Semi-Decentralized Federated Edge Learning with Data and Device Heterogeneity

Yuchang Sun, Jiawei Shao, *Student Member, IEEE*, Yuyi Mao, *Member, IEEE,* Jessie Hui Wang, *Senior Member, IEEE,* and Jun Zhang, *Fellow, IEEE*

***Abstract*—Federated edge learning (FEEL) emerges as a privacy-preserving paradigm to effectively train deep learning models from the distributed data in 6G networks. Nevertheless, the limited coverage of a single edge server results in an insufficient number of participated client nodes, which may impair the learning performance. In this paper, we investigate a novel FEEL framework, namely *semi-decentralized federated edge learning* (SD-FEEL), where multiple edge servers collectively coordinate a large number of client nodes. By exploiting the low-latency communication among edge servers for efficient model sharing, SD-FEEL incorporates more training data, while enjoying lower latency compared with conventional federated learning. We detail the training algorithm for SD-FEEL with three steps, including local model update, intra-cluster, and inter-cluster model aggregations. The convergence of this algorithm is proved on non-independent and identically distributed data, which reveals the effects of key parameters and provides design guidelines. Meanwhile, the heterogeneity of edge devices may cause the straggler effect and deteriorate the convergence speed of SD-FEEL. To resolve this issue, we propose an asynchronous training algorithm with a staleness-aware aggregation scheme, of which, the convergence is also analyzed. The simulations demonstrate the effectiveness and efficiency of the proposed algorithms for SD-FEEL and corroborate our analysis.**

***Index Terms*—Federated learning (FL), mobile edge computing (MEC), non-independent and identically distributed (non-IID) data, device heterogeneity.**

I. Introduction

The recent upsurge of Internet of Things (IoT) applications brings about a drastically increasing number of IoT devices, which is predicted to reach more than 30 billion by 2025 [3]. As a result, an unprecedented volume of data is generated and stored at the edge of the wireless network, which can facilitate the training of powerful machine learning (ML) models to empower various intelligent mobile applications. Meanwhile, as an enabler of the emerging IoT applications, the sixth generation (6G) of wireless networks is envisioned to provide ubiquitous artificial intelligence (AI) services anywhere at any time [4], [5]. To leverage the valuable data resources, a traditional approach is to upload them to a centralized server for training. However, this approach may not be suitable to support ubiquitous AI in 6G networks, since data uploading incurs heavy communication overhead and serious concerns on privacy leakage [6], [7]. Therefore, there is an emerging trend of pushing AI computing to the edge devices [8].

In 2017, Google proposed a privacy-preserving training paradigm, namely federated learning (FL) [9], where the client nodes (e.g., mobile and IoT devices) train models based on local data and periodically upload them to a Cloud-based parameter server (PS) for model aggregation. Since FL requires no data sharing, it substantially avoids privacy leakage. Nevertheless, model uploading between the client nodes and the Cloud-based PS introduces expensive communication costs, which degrades the efficiency of FL. Inspired by the emerging mobile edge computing (MEC) platforms [10], federated edge learning (FEEL) [11] has been proposed to overcome this bottleneck, where an edge-based PS (i.e., the edge server) is deployed to be located near the edge devices as a model aggregator. Despite its great promise in reducing the model uploading latency, the training efficiency of FEEL is far from satisfactory, since the number of client nodes accessible by a single edge server is insufficient due to its limited coverage.

To fully exploit the potential of FEEL, recent works started to incorporate multiple edge servers, each of which is in charge of a number of client nodes. Accordingly, the total amount of training data samples can be significantly increased. To speed up the training process, edge servers collaborate in training by sharing their models. A possible solution is to allow the Cloud to collect and aggregate the models from edge servers [12], [13], which still introduces excessive communication latency. In this work, we will investigate a novel FL architecture to support collaborated learning in 6G networks, namely semi-decentralized federated edge learning (SD-FEEL) [1], [14], which utilizes the low-latency communication among edge servers to realize effective and efficient model aggregation.

A. Related Works

The training efficiency of FEEL faces two bottlenecks, i.e., the limited communication bandwidth and straggler effect. On one hand, the client nodes and edge servers suffer from unstable wireless connections and thus frequent communications will cause a large training latency. To improve the learning efficiency, a control algorithm was proposed in [15] to adaptively determine the global model aggregation frequency given the available resources on client nodes. Besides, Shi

Part of this paper was accepted by the 2022 IEEE Wireless Communications and Networking Conference [1] and submitted to the 2022 IEEE International Conference on Communications [2]. (The corresponding author is J. Zhang.)

Y. Sun, J. Shao, and J. Zhang are with the Department of Electronic and Computer Engineering, the Hong Kong University of Science and Technology, Hong Kong (E-mail: {yuchang.sun, jiawei.shao}@connect.ust.hk, eejzhang@ust.hk). Y. Mao is with the Department of Electronic and Information Engineering, the Hone Kong Polytechnic University, Hong Kong (E-mail: yuyi-eie.mao@polyu.edu.hk). J. H. Wang is with the Institute for Network Sciences and Cyberspace, Tsinghua University, Beijing 100084, China, and also with the Beijing National Research Center for Information Science and Technology, Beijing 100084, China (E-mail: jessiewang@tsinghua.edu.cn).

et al. [16] solved a joint bandwidth allocation and device scheduling problem to maximize the learning performance of FL with the given training time. Moreover, gradient quantization and sparsification techniques were adopted to achieve communication-efficient FEEL [17], [18].

On the other hand, different types of edge devices have heterogeneous computational resources, e.g., processing speed, battery capacity, and memory usage [11]. Particularly, it may take a longer time for the client nodes with less computational resources (namely stragglers) to conduct the same amount of local training, which prolongs the total training time. This device heterogeneity issue can be problematic especially in large-scale implementations. A client selection algorithm for FEEL was proposed in [19], which eliminates the straggling client nodes from global model aggregation. Meanwhile, asynchronous FL has attracted much attention due to its effectiveness in dealing with device heterogeneity [20]–[22]. Xie *et al.* [20] proposed the first asynchronous training algorithm for FL, where the model aggregation is triggered once the PS receives an update from any client node. However, this scheme incurs more frequent communications between the PS and client nodes. To strike a balance between model improvement and training latency, a semi-asynchronous training algorithm for FL was proposed in [21], where the model aggregation is delayed until the PS collects a targeted number of local updates. However, the stale models uploaded from the straggling client nodes are less valuable to the global model aggregation and may even degrade the learning performance. The design in [22] relieved this issue by forcing some client nodes with up-to-date or deprecated local models to synchronize with the PS, while most client nodes stay asynchronous.

When being implemented over wireless networks, the aforementioned benefits of FEEL are hindered by the limited coverage of a single edge server. Recent works considered the cooperation among multiple edge servers in the training to further explore the potential of FEEL. The client-edge-cloud hierarchical FL system, namely HierFAVG, was investigated in [12], [13], where each edge server is responsible for aggregating the models of its associated client nodes, while the Cloud-based PS performs global aggregation to average the models from edge servers periodically. However, the communication with the Cloud-based PS still incurs a high latency. As an alternative to the Cloud-based aggregator, in [23], a fog node (i.e., an edge server) was selected by the Cloud to perform global model aggregation in each communication round. Such a design requires all the fog nodes to be fully connected and may suffer from single-point failure. Besides, a recent work [24] adopted the cell-edge users (i.e., client nodes) as bridges to share models between multiple edge servers. These client nodes may suffer from poor signal quality because of the cell-edge effect [25], which degrades the training performance. Table I summarizes the main characteristics of the above works.

Motivated by the efficient communication among edge servers, this paper investigates a novel FEEL system, *semi-decentralized federated edge learning* (SD-FEEL) [1], [14], where multiple edge servers collectively coordinate a large number of client nodes. The client nodes perform local model updates for several iterations and then upload the model to the associated edge server for *intra-cluster* model aggregation. After several times of intra-cluster model aggregations, multiple rounds of model exchanges and aggregations with the neighboring edge servers, which is termed as *inter-cluster* model aggregation, are conducted by edge servers. Such a semi-decentralized architecture can easily include a huge amount of client nodes and adequately explore their data at a very low cost. The recent work [14] investigated a similar design as SD-FEEL and analyzed its convergence, but only on independent and identically distributed (IID) data. Besides, they assumed only one round of communication among edge servers, which may degrade the training performance due to the model inconsistency among different edge servers.

B. Main Contributions

This paper investigates SD-FEEL at the edge of 6G wireless networks, accounting for the non-IID data and heterogeneous computational resources at different client nodes. Our main contributions are summarized as follows:

- We first propose a generic training algorithm for SD-FEEL, where client nodes and edge servers collaboratively train an ML model through local updates, intra-cluster, and inter-cluster model aggregations. To relieve the model inconsistency among edge clusters, edge servers are allowed to exchange and aggregate models multiple times in each round of inter-cluster model aggregation.
- We analyze the convergence of the training algorithm on non-IID data, with data heterogeneity among clients within the same edge cluster, as well as that among different edge clusters. Based on the analytical results, the impacts of intra-/inter-cluster aggregation periods are discussed. We also investigate how the network topology among edge servers affects the convergence rate. It is found that SD-FEEL converges slowly when the edge servers are sparsely connected, which, however, can be alleviated by multiple times of model sharings in inter-cluster model aggregation.
- To effectively combat device heterogeneity that may significantly hinder the training performance, we propose an asynchronous training algorithm for SD-FEEL, where edge servers can independently set deadlines for local computation at client nodes. Particularly, we design a staleness-aware aggregation scheme to account for the device heterogeneity. For analysis, we decompose the variance incurred by asynchronous training and prove the convergence of the proposed algorithm.
- We conduct extensive simulations on two image classification tasks. The results demonstrate the benefits of SD-FEEL in achieving faster convergence without sacrificing model quality. The simulations also verify our discussions on the effects of various key parameters. Especially, increasing the number of rounds of inter-server communications is found to be an effective strategy to reduce the model inconsistency among edge clusters. Besides, further experiments show that the proposed asynchronous training algorithm improves in the test accuracy when SD-FEEL is under a high degree of device heterogeneity.

TABLE I. COMPARISON OF FEEL SYSTEMS WITH MULTIPLE EDGE SERVERS

System	Communication Among Edge Servers	Communication Between the Cloud and Edge Servers	Latency	Insensitive to Network Topology	Insensitive to the Cell-edge Effect	Convergence Guarantee on Non-IID Data
HierFAVG [12], [13]	✗	✓	High	✓	✓	✓
FogFL [23]	✓	✓	High	✗	✓	✗
FedMes [24]	✗	✗	Low	✓	✗	✗
Multi-level Local SGD [14]	✓	✗	Low	✗	✓	✗
SD-FEEL (Ours)	✓	✗	Low	✓	✓	✓

C. Organizations

The rest of the paper is organized as follows. In Section II, we introduce the SD-FEEL system and propose a generic training algorithm. Section III presents the convergence analysis and corresponding discussions, including the effects of key parameters and comparison of different FEEL systems. In Section IV, we investigate SD-FEEL with device heterogeneity and propose an asynchronous training algorithm, followed by its convergence analysis. We provide the simulation results in Section V and conclude this paper in Section VI.

D. Notations

Throughout this paper, we use bold-face lower-case letters, bold-face upper-case letters, and math calligraphy letters to denote vectors, matrices, and sets, respectively. Besides, $\mathbf{X} \triangleq [\boldsymbol{x}^{(i)}]_{i=1}^{N}$ represents a matrix, of which, the i-th column vector is $\boldsymbol{x}^{(i)}$. The $N \times N$ identity matrix is denoted as $\mathbf{I}_N$, and we define $\mathbf{1}_N \triangleq [1, 1, \ldots, 1]_{1\times N}$. For any $M \times N$ matrix $\mathbf{X}$, $\lambda_i(\mathbf{X})$ represents its i-th largest eigenvalue, the operator norm is denoted as $\|\mathbf{X}\|_{\text{op}} \triangleq \max_{\|\boldsymbol{w}\|=1} \mathbf{X}\boldsymbol{w} = \sqrt{\lambda_{\max}(\mathbf{X}^{\mathsf{T}}\mathbf{X})}$, and the weighted Frobenius norm is defined as $\|\mathbf{X}\|_{\mathbf{M}} \triangleq \sum_{i=1}^{M}\sum_{j=1}^{N} m^{i,j}|x^{i,j}|^2$, where $\mathbf{M} \triangleq \{m^{i,j}\}$. In addition, $\lceil\cdot\rceil$ denotes the ceil function, and $\mathbb{1}\{\cdot\}$ is the binary indicator function. We denote $f(x) = O(g(x))$ when there exists a positive real number M and a real number x_0 such that $|f(x)| \leq Mg(x), \forall x \geq x_0$.

II. SYSTEM MODEL

A. Semi-Decentralized FEEL System

The SD-FEEL system comprises of C client nodes (denoted as set $\mathcal{C}$) and D edge servers (denoted as set $\mathcal{D}$), as shown in Fig. 1. The system can be viewed as D edge clusters, each of which consists of an edge server (denoted as d) and a set of associated client nodes (denoted as $\mathcal{C}_d$). Assume each edge server coordinates at least one client node, and each client node is associated with only one edge server, according to some predefined criteria, e.g., physical proximity and network coverage. Besides, the edge servers are connected with the neighboring servers $\mathcal{N}_d$ via high-speed cables, formulating a connected graph $\mathcal{G}$.

Client node i possesses a set of local training data, denoted as $\mathcal{S}_i = \{\boldsymbol{s}_j^{(i)}\}_{j=1}^{|\mathcal{S}_i|}$, where $\boldsymbol{s}_j^{(i)}$ is the j-th data sample at client node i. The collection of data samples at the set of client nodes $\mathcal{C}_d$ is denoted as $\tilde{\mathcal{S}}_d$, and the training data at all the client nodes is denoted as $\mathcal{S}$. The ratios of data samples are defined as $\hat{m}_i \triangleq \frac{|\mathcal{S}_i|}{|\tilde{\mathcal{S}}_d|}$, $m_i \triangleq \frac{|\mathcal{S}_i|}{|\mathcal{S}|}$, and $\tilde{m}_d \triangleq \frac{|\tilde{\mathcal{S}}_d|}{|\mathcal{S}|}$, respectively. The client nodes collaboratively train an ML model $\boldsymbol{w} \in \mathbb{R}^M$ with M trainable parameters. Denote the loss function associated with a data samples $\boldsymbol{s}$ with model $\boldsymbol{w}$ as $f(\boldsymbol{s}; \boldsymbol{w})$, a typical example of which is the categorical cross-entropy between the predicted label and the ground truth for a classification task. Accordingly, the objective of SD-FEEL is to minimize the value of the loss function over the training data across all the client nodes, i.e., $\min_{\boldsymbol{w}\in\mathbb{R}^M} \left\{F(\boldsymbol{w}) \triangleq \sum_{i\in C} \frac{|\mathcal{S}_i|}{|\mathcal{S}|} F_i(\boldsymbol{w})\right\}$, where $F_i(\boldsymbol{w}) \triangleq \frac{1}{|\mathcal{S}_i|}\sum_{j\in\mathcal{S}_i} f(\boldsymbol{s}_j^{(i)}; \boldsymbol{w})$ is the local loss at client node i.

We use h_i to denote the computational speed of client node i, which is in the unit of floating point operations per second (i.e., FLOPS). Accordingly, the degree of device heterogeneity is characterized by the *heterogeneity gap* $H \triangleq \max_{i,j\in C} \frac{h_i}{h_j}$.

B. Training Algorithm

Assume there are K iterations in the training process, which includes three main procedures: 1) local model update, 2) intra-cluster model aggregation, and 3) inter-cluster model aggregation. While every training iteration consists of local model update, intra-/inter-cluster model aggregations are triggered with the periods of τ_1 and $\tau_1\tau_2$, respectively. We show an illustration of SD-FEEL training for two edge clusters in Fig. 2 and introduce the details as follows. For clarity, we will first present synchronous SD-FEEL, while the asynchronous training will be considered in Section IV.

1) Local Model Update: Denote the model on the client node i at the beginning of the k-th training iteration as $\boldsymbol{w}_{k-1}^{(i)}$. This client node performs model updating based on its local data by using the mini-batch stochastic gradient descent (SGD) algorithm [26], which is expressed as follows:

$$\boldsymbol{w}_k^{(i)} \leftarrow \boldsymbol{w}_{k-1}^{(i)} - \eta g(\boldsymbol{\xi}_k^{(i)}; \boldsymbol{w}_{k-1}^{(i)}), i \in C. \tag{1}$$

Here η is the learning rate, and $g(\boldsymbol{\xi}_k^{(i)}; \boldsymbol{w}_{k-1}^{(i)})$ is the stochastic gradient computed on a randomly-sampled batch of training data $\boldsymbol{\xi}_k^{(i)}$.

2) Intra-cluster Model Aggregation: When the iteration index k is an integer multiple of τ_1, the client nodes upload their local models to the corresponding edge server after completing local training. The edge server aggregates these received models by computing a weighted sum as follows:

$$\hat{\boldsymbol{y}}_k^{(d)} \leftarrow \sum_{i\in\mathcal{C}_d} \hat{m}_i \boldsymbol{w}_k^{(i)}, d \in \mathcal{D}. \tag{2}$$

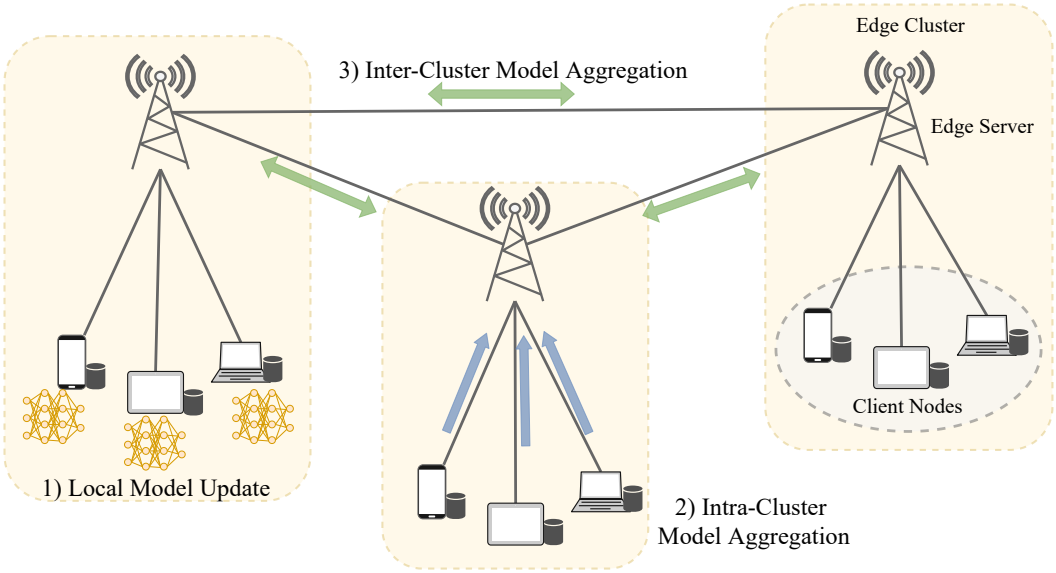


Fig. 1. The semi-decentralized FEEL system.

If k is not an integer multiple of $\tau_1\tau_2$, the edge server directly sends the model $\boldsymbol{y}_k^{(d)}$ to its associated client nodes, i.e.,

$$\boldsymbol{w}_{k+1}^{(i)} \leftarrow \boldsymbol{y}_k^{(d)}, i \in C_d. \tag{3}$$

Otherwise, the inter-cluster model aggregation is triggered.

3) Inter-cluster Model Aggregation: When k is an integer multiple of $\tau_1\tau_2$, each edge server shares its model to the one-hop neighboring edge servers after intra-cluster model aggregation. Each round of inter-cluster model aggregation includes $\alpha \in \{1, 2, \dots\}$ times of model exchanges and aggregations, which can be expressed as follows:

$$\hat{\boldsymbol{y}}_{k,l}^{(d)} \leftarrow \sum_{j \in \mathcal{N}_d \cup \{d\}} p^{j,d} \hat{\boldsymbol{y}}_{k,l-1}^{(j)}, l = 1, 2, \dots, \alpha, d \in \mathcal{D}. \tag{4}$$

Here $\hat{\boldsymbol{y}}_{k,0}^{(d)} = \boldsymbol{y}_k^{(d)}$ is the intra-cluster aggregated model, and we denote $\mathbf{P} \triangleq [p^{j,d}] \in \mathbb{R}^{D \times D}$ as the mixing matrix. To reduce model inconsistency among edge clusters, i.e., to ensure fast convergence to the weighted sum of the distributed models through inter-cluster model aggregation, $\mathbf{P}$ can be chosen as follows [27]:

$$\mathbf{P} = \mathbf{I}_D - \frac{2}{\lambda_1(\tilde{\mathbf{L}}) + \lambda_{D-1}(\tilde{\mathbf{L}})} \tilde{\mathbf{L}}, \tag{5}$$

where $\tilde{\mathbf{L}} \triangleq \mathbf{L}\mathbf{\Omega}^{-1}$, $\mathbf{L}$ is the Laplacian matrix of graph $\mathcal{G}$, and $\mathbf{\Omega} \triangleq \text{diag}(\tilde{m}_1, \tilde{m}_2, \dots, \tilde{m}_D)$. We define $\zeta \triangleq |\lambda_2(\mathbf{P})| \in [0, 1)$. The edge server then updates $\boldsymbol{y}_k^{(d)}$ as $\hat{\boldsymbol{y}}_{k,\alpha}^{(d)}$ and broadcasts it to the associated client nodes according to (3).

After repeating the above steps for K iterations (K is assumed as an integer multiple of $\tau_1\tau_2$), the system enters a consensus phase where the edge servers exchange and aggregate models with their neighboring clusters. After sufficient rounds of such operations, the system will output a model $\sum_{d \in \mathcal{D}} \tilde{m}_d \boldsymbol{y}_K^{(d)}$ and broadcast it to the associated client nodes. The consensus phase takes place only once, which will incur negligible extra overhead. We summarize the training algorithm of SD-FEEL in Algorithm 1.

Algorithm 1: Training Algorithm for SD-FEEL

```
Initialize all client nodes with the same model (i.e., w_0^(i) = w_0, ∀i ∈ C);
for k = 1, 2, ..., K do
    for each client node i ∈ C in parallel do
        Update the local model as w_k^(i) according to (1);
        if mod (k, τ1) = 0 then
            for each edge server d ∈ D in parallel do
                Receive the most updated model from the client nodes in C_d;
                Obtain y_k^(d) by performing intra-cluster model aggregation according to (2);
                if mod (k, τ1τ2) = 0 then
                    Set ŷ_{k,0}^(d) as y_k^(d);
                    for l = 1, ..., α do
                        Share models with N_d and perform inter-cluster model aggregation according to (4);
                        Update y_k^(d) as ŷ_{k,α}^(d) ;
                Broadcast y_k^(d) to the client nodes in C_d;
Enter the consensus phase;
return Σ_{d∈D} m̃_d y_K^(d);
```

III. Theoretical Analysis

A. Convergence Analysis

To facilitate the convergence analysis, we make the following assumptions that are commonly used in the existing literature [28]–[30].

Assumption 1. *For all $i \in C$, we assume:*

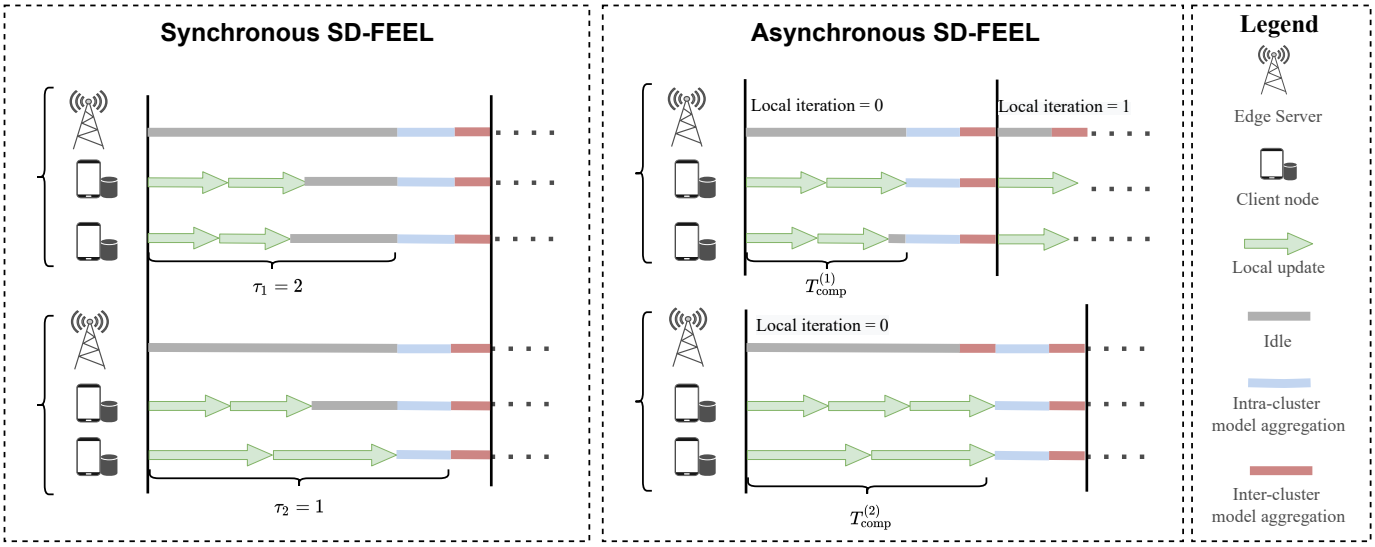


Fig. 2. An illustration of synchronous (left) and asynchronous (right) SD-FEEL. In synchronous SD-FEEL, the client nodes are required to perform the same number of local iterations before intra-cluster and inter-cluster model aggregations, where the fast client nodes stay idle until all the client nodes complete their local training. In asynchronous SD-FEEL, the client nodes in edge cluster $d \in \{1, 2\}$ perform local model updates for a duration of $T^{(d)}_{\text{comp}}$ before uploading the model updates to the associated edge server. The edge server then aggregates the received models from the client nodes in its cluster and shares the aggregated model with its one-hop neighbors.

- ***(Smoothness)*** *The local objective function is L-smooth, i.e.,*

$$\|\nabla F_i(\boldsymbol{w}) - \nabla F_i(\boldsymbol{w}')\|_2 \leq L \|\boldsymbol{w} - \boldsymbol{w}'\|_2, \forall \boldsymbol{w}, \boldsymbol{w}' \in \mathbb{R}^M. \tag{6}$$

- ***(Unbiased and bounded gradient variance)*** *The mini-batch gradient is unbiased, i.e.,*

$$\mathbb{E}_{\boldsymbol{\xi}^{(i)}|\boldsymbol{w}}[g(\boldsymbol{\xi}^{(i)}; \boldsymbol{w})] = \nabla F_i(\boldsymbol{w}), \forall \boldsymbol{w} \in \mathbb{R}^M, \tag{7}$$

and there exists $\sigma > 0$ such that

$$\mathbb{E}_{\boldsymbol{\xi}^{(i)}|\boldsymbol{w}}\left[\left\|g(\boldsymbol{\xi}^{(i)}; \boldsymbol{w}) - \nabla F_i(\boldsymbol{w})\right\|_2^2\right] \leq \sigma^2, \forall \boldsymbol{w} \in \mathbb{R}^M. \tag{8}$$

- ***(Degree of non-IIDness)*** *There exists $\kappa > 0$ such that*

$$\|\nabla F_i(\boldsymbol{w}) - \nabla F(\boldsymbol{w})\|_2 \leq \kappa, \quad \forall \boldsymbol{w} \in \mathbb{R}^M, \tag{9}$$

where κ measures the degree of data heterogeneity across all client nodes. When $\kappa = 0$, it reduces to the IID case.

Denote $\mathbf{W}_k \triangleq [\boldsymbol{w}_k^{(i)}]_{i \in C} \in \mathbb{R}^{M \times C}$ and $\mathbf{G}_k \triangleq [g(\boldsymbol{\xi}^{(i)}; \boldsymbol{w}_k^{(i)})]_{i \in C} \in \mathbb{R}^{M \times C}$. To characterize the process of model uploading and broadcasting between client nodes and edge servers, we define $\mathbf{V} \triangleq [v^{i,d}] \in \mathbb{R}^{C \times D}$ and $\mathbf{B} \triangleq [b^{d,i}] \in \mathbb{R}^{D \times C}$, where $u^{i,d} \triangleq \hat{m}_i \mathbb{1}\{i \in C_d\}$ represents the data ratio of client node i within the d-th edge cluster, and $b^{d,i} = \mathbb{1}\{i \in C_d\}$ denotes the affiliation between edge server d and client node i. Thus, the local models at client nodes evolve according to the following lemma.

Lemma 1. *The local models evolve according to the following expression:*

$$\mathbf{W}_{k+1} = (\mathbf{W}_k - \eta \mathbf{G}_k)\mathbf{T}_k, \; k = 1, 2, \ldots, K, \tag{10}$$

where the transition matrix is given by

$$\mathbf{T}_k = \begin{cases} \mathbf{VB} & \text{if } \operatorname{mod}(k, \tau_1) = 0 \text{ and } \operatorname{mod}(k, \tau_1\tau_2) \neq 0, \\ \mathbf{VP}^{\alpha}\mathbf{B}, & \text{if } \operatorname{mod}(k, \tau_1\tau_2) = 0, \\ \mathbf{I}_C, & \text{otherwise.} \end{cases} \tag{11}$$

Proof. We rewrite (4) in the matrix form as $\hat{\mathbf{Y}}_{k,l} = \mathbf{P}\hat{\mathbf{Y}}_{k,l-1}$, where $\hat{\mathbf{Y}}_{k,l} \triangleq [\hat{\boldsymbol{y}}_{k,l}^{(d)}] \in \mathbb{R}^{M \times D}$. According to this iterative relationship, we have $\hat{\mathbf{Y}}_{k,\alpha} = \mathbf{P}^{\alpha}\hat{\mathbf{Y}}_{k,0}$. Then the proof is completed by following $\hat{\mathbf{Y}}_{k,0} = \mathbf{V}\mathbf{W}_k$ and $\mathbf{W}_{k+1} = \mathbf{B}\hat{\mathbf{Y}}_{k,\alpha}$. □

Denote $\boldsymbol{m} \triangleq [m_i]_{i \in C}$ and $\mathbf{M} \triangleq \boldsymbol{m}\mathbf{1}_C^{\mathrm{T}}$. We define an auxiliary global model as $\boldsymbol{u}_k \triangleq \mathbf{W}_k\boldsymbol{m}$ and the corresponding concentration matrix is $\boldsymbol{u}_k\mathbf{1}_C^{\mathrm{T}} = \mathbf{W}_k\mathbf{M}$. Let $\boldsymbol{u}_k$ evolve as if $\mathbf{G}_k\mathbf{M}$ can be obtained by a centralized PS and used to update the global model in every iteration. Accordingly, we have the following relationship:

$$\boldsymbol{u}_{k+1} = \boldsymbol{u}_k - \eta \mathbf{G}_k \boldsymbol{m}^{\mathrm{T}}. \tag{12}$$

Such a relationship is desired to ensure fast convergence in terms of training iterations, as the gradients computed at all the client nodes can be leveraged to update the global model [15], [31]. Unfortunately, this can be achieved only when edge servers receive local models and reach a consensus in each training iteration (i.e., $\tau_1 = 1$, $\tau_2 = 1$, and $\zeta^{\alpha} = 0$). Comparatively, the model $\mathbf{W}_k$ in SD-FEEL deviates from this desired sequence due to the existence of $\mathbf{T}_k$ in (10). However, there exists an upper bound for the deviation between $\mathbf{W}_k$ and $\boldsymbol{u}_k\mathbf{1}_C^{\mathrm{T}}$, introduced by both mini-batch sampling of SGD (i.e., σ^2) and non-IIDness across client nodes (i.e., κ^2).

Lemma 2. *With Assumption 1, we have:*

$$\begin{aligned} &\frac{1}{K}\sum_{k=1}^{K}\mathbb{E}\left[\left\|\mathbf{W}_k - \boldsymbol{u}_k\mathbf{1}_C^{\mathrm{T}}\right\|_{\mathbf{M}}^2\right] \\ &\leq 2\eta^2 V_1 \sigma^2 + 8\eta^2 V_2 \kappa^2 + \frac{8\eta^2 V_2}{K}\sum_{k=1}^{K} J_k, \end{aligned} \tag{13}$$

where $V_1 \triangleq (\tau_1\tau_2\frac{\zeta^{2\alpha}}{1-\zeta^{2\alpha}}+\frac{\tau_1\tau_2-1}{2})/(1-16\eta^2L^2V_3)$, $V_2 \triangleq V_3/(1-16\eta^2L^2V_3)$, $V_3 \triangleq \tau_1\tau_2(\tau_1\tau_2\Lambda+\frac{\tau_1\tau_2-1}{2}\frac{2-\zeta^\alpha}{1-\zeta^\alpha})$, $\Lambda \triangleq \frac{\zeta^{2\alpha}}{1-\zeta^{2\alpha}}+\frac{2\zeta^\alpha}{1-\zeta^\alpha}+\frac{\zeta^{2\alpha}}{(1-\zeta^\alpha)^2}$, *and* $J_k \triangleq \mathbb{E}[\|\sum_{i\in C} m_i\nabla F_i(\boldsymbol{w}_k^{(i)})\|_2^2]$.

Proof. Since $\boldsymbol{u}_k\mathbf{1}_C^{\mathrm{T}} = \mathbf{W}_k\mathbf{M}$, we have $\mathbf{W}_k - \boldsymbol{u}_k\mathbf{1}_C^{\mathrm{T}} = \mathbf{W}_k(\mathbf{I}_C - \mathbf{M})$, which can be expanded as $\mathbf{W}_0(\mathbf{I}_C - \mathbf{M})\prod_{l=1}^{k-1}\mathbf{T}_l - \eta\sum_{s=1}^{k-1}\mathbf{G}_s\ (\prod_{l=s}^{k-1}\mathbf{T}_l - \mathbf{M})$ using (10). Given that $\boldsymbol{w}_0^{(i)} = \boldsymbol{w}_0, \forall i \in C$, it remains to provide an upper bound for the term $\mathbb{E}[\|-\eta\sum_{s=1}^{k-1}\mathbf{G}_s(\prod_{l=s}^{k-1}\mathbf{T}_l-\mathbf{M})\|_{\mathbf{M}}^2]$. The proof is concluded by bounding accumulated variance of gradients with Assumption 1 and the Jensen's inequality. The complete proof is referred to Appendix C in [1]. □

As aforementioned, the training objective is to output a global model $\boldsymbol{u}_k$ which minimizes the global objective function. We now characterize how the expected loss of $\boldsymbol{u}_k$ changes in two consecutive iterations in the following lemma.

Lemma 3. *With Assumption 1, the expected change of the local loss functions in two consecutive iterations is bounded as follows:*

$$\begin{aligned}&\mathbb{E}[F(\boldsymbol{u}_{k+1})] - \mathbb{E}[F(\boldsymbol{u}_k)]\\ \le& -\frac{\eta}{2}\mathbb{E}\left[\|\nabla F(\boldsymbol{u}_k)\|_2^2\right] + \frac{\eta^2 L}{2}\sum_{i\in C} m_i^2\sigma^2\\ &-\left(\frac{\eta}{2}-\frac{\eta^2 L}{2}\right)J_k + \frac{\eta L^2}{2}\mathbb{E}\left[\|\mathbf{W}_k(\mathbf{I}_C-\mathbf{M})\|_{\mathbf{M}}^2\right].\end{aligned} \tag{14}$$

Proof. Following Lemma 8 in [32], the proof is completed by plugging the right hand side (RHS) of (12) into the first-order Taylor expansion of $\nabla F(\boldsymbol{u}_{k+1})$ and leveraging Assumption 1. Please refer to the detailed proof in [1]. □

With above lemmas, we prove the convergence of Algorithm 1 in the following theorem.

Theorem 1. *If the learning rate* η *satisfies:*

$$1 - \eta L - 8\eta^2L^2V_2 \ge 0, 1 - 16\eta^2L^2V_3 > 0, \tag{15}$$

we have:

$$\begin{aligned}\frac{1}{K}\sum_{k=1}^{K}\mathbb{E}\left[\|\nabla F(\boldsymbol{u}_k)\|_2^2\right] \le& \frac{2\Delta}{\eta K}\\ &+\underbrace{\eta L\sum_{i\in C} m_i^2\sigma^2}_{\Phi_0} + \underbrace{\eta^2L^2(2V_1\sigma^2+8V_2\kappa^2)}_{\Phi(\tau_1,\tau_2,\alpha,\zeta)},\end{aligned} \tag{16}$$

where $\Delta \triangleq \mathbb{E}\left[F(\boldsymbol{u}_1)\right] - F(\boldsymbol{u}^*)$ *and* $\boldsymbol{u}^* \triangleq \arg\min_{\boldsymbol{w}} F(\boldsymbol{w})$.

Proof. We sum up both sides of (14) over $k = 1, 2, \ldots, K$, and divide them by K. Then we apply (13) in its RHS and choose a suitable η that satisfies the conditions in (15) to eliminate J_k. The proof is concluded by rearranging the terms. □

B. Discussions

The result in Theorem 1 provides us with various insights, as presented in this subsection.

Corollary 1. (Convergence rate) If the learning rate is chosen as $\eta = O\left(\frac{1}{L\sqrt{K}}\right)$, (16) can be simplified as follows:

$$\frac{1}{K}\sum_{k=1}^{K}\mathbb{E}\left[\|\nabla F(\boldsymbol{u}_k)\|_2^2\right] \le O\left(\frac{2L\Delta+\Phi_0}{\sqrt{K}}\right) + O\left(\frac{\Phi(\tau_1,\tau_2,\alpha,\zeta)}{K}\right), \tag{17}$$

which implies that Algorithm 1 converges within $O(\frac{1}{\epsilon^2})$ training iterations to find an ϵ-approximate solution (i.e., $\frac{1}{K}\sum_{k=1}^{K}\mathbb{E}\left[\|\nabla F(\boldsymbol{u}_k)\|_2^2\right] \le \epsilon$).

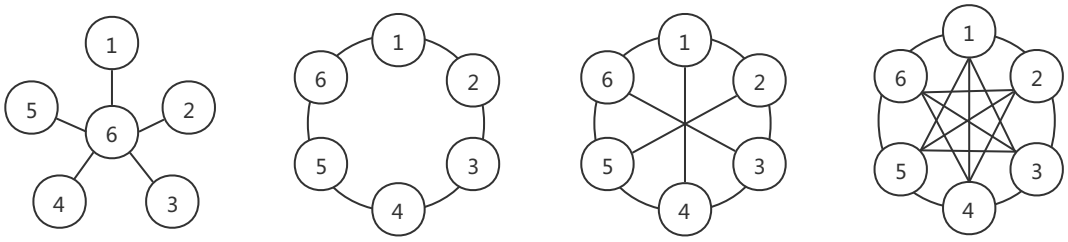
In the results of Theorem 1 and Corollary 1, there are two variance terms, i.e., Φ_0 and $\Phi(\tau_1, \tau_2, \alpha, \zeta)$. The constant term Φ_0 is the same as the bound in previous literature of centralized SGD [33] and FL frameworks [28], [32]. However, infrequent aggregations in SD-FEEL result in model divergence across client nodes, reflected through $\Phi(\tau_1, \tau_2, \alpha, \zeta)$.

Remark 1. (Effect of aggregation periods) By taking the first-order derivative, we see that the term $\Phi(\tau_1, \tau_2, \alpha, \zeta)$ increases with both τ_1 and τ_2. With more local iterations performed, the models on client nodes become biased towards local datasets, which slows down the convergence speed [29], [31]. Therefore, to minimize the error floor in the RHS of (16), we prefer setting $\tau_1 = 1$ and $\tau_2 = 1$. Nevertheless, as will be seen in Section V, with smaller values of τ_1 and τ_2, such frequent aggregations incur a huge latency in both intra-cluster and inter-cluster communications. Thus, τ_1 and τ_2 should be properly selected to achieve fast convergence with respect to the wall-clock time.

Remark 2. (Effect of network among edge servers) The RHS of (16) also increases with ζ and α, Note that when $\zeta^\alpha \to 0$ (i.e., with the fully connected topology or $\alpha \to \infty$), the edge servers can reach consensus (i.e., $\boldsymbol{y}_k^{(d)} \leftarrow \sum_{j\in\mathcal{D}}\tilde{m}_j\boldsymbol{y}_k^{(j)}$) after inter-cluster model aggregation.

- A smaller value of ζ (i.e., a more connected graph among the edge servers) results in faster convergence. Fig. 3 shows several typical topologies formed by six edge servers, including the star ($\zeta = 0.71$), ring ($\zeta = 0.6$), partially connected ($\zeta = 0.33$), and fully connected ($\zeta = 0$) topologies. It is clear that the fully-connected network topology can achieve the best performance with a fixed α.
- Increasing α (i.e., raising the inter-server communication overhead) can reduce the variance terms in the RHS of (16) and thus speed up the convergence. Nevertheless, such benefits diminish as α increases. As will be demonstrated in Section V, there is no additional gain after α is beyond some value.

Remark 3. (Comparison with HierFAVG) If edge servers can achieve a consensus in each training iteration, i.e., $\zeta^\alpha = 0$, our results in Theorem 1 and Corollary 1 reduce to the case of HierFAVG [12]. Accordingly, Corollary 1 indicates that HierFAVG requires fewer iterations to converge. Nevertheless, SD-FEEL adopts inter-server communication to replace the communication between the edge server and the Cloud, which significantly reduces the per-iteration latency. As a result, the total training time of the two architectures depends on the application scenarios. Here we discuss two special cases as follows:



(a) Star (b) Ring (c) Partially-connected (d) Fully-connected

Fig. 3. Typical network topologies of the edge servers.

- When the network among edge servers is fully-connected, SD-FEEL always leads to a better model than HierFAVG within a given training time.
- When the inter-server communication latency is adequately low, the edge servers are allowed to perform sufficient rounds of model sharing to reach consensus. Thus, SD-FEEL converges faster than HierFAVG with respect to the wall-clock time.

When client nodes differ slightly in computation resources, i.e., with a small H, SD-FEEL converges fast with Algorithm 1. Nevertheless, the training efficiency may be degraded if there is a huge disparity in computational speeds, i.e., H is large. In the next section, we propose an asynchronous training algorithm to alleviate such an issue.

IV. Asynchronous Training for SD-FEEL

When client nodes have significantly diverse computational speeds (i.e., $H \gg 1$), performing an equal number of local iterations results in idle time of the fast client nodes since they have to wait for the straggling ones to complete local training. To mitigate such an issue, we propose an asynchronous training algorithm for SD-FEEL where each edge server presets a deadline for local training time and proceeds to the next training iteration once it completes inter-cluster model aggregation of the current iteration.

A. Training Algorithm

Denote the training iteration counter as t, which gives the total number of training iterations completed by edge clusters. Before entering the training process, each edge server presets a deadline for local model updates given the computational resources of the coordinated client nodes. Specifically, edge server d sets the value of $T_{\text{comp}}^{(d)}$ for the client nodes in C_d. The determination of $T_{\text{comp}}^{(d)}$ is beyond the scope of this paper, but in general, it follows the same principles mentioned in Remark 1, namely to ensure effective training as well as avoid large model inconsistency in model aggregations. An illustration of the asynchronous training process is shown in Fig. 2. Now we present the details of three key stages in each iteration as follows.

1) Local Model Update: In the d-th edge cluster, client node i performs $\theta_i = h_i\beta \in \{\theta_{\min}, \theta_{\min}+1, \ldots, \theta_{\max}\}$ local epochs using mini-batch SGD within the duration of $T_{\text{comp}}^{(d)}$ and β is related to the complexity of training task and the batch size. Denote the model of client node i at the beginning of the l-th local training epoch in the t-th global iteration as $\boldsymbol{w}_{t,l}^{(i)}$. Then we have the following expression:

$$\boldsymbol{w}_{t,l+1}^{(i)} \leftarrow \boldsymbol{w}_{t,l}^{(i)} - \eta g(\boldsymbol{\xi}_{t,l}^{(i)}; \boldsymbol{w}_{t,l}^{(i)}), l \in \{0, 1, \ldots, \theta_i - 1\}, i \in C. \quad (18)$$

Given that the varying numbers of local epochs among client nodes may incur model bias [29], client node i normalizes the local updates by θ_i as follows:

$$\boldsymbol{\Delta}_t^{(i)} \triangleq \frac{1}{\theta_i}\left(\boldsymbol{w}_{t,\theta_i}^{(i)} - \boldsymbol{w}_{t,0}^{(i)}\right) = -\frac{\eta}{\theta_i}\sum_{l=0}^{\theta_i-1} g(\boldsymbol{\xi}_{t,l}^{(i)}; \boldsymbol{w}_{t,l}^{(i)}), i \in C. \quad (19)$$

2) Intra-cluster Model Aggregation: Once the deadline $T_{\text{comp}}^{(d)}$ arrives, edge server d collects the normalized model updates from its associated client nodes C_d and computes the weighted average with the weighting factors $\{\hat{m}_i\}$'s. Denote the most updated model at edge server d at the beginning of the t-th iteration as $\boldsymbol{y}_t^{(d)}$. Then such an update is added to $\boldsymbol{y}_t^{(d)}$ as the implementation of gradient descent, which can be expressed as follows:

$$\hat{\boldsymbol{y}}_t^{(d)} \leftarrow \boldsymbol{y}_t^{(d)} + \overline{\theta}_d \sum_{i \in C_d} \hat{m}_i \boldsymbol{\Delta}_t^{(i)}, d \in \mathcal{D}, \quad (20)$$

where $\overline{\theta}_d \triangleq \sum_{i \in C_d} \hat{m}_i \theta_i$ is the weighted average of the numbers of local epochs completed by the client nodes.

3) Inter-cluster Model Aggregation: After intra-cluster model aggregation, edge server d exchanges the local model $\hat{\boldsymbol{y}}_t^{(d)}$ with its neighboring edge servers in $\mathcal{N}_d$. The models maintained by these edge servers are accordingly updated as follows:

$$\boldsymbol{y}_t^{(j)} \leftarrow \sum_{j' \in \mathcal{N}_j \cup \{j\}} p_t^{j',j} \hat{\boldsymbol{y}}_t^{(j')}, j \in \mathcal{N}_d \cup \{d\}, \quad (21)$$

where $\mathbf{P}_t \triangleq \{p_t^{j',j}\} \in \mathbb{R}^{D \times D}$ denotes the mixing matrix in the t-th iteration. Note that the neighboring edge server j maintains a model updated in a previous global iteration $t'(j) < t$, the gap of which is named the *iteration gap*, i.e., $\delta_t^{(j)} \triangleq t - t'(j)$. Since a larger value of $\delta_t^{(j)}$ implies that the

model is more stale and has less value to other edge clusters [20], we design a staleness-aware mixing matrix as follows:

$$p_t^{i,j} = \begin{cases} \frac{\psi(\delta_t^{(j)})}{\Psi_t^{(j)}} & \text{if } j = d \text{ and } i \in \mathcal{N}_d \cup \{d\}, \\ p_t^{j,i} & \text{if } j \in \mathcal{N}_d \text{ and } i = d, \\ 1 - p_t^{d,j} & \text{if } j \in \mathcal{N}_d \text{ and } i = j, \\ 1 & \text{if } j \notin \mathcal{N}_d \cup \{d\} \text{ and } i = j, \\ 0, & \text{otherwise}, \end{cases} \tag{22}$$

where $\psi(x)$ is a general non-increasing function of x and $\Psi_t^{(j)} \triangleq \sum_{i \in \mathcal{N}_j \cup \{j\}} \psi(\delta_t^{(i)})$. Then the model $\boldsymbol{y}_t^{(d)}$ is broadcasted to the client nodes in $\mathcal{C}_d$ according to $\boldsymbol{w}_{t+1,0}^{(i)} \leftarrow \boldsymbol{y}_t^{(d)}, i \in \mathcal{C}_d$. For illustration, consider an example with three edge clusters $d \in \{1, 2, 3\}$ arranging in a sequential order. When edge cluster 1 triggers the inter-cluster model aggregation in the t-th training iteration, the iteration gaps of its neighboring edge cluster ($\mathcal{N}_1 = \{2\}$) is $\delta_t^{(2)} = 2$. The corresponding mixing matrix is $\mathbf{P}_t = [\frac{\psi(0)}{\Psi_t^{(1)}}, \frac{\psi(2)}{\Psi_t^{(1)}}, 0; \frac{\psi(2)}{\Psi_t^{(1)}}, 1 - \frac{\psi(2)}{\Psi_t^{(1)}}, 0; 0, 0, 1]$.

The above steps repeat until timeout or the values of local loss at all the client nodes cannot be further reduced. Assume T is the global iteration index at that time. Then the system enters the consensus phase to output a global model $\sum_{d \in \mathcal{D}} \tilde{m}_d \boldsymbol{y}_T^{(d)}$.

B. Convergence Analysis

We define an auxiliary global model at the t-th training iteration as $\overline{\boldsymbol{y}}_t \triangleq \sum_{d \in \mathcal{D}} \tilde{m}_d \boldsymbol{y}_t^{(d)}$, the evolution of which is expressed as:

$$\overline{\boldsymbol{y}}_{t+1} = \overline{\boldsymbol{y}}_t - \eta \hat{\mathbf{G}}_t \tilde{\boldsymbol{m}}^{\mathrm{T}} \boldsymbol{\Lambda}, \tag{23}$$

where $\hat{\mathbf{G}}_t \triangleq [\sum_{i \in \mathcal{C}_d} \frac{\hat{m}_i}{\theta_i} \sum_{l=0}^{\theta_i - 1} g(\boldsymbol{\xi}_{t,l}^{(i)}; \boldsymbol{w}_{t,l}^{(i)})]_{d \in \mathcal{D}} \in \mathbb{R}^{M \times D}$, $\tilde{\boldsymbol{m}} \triangleq [\tilde{m}_d]_{d \in \mathcal{D}}$, and $\boldsymbol{\Lambda} \triangleq \mathrm{diag}(\overline{\theta}_1, \overline{\theta}_2, \ldots, \overline{\theta}_D)$. Since the client nodes perform different local epochs, we prove the convergence by focusing on the servers' models, in contrast to the client models as in the proof of Section III. The equivalence $\overline{\boldsymbol{y}}_t = \sum_{i \in \mathcal{C}} m_i \boldsymbol{w}_{t+1,0}^{(i)}$ always holds if all edge servers broadcast the models to the associated client nodes.

The analysis basically follows the procedures in subsection III-A, i.e., using accumulated gradients to provide an upper bound for model deviation. Nevertheless, the asynchronous training incurs additional model inconsistency among edge clusters. Specifically, the iteration counter t increases once an edge cluster completes a training iteration. However, the client nodes in other edge clusters are training on stale models as aforementioned. To characterize this phenomenon, we define $\boldsymbol{a}_{\tilde{t}}^{(d)} \triangleq \boldsymbol{a}_{t-\delta_t^{(d)}}^{(d)}$ (respectively $\boldsymbol{a}_{\tilde{t}}^{(i)} \triangleq \boldsymbol{a}_{t-\delta_t^{(d)}}^{(i)}, i \in \mathcal{C}_d$) as the delayed model or gradient at the d-th edge server (respectively the i-th client node) in the t-th iteration. The following lemma shows that $\delta_t^{(d)}$ is upper bounded throughout the whole training process.

Lemma 4. ***(Bounded iteration gap)*** *There exists a constant δ_{max} such that $\delta_t^{(d)} \leq \delta_{max}, \forall t \in \mathbb{N}, d \in \mathcal{D}$.*

Proof. After setting $\{T_{\mathrm{comp}}^{(d)}\}$'s, the training latency for each iteration is fixed as $T_{\mathrm{iter}}^{(d)}$. During one training iteration of the slowest edge cluster j^*, $\delta_{\max} = \sum_{d \in \mathcal{D}} \left(\left\lceil \frac{T_{\mathrm{iter}}^{(d)}}{T_{\mathrm{iter}}^{(j^*)}} \right\rceil - 1 \right)$ gives a maximal value for the number of total iterations that other edge clusters have completed. □

To show the convergence, we begin with upper bounding the expected change of loss functions in two consecutive iterations in Lemma 5.

Lemma 5. *The expected change of the global loss function in two consecutive iterations is bounded as follows:*

$$\begin{aligned} &\mathbb{E}[F(\overline{\boldsymbol{y}}_{t+1})] - \mathbb{E}[F(\overline{\boldsymbol{y}}_t)] \\ &\leq -\frac{1}{2}\eta\theta_{\min}\mathbb{E}\left[\left\|\nabla F(\overline{\boldsymbol{y}}_t)\right\|^2\right] + \frac{1}{2}\eta^2 L\theta_{\max}^2\theta_{\min}^{-1}\sum_{i \in C} m_i^2\sigma^2 \\ &- \frac{\eta}{2}(\theta_{\min} - \eta L\theta_{\max}^2)Q_{\tilde{t}} + \frac{1}{2}\eta\theta_{\min}\underbrace{\mathbb{E}\left[\left\|\nabla F(\overline{\boldsymbol{y}}_t) - \nabla\hat{\mathbf{F}}_{\tilde{t}}\tilde{\boldsymbol{m}}^{\mathrm{T}}\right\|^2\right]}_{\mathcal{E}_t}, \end{aligned} \tag{24}$$

where $Q_{\tilde{t}} \triangleq \mathbb{E}[\|\nabla\hat{\mathbf{F}}_{\tilde{t}}\tilde{\boldsymbol{m}}^{\mathrm{T}}\|^2]$ and $\nabla\hat{\mathbf{F}}_{\tilde{t}} \triangleq [\sum_{i \in \mathcal{C}_d} \frac{\hat{m}_i}{\theta_i} \sum_{l=0}^{\theta_i - 1} \nabla F_i(\boldsymbol{w}_{\tilde{t},l}^{(i)})]_{d \in \mathcal{D}}$.

Proof. Similar with the proof of Lemma 3, by plugging the RHS of (23) into the first-order Taylor expansion of $\nabla F(\overline{\boldsymbol{y}}_{t+1})$ and following Lemma 8 in [32], we conclude the proof. □

The term $\mathcal{E}_t$ in (24) measures the degree to which the gradients collected from client nodes (i.e., $\nabla\hat{\mathbf{F}}_{\tilde{t}}\tilde{\boldsymbol{m}}^{\mathrm{T}}$) deviate from the desired gradient of global model (i.e., $\nabla F(\overline{\boldsymbol{y}}_t)$). Using the L-smoothness and the Jensen's inequality (i.e., $\|\boldsymbol{a} + \boldsymbol{b}\|^2 \leq 2\|\boldsymbol{a}\|^2 + 2\|\boldsymbol{b}\|^2, \forall \boldsymbol{a}, \boldsymbol{b} \in \mathbb{R}^d$), we derive an upper bound the term $\mathcal{E}_t$, which can be decomposed as follows:

$$\begin{aligned} \mathcal{E}_t \leq\ & 2L^2\underbrace{\mathbb{E}\left[\left\|\overline{\boldsymbol{y}}_t - \overline{\boldsymbol{y}}_{\tilde{t}}\right\|^2\right]}_{\mathcal{E}_{t,1}} + 4L^2\underbrace{\sum_{d \in \mathcal{D}} \tilde{m}_d \mathbb{E}\left[\left\|\overline{\boldsymbol{y}}_{\tilde{t}} - \boldsymbol{y}_{\tilde{t}}^{(d)}\right\|^2\right]}_{\mathcal{E}_{t,2}} \\ &+ 4L^2\underbrace{\sum_{d \in \mathcal{D}} \tilde{m}_d \sum_{i \in \mathcal{C}_d} \frac{\hat{m}_i}{\theta_i} \sum_{l=0}^{\theta_i - 1} \mathbb{E}\left[\left\|\boldsymbol{y}_{\tilde{t}}^{(d)} - \boldsymbol{w}_{\tilde{t},l}^{(i)}\right\|^2\right]}_{\mathcal{E}_{t,3}}. \end{aligned} \tag{25}$$

In the RHS of (25), the term $\mathcal{E}_{t,1}$ quantifies the influence of the staleness, which can be upper bounded by using Lemma 4. The term $\mathcal{E}_{t,2}$ measures the inter-cluster model divergence between an edge server and the averaged model $\overline{\boldsymbol{y}}_{\tilde{t}}$, while the term $\mathcal{E}_{t,3}$ measures the weighted sum of the model divergence between edge clusters. After respectively bounding three terms, we obtain the following lemma.

Lemma 6. *With Assumption 1, we have:*

$$\begin{aligned} \frac{1}{T}\sum_{t=0}^{T-1}\mathcal{E}_t \leq\ & A(\theta_{\min}, \theta_{\max}, \delta_{\max})\sigma^2 \\ &+ B(\theta_{\min}, \theta_{\max}, \delta_{\max})\kappa^2 + C(\theta_{\max}, \delta_{\max})Q_{\tilde{t}}, \end{aligned} \tag{26}$$

where $A(\theta_{\min}, \theta_{\max}, \delta_{\max}) \triangleq 4\eta^2L^2\delta_{\max}^2\theta_{\max}^2\theta_{\min}^{-1}U_4 + \frac{4\eta^2L^2(\theta_{\max}-1)}{1-2\eta^2L^2U_2} + 8\eta^2L^2\theta_{\max}^2\theta_{\min}^{-1}\frac{U_3}{T}\sum_{t=0}^{T-1}\sum_{s=1}^{t-1}\rho_{s,t-1}^2$, $B(\theta_{\min}, \theta_{\max}, \delta_{\max}) \triangleq 8\eta^2L^2\delta_{\max}^2\theta_{\max}^2\theta_{\min}^{-1}U_4 + \frac{24\eta^2L^2U_2}{1-2\eta^2L^2U_2} + 16\eta^2L^2\theta_{\max}^2\theta_{\min}^{-1}U_3\frac{1}{T}\sum_{t=0}^{T-1}(\sum_{s=1}^{t-1}\rho_{s,t-1})^2$ $C(\theta_{\max}, \delta_{\max}) \triangleq$

$8\eta^2L^2\delta_{\max}^2\theta_{\max}U_4+16\eta^2L^2\theta_{\max}^2U_3\frac{1}{T}\sum_{t=s+1}^{T-1}\rho_{s,t-1}(\sum_{l=1}^{t-1}\rho_{l,t-1})$, $U_2 \triangleq \theta_{\max}(\theta_{\max}-1)$, $U_3 \triangleq \frac{1+4\eta^2L^2U_2}{1-2\eta^2L^2U_2}$, $U_4 \triangleq \frac{1+22\eta^2L^2U_2}{1-2\eta^2L^2U_2}$, *and* $\rho_{s,t-1} \triangleq \|\prod_{l=s}^{t-1}\mathbf{P}_l - \mathbf{M}\|_{\text{op}}$.

Proof. The proof is obtained by respectively bounding the three terms in the RHS of (25). Please refer to Appendix A. □

We are now ready to prove the convergence of the asynchronous training algorithm.

Theorem 2. *With Assumption 1, if the learning rate η satisfies*

$$1-\eta L\theta_{\max}^2\theta_{\min}^{-1}-C(\theta_{\max},\delta_{\max})\geq 0, 1-2\eta^2L^2U_2>0, \quad (27)$$

we have

$$\begin{aligned}&\frac{1}{T}\sum_{t=0}^{T-1}\mathbb{E}\left[\left\|\nabla F(\overline{\boldsymbol{y}}_t)\right\|^2\right]\\ &\leq \frac{2\{\mathbb{E}[F(\boldsymbol{y}_0)]-\mathbb{E}[F(\overline{\boldsymbol{y}}_T)]\}}{\eta\theta_{\min}U_1T}+\frac{\eta L\theta_{\max}^2}{U_1\theta_{\min}^2}\sum_{i\in C}m_i^2\sigma^2 \qquad (28)\\ &+A(\theta_{\min},\theta_{\max},\delta_{\max})\frac{\sigma^2}{U_1}+B(\theta_{\min},\theta_{\max},\delta_{\max})\frac{\kappa^2}{U_1},\end{aligned}$$

where $U_1 \triangleq \frac{1-14\eta^2L^2U_2}{1-2\eta^2L^2U_2}$.

Proof. Summing up both sides of (24), plugging (26) into the RHS, and then choosing the learning rate as (27) to eliminate $Q_{\tilde{t}}$, we conclude the proof. □

Remark 4. (Convergence rate) If we choose the learning rate as $\eta=O\left(\frac{1}{L\sqrt{T}}\right)$, the RHS of (28) decreases at a speed of $O(\frac{1}{\sqrt{T}})$ and approaches zero when $T\to\infty$, which ensures the convergence.

V. Simulations

A. Settings

We simulate an SD-FEEL system with 50 client nodes and 10 edge servers. It is assumed that each edge server coordinates five client nodes. We consider a ring topology of edge servers unless otherwise specified. We evaluate SD-FEEL on two benchmark datasets for image classification, i.e., the MNIST [34] and CIFAR-10 [35] datasets, each of which has ten classes of labels. For the non-IID setting, we adopt the skewed label partition [36] for the MNIST dataset, where each client node has c (with a default value of 2) random classes of data samples. For the CIFAR-10 dataset, we utilize a Dirichlet distribution $\text{Dir}_{50}(\beta)$ to sample the probabilities $\{p'_{l,i}\}$'s, which is the proportion of the training samples of class l assigned to the i-th client node [37], where β (with a default value of 0.5) is the concentration parameter and a smaller value of β results in a more uneven local distribution across the client nodes. Following [12], we train a convolutional neural network (CNN) with two 5×5 convolutional layers and $M=21,840$ trainable parameters on the MNIST dataset, and another CNN with six convolutional layers that consists of $M=5,852,170$ trainable parameters on the CIFAR-10 dataset. The mini-batch SGD is employed with a batch size of 10, and the learning rate is set as 0.01 and 0.1 for the MNIST and CIFAR-10 datasets, respectively.

To demonstrate the effectiveness of SD-FEEL, we adopt three FL schemes as baselines, including 1) FedAvg [9], 2) HierFAVG [12], and 3) FEEL [11]. We implement the three baseline schemes based on FedML [38], which is an open-source research library for FL. Note that FEEL is an edge-assisted FL scheme with a single edge server randomly scheduling five client nodes in each iteration. For fair comparisons, each edge server in HierFAVG, FEEL, and SD-FEEL is assumed to have equal number of orthogonal uplink wireless channels.

B. Training Latency

The latency of K training iterations can be calculated as $T_{\text{tot}} = K\left(T_{\text{comp}}^{\text{ct}}+\frac{1}{\tau_1}T_{\text{comm}}^{\text{ct-sr}}+\frac{\alpha}{\tau_1\tau_2}T_{\text{comm}}^{\text{sr-sr}}\right)$, where $T_{\text{comp}}^{\text{ct}}$ is the computation latency for each local iteration at client nodes, $T_{\text{comm}}^{\text{ct-sr}}$ denotes the model uploading latency from a client node to its associated edge server, and $T_{\text{comm}}^{\text{sr-sr}}$ is the model transmission latency between neighboring edge servers. Following [39], the averaged computation time is assumed to $T_{\text{comp}}^{\text{ct}}=\frac{N_{\text{MAC}}}{C_{\text{CPU}}}$, where N_{MAC} is the number of the floating-point operations (FLOPs) for one local iteration, and $C_{\text{CPU}}=10$ GFLOPS denotes the CPU's computing bandwidth for the slowest device. The numbers of FLOPs required for local training at each iteration are $N_{\text{MAC}}=487.54$ KFLOPs for the MNIST dataset and $N_{\text{MAC}}=138.4$ MFLOPs for the CIFAR-10 dataset[1]. The communication latency is expressed as $T_{\text{comm}}=\frac{M_{\text{bit}}}{R}$ with $M_{\text{bit}}=32M$ bits and R denoting the transmission rate. Specifically, we assume that the client nodes communicate with the associated edge servers using orthogonal channels and there is no inter-cluster interference [16]. The transmission rate is assumed to be $R^{\text{ct-sr}}=B\log_2(1+\text{SNR})\approx5$ Mbps, where $B=1$ MHz and $\text{SNR}=15$ dB. The edge server communicates with the neighboring edge servers via high-speed links with the bandwidth of 50 Mbps [40], and the bandwidth from the edge servers to the Cloud is set as 5 Mbps. Accordingly, the transmission rate from the client nodes to the Cloud is given by $R^{\text{ct-cd}}=2.5$ Mbps.

C. Results

1) Convergence Speed and Generalization Performance: We show the training loss with respect to the training time for both the MNIST and CIFAR-10 datasets in Fig. 4. We see that the training loss of SD-FEEL drops rapidly in the initial training stage, and SD-FEEL converges at around 40 seconds and 250 minutes for the MNIST and CIFAR-10 datasets, respectively. Comparatively, HierFAVG and FedAvg fall behind due to the high communication latency with the Cloud-based PS. Besides, despite the low uplink transmission delay, FEEL has an unstable and slower training process as the edge server has access to a limited number of accessible client nodes in each iteration. Fig. 5 shows the generalization

[1]These values are calculated using OpCounter, which is an open-source model analysis library available at https://github.com/Lyken17/pytorch-OpCounter.

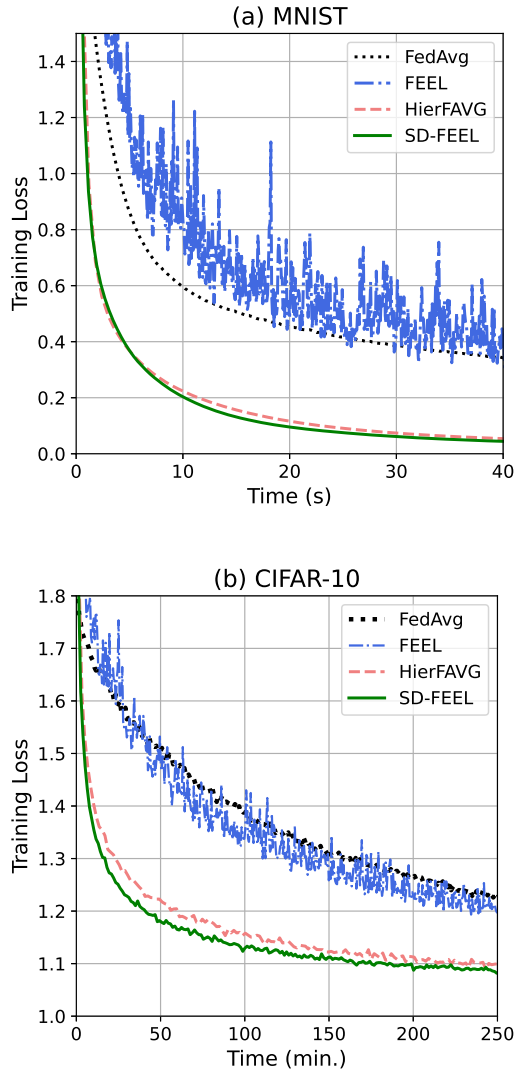


Fig. 4. Training loss over time on the (a) MNIST ($\tau_1 = 5$, $\tau_2 = 1$, and $\alpha = 1$) and (b) CIFAR-10 ($\tau_1 = 2$, $\tau_2 = 1$, and $\alpha = 5$) datasets.

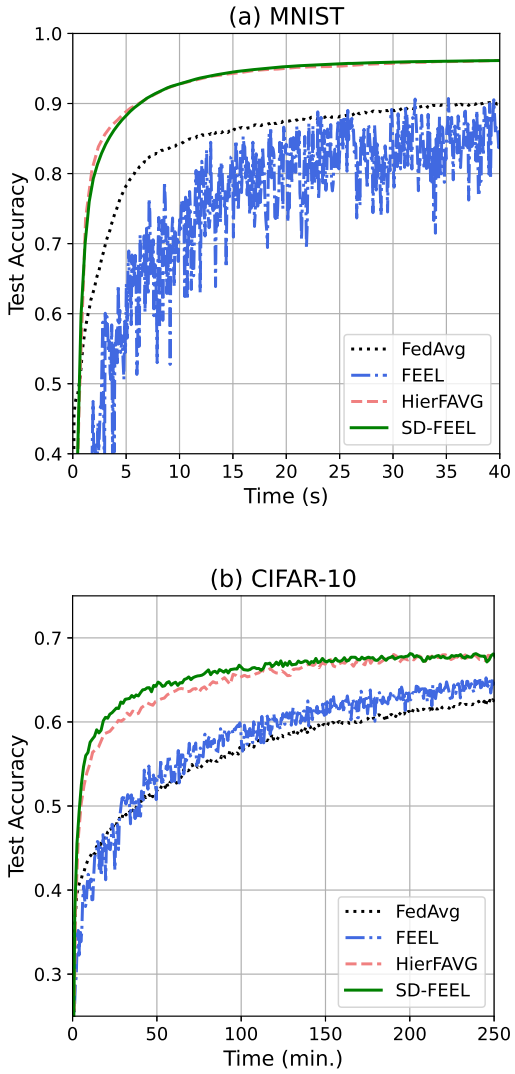


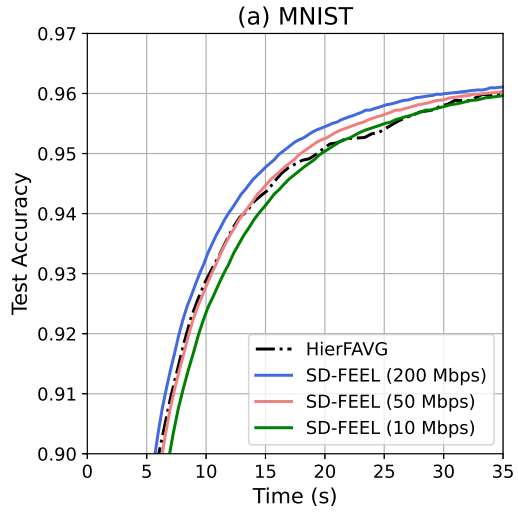
Fig. 5. Test accuracy over time on the (a) MNIST ($\tau_1 = 5$, $\tau_2 = 1$, and $\alpha = 1$) and (b) CIFAR-10 ($\tau_1 = 2$, $\tau_2 = 1$, and $\alpha = 5$) datasets.

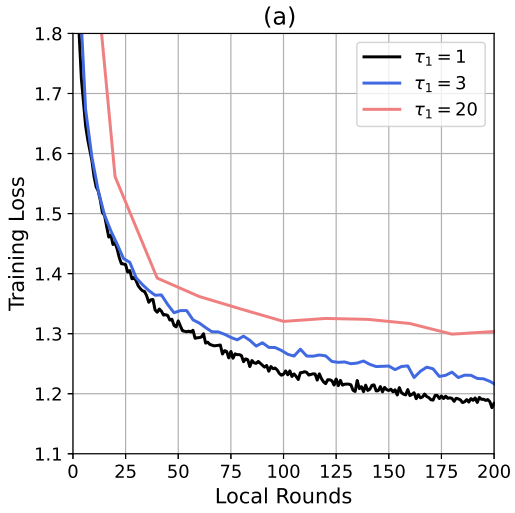
performance in terms of the test accuracy over time. Within the given training time, the test accuracy of SD-FEEL improves over FedAvg and FEEL due to efficient communication across edge servers, while their learned models are still unusable. In addition, we notice that in Fig. 4(a) and Fig. 5(a), SD-FEEL and HierFAVG have a small gap in terms of training loss and test accuracy, respectively, since the computation latency dominates the training time for this task on the MNIST dataset.

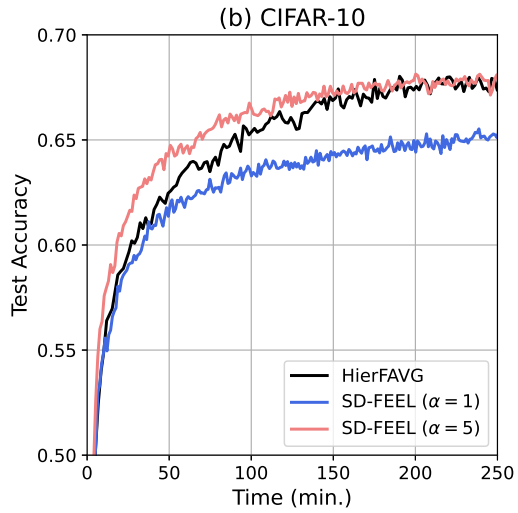
To further compare the performance of SD-FEEL and HierFAVG, Fig. 6(a) shows the test accuracy over time with different inter-server communication rates, i.e., $R_{\text{comm}}^{\text{sr-sr}} =$ 10 Mbps, 50 Mbps, and 200 Mbps. When edge servers share models with a slower rate (e.g., 10 Mbps), SD-FEEL reaches a lower test accuracy than HierFAVG. Correspondingly, a high communication speed among edge servers (e.g., 200 Mbps) ensures SD-FEEL to converge faster. Besides, in Fig. 6(b), we see that with a sparsely-connected network (i.e., the ring topology), SD-FEEL may have a slower convergence speed due to the model inconsistency among edge servers, which can be alleviated through multiple rounds of communications in inter-cluster model aggregation. These observations verify the discussion in Remark 3.

2) Impacts of Parameters: We investigate how the aggregation period τ_1 affects the learning performance on the MNIST dataset by showing the relationship between the training loss and training iterations (respectively training time) in Fig. 7(a) (respectively Fig. 7(b)). We fix $\tau_2 = 1$ and evaluate SD-FEEL at $\tau_1 = 1, 3$, and 20. Fig. 7(a) shows that a smaller value of τ_1 leads to a lower training loss within the given training iterations, as explained in Remark 1. This conclusion, however, is invalid when considering the training time, as shown in Fig. 7(b). Since less frequent communications between client nodes and edge servers can reduce the total latency, a larger value of τ_1 may be preferred. The inter-cluster model aggregation period τ_2 has similar behaviors, and the results are omitted due to space limitation.

We also evaluate the test accuracy of SD-FEEL on different







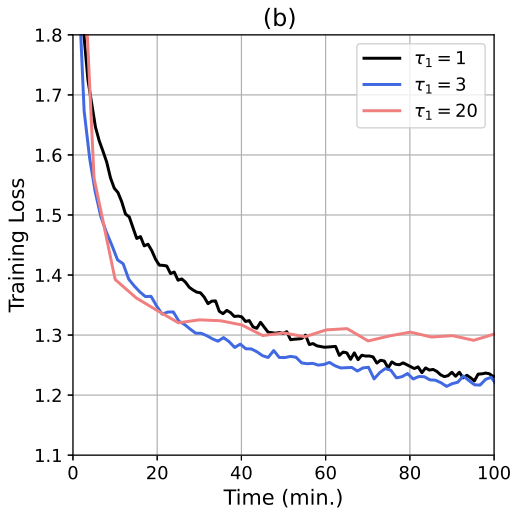


Fig. 6. Test accuracy over time on the (a) MNIST ($\tau_1 = 1$, $\tau_2 = 1$, and $\alpha = 1$) and (b) CIFAR-10 ($\tau_1 = 2$, $\tau_2 = 1$, $\alpha = 1$, and $R_{\text{comm}}^{\text{sr-sr}} = 50$ Mbps) datasets.

Fig. 7. Training loss of SD-FEEL ($\tau_2 = 1$ and $\alpha = 1$) over (a) iterations and (b) time on the CIFAR-10 dataset.

network topologies of the edge servers. As shown in Fig. 8, with $\alpha = 1$ round of communication, a more connected topology leads to a higher test accuracy within the given number of training iterations. This is because more information can be received from neighboring edge clusters in each round of inter-cluster model aggregation, as discussed in Remark 2. Besides, as α increases in the ring topology, the training speed becomes faster in terms of training iterations since more information is collected from neighboring edge clusters. When $\alpha = 10$ (respectively $\alpha = 15$), SD-FEEL with the ring topology leads to a comparable performance with fully-connected topology on the MNIST dataset (respectively CIFAR-10 dataset), which corroborates the discussion in Remark 2.

3) Effect of DAta and Device Heterogeneity: We test SD-FEEL with different degrees of data and device heterogeneity. We start with the effect of data heterogeneity in Fig. 9. According to Fig. 9(a), when each client node has more classes of data samples, the local training is less biased and thus SD-FEEL has a faster learning speed. Similarly in Fig. 9(b), a smaller value of β leads to a higher degree of data heterogeneity and thus slows down the training significantly.

To investigate the effect of device heterogeneity, we next compare synchronous SD-FEEL (denoted as *Sync.*), asynchronous SD-FEEL (denoted as *Async.*), and asynchronous SD-FEEL with a constant mixing matrix (denoted as *Vanilla Async.*) in Fig. 10. The computation deadline is chosen such that each client node is able to compute at least 100 (respectively 1000) batches of data samples on the MNIST (respectively CIFAR-10) dataset. Besides, we adopt $\psi(\delta_t^{(j)}) = \frac{1}{2(\delta_t^{(j)}+1)}$ to calculate the mixing matrix for inter-cluster model aggregation. According to Fig. 10, we observe that asynchronous SD-FEEL with a constant mixing matrix has a slightly slower convergence than synchronous training. Nevertheless, with the proposed staleness-aware mixing matrix, asynchronous SD-FEEL performs much better, which demonstrates the necessity of our proposed staleness-aware

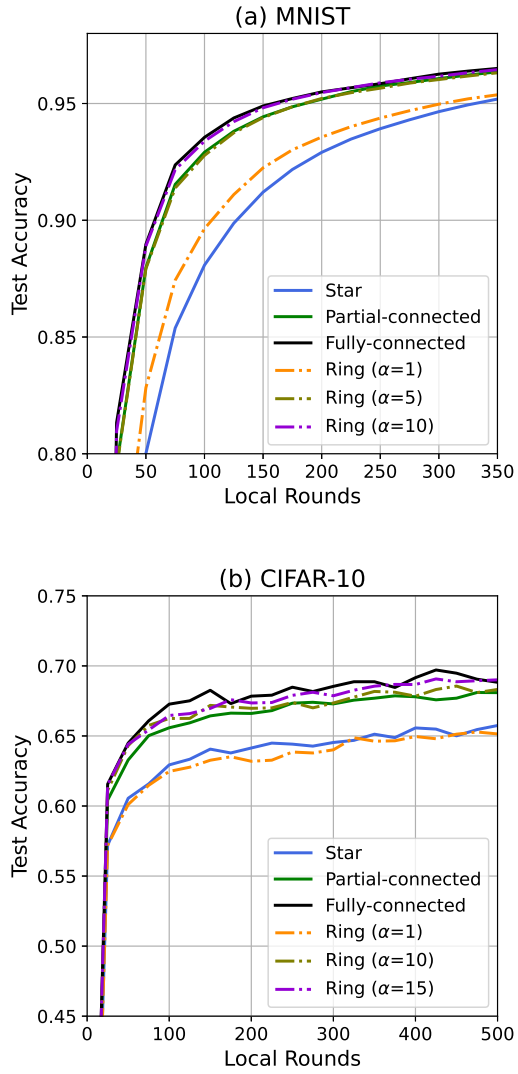


Fig. 8. Test accuracy over iterations with different network topologies ($\tau_1 = 5$ and $\tau_2 = 5$) on the (a) MNIST and (b) CIFAR-10 datasets.

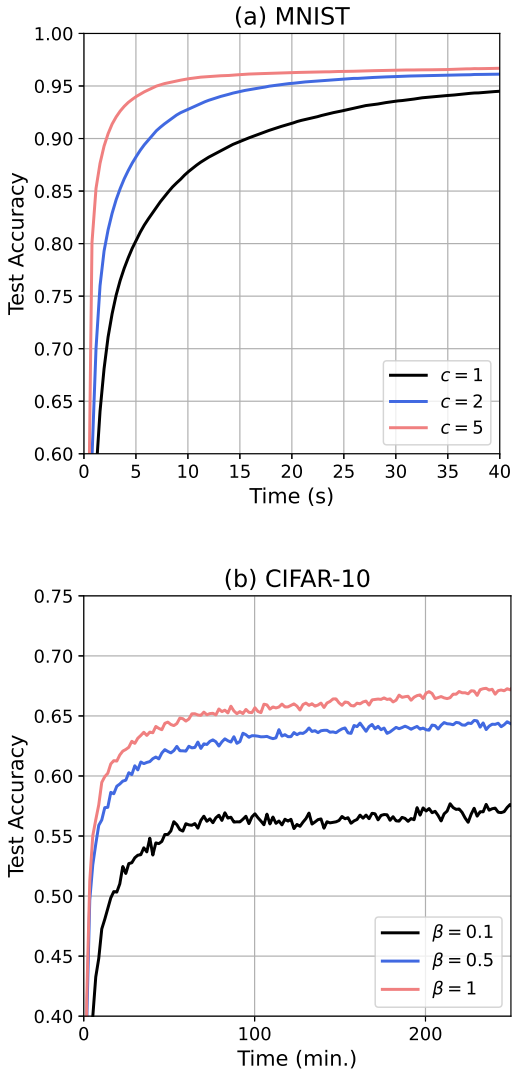


Fig. 9. Test accuracy over iterations with varying degrees of non-IIDness ($\tau_1 = 5$, $\tau_2 = 1$, $\alpha = 1$, and $H = 1$) on the (a) MNIST and (b) CIFAR-10 datasets.

mixing matrix.

Fig. 11 shows the test accuracy over time with different degrees of device heterogeneity on the CIFAR-10 dataset. When the computational resources are quite uneven (i.e., with a larger H), the convergence speed of synchronous SD-FEEL is degraded, since slower client nodes require more time for local training. Comparatively, the training efficiency of SD-FEEL is improved with the asynchronous training algorithm, which is more notable as the device heterogeneity becomes more significant. This is because the fast client nodes are allowed to perform more local training and thus have less idle time. With the limited training time, asynchronous SD-FEEL obtains an improvement in the test accuracy over synchronous training. However, if given a sufficiently long training time such that the data at the slower client nodes are fully exploited, synchronous SD-FEEL is able to achieve a higher test accuracy.

VI. Conclusions

In this paper, we investigated a novel FL system for privacy-preserving distributed learning in 6G networks, named semi-decentralized federated edge learning (SD-FEEL). It enjoys high training efficiency by employing low-latency communication among multiple edge servers. We presented the training algorithm for SD-FEEL with a convergence analysis on non-IID data. Then the effects of various parameters on the training performance of SD-FEEL were discussed. Moreover, to combat the device heterogeneity, we proposed an asynchronous training algorithm for SD-FEEL, followed by its convergence analysis. Simulation results demonstrated the benefits of SD-FEEL and corroborated our analysis. For future works, it is worth considering the performance of SD-FEEL under varying channel conditions. Further investigations are also needed for the selection of key algorithmic parameters as well as practical deployment.

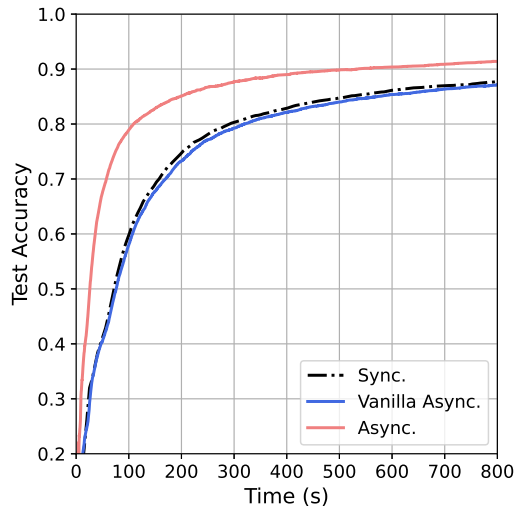


Fig. 10. Test accuracy over time on the MNIST dataset ($H = 10$).

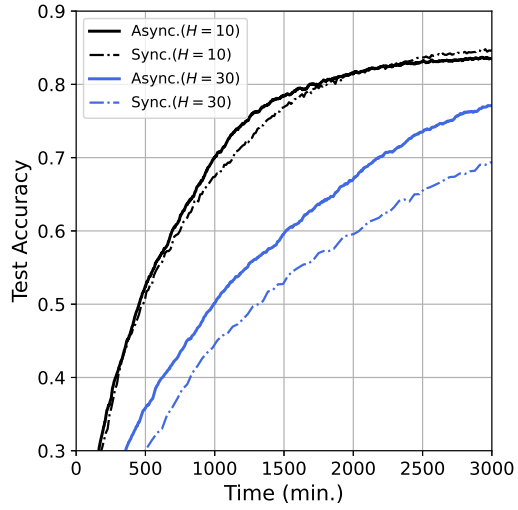


Fig. 11. Test accuracy over time on the CIFAR-10 dataset.

REFERENCES

[1] Y. Sun, J. Shao, Y. Mao, J. H. Wang, and J. Zhang, "Semi-decentralized federated edge learning for fast convergence on non-IID data." [Online]. Available: https://arxiv.org/pdf/2104.12678.pdf.

[2] Y. Sun, J. Shao, Y. Mao, and J. Zhang, "Asynchronous semi-decentralized federated edge learning for heterogeneous clients." [Online]. Available: https://arxiv.org/abs/2112.04737.pdf.

[3] K. L. Lueth, "State of the IoT 2020: 12 billion IoT connections, surpassing non-IoT for the first time." [Online]. Available: https://iot-analytics.com/state-of-the-iot-2020-12-billion-iot-connections-surpassing-non-iot-for-the-first-time/.

[4] K. B. Letaief, W. Chen, Y. Shi, J. Zhang, and Y.-J. A. Zhang, "The roadmap to 6G: Ai empowered wireless networks," *IEEE Commun. Mag.*, vol. 57, no. 8, pp. 84–90, Aug. 2019.

[5] D. C. Nguyen, M. Ding, P. N. Pathirana, A. Seneviratne, J. Li, D. Niyato, O. Dobre, and H. V. Poor, "6G Internet of Things: A comprehensive survey," *IEEE Internet Things J.*, vol. 9, no. 1, pp. 359–383, Jan. 2022.

[6] F. Meneghello, M. Calore, D. Zucchetto, M. Polese, and A. Zanella, "IoT: Internet of threats? A survey of practical security vulnerabilities in real IoT devices," *IEEE Internet Things J.*, vol. 6, no. 5, pp. 8182–8201, Oct. 2019.

[7] V. Hassija, V. Chamola, V. Saxena, D. Jain, P. Goyal, and B. Sikdar, "A survey on IoT security: application areas, security threats, and solution architectures," *IEEE Access*, vol. 7, pp. 82 721–82 743, Jun. 2019.

[8] Y. Shi, K. Yang, T. Jiang, J. Zhang, and K. B. Letaief, "Communication-efficient edge AI: Algorithms and systems," *IEEE Commun. Surveys Tuts.*, vol. 22, no. 4, pp. 2167–2191, 4th Quart. 2020.

[9] 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)*, Ft. Lauderdale, FL, USA, Apr. 2017.

[10] Y. Mao, C. You, J. Zhang, K. Huang, and K. B. Letaief, "A survey on mobile edge computing: The communication perspective," *IEEE Commun. Surveys Tuts.*, vol. 19, no. 4, pp. 2322–2358, 4th Quart. 2017.

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

[12] L. Liu, J. Zhang, S. Song, and K. B. Letaief, "Client-edge-cloud hierarchical federated learning," in *Proc. IEEE Int. Conf. Commun. (ICC)*, Dublin, Ireland, Jun. 2020.

[13] J. Wang, S. Wang, R.-R. Chen, and M. Ji, "Local averaging helps: Hierarchical federated learning and convergence analysis." [Online]. Available: https://arxiv.org/pdf/2010.12998.pdf.

[14] T. Castiglia, A. Das, and S. Patterson, "Multi-level local SGD: Distributed SGD for heterogeneous hierarchical networks," in *Proc. Int. Conf. Learn. Repr. (ICLR)*, Virtual Event, May 2020.

[15] 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.

[16] W. Shi, S. Zhou, Z. Niu, M. Jiang, and L. Geng, "Joint device scheduling and resource allocation for latency constrained wireless federated learning," *IEEE Trans. Wireless Commun.*, vol. 20, no. 1, pp. 453–467, Jan. 2021.

[17] J. Mills, J. Hu, and G. Min, "Communication-efficient federated learning for wireless edge intelligence in IoT," *IEEE Internet Things J.*, vol. 7, no. 7, pp. 5986–5994, Jul. 2020.

[18] M. M. Amiri and D. Gündüz, "Federated learning over wireless fading channels," *IEEE Trans. Wireless Commun.*, vol. 19, no. 5, pp. 3546–3557, May 2020.

[19] T. Nishio and R. Yonetani, "Client selection for federated learning with heterogeneous resources in mobile edge," in *Proc. IEEE Int. Conf. Commun. (ICC)*, Shanghai, China, May 2019.

[20] C. Xie, S. Koyejo, and I. Gupta, "Asynchronous federated optimization," in *Proc. Wkshop. Optim. Mach. Learn.*, Virtual Event, Dec. 2020.

[21] Q. Ma, Y. Xu, H. Xu, Z. Jiang, L. Huang, and H. Huang, "FedSA: A semi-asynchronous federated learning mechanism in heterogeneous edge computing," *IEEE J. Sel. Areas Commun.*, vol. 39, no. 12, pp. 3654–3672, Dec. 2021.

[22] W. Wu, L. He, W. Lin, R. Mao, C. Maple, and S. Jarvis, "SAFA: A semi-asynchronous protocol for fast federated learning with low overhead," *IEEE Trans. Comput.*, vol. 70, no. 5, pp. 655–668, May 2020.

[23] R. Saha, S. Misra, and P. K. Deb, "FogFL: Fog assisted federated learning for resource-constrained IoT devices," *IEEE Internet Things J.*, vol. 8, no. 10, pp. 8456–8463, May 2021.

[24] D.-J. Han, M. Choi, J. Park, and J. Moon, "FedMes: Speeding up federated learning with multiple edge servers," *IEEE J. Sel. Areas Commun.*, vol. 39, no. 12, pp. 3870–3885, Dec. 2021.

[25] X. You, D. Wang, P. Zhu, and B. Sheng, "Cell edge performance of cellular mobile systems," *IEEE J. Sel. Areas Commun.*, vol. 29, no. 6, pp. 1139–1150, Jun. 2011.

[26] O. Dekel, R. Gilad-Bachrach, O. Shamir, and L. Xiao, "Optimal distributed online prediction using mini-batches," *J. Mach. Learn. Res.*, vol. 13, no. 1, pp. 165–202, Jan. 2012.

[27] R. Elsässer, B. Monien, and R. Preis, "Diffusion schemes for load balancing on heterogeneous networks," *Theory Comput. Syst.*, vol. 35, no. 3, pp. 305–320, Jun. 2002.

[28] W. Liu, L. Chen, and W. Zhang, "Decentralized federated learning: Balancing communication and computing costs." [Online]. Available: https://arxiv.org/pdf/2107.12048.pdf.

[29] J. Wang, Q. Liu, H. Liang, G. Joshi, and H. V. Poor, "Tackling the objective inconsistency problem in heterogeneous federated optimization," in *Proc. 34th Conf. Adv. Neural Inf. Process. Syst.*, Virtual Event, Dec. 2020.

[30] Y. Wang, Y. Xu, Q. Shi, and T.-H. Chang, "Quantized federated learning under transmission delay and outage constraints," *IEEE J. Sel. Areas Commun.*, to appear.

[31] J. Wang and G. Joshi, "Adaptive communication strategies to achieve the best error-runtime trade-off in local-update SGD," in *Proc. Mach. Learn. Syst.*, Palo Alto, CA, USA, Mar. 2019.

[32] H. Tang, X. Lian, M. Yan, C. Zhang, and J. Liu, "D^2: Decentralized training over decentralized data," in *Proc. Int. Conf. Mach. Learn. (ICML)*, Stockholm, Sweden, Jul. 2018.
[33] L. Bottou, F. E. Curtis, and J. Nocedal, "Optimization methods for large-scale machine learning," *SIAM Rev.*, vol. 60, no. 2, pp. 223–311, Aug. 2018.
[34] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proc. IEEE*, vol. 86, no. 11, pp. 2278–2324, Nov. 1998.
[35] A. Krizhevsky and G. Hinton, "Learning multiple layers of features from tiny images." [Online]. Available: https://www.cs.toronto.edu/~kriz/cifar.html.
[36] K. Hsieh, A. Phanishayee, O. Mutlu, and P. Gibbons, "The non-IID data quagmire of decentralized machine learning," in *Proc. Int. Conf. Mach. Learn. (ICML)*, Virtual Event, Jul. 2020.
[37] M. Yurochkin, M. Agarwal, S. Ghosh, K. Greenewald, N. Hoang, and Y. Khazaeni, "Bayesian nonparametric federated learning of neural networks," in *Proc. Int. Conf. Mach. Learn. (ICML)*, Long Beach, CA, USA, Jun. 2019.
[38] C. He, S. Li, J. So, X. Zeng, M. Zhang, H. Wang, X. Wang, P. Vepakomma, A. Singh, H. Qiu *et al.*, "FedML: A research library and benchmark for federated machine learning." [Online]. Available: https://arxiv.org/pdf/2007.13518.pdf.
[39] V. Smith, C.-K. Chiang, M. Sanjabi, and A. Talwalkar, "Federated multi-task learning," in *Proc. 31st Adv. Neural Inf. Process. Syst. (NeurIPS)*, Long Beach, CA, USA, Dec. 2017.
[40] L. Hu, G. Sun, and Y. Ren, "CoEdge: Exploiting the edge-cloud collaboration for faster deep learning," *IEEE Access*, vol. 8, pp. 100 533–100 541, May 2020.

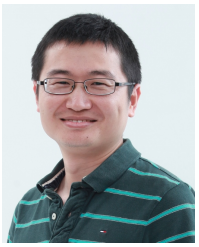
Yuchang Sun (Student member, IEEE) received the B.Eng. degree in electronic and information engineering from Beijing Institute of Technology in 2020. She is currently pursuing a Ph.D. degree at Hong Kong University of Science and Technology. Her research interests include federated learning and distributed optimization.

Jiawei Shao (Student member, IEEE) received the B.Eng. degree in telecommunication engineering from Beijing University of Posts and Telecommunications in 2019. He is currently pursuing a Ph.D. degree at Hong Kong University of Science and Technology. His research interests include edge intelligence and distributed learning.

Yuyi Mao (Member, IEEE) received the B.Eng. degree in information and communication engineering from Zhejiang University, Hangzhou, China, in 2013, and the Ph.D. degree in electronic and computer engineering from The Hong Kong University of Science and Technology, Hong Kong, in 2017. He was a Lead Engineer with the Hong Kong Applied Science and Technology Research Institute Co., Ltd., Hong Kong, and a Senior Researcher with the Theory Lab, 2012 Labs, Huawei Tech. Investment Co., Ltd., Hong Kong. He is currently a Research Assistant Professor with the Department of Electronic and Information Engineering, The Hong Kong Polytechnic University, Hong Kong. His research interests include wireless communications and networking, mobile-edge computing and learning, and wireless artificial intelligence.

He was the recipient of the 2021 IEEE Communications Society Best Survey Paper Award and the 2019 IEEE Communications Society and Information Theory Society Joint Paper Award. He was also recognized as an Exemplary Reviewer of the IEEE Transactions on Communications and the IEEE Wireless Communications Letters in 2020 and 2019, respectively.

Jessie Hui Wang (Senior member, IEEE) received the B.S. and M.S. degrees in computer science from Tsinghua University and the Ph.D. degree in information engineering from The Chinese University of Hong Kong in 2007. She is currently an Associate Professor with Tsinghua University. Her research interests include Internet routing, cloud computing, edge computing, network measurement, and Internet economics.



Jun Zhang (Fellow, IEEE) received the B.Eng. degree in Electronic Engineering from the University of Science and Technology of China in 2004, the M.Phil. degree in Information Engineering from the Chinese University of Hong Kong in 2006, and the Ph.D. degree in Electrical and Computer Engineering from the University of Texas at Austin in 2009. He is an Associate Professor in the Department of Electronic and Computer Engineering at the Hong Kong University of Science and Technology. His research interests include wireless communications and networking, mobile edge computing and edge AI, and cooperative AI.

Dr. Zhang co-authored the book Fundamentals of LTE (Prentice-Hall, 2010). He is a co-recipient of several best paper awards, including the 2021 Best Survey Paper Award of the IEEE Communications Society, the 2019 IEEE Communications Society & Information Theory Society Joint Paper Award, and the 2016 Marconi Prize Paper Award in Wireless Communications. Two papers he co-authored received the Young Author Best Paper Award of the IEEE Signal Processing Society in 2016 and 2018, respectively. He also received the 2016 IEEE ComSoc Asia-Pacific Best Young Researcher Award. He is an Editor of IEEE Transactions on Communications, and was an editor of IEEE Transactions on Wireless Communications (2015-2020). He served as a MAC track co-chair for IEEE Wireless Communications and Networking Conference (WCNC) 2011 and a co-chair for the Wireless Communications Symposium of IEEE International Conference on Communications (ICC) 2021.