance to the theoretical analysis in Theorem 1.

VII. Conclusion

In this paper, we have developed a new FL framework via indirect server-client communications to support distributed machine learning in scenarios without a communication infrastructure. Two novel algorithms have been proposed that utilize mobile transporters to disseminate global models and collect local models via device-to-device communications. We have carried out a novel convergence analysis of these algorithms under arbitrary transporter routes, for non-convex loss functions and non-i.i.d. data distributions. The result offers a principled guideline for the joint client assignment and route design.

References

[1] S. U. Stich, "Local sgd converges fast and communicates little," *arXiv preprint arXiv:1805.09767*, 2018.

[2] B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. y Arcas, "Communication-efficient learning of deep networks from decentralized data," in *Artificial intelligence and statistics*. PMLR, 2017, pp. 1273–1282.

[3] M. F. Balcan, A. Blum, S. Fine, and Y. Mansour, "Distributed learning, communication complexity and privacy," in *Conference on Learning Theory*. JMLR Workshop and Conference Proceedings, 2012, pp. 26–1.

[4] D. Avdiukhin and S. Kasiviswanathan, "Federated learning under arbitrary communication patterns," in *International Conference on Machine Learning*. PMLR, 2021, pp. 425–435.

[5] D. Basu, D. Data, C. Karakus, and S. Diggavi, "Qsparse-local-sgd: Distributed sgd with quantization, sparsification and local computations," *Advances in Neural Information Processing Systems*, vol. 32, 2019.

[6] K. Yuan, Q. Ling, and W. Yin, "On the convergence of decentralized gradient descent," *SIAM Journal on Optimization*, vol. 26, no. 3, pp. 1835–1854, 2016.

[7] 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 *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 33, no. 01, 2019, pp. 5693–5700.

[8] S. Lin and B. W. Kernighan, "An effective heuristic algorithm for the traveling-salesman problem," *Operations research*, vol. 21, no. 2, pp. 498–516, 1973.

[9] M. M. Flood, "The traveling-salesman problem," *Operations research*, vol. 4, no. 1, pp. 61–75, 1956.

[10] M. Jünger, G. Reinelt, and G. Rinaldi, "The traveling salesman problem," *Handbooks in operations research and management science*, vol. 7, pp. 225–330, 1995.

[11] G. A. Croes, "A method for solving traveling-salesman problems," *Operations research*, vol. 6, no. 6, pp. 791–812, 1958.

[12] H. Xiao, K. Rasul, and R. Vollgraf, "Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms," *arXiv preprint arXiv:1708.07747*, 2017.

[13] Y. Netzer, T. Wang, A. Coates, A. Bissacco, B. Wu, and A. Y. Ng, "Reading digits in natural images with unsupervised feature learning," 2011.

[14] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.

[15] H.-Y. Chen and W.-L. Chao, "Fedbe: Making bayesian model ensemble applicable to federated learning," *arXiv preprint arXiv:2009.01974*, 2020.