In-network Computation for Large-scale Federated Learning over Wireless Edge Networks

SCHOLARONE™
Manuscripts

In-network Computation for Large-scale Federated Learning over Wireless Edge Networks

Thinh Quang Dinh, Diep N. Nguyen, Dinh Thai Hoang, Pham Tran Vu, and Eryk Dutkiewicz

✦

Abstract—Most conventional Federated Learning (FL) models are using a star network topology where all users aggregate their local models at a single server (e.g., a cloud server). That causes significant overhead in terms of both communications and computing at the server, delaying the training process, especially for large scale FL systems with straggling nodes. This paper proposes a novel edge network architecture that enables decentralizing the model aggregation process at the server, thereby significantly reducing the training delay for the whole FL network. Specifically, we design a highly-effective in-network computation framework (INC) consisting of a user scheduling mechanism, an in-network aggregation process (INA) which is designed for both primal- and primal-dual methods in distributed machine learning problems, and a network routing algorithm with theoretical performance bounds. The in-network aggregation process, which is implemented at edge nodes and cloud node, can adapt two typical methods to allow edge networks to effectively solve the distributed machine learning problems. Under the proposed INA, we then formulate a joint routing and resource optimization problem, aiming to minimize the aggregation latency. The problem turns out to be NP-hard, and thus we propose a polynomial time routing algorithm which can achieve near optimal performance with a theoretical bound. Simulation results showed that the proposed algorithm can achieve more than 99% of the optimal solution and reduce the FL training latency, up to 5.6 times w.r.t other baselines. The proposed INC framework can not only help reduce the FL training latency but also significantly decrease cloud's traffic and computing overhead. By embedding the computing/aggregation tasks at the edge nodes and leveraging the multi-layer edge-network architecture, the INC framework can liberate FL from the star topology to enable large-scale FL.

Index Terms—Mobile Edge Computing, Federated Learning, In-network Computation, Large-scale Distributed Learning

1 INTRODUCTION

Machine learning that enables intelligent systems has emerged as a key technology benefiting many aspects of modern society [2]. Currently, most data-driven systems collect data from local devices and then train the data at centralized cloud servers. However, user privacy, applications' latency and network overhead traffic are major concerns of this centralized solution. To address these issues, collaborative learning schemes have received considerable attention where mobile users (MUs) can build and share their machine learning models instead of sending their raw

data to a centralized data center [3]. Out of these distributed learning schemes, Federated Learning (FL) has recently emerged as a promising candidate [4]. In FL, each MU iteratively computes its local model using its local training data. This local model is then sent to a cloud server and aggregated together with other local models, contributed by other MUs to produce a global model. The global model is then sent back to all the MUs and then these MUs will use it to update their new local models accordingly. This process is repeated until it is converged or an accuracy level of the learning model is reached. Since no user's raw data is exchanged, FL facilitates machine learning in many aspects from data storage, training to data acquisition, and privacy preservation.

Although communications overhead is significantly saved by sharing local models instead of local raw data, communication cost is still a major bottleneck of FL. This is because to aggregate the global model, the centralized server generally needs to connect with a huge number of users for all their updates via star network topology [4]–[10]. Moreover, for advanced deep learning models [11], [12] which contain a vast amount of parameters, the size of models exchanged in networks could be very large. For example, the BERT model [11] in Natural Language Processing area is up to 1.3 GB, while VGG16 [12] in Computer Vision is more than 500 MB. Thus, with VGG16, it can cost about 500 TB for each user until the global model is converged [13]. Consequently, high communication cost can lead to (a) high transmission latency and (b) high traffic at the cloud server. To address the issues, new network architectures are needed be investigated.

High computation cost is another challenge of conventional FL models using the star network topology. Let us assume that vectors or matrices are used to store users' models for each FL iteration. With VGG16, vectors or matrices contain more than 138 million elements. As a result, when the number of users grows, aggregation operations at the centralized server can be very computationally costly [14], [15] in terms of processing and memory resources. For that, novel network topologies should be considered to enable large-scale FL systems.

Edge Computing (EC), an emerging distributed network architecture [3] that aims to bring cloud functions and resources (including computing, storage and networking

capacities) closer to end users can be a promising solution to large scale FL. Since edge nodes (EN) possess both computation and communication capacities, edge networks can effectively support the cloud server to decentralize its communication and computing burden of model aggregation in very large scale FL networks. To exploit potential benefits of edge networks for FL models, it is necessary to develop a distributed in-network aggregation (INA) functionality that can be implemented at edge nodes of EC to liberate FL from the conventional start network topology.

In-network computation (INC) is a process of gathering and routing information through a multi-hop network, then processing data at intermediate nodes with the objective of optimizing the resource consumption [16]. Currently, the concept is well-studied for big data analytics such as MapReduce [17], Pregel [18] and DryadLINQ [19] for distributed data clusters. Three basic components of an in-network computation solution are: suitable networking protocols, effective aggregation functions and efficient ways of representing data [16]. However, INC frameworks for FL is under-studied. Moreover, edge nodes will be densely deployed in future networks [20], one can exploit this diversity by considering a scenario where an MU can associate with multiple nearby edge nodes, instead of only a single edge node. Under such a scenario, the problems of network routing and network resource allocation become more challenging.

Regarding machine learning models, e.g., classification or regression, they can be learned by solving regularized loss minimization problems in two ways: (a) by using primal methods such as Stochastic Gradient Descent (SGD) [4] or (2) by using the primal-dual methods such as Stochastic Dual Coordinate Ascent (SDCA) [21], [22]. Most works in FL are limited in one method [4]–[10], [23]–[27]. From experimental and theoretical results [9], [28], it is noted that the primal methods cost less resources, e.g. computing and storage, and can faster reach a moderate accuracy level, while the primal-dual method can provide better convergence and accuracy in the long run. As a result, to achieve high accuracy with faster convergence, instead of using vanilla SDCA, authors in [28] implemented SGD in initial rounds before executing SDCA. Therefore, a general in-network computation protocol, which can adapt to both the FL methods, is required.

Given the above, this paper proposes a novel edge network architecture and its associated in-network computation framework that enable decentralizing the model aggregation process at the server, thereby significantly reducing the training delay for the whole FL network. To this end, a highly-effective in-network computation protocol is proposed with three main components: a user scheduling mechanism, a novel in-network aggregation process which is designed for both primal- and primal-dual methods in distributed machine learning problems, and a network routing algorithm with theoretical performance bounds. The major contributions of this work are summarized as follows.

1) Propose a novel edge network architecture to decentralize the communication and computing burden of cloud node in FL aiming at minimizing the whole network latency. Such a network architecture is enabled by a novel in-network computation protocol consisting of an effective in-network aggregation process, a routing algorithm and a user scheduling.

2) Design a new user scheduling mechanism which allows a group of users to be able to send their models in advance instead of waiting for the last user finishing its local update. Based on the power law distribution of user data, this mechanism could mitigate straggler effect with a theoretical performance bound.

3) Develop a novel in-network aggregation process that instructs on how data are processed at edge nodes and cloud node to decentralize the global model aggregation process. The INA is designed for both primal- and dual-primal solutions to a given FL network.

4) Optimize the joint routing and resource allocation for MUs and ENs under the proposed architecture, thereby minimizing FL training time of the whole network. We show that the resulting mixed integer non-linear programming problem is NP-hard then propose an effective solution based on randomized rounding techniques. Simulations show that the proposed solution can achieve more than 99% of the optimal solution.

The rest of the paper is organized as follows. In Section 2, we present the related works. The system model is introduced in Section 3. Then, the user scheduling mechanism is proposed in Section 4. Section 5 presents our in-network aggregation process. Next, we formulate the network routing and resource allocation frameworks in Section 6, and propose our solution in Section 7. We then present the numerical results in Section 8 and final conclusions are drawn in Section 9.

2 RELATED WORKS

Although sharing raw local data is not required, high communication cost still remains a major obstacle in FL systems, especially given its canonical star network topology. Solutions to this issue can be divided into several directions such as compressing the local models using quantization, e.g., [5], [6], skipping unnecessary gradient calculations or global updates [4], [7], and selecting a promising user subset of each global update [8]–[10]. For compressing local models, MPEG-7 part 17 [5] has become a universal neural network compression standard, which is not only limited in the domain of multimedia. However, among these works [4]–[10], authors usually assumed that MUs can be directly connected to a single server. This star network topology assumption is impractical since it does not reflect the hierarchical structure of large-scale wireless networks. Moreover, for a large number of users, such a star topology cannot help scale up the FL system.

Edge computing is a potential infrastructure which can help to address the high communication cost of FL. Even though there are many works where edge networks are complementary to the cloud [29]–[32], the capacities of edge networks are under explored in existing FL works [4]–[10], [23]–[26]. To decentralize and redistribute the model aggregation process of FL at cloud server to edge nodes in edge networks, this article leverages the "in-network computing" concept. Although such a concept has been well-studied for traditional machine learning paradigms where users' raw data are collected and stored at big data clusters [17]–[19],

INC for distributed learning paradigms, e.g., FL, especially with the aid of edge computing, has not been visited.

There have been early works proposing decentralizing aggregation processes for different network architectures such hierarchical topology [27], [33], ring topology [34], and random graph [35]. However, in those works, the proposed decentralized aggregation processes can be considered as variants of either FedAvg or CoCoA. In our paper, the proposed in-network aggregation process, a component of our INC solution, can adapt to both primal and primal-dual methods. Moreover, existing works mostly focused on designing aggregation processes while network resource management and routing problems were not jointly considered. For example, in [27], users are assumed to connect with a single edge node without any alternative links. In reality with dense edge networks, MUs can associate to more than one edge nodes. Such a practical scenario calls for optimal network routing and resource allocation solution, aiming at minimizing the system latency. In this article, under the proposed INC, we formulate a joint routing and resource optimization problem, aiming to minimize the aggregation latency. The problem turns out to be NP-hard, and thus we design a polynomial time routing algorithm which can achieve near optimal performance with a proven theoretical bound. Last but not least, to address slow workers that stagnate the learning systems in these existing works [4]–[10], [23]–[27], [36], [37] (known as the straggler effect), we propose a user scheduling scheme that is the third component of our INC solution. In short, the key novelty of our work is to leverage edge networks to enable large-scale FL and liberate FL from its canonical start topology. All aforementioned works that aim to decentralize the FL process did not leverage edge networks nor address the associated challenges (e.g., routing, resource allocation, and the straggling effect).

3 SYSTEM MODEL

Let us consider a set of K mobile users MUs, denoted by $\mathcal{K} = \{1, \ldots, K\}$ with local datasets $\mathcal{D}_k = \{\mathbf{x}_i \in \mathbb{R}^d, y_i\}_{i=1}^{n_k}$ with n_k data points. These MUs participate in a distributed learning process by learning a shared global model from and sharing their local models (without sharing their raw data) with a cloud server. These MUs are co-located and supported by an MEC network consisting of a set of M edge nodes (ENs), denoted by $\mathcal{M} = \{1, \ldots, M\}$. The edge nodes, controlled by the MEC operator, can be small cell base stations with their own communications and computing capacities [30]. ENs are connected with a cloud server via a macro base station. Without loss of generality, we can consider the cloud server as a special EN, i.e., EN 0, co-located with the macro base station. MUs can connect to the cloud server either through ENs or directly with the macro base station. Each MU can be associated with one or more ENs [38]. For example, in Fig. 1, MU 4, which lies in the overlapping coverage areas of EN 1 and EN 2, can upload its local model to either one of these ENs.

3.1 Federated Learning

To construct and share the global model, the goal is to find the model parameter $\mathbf{w} \in \mathbb{R}^d$ which minimizes the follow-
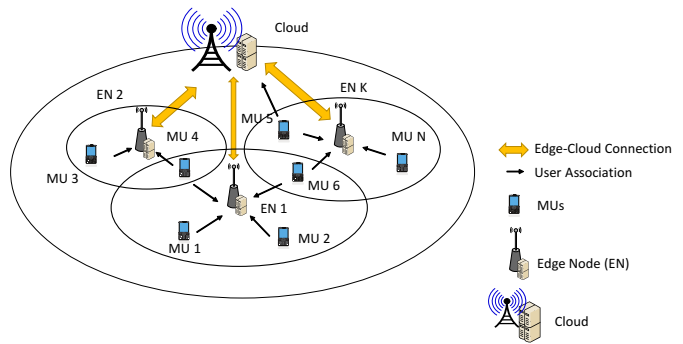

Fig. 1. FL-enabled Edge Computing Network Architecture.

ing global loss function in a distributed manner where data is locally stored at MUs:

$$\min_{\mathbf{w} \in \mathbb{R}^d} \left\{ P(\mathbf{w}) = \frac{1}{n} \sum_{i=1}^{n} l_i(\mathbf{x}_i^T \mathbf{w}) + \xi r(\mathbf{w}) \right\}, \quad (1)$$

where $n = \sum_{i=1}^{K} n_k$, ξ is the regularizing parameter and $r(\mathbf{w})$ is a deterministic penalty function [39]. Here, l_i is the loss function at data sample i. To solve the problem (1), one can either implement the (a) primal only method or the (b) primal-dual method. For example, FedAvg is a variant of the former one [4], while CoCoA is a variant of the latter [22]. Since both FedAvg and CoCoA are widely adopted, e.g., [4], [8]–[10], [22]–[26], in this paper, we consider both of them in our design.

For FedAvg [4], at each iteration t, the cloud broadcasts $\mathbf{w}^t$ to all MUs. Based on the latest global model, each MU learns its local parameters $\mathbf{w}_k^t$ according to the Stochastic Gradient Descent update rule aiming at minimizing the objective function $P(\mathbf{w})$ by only using local information and the parameter values in $\mathbf{w}_k^t$ [4]:

$$\mathbf{w}_k^t = \mathbf{w}_k^t - \eta(\nabla l_i(\mathbf{w}_k^t) + \nabla r(\mathbf{w}^t)). \quad (2)$$

Here, ∇ stands for partial derivative computation. The resulting local model updates are forwarded to the cloud for computing the new global model as follows:

$$\mathbf{w}^{t+1} = \frac{1}{n} \sum_{k=1}^{K} n_k \mathbf{w}_k^t. \quad (3)$$

For CoCoA, the goal is to find the model parameter $\mathbf{v} \in \mathbb{R}^d$ which is related to model $\mathbf{w}$ via the dual relationship. Following [21], using the Fenchel-Rockafeller duality, we rewrite the local dual optimization problem of (1) as follows:

$$\max_{\boldsymbol{\alpha} \in \mathbb{R}^n} \left\{ G(\boldsymbol{\alpha}) = -\sum_{i=1}^{n} \frac{l_i^*(-\alpha_i)}{n} - \xi r^*(\frac{1}{\xi n} \mathbf{X} \boldsymbol{\alpha}) \right\}, \quad (4)$$

where $\{\alpha_i\}_i^n \in \mathbb{R}$ represents the set of the dual variables, $\mathbf{X} = [\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n] \in \mathbb{R}^{d \times n}$ is the total data set, $l_i^*(\cdot)$ and $r^*(\cdot)$ are the convex conjugate functions of $l_i(\cdot)$ and $r(\cdot)$. Following [21], $l_i(\cdot)$ is assumed to be convex with $1/\mu$-smoothness, $r(\cdot)$ is assumed to be 1-strongly convex. At each iteration t, the cloud broadcasts $\mathbf{v}^t$ to all MUs. Based on Stochastic Dual Coordinated Ascent method, each MU

TABLE 1
Notation Used Throughout the Paper

Notation	Definition	Notation	Definition
k	index of MU	D	the data size of the global model
$\mathcal{K}$	set of users	W^d	the downlink data rate capacity
$\mathcal{D}_k$	local dataset of user k	T^d	the downlink latency for broadcasting global model
n_k	number of data points of user k	a_{km}	the aggregation routing variable
n	total number of data points of all users	r_{km}	the uplink data rate allocation variable
i	index of data point	$B_m^{\text{fr}}, B_m^{\text{bk}}$	the uplink fronthaul and backhaul data rate capacity of edge node m
m	index of EN	$T_m^{\text{u,fr}}$	the uplink fronthaul latency of edge node m and its users
$\mathcal{M}$	set of ENs	γ_m	the transmission latency between edge node m and the cloud node
t	index of time	T_m^{u}	the uplink latency of users associated with edge node m
$\mathbf{w}^t$	the global model parameter of FL in primal at learning round t	T^{u}	the total uplink latency of the whole network
$\mathbf{w}_k^t$	the local model parameter of FL at user k in primal at learning round t	t_k^{cp}	local training update at MU k
$\mathbf{v}^t$	the global model parameter of FL in primal-dual formulation at learning round t	T	the training time in one learning round
$\Delta\mathbf{v}_k^t$	the local model parameter of FL at user k in primal-dual formulation at learning round t	s	user scheduling scheme
ψ	the generalized global model of $\mathbf{w}$ and $\mathbf{v}$	$\mathcal{P}_1, \mathcal{P}_2$	user partition 1 and 2
$\boldsymbol{\alpha}$	the dual variables of FL in in primal-dual formulation	j	index of user partition
$P(\mathbf{w})$	the global loss function	z	the indicator if FL system is using primal-dual method
l_i	the loss function at data point i	ϕ_k^t	the local message that user k sends in learning round t
r	a deterministic penalty function	$\varphi_{m,j}^t$	the message that EN m sends to cloud for user partition $\mathcal{P}_j$ at learning round t
ξ	the regularizing parameter	$\boldsymbol{\lambda}_j^t$	the weight and the parameters of the aggregated model for user partition $\mathcal{P}_j$ at learning round t
$G(\boldsymbol{\alpha})$	the Fenchel-Rockafeller dual form of the global loss function $P(\mathbf{w})$		

updates a mapping of dual variables $\Delta\mathbf{v}_k^t$ as follows:

$$\Delta\mathbf{v}_k^t = \frac{1}{\xi n}\mathbf{X}_{[k]}\mathbf{h}_{[k]}^t, \qquad (5)$$

where $\mathbf{X}_{[k]}$ is denoted for the matrix consisting of only the columns corresponding to data samples $i \in \mathcal{D}_k$, padded with zeros in all other columns, $\mathbf{h}_{[k]}^t$ is denoted for the iterative solution obtained by solving the approximated problem of (4) using only data samples $i \in \mathcal{D}_k$, which is defined in [21]. The new global model is then aggregated at the cloud as follows:

$$\mathbf{v}^{t+1} = \mathbf{v}^t + \frac{1}{K}\sum_{k=1}^{K}\Delta\mathbf{v}_k^t. \qquad (6)$$

We summarize the general procedure of these two FL methods as follows:

1 *Global Model Broadcasting*: The cloud broadcasts the latest global model to the MUs, either $\mathbf{w}^t$ or $\mathbf{v}^t$.
2 *Local Model Updating*: Each MU performs local training following either (2) or (5).
3 *Global Model Aggregation*: Local models are then sent back to the cloud. Let ψ^t denote the generalized global model. The new value of global model is then computed at the cloud by following (7).
4. Steps 1-3 are repeated until convergence.

5

$$\boldsymbol{\psi}^{t+1} = \begin{cases} \frac{1}{n} \sum_{k=1}^{K} n_k \mathbf{w}_k^t & \text{if we choose the primal method,} \\ \boldsymbol{\psi}^t + \frac{1}{K} \sum_{k=1}^{K} \Delta \mathbf{v}_k^t & \text{if we choose the primal-dual method.} \end{cases} \tag{7}$$

3.2 Communication Model

We then introduce the communication model for multi-user access. For each FL iteration, the cloud node will select a set of users $\mathcal{K}^t$ at each iteration t. Here, how to select the best MUs at each learning round is out of the scope of this paper [1]. All users consent about their models' structure, such as a specific neural network design. Hence, let D denote the data size of model parameters:

$$D = (d+1) \times \text{Codeword length,} \tag{8}$$

where d is defined as the number of parameters mentioned as the length of $\mathbf{w}_k^t$ or $\Delta \mathbf{v}_k^t$.

3.2.1 Global Model Broadcasting

Since the downlink data rate capacity W^d of the cloud node is much larger than that of an edge node, all users will listen to the cloud node at the model broadcasting step. The latency for broadcasting the global model is $T^d = \frac{D}{W^d}$.

3.2.2 Global Model Aggregation

Let a_{km} be the aggregation routing variable, where

$$a_{km} = \begin{cases} 1 & \text{if MU } k\text{'s local model is directly sent to edge} \\ & \text{node } m, \forall m \in \mathcal{M}, \\ 0 & \text{otherwise.} \end{cases} \tag{9}$$

Let $\mathbf{a}_m = [a_{1m}, a_{2m}, \cdots, a_{Km}]^T$ denote the uplink association vector of edge node m. We denote $\mathbf{A} = \{a_{km}\} \in \{0,1\}^{K \times (M+1)}$ to be the uplink routing matrix, and $\tilde{\mathbf{a}} = [\mathbf{a}_0^T, \mathbf{a}_1^T, \ldots, \mathbf{a}_M^T]^T$ to be the column vector corresponding to $\mathbf{A}$. Let $\mathcal{K}_m^t$ denote the set of user associated with edge node m at iteration t, then we have $\bigcup_m \mathcal{K}_m^t = \mathcal{K}^t$, and $|\mathcal{K}^t| = \sum_m |\mathcal{K}_m^t|$, where $|\cdot|$ denotes the cardinality of a set. Here, $\mathcal{K}_m^t$ may change over t to adapt with the change of wireless channels.

Let r_{km} denote the uplink data rate between user k and edge node m. For ease of exposition, let r_{k0} denote the uplink data rate between the cloud node and user k. Let $\mathbf{r}_m = [r_{1m}, r_{2m}, \ldots, r_{Km}]^T$ denote the uplink bandwidth allocation vector corresponding to edge node m, with $m = 0$ for cloud node. We denote $\mathbf{R} = r_{km} \in \mathbb{R}^{K \times (M+1)}$ as the uplink bandwidth allocation matrix.

Let B_m^{fr}, and B_m^{bk} denote the uplink fronthaul and backhaul data rate capacity of edge node m. Then, uplink communication latency between edge node m and its associated users is the longest latency of a given user:

$$T_m^{\text{u,fr}} = \max_{k \in \mathcal{K}_m} \left\{ D \frac{a_{km}}{r_{km}} \right\}, \text{ where } r_{km} \leq B_m^{\text{fr}}, \forall m \in \mathcal{M} \setminus \{0\}. \tag{10}$$

After edge nodes receive local models, there are two methods considered in this paper. Each edge node can help the

cloud node to aggregate the local model, then send the aggregated result to the cloud node. Alternatively, edge nodes just forward received models to the cloud. Let γ_m denote the transmission latency between edge node m and the cloud node. Without in-network computation protocols, γ_m is computed as followed

$$\gamma_m = \frac{D \sum_{k \in \mathcal{K}_m} a_{km}}{B_m^{\text{bk}}}. \tag{11}$$

The uplink latency of users associated with edge node m is

$$T_m^{\text{u}} = \max_{k \in \mathcal{K}_m} \left\{ T_m^{\text{u,fr}} \right\} + \gamma_m. \tag{12}$$

For users associated with the cloud node, let W^u denote the uplink communication capacity of the cloud node. Thus, the uplink latency of these users is the longest latency of a given user

$$T_0^{\text{u}} = \max_{k \in \mathcal{K}_0} \left\{ D \frac{a_{k0}}{r_{k0}} \right\}, \forall k \in \mathcal{K}_0, \text{ where } r_{k0} \leq W^u. \tag{13}$$

Hence, the total uplink latency of the whole network is

$$T^{\text{u}} = \max \left\{ \max_m \left\{ T_m^{\text{u}} \right\}, T_0^{\text{u}} \right\}. \tag{14}$$

Remark 1. *Selecting the primal method or the primal-dual method does not impact the amount of data sent by each user per iteration. The reason is that instead of sending $\mathbf{w}_k^t \in \mathbb{R}^d$, each user sends $\Delta \mathbf{v}_k^t \in \mathbb{R}^d$ having the same size d. Thus, in the scenarios where the system chooses primal method in initial iterations and primal-dual method later as in [28] the computation of the total uplink latency remains unchanged.*

3.3 Local Processing Model

Let c_k (cycles/sample) be the number of processing cycles of MU k to execute one sample of data, which assumes to be measured offline and known a prior [25]. Denoting the central processing unit (CPU) frequency of MU k by f_k (cycles/s), the computation time for the local training update at MU k over L local iterations is given by

$$t_k^{\text{cp}} = L_k \frac{c_k n_k}{f_k}. \tag{15}$$

Here, L_k depends on the number of training passes that each client makes over its local dataset on each round, number of local data samples, and the local minibatch size [4]. Note that t_k^{cp} can be estimated by each MU before joining FL. Thus, the MEC knows t_k^{cp} as a prior. Since data generated by mobile users usually follow the power law [40], [41], we assume that t_k^{cp} also follows the power law. Let t_k^{cp} be lower bounded and upper bounded by $t_{\text{min}}^{\text{cp}}$ and $t_{\text{max}}^{\text{cp}}$, respectively. The probability density of computing time is

$$p(t^{\text{cp}}) = \frac{\beta - 1}{t_{\text{min}}^{\text{cp}}} \left(\frac{t^{\text{cp}}}{t_{\text{min}}^{\text{cp}}} \right)^{-\beta}. \tag{16}$$
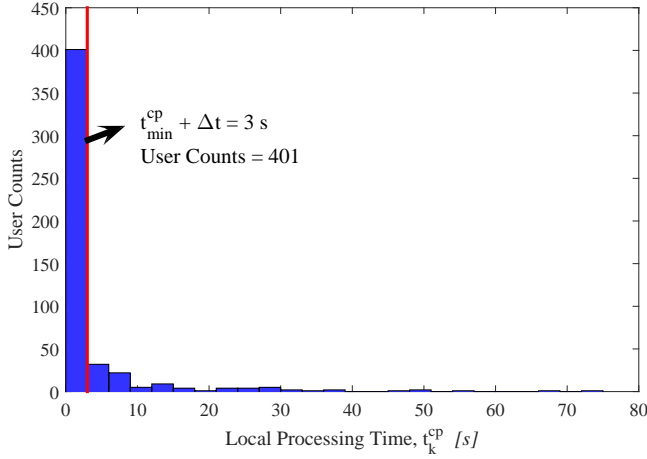
---

1. The learner selection in FL can be based on the quality or significance of information or location learners [8]–[10].

Fig. 2. An example of the distribution of t_k^{cp}, where $K = 500$, $t_{\min}^{\mathrm{cp}} = 0.2$s, $t_{\max}^{\mathrm{cp}} = 80$s and $\beta = 1.6$.

With several real datasets [41], it is observed that $\beta \in [1.47, 2.46]$. Here, we set β at 1.6, $t_{\min}^{\mathrm{cp}}$ at 0.2. We also limit the maximum value of t^{cp} at 80s.

4 IN-NETWORK COMPUTATION USER SCHEDUL-ING

To minimize the training time of the FL systems, we aim to minimize the latency that includes both up/down-link communication and computing latency for each FL iteration. Specifically, the latency per iteration depends on how we schedule the MUs to upload their updates, how the local model/updates from MUs are routed to the server, and how allocate communications resources for each up-link. Let $s \in \mathbb{S}$ denote a user scheduling scheme and $\mathbb{S}$ denote the set of all feasible scheduling schemes. The training time minimization problem can be formulated as follows.

$$\min_{s, \mathbf{A}, \mathbf{R}} T(s, \mathbf{A}, \mathbf{R}), \qquad (17)$$

where $T(s, \mathbf{A}, \mathbf{R})$ is the training time of one iteration. For the widely accepted conventional scheduling scheme, de-noted by s_0, the network operator begins aggregating users' models after all users complete their local update step [25]. Its training time for one FL iteration is computed as follows :

$$T(s_0, \mathbf{A}, \mathbf{R}) = T^{\mathrm{d}} + \max_k \{t_k^{\mathrm{cp}}\} + T^{\mathrm{u}}(\mathbf{A}, \mathbf{R}). \qquad (18)$$

This scheduling may suffer serious straggler effect due to slow MUs. We observe that with different distributions of t_k^{cp}, we can use different suitable user scheduling schemes. Let us consider two extreme cases where t_k^{cp} is either densely or dispersedly distributed. We apply two corre-sponding scheduling schemes with the upper bounds of the training time.

Remark 2. *If the local computing time of users is densely distributed, i.e., $t_{\max}^{\mathrm{cp}} - t_{\min}^{\mathrm{cp}} \leq \epsilon_0 T^{\mathrm{u}}(\mathbf{A}, \mathbf{R}), \epsilon_0 \in (0, 1)$, the network operator could wait for all users finishing updating local*

model, *i.e., the s_0 scheduling scheme. The training time in one iteration is then bounded by*

$$T(s_0, \mathbf{A}, \mathbf{R}) \leq T^{\mathrm{d}} + t_{\min}^{\mathrm{cp}} + (1 + \epsilon_0) T^{\mathrm{u}}(\mathbf{A}, \mathbf{R}). \qquad (19)$$

We observe that if the difference of local processing time between the slowest and the fastest client is not sig-nificant, i.e., $t_{\max}^{\mathrm{cp}} - t_{\min}^{\mathrm{cp}} \leq \epsilon_0 T^{\mathrm{u}}(\mathbf{A}, \mathbf{R}), \epsilon_0 \in (0, 1)$, the user scheduling does not highly impact the training time in one iteration. For example, given 100 users and the aggregation time of collecting all models, T^{u}, is 100 seconds, if 80 users finish local processing at the same time after 79 seconds and 20 users finish local processing at the same time after 80 seconds, the training time in one iteration cannot be sooner than $T^{\mathrm{d}} + 79 + T^{\mathrm{u}}(\mathbf{A}, \mathbf{R})$, but can not later than $T^{\mathrm{d}} + 80 + T^{\mathrm{u}}(\mathbf{A}, \mathbf{R})$. Since the upper-bound of the difference of aggregation time between the earliest case and the slowest case is 1 second which is very small compared with the aggregation time. In this case, it does not impact much on the training time.

Remark 3. *Let us rank the local processing time t_k^{cp} in an increasing order. If the local processing time of users is dispersedly distributed, i.e., $\min |t_q^{\mathrm{cp}} - t_{q+1}^{\mathrm{cp}}| \geq \frac{D}{W^u}, \forall q$, where q is the index of the rank, the centralized/cloud server can collect users' models one by one. The training time in one iteration is then*

$$T(s_1, \mathbf{A}, \mathbf{R}) = T^{\mathrm{d}} + t_{\max}^{\mathrm{cp}} + \frac{D}{W^u}. \qquad (20)$$

In the sequel, we focus on addressing the scenario where $t_{\max}^{\mathrm{cp}} - t_{\min}^{\mathrm{cp}} > \epsilon_0 T^{\mathrm{u}}(\mathbf{A}, \mathbf{R})$ and $\exists k, k', |t_k^{\mathrm{cp}} - t_{k'}^{\mathrm{cp}}| < \frac{D}{W^u}$. As an illustrative example, Fig. 2 shows a distribution of t_k^{cp} of 500 users following the power law distribution, where $t_{\min}^{\mathrm{cp}}$ and $t_{\max}^{\mathrm{cp}}$ are 0.2s and 80s, respectively. As can be observed, within a small amount of time $\Delta t \ll t_{\max}^{\mathrm{cp}}$ during $[t_{\min}^{\mathrm{cp}}, t_{\min}^{\mathrm{cp}} + \Delta t]$, where $t_{\min}^{\mathrm{cp}} + \Delta t = 3$s, most users (401 out of 500) finish their local update. Meanwhile, for the ResNet152's model size $D = 232$MB [42], the minimum uplink latency if all users directly send their model to the cloud via its uplink wireless channel is $T^{\mathrm{u}} = \frac{KD}{W^u} = 464$s. Since $t_{\min}^{\mathrm{cp}} + \Delta t = 3$s is much smaller than T^{u}, while $t_{\max}^{\mathrm{cp}}$ is comparable with T^{u}, it significantly stagnates the FL system if the server only starts the model aggregation after the last user completes its local update, i.e., using the scheduling s_0. Given this observation, we propose an in-network user scheduling mechanism to mitigate the straggler effect in FL systems in the next section.

4.1 Bipartition User Scheduling Scheme

Our proposed bipartition user scheduling is summarized in Algorithm 1. Specifically, let $\mathcal{P}_1^t = \{k \in \mathcal{K}^t | t_k^{\mathrm{cp}} \in [t_{\min}^{\mathrm{cp}}, t_{\min}^{\mathrm{cp}} + \Delta t]\}$ be the user partition where $t_k^{\mathrm{cp}} \in [t_{\min}^{\mathrm{cp}}, t_{\min}^{\mathrm{cp}} + \Delta t]$ and $\mathcal{P}_2^t = \mathcal{K}^t \setminus \mathcal{P}_1^t$ denote the user partition of the rest users, at iteration t. Let $T^{\mathrm{u}}(\mathbf{A}_{\mathcal{P}_j^t}, \mathbf{R}_{\mathcal{P}_j^t}, \mathcal{P}_j^t)$ denote the uplink latency which is returned by a network routing algorithm by aggregating of all users' models of $\mathcal{P}_j^t$ under the corresponding uplink routing and bandwidth allocation matrices $\mathbf{A}_{\mathcal{P}_j^t}, \mathbf{R}_{\mathcal{P}_j^t}$, for $j \in \{1, 2\}$. At each iteration t, we first begin aggregating the global model of $\mathcal{P}_1^t$ at time $t_{\min}^{\mathrm{cp}} + \Delta t$. The time required for these users in $\mathcal{P}_1^t$ to complete the aggregation process is $t_{\mathcal{P}_1^t} = T^{\mathrm{d}} + t_{\min}^{\mathrm{cp}} + \Delta t + T^{\mathrm{u}}(\mathbf{A}_{\mathcal{P}_1^t}, \mathbf{R}_{\mathcal{P}_1^t}, \mathcal{P}_1^t)$. If $t_{\mathcal{P}_1^t} > T^{\mathrm{d}} + t_{\max}^{\mathrm{cp}}$, we aggregate $\mathcal{P}_2^t$.

Algorithm 1 Bipartition User Scheduling Scheme

Input: $\mathcal{K}^t$, $\{t_k^{\mathrm{cp}}, \forall k \in \mathcal{K}^t\}$, Δt and a network routing algorithm.

Output: T'

1: Partition users into $\mathcal{P}_1 = \{k \in \mathcal{K}^t | t_k^{\mathrm{cp}} \in [t_{\min}^{\mathrm{cp}} + \Delta t]\}$ and $\mathcal{P}_2 = \mathcal{K}^t \setminus \mathcal{P}_1$.

2: Begin aggregating $\mathcal{P}_1$ at time $t_{\min}^{\mathrm{cp}} + \Delta t$. Then, compute $t_{\mathcal{P}_1} = t_{\min}^{\mathrm{cp}} + \Delta t + T^{\mathrm{u}}(\mathbf{A}_{\mathcal{P}_1}, \mathbf{R}_{\mathcal{P}_1}, \mathcal{P}_1)$.

3: **if** $t_{\mathcal{P}_1} > T^{\mathrm{d}} + t_{\max}^{\mathrm{cp}}$ **then**

4:      Begin aggregating $\mathcal{P}_2$ at the time $t_{\mathcal{P}_1}$.

5: **else**

6:      Begin aggregating $\mathcal{P}_2$ at the time $T^{\mathrm{d}} + t_{\max}^{\mathrm{cp}}$.

7: **end if**

Otherwise, the rest users will wait until the slowest user finishing local processing, i.e., $T^{\mathrm{d}} + t_{\max}^{\mathrm{cp}}$.

4.2 System's Latency Analysis

We now analyze the training time of the FL system in one iteration resulting from the proposed user scheduling scheme. This bipartition scheduling scheme's whole network latency of one FL iteration is computed by

$$T(s_b, \mathbf{A}, \mathbf{R}) = T^{\mathrm{d}} + \max\{t_{\mathcal{P}_1^t}, t_{\max}^{\mathrm{cp}}\} + T^{\mathrm{u}}(\mathbf{A}_{\mathcal{P}_2^t}, \mathbf{R}_{\mathcal{P}_2^t}, \mathcal{P}_2^t), \tag{22}$$

where s_b is the proposed bipartition user scheduling scheme.

Theorem 1. *If there exist $\epsilon_1, \epsilon_2, \epsilon_3 \in (0,1)$ and $\epsilon_0 > \epsilon_2 + \epsilon_3$ that $|\mathcal{P}_2| \le \epsilon_1 K$, $T^{\mathrm{u}}(\mathbf{A}_{\mathcal{P}_2^t}, \mathbf{R}_{\mathcal{P}_2^t}, \mathcal{P}_2^t) \le \epsilon_2 T^{\mathrm{u}}(\mathbf{A}, \mathbf{R})$ and $\Delta t \le \epsilon_3 T^{\mathrm{u}}(\mathbf{A}, \mathbf{R})$, $T(s_b, \mathbf{A}, \mathbf{R})$ is upper bounded by*

$$T(s_b, \mathbf{A}, \mathbf{R}) \le T^{\mathrm{d}} + \max\left\{t_{\min}^{\mathrm{cp}} + (1 + \epsilon_2 + \epsilon_3)T^{\mathrm{u}}(\mathbf{A}, \mathbf{R}),\right.$$
$$\left. t_{\max}^{\mathrm{cp}} + \epsilon_2 T^{\mathrm{u}}(\mathbf{A}, \mathbf{R})\right\} < T(s_0, \mathbf{A}, \mathbf{R}). \tag{23}$$

Proof. If $t_{\mathcal{P}_1^t} > T^{\mathrm{d}} + t_{\max}^{\mathrm{cp}}$,

$$T(s_b, \mathbf{A}, \mathbf{R}) = T^{\mathrm{d}} + t_{\min}^{\mathrm{cp}} + \Delta t + T^{\mathrm{u}}(\mathbf{A}_{\mathcal{P}_1^t}, \mathbf{R}_{\mathcal{P}_1^t}, \mathcal{P}_1^t)$$
$$+ T^{\mathrm{u}}(\mathbf{A}_{\mathcal{P}_2^t}, \mathbf{R}_{\mathcal{P}_2^t}, \mathcal{P}_2^t)$$
$$\le T^{\mathrm{d}} + t_{\min}^{\mathrm{cp}} + (1 + \epsilon_2 + \epsilon_3)T^{\mathrm{u}}(\mathbf{A}, \mathbf{R})$$
$$< T^{\mathrm{d}} + t_{\max}^{\mathrm{cp}} + T^{\mathrm{u}}(\mathbf{A}, \mathbf{R}) = T(s_0, \mathbf{A}, \mathbf{R}). \tag{24}$$

Otherwise,

$$T(s_b, \mathbf{A}, \mathbf{R}) = T^{\mathrm{d}} + t_{\max}^{\mathrm{cp}} + T^{\mathrm{u}}(\mathbf{A}_{\mathcal{P}_2^t}, \mathbf{R}_{\mathcal{P}_2^t}, \mathcal{P}_2^t)$$
$$\le T^{\mathrm{d}} + t_{\max}^{\mathrm{cp}} + \epsilon_2 T^{\mathrm{u}}(\mathbf{A}, \mathbf{R})$$
$$< T^{\mathrm{d}} + t_{\max}^{\mathrm{cp}} + T^{\mathrm{u}}(\mathbf{A}, \mathbf{R}) = T(s_0, \mathbf{A}, \mathbf{R}). \tag{25}$$

From (24) and (25), the upper bound of the whole network latency of proposed user scheduling T' is achieved which is always smaller than that of the conventional user scheduling T. $\square$

Let us study a simple example where there is a star network consisting of $K = 500$ users and a single cloud node with $W^u = W^d = 2$Gbps. We assume that ResNet152's model is considered. Reusing the distribution in Fig. 2, the aggregation time without the proposed user scheduling is $T(s_0, \mathbf{A}, \mathbf{R}) = D/W^d + t_{\max}^{\mathrm{cp}} + KD/W^u = 0.928 +$

80 + 464 = 544.928s. With the proposed user scheduling, $t_{\min}^{\mathrm{cp}} + \Delta t = 3$s and $|\mathcal{P}_1| = 401$ users, and $t_{\mathcal{P}_1} = 0.928 + 3 + 372.128 = 376.056$s. Since $t_{\mathcal{P}_1} > T^{\mathrm{d}} + t_{\max}^{\mathrm{cp}} = 0.928 + 80 = 80.928$s, the aggregation time is $T(s_b, \mathbf{A}, \mathbf{R}) = T^{\mathrm{d}} + t_{\min}^{\mathrm{cp}} + \Delta t + T^{\mathrm{u}}(\mathbf{A}_{\mathcal{P}_1^t}, \mathbf{R}_{\mathcal{P}_1^t}, \mathcal{P}_1^t) + T^{\mathrm{u}}(\mathbf{A}_{\mathcal{P}_2^t}, \mathbf{R}_{\mathcal{P}_2^t}, \mathcal{P}_2^t) = t_{\mathcal{P}_1} + (K - |\mathcal{P}_1|)D/W^u = 376.056 + 91.872 = 467.928$s. Thus, $T(s_b, \mathbf{A}, \mathbf{R}) < T(s_0, \mathbf{A}, \mathbf{R})$. In another case, when $K = 50$ and $|\mathcal{P}_1| = 40$ users, $T(s_0, \mathbf{A}, \mathbf{R}) = D/W^d + t_{\max}^{\mathrm{cp}} + KD/W^u = 0.928 + 80 + 46.4 = 127.328$s and $t_{\mathcal{P}_1} = 0.928 + 3 + 37.2128 = 41.1408$s. Here $t_{\mathcal{P}_1} < T^{\mathrm{d}} + t_{\max}^{\mathrm{cp}}$, the aggregation time is thus $T(s_b, \mathbf{A}, \mathbf{R}) = T^{\mathrm{d}} + t_{\max}^{\mathrm{cp}} + T^{\mathrm{u}}(\mathbf{A}_{\mathcal{P}_2^t}, \mathbf{R}_{\mathcal{P}_2^t}, \mathcal{P}_2^t) = 0.928 + 80 + 9.28 = 90.208$s. It is also smaller than $T(s_0, \mathbf{A}, \mathbf{R})$.

5 IN-NETWORK AGGREGATION DESIGN

We now introduce the in-network aggregation design that allows edge nodes to support the server for addictive weighting users' local models. Back to early 2000s, the concept of in-network computation was well-studied for wireless sensor networks (WSNs), e.g., [16] due to sensors' limited communications, computing, storage capabilities. The core idea of in-network computation is to design data structures to better represent the information collected/generated at each sensor for each specific application [16]. Analogously, under FL, mobile users also do not transmit their raw data to the server. We then can interpret their local models as their data representation. To obtain the global model, defined in (7), we first design the user packet which plays the role of data representation in a in-network computation solution. This packet design can be tailored to adapt with two FL schemes. Specifically, let $\phi_k^t = \{\phi_k^t[0], \phi_k^t[1]\}$ denote the local message of users k at iteration t which are generated by following (21).

5.1 Aggregation Function

In this section, we propose a general aggregation function which is suitable for the two methods solving FL problems. If we use the primal method, we observe that

$$n = \sum_k n_k = \sum_k \phi_k^t[0]. \tag{26}$$

Similarly, for the primal-dual method, we also observe that

$$|\mathcal{K}^t| = \sum_k \phi_k^t[0]. \tag{27}$$

Hence, let z be a hyperparameter where $z = 1$ if we use the primal-dual method and $z = 0$ otherwise. To preserve the return of (7), the global model of the cloud at each iteration can be computed from the users' packets designed in (21) as follows

$$\boldsymbol{\psi}^{t+1} = z\boldsymbol{\psi}^t + \frac{\sum_k \phi_k^t[0]\phi_k^t[1]}{\sum_k \phi_k^t[0]}. \tag{28}$$

Thus, we now can use (28) as the aggregation function generalized for both primal and primal-dual methods solving FL.

5.2 In-network Aggregation Process with Bipartition User Scheduling Scheme

Under the proposed bipartition scheduling scheme s_b, users are scheduled to aggregate their models in two partitions.

$$\{\phi_k^t[0], \phi_k^t[1]\} = \begin{cases} \{n_k, \mathbf{w}_k^t\}, & \text{if we choose the primal method,} \\ \{1, \Delta\mathbf{v}_k^t\}, & \text{if we choose the primal-dual method.} \end{cases} \quad (21)$$
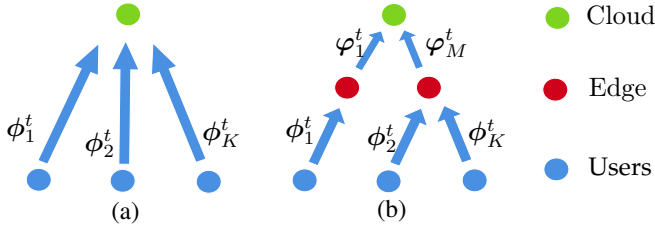

Fig. 3. The logical view of (a) conventional network model and (b) multi-tier edge network model with INA.

In this section, we propose a novel in-network aggregation (INA) process, as illustrated in Fig. 3, that enable edge nodes to support the cloud node in decentralizing the aggregation process. Let $\chi_{m,j}^t$ denote the local aggregated model of edge node m, for user partition $\mathcal{P}_j$, such that

$$\chi_{m,j}^t = \frac{\sum_{k \in \mathcal{K}_m, k \in \mathcal{P}_j} \phi_k^t[0]\phi_k^t[1]}{\sum_{k \in \mathcal{K}_m, k \in \mathcal{P}_j} \phi_k^t[0]}. \quad (29)$$

Let $\boldsymbol{\varphi}_{m,j}^t = \{\varphi_{m,j}^t[0], \boldsymbol{\varphi}_{m,j}^t[1]\}$ denote the message edge node m sends to the cloud node, for user partition $\mathcal{P}_j$, such that

$$\{\varphi_{m,j}^t[0], \boldsymbol{\varphi}_{m,j}^t[1]\} = \Big\{ \sum_{k \in \mathcal{K}_m, k \in \mathcal{P}_j} \phi_k^t[0], \chi_{m,j}^t \Big\}. \quad (30)$$

Let $\boldsymbol{\lambda}_j^t = \{\lambda_j^t[0], \boldsymbol{\lambda}_j^t[1]\}$ denote the weight and the parameters of the aggregated model for user partition $\mathcal{P}_j$. After all edge nodes' messages $\boldsymbol{\varphi}_{m,j}^t$ are sent to the cloud, we compute $\boldsymbol{\lambda}_j^t$ as follows:

$$\begin{cases} \lambda_j^t[0] &= \sum_{k \in \mathcal{K}_0, k \in \mathcal{P}_j} \phi_k^t[0] + \sum_m \varphi_{m,j}^t[0], \\ \boldsymbol{\lambda}_j^t[1] &= \frac{\sum_{k \in \mathcal{K}_0, k \in \mathcal{P}_j} \phi_k^t[0]\phi_k^t[1] + \sum_m \varphi_{m,j}^t[0]\boldsymbol{\varphi}_{m,j}^t[1]}{\sum_{k \in \mathcal{K}_0, k \in \mathcal{P}_j} \phi_k^t[0] + \sum_m \varphi_{m,j}^t[0]}. \end{cases} \quad (31)$$

Proposition 1. *In order to preserve the return of (28), the global model is computed as follows,*

$$\boldsymbol{\psi}^{t+1} = z\boldsymbol{\psi}^t + \frac{\sum_j \lambda_j^t[0]\boldsymbol{\lambda}_j^t[1]}{\sum_j \lambda_j^t[0]}. \quad (32)$$

Proof. We have:

$$\sum_j \lambda_j^t[0]$$

$$= \sum_{k \in \mathcal{K}_0, k \in \mathcal{P}_1} \phi_k^t[0] + \sum_m \varphi_{m,1}^t[0] + \sum_{k \in \mathcal{K}_0, k \in \mathcal{P}_2} \phi_k^t[0] + \sum_m \varphi_{m,2}^t[0]$$

$$= \sum_{k \in \mathcal{K}_0, k \in \mathcal{P}_1} \phi_k^t[0] + \sum_{k \in \mathcal{K}_m, k \in \mathcal{P}_1} \phi_k^t[0] + \sum_{k \in \mathcal{K}_0, k \in \mathcal{P}_2} \phi_k^t[0] + \sum_{k \in \mathcal{K}_m, k \in \mathcal{P}_2} \phi_k^t[0]$$

$$= \sum_k \phi_k^t[0]. \quad (33)$$

We also have

$$\sum_j \lambda_j^t[0]\boldsymbol{\lambda}_j^t[1]$$

$$= \sum_j \Big(\sum_{k \in \mathcal{K}_0, k \in \mathcal{P}_j} \phi_k^t[0] + \sum_m \varphi_{m,j}^t[0] \Big)$$
$$\Big(\frac{\sum_{k \in \mathcal{K}_0, k \in \mathcal{P}_j} \phi_k^t[0]\phi_k^t[1] + \sum_m \varphi_{m,j}^t[0]\boldsymbol{\varphi}_{m,j}^t[1]}{\sum_{k \in \mathcal{K}_0, k \in \mathcal{P}_j} \phi_k^t[0] + \sum_m \varphi_{m,j}^t[0]} \Big)$$

$$= \sum_j \Big(\sum_{k \in \mathcal{K}_0, k \in \mathcal{P}_j} \phi_k^t[0]\phi_k^t[1] + \sum_m \varphi_{m,j}^t[0]\boldsymbol{\varphi}_{m,j}^t[1] \Big)$$

$$= \sum_j \Big(\sum_{k \in \mathcal{K}_0, k \in \mathcal{P}_j} \phi_k^t[0]\phi_k^t[1]$$
$$+ \sum_m \Big(\sum_{k \in \mathcal{K}_m, k \in \mathcal{P}_j} \phi_k^t[0] \frac{\sum_{k \in \mathcal{K}_m, k \in \mathcal{P}_j} \phi_k^t[0]\phi_k^t[1]}{\sum_{k \in \mathcal{K}_m, k \in \mathcal{P}_j} \phi_k^t[0]} \Big) \Big)$$

$$= \sum_j \Big(\sum_{k \in \mathcal{K}_0, k \in \mathcal{P}_j} \phi_k^t[0]\phi_k^t[1] + \sum_m \sum_{k \in \mathcal{K}_m, k \in \mathcal{P}_j} \phi_k^t[0]\phi_k^t[1] \Big)$$

$$= \sum_k \phi_k^t[0]\phi_k^t[1]. \quad (34)$$

From (33) and (34), the return of (28) is preserved when the global model is aggregated using (32). $\square$

Theorem 2. *(Theorem 2 of [26] as well as Theorem 4.2 and Theorem 4.3 of [21]) Assuming that $l_i(\cdot)$ is convex with $1/\mu$-smoothness and $r(\cdot)$ is 1-strongly convex. If the primal method is used, given that $\mathbf{w}^*$ is the global optimal model, after ι learning rounds, we have the following convergence upperbound:*

$$P(\mathbf{w}^\iota) - P(\mathbf{w}^*) \le \theta_g, \quad (35)$$

where θ_g is the global accuracy. Here, ι is upperbounded by $\mathcal{O}(1/\theta_g)$. On the other hand, if the primal-dual method is used, after ι' learning rounds, we have the following convergence upperbound:

$$\mathbb{E}\Big[P(\mathbf{w}(\boldsymbol{\alpha}^{\iota'})) - G(\boldsymbol{\alpha}^{\iota'}) \Big] \le \theta_g. \quad (36)$$

Here, ι' is upperbounded by $\frac{\mathcal{O}(\log(1/\theta_g))}{1-\theta_l}$, where θ_l is the local accuracy defining at Assumption 4.1 of [21].

Proof. From Proposition 1, it can be seen that the INA process always remains the returns of the FL aggregated model for both FedAvg and CoCoA. Thus, if FedAvg is used, Eq. (35) holds by following Theorem 2 of [26]. Similarly, if CoCoA is used, Eq. (36) holds by following Theorem 4.2 and Theorem 4.3 of [21]. $\square$

Remark 4. *In a special case where no user is directly associated with the cloud node, the proposed edge network architecture and the INA process could reduce the traffic and computing overhead at the cloud node by a factor of K/M in comparison with*

conventional FL star network topology, where K is the number of users and M is the number of edge nodes.

Proof. The traffic overhead at the cloud node is proportional to the number of models the cloud node received. For computing overhead, we also assume that the aggregation operation at the cloud runs linearly to the number of received models. Hence, under the conventional FL star network topology, the cloud node receives K models from its users.

In contrast, with the proposed edge network architecture using the INA, when no user is directly associated with the cloud node, i.e., $\mathcal{K}_0 = \emptyset$, the cloud node only receives M aggregated models from the edge nodes. As a result, in practice where $K >> M$, the two mentioned overheads are reduced by a factor of K/M. □

Under the proposed INA process, the transmission latency γ_m between an edge node m and the cloud node is revised. If there is no user associate with an edge node m, i.e., $\sum_k a_{km} = 0$, γ_m is zero. Otherwise, since edge node m only needs to send the aggregated model the cloud node, γ_m is revised as follows

$$\gamma_m = \min\left\{ \frac{D}{B_m^{\mathrm{bk}}}, \frac{D\sum_{k\in\mathcal{K}_m} a_{km}}{B_m^{\mathrm{bk}}} \right\}. \tag{37}$$

6 NETWORK ROUTING AND RESOURCE ALLOCATION FRAMEWORK FOR FL

Following the user scheduling scheme and the INA process proposed above, in this section, we aim to optimize the network' resource allocation and the routing matrices to minimize the uplink aggregation latency in one iteration for a given partition $\mathcal{P}_j^t$. When the proposed INA process is considered, the joint routing and resource optimization problem for users in $\mathcal{P}_j^t$ can be written as follows:

$$\mathscr{F}_1^u : \min_{\mathbf{A},\mathbf{R}} T^{\mathrm{u}},$$

$$\text{s.t.} \quad \sum_{m=0}^{M} a_{km} = 1, \forall k \in \mathcal{P}_j^t, \tag{38a}$$

$$\sum_{k\in K_0} r_{k0} \le W^u, \tag{38b}$$

$$\sum_{k\in\mathcal{K}_m} r_{km} \le B_m^{\mathrm{fr}}, \forall m \in \mathcal{M}\setminus\{0\}, \tag{38c}$$

$$a_{km} \in \{0,1\}, \tag{38d}$$

$$r_{k0} \in [0, W^u], \text{ and } r_{km} \in [0, B_m^{\mathrm{fr}}], \forall m \in \mathcal{M}\setminus\{0\}. \tag{38e}$$

The constraints (38a) guarantee that a user can associate with only one edge node in one iteration[2]. The constraints (38b) and (38c) ensure that total users' data rates associated with each edge node or the cloud node must not exceed its bandwidth capacity. Here, the transmission latency between

2. A more general network model could be considered. The binary constraints can be replaced by $a_{km} \in [0,1]$. However, in that scenario, the proposed in-network aggregation process at edge nodes could run with incomplete models. Those models are then forwarded to the cloud to be aggregated. Consequently, the network suffers extra latency and traffic as penalties. In practice, as D and K can be significantly large, the penalties make the solution of a_{km} close to binary.

an edge node m and the cloud node is computed as in (37). The mixed integer non-linear programming problem $\mathscr{F}_1^u$ is actually NP-Hard. We then propose a highly efficient randomized rounding solution for practical implementation in the next section.

Proposition 2. $\mathscr{F}_1^u$ *is a NP-Hard problem.*

Proof. To prove that $\mathscr{F}_1^u$ is a NP-Hard problem, we first introduce Lemma 1 and Lemma 2. These two lemmas allow us to transform $\mathscr{F}_1^u$ into an equivalent Integer Linear Programming, which is then proven to be NP-Hard. Hence, $\mathscr{F}_1^u$ is also NP-Hard.

Lemma 1. *Given any uplink routing matrix* $\mathbf{A}$, *with* $|\mathcal{K}_0| = \sum_{k\in\mathcal{K}_0} a_{k0} > 0$, *for problem* $\mathscr{F}_1^u$, *at the cloud node, the uplink latency for users associated with the cloud node satisfies*

$$T_0^{\mathrm{u}} = \max_{k\in\mathcal{K}_0}\left\{ D\frac{a_{k0}}{r_{k0}} \right\} \ge \frac{D|\mathcal{K}_0|}{W^u}, \forall k \in \mathcal{K}_0. \tag{39}$$

Here, the equality happens when $r_{10} = \ldots = r_{|\mathcal{K}_0|0} = \frac{W^u}{|\mathcal{K}_0|}$.

Lemma 2. *Given any uplink routing matrix* $\mathbf{A}$, *with* $|\mathcal{K}_m| = \sum_{k\in\mathcal{K}_m} a_{km} > 0$, *for problem* $\mathscr{F}_1^u$, *at each edge node* m, *the uplink latency for users associated with edge node* m *satisfies*

$$T_m^{\mathrm{u}} = \max_{k\in\mathcal{K}_m}\left\{ D\frac{a_{km}}{r_{km}} \right\} + \min\left\{ \frac{D}{B_m^{\mathrm{bk}}}, \frac{D\sum_{k\in\mathcal{K}_m} a_{km}}{B_m^{\mathrm{bk}}} \right\}$$

$$\ge \frac{D|\mathcal{K}_m|}{B_m^{\mathrm{fr}}} + \min\left\{ \frac{D}{B_m^{\mathrm{bk}}}, \frac{D|\mathcal{K}_m|}{B_m^{\mathrm{bk}}} \right\}. \tag{40}$$

The equality happens when $r_{1m} = \ldots = r_{|\mathcal{K}_m|m} = \frac{B_m^{\mathrm{fr}}}{|\mathcal{K}_m|}$.

Following Lemma 1 and Lemma 2, we can observe that the network operator only needs to optimize the uplink routing matrix while the uplink data rates for users associated with edge nodes or cloud node will be equally allocated among their directly associated users. If there exists m such that $|\mathcal{K}_m| = 0$, we can arbitrarily set the value of $\mathbf{r}_m$ and set the value of T_m^{u} as 0. As a result, $\mathscr{F}_1^u$ is reduced to

$$\mathscr{F}_2^u : \min_{\mathbf{A}} \max\left\{ \max_m\left\{ D\frac{\sum_{k\in\mathcal{K}_m} a_{km}}{B_m^{\mathrm{fr}}} \right. \right.$$

$$\left. + \min\left\{ \frac{D}{B_m^{\mathrm{bk}}}, \frac{D\sum_{k\in\mathcal{K}_m} a_{km}}{B_m^{\mathrm{bk}}} \right\}, D\frac{\sum_{k\in\mathcal{K}_0} a_{k0}}{W^u} \right\},$$

$$\text{s.t. } (38a) \text{ and } (38d), \tag{41}$$

where the optimal solution in $\mathscr{F}_2^u$ is also the optimal solution in $\mathscr{F}_1^u$. Consider a special case when the edge-to-cloud latency are negligible, i.e. $B_m^{\mathrm{bk}} \to \infty$. For the convenience of notations, let $B_0^{\mathrm{fr}} \triangleq W^u$. We have

$$\mathscr{F}_2^u : \min_{\mathbf{A}}\left\{ \max_m\left\{ D\frac{\sum_{k\in\mathcal{K}_m} a_{km}}{B_m^{\mathrm{fr}}} \right\} \right\},$$

$$\text{s.t. } (38a) \text{ and } (38d), \tag{42}$$

This formulation is mathematically similar to the makespan minimization problem for parallel machines, which is NP-Hard [43], where the makespan is the completion time of the last task. The special case is NP-hard and hence, so is $\mathscr{F}_2^u$.

Since $\mathscr{F}_2^u$ is an NP-Hard problem, $\mathscr{F}_1^u$ is also an NP-Hard problem. □

If the proposed in-network computation protocol is not considered, the aggregation latency-minimized routing framework is formulated as followed

$$\mathscr{Q}_1^u : \min_{\mathbf{A},\mathbf{R}} T^{\mathrm{u}},$$

$$\text{s.t. } (38a) - (38e), \tag{43}$$

where the transmission latency between an edge node m and the cloud node is computed as in (11).

Proposition 3. *$\mathscr{Q}_1^u$ is a NP-Hard problem*

Proof. Similar to the proof of Proposition 2, we first introduce Lemma 3 and Lemma 4. The proofs of these lemmas are similar to those of Lemma 1 and Lemma 2, hence omitted.

Lemma 3. *Given any uplink routing matrix $\mathbf{A}$, with $|\mathcal{K}_0| = \sum_{k\in\mathcal{K}_0} a_{k0} > 0$, for problem $\mathscr{Q}_1^u$, at the cloud node, we observe that T_0^{u} satisfies*

$$T_0^{\mathrm{u}} = \max_{k\in\mathcal{K}_0} \left\{ D\frac{a_{k0}}{r_{k0}} \right\} \geq \frac{D|\mathcal{K}_0|}{W^u}, \forall k \in \mathcal{K}_0. \tag{44}$$

Here, the equality happens when $r_{10} = \ldots = r_{|\mathcal{K}_0|0} = \frac{W^u}{|\mathcal{K}_0|}$.

Lemma 4. *Given any uplink routing matrix $\mathbf{A}$, with $|\mathcal{K}_m| = \sum_{k\in\mathcal{K}_m} a_{km} > 0$, for problem $\mathscr{Q}_1^u$, at each edge node m, we observe that T_m^{u} satisfies*

$$T_m^{\mathrm{u}} = \max_{k\in\mathcal{K}_m} \left\{ D\frac{a_{km}}{r_{km}} \right\} + \frac{D|\mathcal{K}_m|}{B_m^{\mathrm{bk}}} \geq \frac{D|\mathcal{K}_m|}{B_m^{\mathrm{fr}}} + \frac{D|\mathcal{K}_m|}{B_m^{\mathrm{bk}}}, \tag{45}$$

where $|\mathcal{K}_m| = \sum_{k\in\mathcal{K}_m} a_{km}$. The equality happens when $r_{1m} = \ldots = r_{|\mathcal{K}_m|m} = \frac{B_m^{\mathrm{fr}}}{|\mathcal{K}_m|}$.

Following Lemma 3 and Lemma 4, we see that the network operator needs to only optimize the uplink routing matrix, while the uplink data rates for users associated with edge nodes or cloud node will be equally allocated. As a result, $\mathscr{Q}_1^u$ is reduced to

$$\mathscr{Q}_2^u : \min_{\mathbf{A}} \max \left\{ \max_m \left\{ D\frac{\sum_{k\in\mathcal{K}_m} a_{km}}{B_m^{\mathrm{fr}}} + D\frac{\sum_{k\in\mathcal{K}_m} a_{km}}{B_m^{\mathrm{bk}}} \right\}, \right.$$
$$\left. D\frac{\sum_{k\in\mathcal{K}_0} a_{k0}}{W^u} \right\},$$

$$\text{s.t. } (38a) \text{ and } (38d), \tag{46}$$

where the optimal solution in $\mathscr{Q}_2^u$ is also the optimal solution in $\mathscr{Q}_1^u$. Similar to Proposition 2, we consider a special case when the edge-to-cloud latency are negligible, i.e. $B_m^{\mathrm{bk}} \to \infty$. Here, we see that $\mathscr{Q}_2^u$ is a NP-Hard problem, so is $\mathscr{Q}_1^u$. □

7 RANDOMIZED ROUNDING BASED SOLUTION

7.1 With In-Network Computation Protocol

In this section, we present an approximation algorithm for the main problem that leverages a randomized rounding

technique. The proposed algorithm is summarized in Algorithm 2. First, we introduce auxiliary variables y and γ_m into $\mathscr{F}_2^u$ such that

$$y \geq \max \left\{ \max_m \left\{ D\frac{\sum_{k\in\mathcal{K}_m} a_{km}}{B_m^{\mathrm{fr}}} + \gamma_m \right\}, D\frac{\sum_{k\in\mathcal{K}_0} a_{k0}}{W^u} \right\}, \tag{47}$$

$$\gamma_m \leq \min \left\{ \frac{D}{B_m^{\mathrm{bk}}}, \frac{D\sum_{k\in\mathcal{K}_m} a_{km}}{B_m^{\mathrm{bk}}} \right\}, \forall m \tag{48}$$

Problem $\mathscr{F}_2^u$ is then equivalently rewritten as

$$\mathscr{F}_3^u : \min_{y,\{\gamma_m\},\{a_{km}\}} y,$$

$$\text{s.t. } y \geq D\frac{\sum_{k\in\mathcal{K}_m} a_{km}}{B_m^{\mathrm{fr}}} + \gamma_m, \forall m, \tag{49a}$$

$$y \geq D\frac{\sum_{k\in\mathcal{K}_0} a_{k0}}{W^u}, \tag{49b}$$

$$\gamma_m \leq \frac{D}{B_m^{\mathrm{bk}}}, \forall m \in \mathcal{M} \setminus \{0\}, \tag{49c}$$

$$\gamma_m \leq D\frac{\sum_{k\in\mathcal{K}_0} a_{k0}}{W^u}, \forall m \in \mathcal{M} \setminus \{0\}, \tag{49d}$$

$$(38a) \text{ and } (38d).$$

The Algorithm 2 starts by solving the Linear Relaxation (LR) of $\mathscr{F}_3^u$. Specifically, it relaxes the variables a_{km} to be fractional, rather than integer. The Linear Relaxation of $\mathscr{F}_3^u$ can be expressed as follows:

$$\mathscr{F}_4^u : \min_{y,\{\gamma_m\},\{a_{km}\}} y,$$

$$\text{s.t. } (49a) - (49d), (38a), \text{ and } a_{km} \in [0,1]. \tag{50}$$

Let $z^\dagger = [\tilde{\mathbf{a}}^\dagger, y^\dagger, \gamma_1^\dagger, \ldots, \gamma_M^\dagger]$ denote the optimal solution of $\mathscr{F}_4^u$. We first transform $\tilde{\mathbf{a}}^\dagger$ to the equivalent fractional matrix $\mathbf{A}^\dagger$, whose elements are in $[0,1]$. If all components of $\mathbf{A}^\dagger$ are binary, it is also the optimal solution to $\mathscr{F}_2^u$. Otherwise, to recover binary characteristic of $\mathbf{A}$, for each row of $\mathbf{A}^\dagger$, we randomly round the element a_{km} to 1 with probability $a_{km}^\dagger$. The decision is done in an exclusive manner for satisfying constraint (38a). It means that for each row k, only one element of the row is one, the rest are zeros. For example, let us consider 2 edge nodes and at row k, assume $a_{k0} = 0.7$, $a_{k1} = 0.1$ and $a_{k2} = 0.2$. We construct 3 intervals, $0 : [0, 0.7]$, $1 : (0.7, 0.8]$ and $2 : (0.8, 1]$. We then randomly pick a number uniformly distributed in $[0, 1]$. If the value is in interval 1, we set $a_{k1} = 1$, and the rest are zeros. The random decision is made independently for each k. Following this procedure, we obtain the solution $\mathbf{A}^{(\mathrm{Alg})}$. Then, $r_{1,0}^{(\mathrm{Alg})} = \ldots = r_{|\mathcal{K}_0|,0}^{(\mathrm{Alg})} = \frac{W^u}{|\mathcal{K}_0|}$ and $r_{1m}^{(\mathrm{Alg})} = \ldots = r_{|\mathcal{K}_m|m}^{(\mathrm{Alg})} = \frac{B_m^{\mathrm{fr}}}{|\mathcal{K}_m|}, \forall m$. The complexity of this algorithm is $O(\nu^{3.5}\Omega^2)$, where $\nu = K(M+1)+1$, Ω is the number of the bits in the input [44]. Here, we provide the guarantee on the quality of the aggregation latency returned by Algorithm 2.

Theorem 3. *The aggregation latency returned by Algorithm 2 is at most $\frac{2\ln K}{y^\dagger} + 3$ times higher than the optimal with high probability $1 - 1/K$, under the assumption $y^\dagger > \ln(K)$, where $y^\dagger$ is the optimal value of the $\mathscr{F}_4^u$ and also the lower bound of the optimal, and K is the number of users.*

Algorithm 2 Randomized Routing Algorithm for Low Latency Federated Learning

---

Input: D, B_m^{fr}, B_m^{bk}, W^u and $\mathcal{K}^t$.
Output: $\mathbf{A}^{(\text{Alg})}$, $\mathbf{R}^{(\text{Alg})}$
1: Solve $\mathscr{F}_4^u$ to achieve $\mathbf{A}^\dagger$.
2: **if** $\mathbf{A}^\dagger$ is binary **then**
3: $\quad$ $\mathbf{A}^{(\text{Alg})} = \mathbf{A}^\dagger$
4: **else**
5: $\quad$ **for** $k = 1$ to $k = K$ **do**
6: $\quad\quad$ $a_{km}^{(\text{Alg})} = 1$ with probability $a_{km}^\dagger$ with exclusive manner based on constraint (38a)
7: $\quad$ **end for**
8: **end if**
9: Then,

$$r_{1,0}^{(\text{Alg})} = \ldots = r_{|\mathcal{K}_0|,0}^{(\text{Alg})} = \frac{W^u}{|\mathcal{K}_0|}$$

$$r_{1m}^{(\text{Alg})} = \ldots = r_{|\mathcal{K}_m|m}^{(\text{Alg})} = \frac{B_m^{\text{fr}}}{|\mathcal{K}_m|}, \forall m \text{ such that } |\mathcal{K}_m| \neq 0.$$
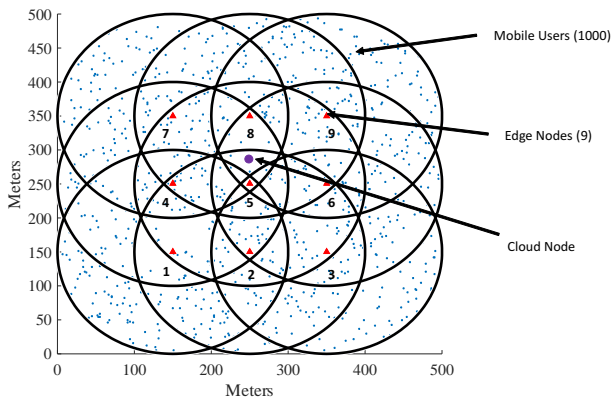

Fig. 4. The network setup.

Proof. Let $y^\dagger$ denote the optimal value of the $\mathscr{F}_4^u$ and $U = 6\ln(K)y^\dagger$. For $\delta > 0$, applying the Chernoff bound, we have

$$\Pr\left[T_m^{\text{u}} > (1+\delta)y^\dagger\right] \leq e^{-\frac{\delta^2 y^\dagger}{2+\delta}}. \quad (51)$$

Next, we upper bound the right hand side of the above inequality by $1/K^2$. In order to achieve this condition, the δ value must satisfy the following condition:

$$\delta \geq \frac{\ln(K)}{y^\dagger} + \sqrt{\frac{\ln^2(K)}{y^{\dagger 2}} + \frac{4\ln(K)}{y^\dagger}}, \quad (52)$$

with the assumption $y^\dagger > \ln(K)$. The above condition holds if we pick $\delta = \frac{2\ln(K)}{y^\dagger} + 2$. Then, by applying the union bound, we get

$$\Pr[\exists m | T_m^{\text{u}} > (1+\delta)y^\dagger] \leq \sum \Pr[T_m^{\text{u}} > (1+\delta)y^\dagger]$$
$$\leq \frac{M+1}{K^2} \leq \frac{1}{K}. \quad (53)$$

Consequently, with high probability $1 - \frac{1}{K}$, the resulting aggregation latency is at most $1 + \delta = \frac{2\ln K}{y^\dagger} + 3$ times worse than that of the optimal solution. $\square$

In practice, the number of users K is large, for example 5000. It is reasonable to assume that the lower bound of

latency is higher than $\ln K = 8.5$ s. In the simulations below, we observe that the assumption holds for practical settings.

7.2 Without In-Network Computation Protocol

In this section, we aim to find the lower bound of network latency when the proposed in-network computation protocol is not considered. First, we introduce an auxiliary variable y such that

$$y \geq \max\left\{ \max_m \left\{ D\frac{\sum_{k \in \mathcal{K}_m} a_{km}}{B_m^{\text{fr}}} + D\frac{\sum_{k \in \mathcal{K}_m} a_{km}}{B_m^{\text{bk}}} \right\}, \right.$$
$$\left. D\frac{\sum_{k \in \mathcal{K}_0} a_{k0}}{W^u} \right\}. \quad (54)$$

Hence, problem $\mathscr{Q}_2^u$ is equivalently transformed to

$$\mathscr{Q}_3^u: \min_{y, \{a_{km}\}} y,$$
$$\text{s.t. } y \geq D\frac{\sum_{k \in \mathcal{K}_m} a_{km}}{B_m^{\text{fr}}} + D\frac{\sum_{k \in \mathcal{K}_m} a_{km}}{B_m^{\text{bk}}}, \forall m, \quad (55a)$$
$$y \geq D\frac{\sum_{k \in \mathcal{K}_0} a_{k0}}{W^u}, \quad (55b)$$
$$(38a) \text{ and } (38d).$$

Like Algorithm 2, we relax the variables a_{km} to be fractional, rather than integer. The Linear Relaxation formulation of $\mathscr{Q}_3^u$ can be expressed as follows:

$$\mathscr{Q}_4^u: \min_{y, \{a_{km}\}} y,$$
$$\text{s.t. } (55a), (55b), (38a), \text{ and } a_{km} \in [0, 1]. \quad (56)$$

Let $y^{\dagger\dagger}$ denote the optimal solution of $\mathscr{Q}_4^u$. $y^{\dagger\dagger}$ is the lower bound of the uplink latency without the proposed in-network computation protocol. We use $y^{\dagger\dagger}$ to proxy the aggregation latency when INA is not considered.

8 NUMERICAL RESULTS

In this section, we carry out simulations to evaluate the performance of the proposed frameworks. We consider a similar setup as in [30], depicted in Fig. 4. Here, $M = 9$ edge nodes are regularly deployed on a grid network inside a 500×500 m^2 area. $K = 1000$ mobile users are uniformly distributed over the edge nodes' coverage regions (each of 150m radius). All mobile users are within the coverage of the cloud node. For each edge node m, we set the uplink fronthaul capacity to $B_m^{\text{fr}} = 1$Gbps, the backhaul capacity to $B_m^{\text{bk}} = 1$Gbps. These settings are inspired by WiFi IEEE 802.11ac standards [45], and data centers interconnection using optical fibers [46]. We also set the cloud uplink and downlink capacities $W^u = W^d = 2$Gbps. As in [25], we set $t_{\min}^{\text{cp}}$ and $t_{\max}^{\text{cp}}$ at 0.2s and 80s, respectively. For model aggregation, by default, we investigate our system using ResNet152's model size, i.e., $D = 232$ MB [42]. In later simulations, we also investigate our system with different model sizes.

8.1 Algorithm Comparison - Latency Reduction

Fig. 5 compares the one-iteration training time of different algorithms versus the number of users K. The proposed

TABLE 2
Default Parameter Setup.

Parameter	Value
M	9
K	1000
EN's coverage	150 m
EN to EN distance	100 m
B_m^{fr}	1Gbps
B_m^{bk}	1Gbps
W^u	2 Gbps
W^d	2 Gbps
$t_{\min}^{\mathrm{cp}}, t_{\max}^{\mathrm{cp}}$	0.2 s, 8s
D	232 MB

TABLE 3
Time Complexity.

Frameworks	Complexity
"All to Cloud"	$O(1)$
"Select Highest Capacity EN"	$O(KM)$
"Select Nearest EN"	$O(KM)$
"INC Solution"	$O(\nu^{3.5}\Omega^2)$
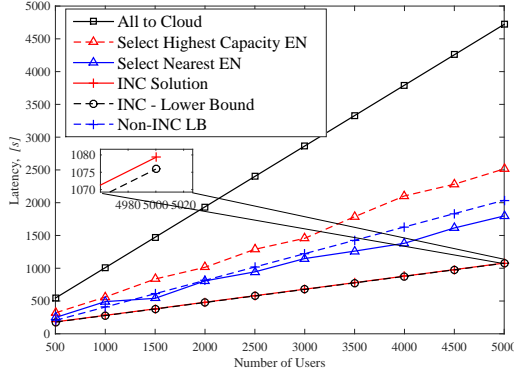
Fig. 5. Algorithm comparison with respect to different number of users.

Fig. 6. Algorithm comparison with respect to different models.

INC protocol is compared with three other baseline methods, namely:

1. "Only Cloud": All K users' models are directly aggregated via their direct link to the cloud. Here, models are not forwarded to any edge nodes (ENs).
2. "Select Highest Capacity EN": In this greedy-manner framework, each user selects the neighbor EN with the highest uplink data rate capacity.
3. "Select Nearest EN": In this greedy-manner framework, each user selects the nearest neighbor EN.
4. "INC Solution": K users can associate with the cloud node and edge nodes with INC protocol. The network routing problem $\mathscr{F}_3^u$ is solved by using Algorithm 2.
5. "INC-Lower Bound": In this scenario, we use Linear Relaxation to solve $\mathscr{F}_3^u$. This scenario will provide the lower bound of network latency if the proposed INC protocol is considered.
6. "Non-INC LB": K users can associate with the cloud node and edge nodes without the INC protocol. By using Linear Relaxation to solve $\mathscr{Q}_3^u$, this method will provide the lower bound of network latency if INA process is not implemented at edge nodes and the cloud node.

Here, users are scheduled by following Algorithm 1. As can be observed in Fig. 5, our proposed algorithm can achieve near optimal performance. When $K = 5000$, the latency obtained by the proposed solution is approximately 0.7% higher than that of the *INC LB*. It implies that our proposed solution can achieve the performance almost the same as that of the lower bound solution even with a high number of users. *Only Cloud* has the worst performance.

For example, when $K = 5000$, the network latency of *Only Cloud* is 4721s which is is 1.8 times higher than that of the second worst one, *Select Highest Capacity EN*, 2516s. The second framework, "Select Highest Capacity EN", provides the second lowest performance due to resource contention. Since users select the highest EN with the highest capacity EN of their neighbor EN set, user locating in the overlapping region of the same set of ENs will select the same EN. As a result, there is a resource contention in the chosen EN. The other greedy-manner, "Select Nearest EN", can achieve better performance. Because users' positions are randomly distributed, the number of users in each EN's coverage is approximately the same. Thus, this framework can achieve an approximated load-balancing solution which is significantly better than the second framework. Last but not least, the gaps between our proposed algorithm with other baselines increase as the number of users K increases. This clearly shows that our proposed solution is significantly beneficial for very large scale federated learning networks.

Since our algorithm requires to compute the Linear Relaxation results before conducting randomization, its time complexity is higher than those of the baseline methods. Table 3 summarizes the time complexity of the four frameworks.

In Fig. 6, consider *Only Cloud*, *Non-INC LB* and our *INC solution*, we evaluate the network latency with or without user scheduling mechanism in different models. In the case when the user scheduling mechanism is not considered, all users need to wait until the slowest user finishing it local processing step. They are VGG16, ResNet152, Xception and DenseNet121 whose model sizes are 528MB, 232MB, 88MB and 33MB, respectively. The distribution of $p(t^{\mathrm{cp}})$ is remained unchanged since we want to focus only on the variation of aggregation latency as D changes. Here, we choose
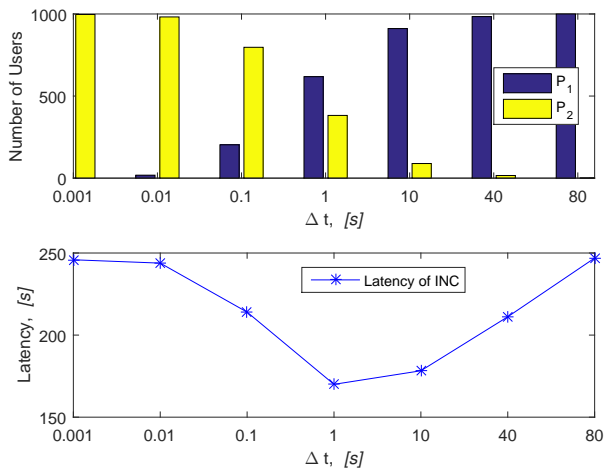
Fig. 7. Latency of one learning round and the number of users in each partition change w.r.t Δt.

the default setting with $K = 1000$. We observe that our user scheduling mechanism can mitigate straggler effects due to slow workers. For example, when we consider ResNet152 model, *INC solution* saves 28.49% in comparison with the case where user scheduling mechanism is not considered. Moreover, since the aggregation latency decreases as the model's size D decreases, we observe that the latency saving of *INC solution w.o User Scheduling* to *INC solution w. User Scheduling* increases then decreases. For example, the saving is 14.63%, 28.49%, 38.18 and 20.43% for VGG16, ResNet152, Xception and DenseNet121, respectively. It implies that as the communication latency decreases, the contribution of the slowest node's computing delay increases. Until a certain value D, the slowest node's computing delay, which is fixed in this simulation, becomes the major part of the whole network delays. Hence, the saving decreases again.

8.2 Impacts of Δt

In this subsection, we investigate the impacts of Δt on the latency of one learning round. Here, we used the default setting with number of users $K = 1000$. As we can see in the above figure, as Δt increases, the number of users in P_1 increases because all users with $t_k^{\mathrm{cp}} \leq t_{\min}^{\mathrm{cp}} + \Delta t$ are assigned to P_1. It is also the reason why the number of users in P_2 decreases. As can be seen in Fig. 7, we observe that with small Δt, most of users wait until $t_{\max}^{\mathrm{cp}}$ to be collected in partition P_2. Thus, we can see high latency with small Δt. As Δt increases, more users go to P_1, thus the latency decreases. When Δt is big enough, the latency rises again because most of users are in P_1 and their local models have to wait after $t_{\min}^{\mathrm{cp}} + \Delta t$ to be collected. When $\Delta t = t_{\max}^{\mathrm{cp}} = 80$s, the latency is highest since all users' models start to be collected after $t_{\max}^{\mathrm{cp}}$.

8.3 Traffic and Computation Reduction at the Cloud Node

In this subsection, using ResNet152's model setting, we investigate the uplink traffic and the number of models needed to be aggregated at the cloud node in one learning
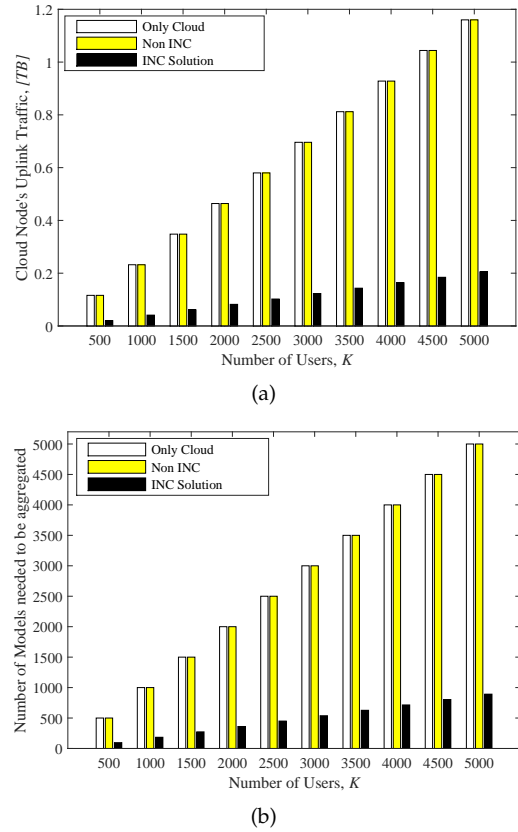

(a)

(b)

Fig. 8. Cloud node's uplink traffic and computing load.

iteration. We compare three schemes: *Only Cloud*, *Non-INC LB* and our *INC solution*. The number of models needed to be aggregated at the cloud node is proportional to the number of computations here. In Figs. 8, the uplink traffic and the number of computations of *Non-INC LB* at the cloud node is equal to those of *Only Cloud*. It is because all models need to be sent to the cloud before being aggregated and edge nodes only forward the models from users to the cloud without INA process. Meanwhile, with *INC solution*, the two metrics are significantly reduced. For example, when $K = 5000$, the traffic is 0.2TB for our scheme and 1.16TB for the two others. Our scheme achieves more than 5 times lower traffic than the others. In this simulation, with ResNet152, the number of parameters of each local models is $60,419,944$. As a result, without the proposed INA process, the cloud node needs to aggregate K models whose sizes are more 60 millions elements. This will consume a huge amount of processing and memory resources. As can be seen, the *INC solution* can reduce the number of models needed to be aggregated at the cloud by more than 5 times.

8.4 Impacts of the number of additional edge nodes' connectivities

In this subsection, we investigate the impacts of the number extra edge nodes users on the straggler effects due to bad communication links. In this simulation, our system suffers straggler effects when a user's all wireless connections are bad. As a result, its model is not able to be aggregated at the cloud node. Without loss of generality, we assume that the networking components of servers, edge nodes or cloud
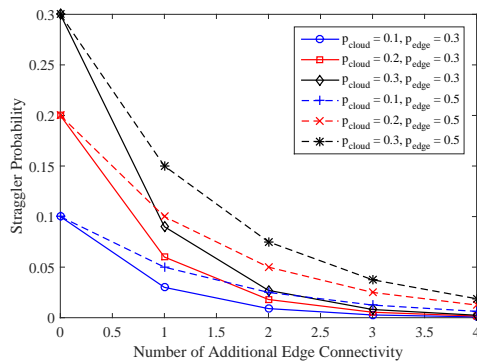
Fig. 9. Straggler effect's probability.

REFERENCES

[1] T. Q. Dinh, D. N. Nguyen, D. T. Hoang, T. V. Pham, and E. Dutkiewicz, "Enabling large-scale federated learning over wireless edge networks," in *in Proc. IEEE GLOBECOM*, Madrid, Spain, Dec. 2021.

[2] W. Saad, M. Bennis, and M. Chen, "A vision of 6G wireless systems: Applications, trends, technologies, and open research problems," *IEEE Netw.*, vol. 34, no. 3, pp. 134–142, May 2020.

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

[4] 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. (AISTATS)*, vol. 54, Apr. 2017, pp. 1273–1282.

[5] S. Wiedemann, H. Kirchhoffer, S. Matlage, P. Haase, A. Marban, T. Marin, D. Neumann, T. Nguyen, H. Schwarz, T. Wiegand, D. Marpe, and W. Samek, "Deepcabac: A universal compression algorithm for deep neural networks," *IEEE J. Sel. Topics Signal Process.*, vol. 14, no. 4, pp. 700–714, May 2020.

[6] Y. Lin, S. Han, H. Mao, Y. Wang, and W. J. Dally, "Deep gradient compression: Reducing the communication bandwidth for distributed training," 2020. [Online]. Available: arXiv:1712.01887

[7] T. Chen, G. Giannakis, T. Sun, and W. Yin, "Lag: Lazily aggregated gradient for communication-efficient distributed learning," in *Advances in Neural Information Processing Systems (NIPS)*, Montreal, Canada, Dec. 2018, pp. 5050–5060.

[8] W. Xia, T. Q. S. Quek, K. Guo, W. Wen, H. H. Yang, and H. Zhu, "Multi-armed bandit-based client scheduling for federated learning," *IEEE Trans. Wireless Commun.*, vol. 19, no. 11, pp. 7108–7123, Nov. 2020.

[9] H. H. Yang, Z. Liu, T. Q. S. Quek, and H. V. Poor, "Scheduling policies for federated learning in wireless networks," *IEEE Trans. Commun.*, vol. 68, no. 1, pp. 317–333, Jan. 2020.

[10] M. Chen, H. V. Poor, W. Saad, and S. Cui, "Convergence time minimization of federated learning over wireless networks," in *Proc. IEEE ICC*, Dublin, Ireland, Jul. 2020, pp. 1–6.

[11] I. Tenney, D. Das, and E. Pavlick, "BERT rediscovers the classical NLP pipeline," in *Proc. 57th Annu. Meeting Assoc. for ACL*, Florence, Italy, Jul. 2019, pp. 4593–4601.

[12] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," 2015. [Online]. Available: arXiv:1409.1556

[13] Wojciech Samek and Deniz Gunduz, "Distributed deep learning: Concepts, methods & applications in wireless networks," 2020, IEEE GLOBECOM 2020 Tutorial. [Online]. Available: http://www.federated-ml.org/tutorials/globecom2020/part2.pdf

[14] Z. Qian, X. Chen, N. Kang, M. Chen, Y. Yu, T. Moscibroda, and Z. Zhang, "MadLINQ: Large-scale distributed matrix computation for the cloud," in *Proc. ACM Eur. Conf. Comput. Syst.*, Bern, Switzerland, Apr. 2012, pp. 197–210.

[15] E. Jonas, Q. Pu, S. Venkataraman, I. Stoica, and B. Recht, "Occupy the cloud: Distributed computing for the 99%," in *Proc. Symp. Cloud Comput.*, Santa Clara, California, USA, Sep. 2017, pp. 445–451.

[16] E. Fasolo, M. Rossi, J. Widmer, and M. Zorzi, "In-network aggregation techniques for wireless sensor networks: a survey," *IEEE Wireless Commun.*, vol. 14, no. 2, pp. 70–87, Apr. 2007.

[17] J. Dean and S. Ghemawat, "MapReduce: Simplified data processing on large clusters," *Commun. ACM*, vol. 51, no. 1, pp. 107–113, Jan. 2008.

[18] G. Malewicz, M. H. Austern, A. J. Bik, J. C. Dehnert, I. Horn, N. Leiser, and G. Czajkowski, "Pregel: A system for large-scale graph processing," in *Proc. ACM SIGMOD*, Indianapolis, Indiana, USA, 2010, pp. 135–146.

[19] Y. Yu, M. Isard, D. Fetterly, M. Budiu, U. Erlingsson, P. K. Gunda, and J. Currey, "DryadLINQ: A system for general-purpose distributed data-parallel computing using a high-level language," in *Proc. USENIX Conf. Oper. Syst. Des. Implement. (OSDI)*, Dec. 2008, pp. 1–14.

[20] M. Chen and Y. Hao, "Task offloading for mobile edge computing in software defined ultra-dense network," *IEEE J. Sel. Areas Commun.*, vol. 36, no. 3, pp. 587–597, Mar. 2018.

node, have probabilities of being faulted. We define p_{cloud} as the outage probability of the cloud. Similarly, we define p_{edge} as the outage probability of a given edge node (here we assume all edge nodes have the same outage probability). Thus, the probability that the system suffers the straggler effect P_s is computed as

$$\mathrm{P_s} = p_{\text{cloud}} p_{\text{edge}}^v, \tag{57}$$

where v is the number of extra edge nodes users can connect. As can be observed in Fig. 9, increasing the number of additional edge connectivities can significantly mitigate straggler effects. With $p_{\text{cloud}} = 0.3$, we can decrease the straggle effects 4 times and 12 times by providing two additional edge connections for users, with the well-being probabilities of edge node's networking component are 0.5 and 0.7, respectively.

9 CONCLUSION AND FUTURE DIRECTIONS

In this article, we proposed a novel edge network architecture to decentralize the communications and computing burden of cloud node in Federated Learning. To that end, we designed an in-network computation protocol (INC) consisting of a user scheduling mechanism, an in-network aggregation process (INA), and an routing algorithm. The in-network aggregation process, which is implemented at edge nodes and cloud node, can adapt two typical methods to solve the distributed machine learning problems. Under the proposed in-network aggregation (INA) framework, we then formulated a joint routing and resource optimization problem, aiming to minimize the aggregation latency. The problem is proved to be NP-Hard. We then derived its near-optimal solution using random rounding with proven performance guarantee. Simulation results showed that the proposed algorithm can achieve more than 99 % of the optimal solution and significantly outperforms all other baseline schemes without INA. The proposed scheme becomes even more effective (in reducing the latency and straggler effects) when more edge nodes are available. Moreover, we also showed that the INA framework not only help reduce training latency in FL but also reduce significantly reduce the traffic load to the cloud node. By embedding the computing/aggregation tasks at edge nodes and leveraging the multi-layer edge-network architecture, the INA framework can enable large-scale FL.

[21] C. Ma, J. Konecny, M. Jaggi, V. Smith, M. I. Jordan, P. Richtarik, and M. Takac, "Distributed optimization with arbitrary local solvers," *Optimization Methods and Software*, vol. 32, no. 4, pp. 813–848, Jul. 2017.

[22] M. Jaggi, V. Smith, M. Takac, J. Terhorst, S. Krishnan, T. Hofmann, and M. I. Jordan, "Communication-efficient distributed dual coordinate ascent," in *Advances in Neural Information Processing Systems (NIPS)*, vol. 27, Montreal, Canada, Dec. 2014, pp. 3068–3076.

[23] K. Yang, T. Jiang, Y. Shi, and Z. Ding, "Federated learning via over-the-air computation," *IEEE Trans. Wireless Commun.*, vol. 19, no. 3, pp. 2022–2035, Jan. 2020.

[24] X. Cao, G. Zhu, J. Xu, and K. Huang, "Optimized power control for over-the-air computation in fading channels," *IEEE Trans. Wireless Commun.*, vol. 19, no. 11, pp. 7498–7513, Aug. 2020.

[25] N. H. Tran, W. Bao, A. Zomaya, M. N. H. Nguyen, and C. S. Hong, "Federated learning over wireless networks: Optimization model design and analysis," in *Proc. IEEE INFOCOM*, Paris, France, Jun. 2019, pp. 1387–1395.

[26] S. Wang, T. Tuor, T. Salonidis, K. K. Leung, C. Makaya, T. He, and K. Chan, "Adaptive federated learning in resource constrained edge computing systems," *IEEE J. Sel. Areas Commun.*, vol. 37, no. 6, pp. 1205–1221, Mar. 2019.

[27] L. Liu, J. Zhang, S. H. Song, and K. B. Letaief, "Client-edge-cloud hierarchical federated learning," in *Proc. IEEE ICC*, Dublin, Ireland, Jul. 2020, pp. 1–6.

[28] S. Shalev-Shwartz and T. Zhang, "Stochastic dual coordinate ascent methods for regularized loss," *J. Mach. Learn. Res.*, vol. 14, no. 1, pp. 567–599, Feb. 2013.

[29] L. Jiao, A. M. Tulino, J. Llorca, Y. Jin, and A. Sala, "Smoothed online resource allocation in multi-tier distributed cloud networks," *IEEE/ACM Trans. Netw.*, vol. 25, no. 4, pp. 2556–2570, Aug. 2017.

[30] K. Poularakis, J. Llorca, A. M. Tulino, I. Taylor, and L. Tassiulas, "Service placement and request routing in mec networks with storage, computation, and communication constraints," *IEEE/ACM Trans. Netw.*, vol. 28, no. 3, pp. 1047–1060, Jun. 2020.

[31] T. Q. Dinh, B. Liang, T. Q. S. Quek, and H. Shin, "Online resource procurement and allocation in a hybrid edge-cloud computing system," *IEEE Trans. Wireless Commun.*, vol. 19, no. 3, pp. 2137–2149, Mar. 2020.

[32] T. T. Vu, N. V. Huynh, D. T. Hoang, D. N. Nguyen, and E. Dutkiewicz, "Offloading energy efficiency with delay constraint for cooperative mobile edge computing networks," in *in Proc. IEEE GLOBECOM*, 2018, pp. 1–6.

[42] Keras. Keras applications. [Online]. Available: https://keras.io/api/applications/

[33] S. H. *et al.*, "Multi-stage hybrid federated learning over large-scale d2d-enabled fog networks," *IEEE/ACM Trans. on Netw.*, vol. Earl Access, 2022.

[34] J. woo Lee *et al.*, "TornadoAggregate: Accurate and Scalable federated learning via the ring-based architecture," in *Conference on Artificial Intelligence (AAAI)*, 2021.

[35] A. Bellet, A. Kermarrec, and E. Lavoie, "D-cliques: Compensating for data heterogeneity with topology in decentralized federated learning," 2021. [Online]. Available: arXiv:2104.07365

[36] Y. Saputra, D. Nguyen, H. Dinh, Q.-V. Pham, E. Dutkiewicz, and W.-J. Hwang, "Federated learning framework with straggling mitigation and privacy-awareness for ai-based mobile application services," *IEEE Transactions on Mobile Computing*, pp. 1–1, 2022.

[37] Y. M. Saputra, H. T. Dinh, D. Nguyen, L.-N. Tran, S. Gong, and E. Dutkiewicz, "Dynamic federated learning-based economic framework for internet-of-vehicles," *IEEE Transactions on Mobile Computing*, pp. 1–1, 2021.

[38] Y. M. Saputra, D. T. Hoang, D. N. Nguyen, and E. Dutkiewicz, "A novel mobile edge network architecture with joint caching-delivering and horizontal cooperation," *IEEE Transactions on Mobile Computing*, vol. 20, no. 1, pp. 19–31, 2021.

[39] C.-H. Zhang, "Nearly unbiased variable selection under minimax concave penalty," *Ann. Statist.*, vol. 38, no. 2, pp. 894–942, Apr. 2010.

[40] L. Muchnik, S. Pei, L. C. Parra, S. D. S. Reis, J. S. Andrade Jr, S. Havlin, and H. A. Makse, "Origins of power-law degree distribution in the heterogeneity of human activity in social networks," *Sci. Rep.*, vol. 3, no. 1, p. 1783, May 2013.

[41] D. R. Bild, Y. Liu, R. P. Dick, Z. M. Mao, and D. S. Wallach, "Aggregate characterization of user behavior in twitter and analysis of the retweet graph," *ACM Trans. Internet Technol.*, vol. 15, no. 1, pp. 1–24, Mar. 2015.

[43] M. L. Pinedo, *Scheduling: Theory, Algorithms, and Systems*, 3rd ed. Springer Publishing Company, Incorporated, 2008.

[44] Y. Nesterov and A. Nemirovskii, *Interior-point polynomial algorithms in convex programming*. SIAM, 1994.

[45] E. Khorov, A. Kiryanov, A. Lyakhov, and G. Bianchi, "A tutorial on IEEE 802.11ax high efficiency WLANs," *IEEE Commun. Surveys Tuts.*, vol. 21, no. 1, pp. 197–216, Sep. 2019.

[46] Y. Cheng, "Optical interconnects for next generation datacenters," Ph.D. dissertation, KTH Royal Institute of Technology, 2019.

1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60

In-network Computation for Large-scale Federated Learning over Wireless Edge Networks

Thinh Quang Dinh, Diep N. Nguyen, Dinh Thai Hoang,
Pham Tran Vu, and Eryk Dutkiewicz

Paper ID: TMC-2021-09-0779.R1 (Minor Revision)

We would like to thank the editor for the time and effort to gather insightful reviews for our submission to the IEEE Transactions on Mobile Computing (Paper ID: TMC-2021-09-0779.R1: *In-network Computation for Large-scale Federated Learning over Wireless Edge Networks*). We also would like to thank the reviewers for their thoughtful and constructive comments that significantly help improve the quality of the paper in this revised version. In response to the comments from the reviewers, we have carefully revised the work to better highlight the work's contributions. Additionally, we have added new paragraph, following the suggestion of the editor and the Reviewer 1 in the Section II. The followings are the detailed answers to the comments.

<u>Note</u>: In the response, we provide a separate list of references used to support our answers to the reviewers' comments. Moreover, we refer to the references and equations locally in which the numbers may be different from those in the revised manuscript.

2 _____ _

1 Responses to the editors

> **Comment 1**
>
> According to Reviewer 1's remaining comment, it would be nice if the authors can add some discussion (e.g., a paragraph) to highlight the main challenges and novelty in this work.

Response: We thank the editors for the time and effort to gather insightful reviews. We have carefully revised the paper based on the Reviewer 1' remaining comment. As suggested by the Editor, we are going to elaborate the last paragraphs of Section II to highlight the main challenges and novelty in this work with other studies related to decentralizing FL process. The key point here is that our work leverages edge networks to enable large-scale FL and liberate FL from its canonical start topology. All existing works that aim to decentralize the FL process did not leverage edge networks nor address its associated challenges (e.g., routing, resource allocation, and the straggling effect). Again, we highly appreciate the Editor's time and efforts in coordinating the review process and giving us thoughtful comments/suggestions.

2 Responses to the Comments from Review 1

> **Comment 1**
>
> My major concerns have been addressed in this version. However, this idea of proposed decentralized aggregation structure to replace the naive centralized (star) structure is not much novel. Many works have been proposed to optimize the decentralized structure, e.g., hierarchical (tree based) structure, ring structure, even random graph structure. Besides, the performance results in this paper are obtained from simulations, and real implementation and deployment of the proposed solutions are missing, which limits the contribution of this work.

Response: We first would like to thank the reviewer for your valuable comment. The comment helps us to improve the quality of our paper in the second revised manuscript. We agree with the Reviewer that many works have been proposed to optimize the decentralized structure, e.g., hierarchical (tree based) structure, ring structure, even random graph structure. However, the key novelty of our work here is to leverage edge networks to enable large-scale FL and liberate FL from its canonical start topology. All aforementioned works that aim to decentralize the FL process did not leverage edge networks nor address the associated challenges (e.g., routing, resource allocation, and the straggling effect).

Moreover, since communication and computation cost are the two main bottlenecks of FL systems, many works studied different approaches to decentralize the aggregation process of Federated Learning (FL), such as hierarchical (tree based) structure [R2], ring structure [R3], or even random graphs [R4]. However, below are challenges of FL systems that have not been tackled.

2. Responses to the Comments from Review 1

<div align="right">3</div>

1. First of all, there are two main categories of learning methods for FL, primal methods like FedAvg [R1] or primal-dual methods like CoCoA [R5]. However, in those mentioned works, [R2-R4], the proposed decentralized aggregation processes can be considered as a variant of either FedAvg or CoCoA. Similarly, most of FL works using star topology also considered either FedAvg or CoCoA [R6-R8]. In our paper, the proposed in-network aggregation process, a component of our In-network Computation (INC) solution, can adapt to both primal and primal-dual methods.

2. Second, in those works [R2-R4], authors mostly focused on designing aggregation processes while network resource management problems were not considered. In our papers, we also proposed a network routing and resource allocation algorithm to minimize the system latency. Beside intensive numerical results showing that our proposed algorithm can achieve near optimal performance, we also would like to highlight that the theoretical performance bound of our algorithm is also provided.

3. Last but not least, in those works [R2-R4], slow workers that stagnate the learning systems, i.e. straggler effect, are not addressed. For that, we propose a user scheduling scheme to mitigate the effect as the third component of our INC solution.

Regarding the methods to evaluate the proposed framework, we largely agree with the reviewer that it would be ideal to validate it with real implementation and deployment. However, given our limited resources, we had to rely on simulations. We would like to note that such a evaluation method is widely adopted in the research community [R1-R8]. Moreover, the theoretical performance bound of our algorithm is also provided. Therefore, we wish to affirm that our theoretical and numerical results here are significant and reliable to support the underlying claims. As the next step, we are working with Amazon to explore testbed implementation of the proposed framework.

The above explanation have been incorporated into this paper by elaborating the last paragraphs in Section II. The modified texts are highlighted in the second revised manuscript on pages 2-3:

Edge computing is a potential infrastructure which can help to address the high communication cost of FL. Even though there are many works where edge networks are complementary to the cloud [29]–[32], the capacities of edge networks are under explored in existing FL works [4]–[10], [23]–[26]. To decentralize and redistribute the model aggregation process of FL at cloud server to edge nodes in edge networks, this article leverages the âin-network computing" concept. Although such a concept has been well-studied for traditional machine learning paradigms where usersâ raw data are collected and stored at big data clusters [17]â[19], INC for distributed learning paradigms, e.g., FL, especially with the aid of edge computing, has not been visited.

There have been early works proposing decentralizing aggregation processes for different network architectures such hierarchical topology [27], [33], ring topology [34], and random graph [35]. However, in those works, the proposed decentralized aggregation processes can be considered as variants of either FedAvg or CoCoA. In our paper, the proposed in-network aggregation process, a component of our INC solution, can adapt to both primal and primal-dual methods. Moreover, existing works mostly focused on designing aggregation processes while network resource management and routing problems were not jointly considered. For example, in [27], users are assumed to connect with a single edge node without any alternative

4   _

links. In reality with dense edge networks, MUs can associate to more than one edge nodes. Such a practical scenario calls for optimal network routing and resource allocation solution, aiming at minimizing the system latency. In this article, under the proposed INC, we formulate a joint routing and resource optimization problem, aiming to minimize the aggregation latency. The problem turns out to be NP-hard, and thus we design a polynomial time routing algorithm which can achieve near optimal performance with a proven theoretical bound. Last but not least, to address slow workers that stagnate the learning systems in these existing works [4]–[10], [23]–[27], [36], [37] (known as the straggler effect), we propose a user scheduling scheme that is the third component of our INC solution. In short, the key novelty of our work is to leverage edge networks to enable large- scale FL and liberate FL from its canonical start topology. All aforementioned works that aim to decentralize the FL process did not leverage edge networks nor address the associated challenges (e.g., routing, resource allocation, and the straggling effect).

3. Responses to the Comments from Review 2   5

3   Responses to the Comments from Review 2

> **Comment 1**
>
> I found the revision satisfactory and suggest accepting the paper in its current form

6 —

4 Responses to the Comments from Review 3

Comment 1

The authors have addressed all my previous comments.

References

[R1] 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. (AISTATS)*, vol. 54, Apr. 2017, pp. 1273–1282.

[R2] S. Hosseinalipour *et al.*, "Multi-Stage Hybrid Federated Learning Over Large-Scale D2D-Enabled Fog Networks," *IEEE/ACM Trans. on Netw., Early Access*

[R3] Jin-woo Lee *et al.* "TornadoAggregate: Accurate and Scalable Federated Learning via the Ring-Based Architecture" in *AAAI*, 2021.

[R4] Aurélien Bellet and Anne-Marie Kermarrec and Erick Lavoie, "D-Cliques: Compensating for Data Heterogeneity with Topology in Decentralized Federated Learning," 2021. [Online]. Available: arXiv:2104.07365

[R5] M. Jaggi, V. Smith, M. Takac, J. Terhorst, S. Krishnan, T. Hofmann, and M. I. Jordan, "Communication-efficient distributed dual coordinate ascent," in *Advances in Neural Information Processing Systems (NIPS)*, vol. 27, Montreal, Canada, Dec. 2014, pp. 3068–3076.

[R6] W. Xia, T. Q. S. Quek, K. Guo, W. Wen, H. H. Yang, and H. Zhu, "Multi-armed bandit-based client scheduling for federated learning," *IEEE Trans. Wireless Commun.*, vol. 19, no. 11, pp. 7108–7123, Nov. 2020.

[R7] N. H. Tran, W. Bao, A. Zomaya, M. N. H. Nguyen, and C. S. Hong, "Federated learning over wireless networks: Optimization model design and analysis," in *Proc. IEEE INFOCOM*, Paris, France, Jun. 2019, pp. 1387–1395.

[R8] S. Wang, T. Tuor, T. Salonidis, K. K. Leung, C. Makaya, T. He, and K. Chan, "Adaptive federated learning in resource constrained edge computing systems," *IEEE J. Sel. Areas Commun.*, vol. 37, no. 6, pp. 1205–1221, Mar. 2019.

In-network Computation for Large-scale Federated Learning over Wireless Edge Networks

IEEE Transactions on Mobile Computing (TMC)

Differences between the Current Submission and the IEEE GLOBECOM 2021 Paper [1]

Thinh Quang Dinh, Diep N. Nguyen, Dinh Thai Hoang,

Pham Tran Vu, and Eryk Dutkiewicz

In this letter, we explain the differences between the current TMC submission and its preliminary version that has been accepted at the IEEE GLOBECOM 2021 Conference [1]. This TMC submission has extended the preliminary findings of the IEEE GLOBECOM paper significantly with new aspects, theoretical, numerical results, and insights. Please note that the TMC submission is 13 pages containing the appendices (in double-column format) whereas the IEEE GLOBECOM version is 6 pages (in double-column format). The major differences between the two versions are summarized as follows:

1) We propose a new edge network architecture where users can directly associate with the cloud node. In the conference version, users can only associate with edge nodes where the cloud node aggregates a fixed number of aggregated models from edge nodes. As a result, it is more challenging to design the In-network aggregation process and the routing algorithm since the cloud node needs to aggregate not only aggregated models from edge nodes but also local models from users whose numbers may change over time.

2) We consider the distribution of users' local processing time following the power law. As a result, a user scheduling scheme is added into the In-network computation solution to reduce the training time in one iteration with theoretical bound. In the conference version, all users' models are aggregated at the same starting time. Thus, the user scheduling could mitigate the straggler effect in Federated Learning (FL).

3) We design a new In-network aggregation (INA) process in the In-network computation solution which adapting (a) the new edge network architecture, (b) both primal methods and primal-dual methods in Federated Learning. In the conference version, the INA process only adapts the primal

methods. The new INA process is suitable to not only FL systems using either primal or primal-dual methods but also FL systems initial using a primal method then using a primal-dual method later.

4) The routing algorithm is revised in order to adapt with the new edge network architecture and the recent added user scheduling scheme. We also derive the performance guarantee of the routing algorithm.

5) The TMC paper contains significant new figures and numerical results that have not been presented in [1]. Specifically, we have added simulation results to evaluate the benefits of the user scheduling schemes. Through the simulation results, we show that our proposed In-network computation can outperform baseline methods in training time, the cloud node's uplink traffic and computing load where the number of users are examined in larger scales. We provide an additional simulation to show to decreasing in straggler probability as the number of additional edge connectivity per user increases.

6) Last but not least, we have extensively revised the Section *Introduction* with many new literature reviews to strongly motivate our key contributions in using In-network Computation for large scale Federated Learning over Wireless Edge Networks. Several new references from top journals and conferences have also been added to ensure the timeliness of our manuscript as described in the Section I.

Finally, we also intensively revise the writing of [1] to reflect the above changes and new contributions in this TMC submission.

REFERENCES

[1] T. Q. Dinh, D. N. Nguyen, D. T. Hoang, T. V. Pham, and E. Dutkiewicz, "Enabling large-scale federated learning over wireless edge networks," in *in Proc. IEEE GLOBECOM*, Madrid, Spain, Dec. 2021.