Accelerating Federated Learning via Parallel Servers: A Theoretically Guaranteed Approach

Xuezheng Liu, Zhicong Zhong, Yipeng Zhou, *Member, IEEE*, Di Wu, *Senior Member, IEEE*, Xu Chen, Min Chen, *Fellow, IEEE*, and Quan Z. Sheng, *Member, IEEE*

***Abstract*— With the growth of participating clients, the centralized parameter server (PS) will seriously limit the scale and efficiency of Federated Learning (FL). A straightforward approach to scale up the FL system is to construct a Parallel FL (PFL) system with multiple parallel PSes. However, it is unclear whether PFL can really accelerate FL or reduce the training time of FL. Even if the answer is yes, it is non-trivial to design a highly efficient parameter average algorithm for a PFL system. In this paper, we propose a completely parallelizable FL algorithm called P-FedAvg under the PFL architecture. P-FedAvg extends the well-known FedAvg algorithm by allowing multiple PSes to cooperate and train a learning model together. In P-FedAvg, each PS is only responsible for a fraction of total clients, but PSes can mix model parameters in a dedicatedly designed way so that the FL model can well converge. Different from heuristic-based algorithms, P-FedAvg is with theoretical guarantees. To be rigorous, we theoretically analyze the convergence rate of P-FedAvg in terms of the number of conducted iterations, the communication cost of each global iteration and the optimal weights for each PS to mix parameters with its neighbors. Based on theoretical analysis, we conduct a case study on five typical overlay topolgoies formed by PSes to further examine the communication efficiency under different topologies, and investigate how the overlay topology affects the convergence rate, communication cost and robustness of a PFL system. Lastly, we perform extensive experiments with real datasets to verify our analysis and demonstrate that P-FedAvg can significantly speed up FL than traditional FedAvg and other competitive baselines. We believe that our work can help to lay a theoretical foundation for building more efficient PFL systems.**

***Index Terms*— Parallel federated learning, convergence rate, network topology, mixing matrix.**

I. Introduction

THE past decade has witnessed the tremendous success achieved by machine learning. However, an arising concern threatening the advances of machine learning is the potential leakage of user privacy because data samples collected to train machine learning models may contain end users' sensitive and confidential information [2]–[4]. To reconcile the concern on data privacy leakage, the *Federated Learning* (FL) framework is devised. In a typical FL system, a centralized parameter server (PS) is deployed to coordinate the learning process for a number of decentralized clients. Instead of collecting original data samples from clients, only intermediate computations (*e.g.*, model parameters) are gathered from clients via the Internet. To facilitate the collaboration of multiple clients, various model average algorithms are designed, such as the FedAvg algorithm [5].

In vanilla FL, a single centralized PS is responsible for coordinating clients. For a large-scale FL system, it turns out that the communication between multiple decentralized clients and the single PS would be the bottleneck and the learning process can be retarded considerably due to the following two reasons [6], [7]. *Firstly*, for clients located in different geographic areas, it is difficult to establish a fast network to connect all of them with a single PS. *Secondly*, the communication capacity of a single PS is limited while the client population can be a huge number [8]. Thereby, the single PS can only interact with a small portion of all clients in each global iteration, and hence the overall learning efficiency can be low.

To overcome the bottleneck of a single PS in FL, a straightforward solution is to build a parallel FL (PFL) system by deploying multiple decentralized and parallel PSes in the system. Fig. 1 shows a simple PFL system with clients distributed in three cities. By deploying three PSes such as edge servers in different cities, the PFL system can cover all clients completely and exclusively. Meanwhile, PSes exchange model parameters with each other so that the model can finally converge. However, constructing an efficient PFL faces quite a few challenges: *1) In theory, it is unclear whether PFL will*

Manuscript received 17 August 2021; revised 19 December 2021; accepted 29 March 2022; approved by IEEE/ACM TRANSACTIONS ON NETWORKING Editor G. Joshi. Date of publication 5 May 2022; date of current version 17 October 2022. This work was supported in part by the National Natural Science Foundation of China under Grant U1911201 and Grant U2001209, in part by the Science and Technology Planning Project of Guangdong Province under Grant 2021A0505110008, in part by the Program for Guangdong Introducing Innovative and Entrepreneurial Teams under Grant 2017ZT07 × 355, and in part by the Pearl River Talent Recruitment Program under Grant 2017GC010465. This is an extended version of a paper that appeared in the IEEE International Conference on Computer Communications (INFOCOM 2021) [DOI: 10.1109/INFOCOM42981.2021.9488877]. *(Corresponding author: Di Wu.)*

Xuezheng Liu, Di Wu, and Xu Chen are with the School of Computer Science and Engineering, Sun Yat-sen University, Guangzhou 510275, China, and also with the Guangdong Key Laboratory of Big Data Analysis and Processing, Guangzhou 510006, China (e-mail: lxuezh@mail2.sysu.edu.cn; wudi27@mail.sysu.edu.cn; chenxu35@mail.sysu.edu.cn).

Zhicong Zhong is with the Department of Computer Science and Engineering, Chinese University of Hong Kong, Hong Kong 999077, China (e-mail: zhiczhong3@mail2.sysu.edu.cn).

Yipeng Zhou and Quan Z. Sheng are with the Faculty of Science and Engineering, School of Computing, Macquarie University, Sydney, NSW 2109, Australia (e-mail: yipeng.zhou@mq.edu.au; michael.sheng@mq.edu.au).

Min Chen is with the School of Computer Science and Technology, Huazhong University of Science and Technology, Wuhan 430074, China (e-mail: minchen@ieee.org).

This article has supplementary downloadable material available at https://doi.org/10.1109/TNET.2022.3168939, provided by the authors.

Digital Object Identifier 10.1109/TNET.2022.3168939

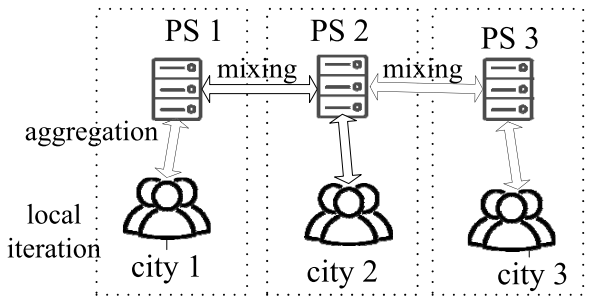


Fig. 1. A simple PFL system with three PSes located in three different cities to serve their clients.

converge or lower the convergence rate than the traditional FL; 2) Even if PFL can converge, a highly efficient model average algorithm is still unknown for PFL; 3) To speed up the convergence, what is the optimal network topology to connect these PSes? 4) How should a PS mix parameters exchanged with its neighbor PSes?

In this paper, we propose a parallel FL algorithm called *P-FedAvg* for the PFL system, which is an extension of the FedAvg algorithm [1]. [1] The *P-FedAvg* algorithm works as follows. Each client conducts a number of local iterations before its model parameters are uploaded to its PS. After collecting model parameters from selected clients, each PS conducts a round of global iteration by aggregating model parameters uploaded from its clients and then mixing model parameters with its neighbors. Then, each PS distributes the mixed model parameters to its covered clients to kick off a new round of global iteration.

To prove the feasibility of PFL, we formally prove the convergence of P-FedAvg in terms of the number of iterations. By taking the communication time cost of each global iteration into account, we prove that PFL can accelerate FL or reduce the training time of FL when communication is the bottleneck of the FL system. The reason lies in that our method can distribute the communication load of a single PS in traditional FL to multiple parallel PSes in PFL. Furthermore, we continue to optimize the weights for PSes to mix their parameters with neighbor PSes so as to maximize the final model accuracy. We also explore how the overlay topology connecting PSes affects the convergence rate, the system robustness and the communication cost through case study with five typical overlay topologies. Finally, extensive experiments are conducted by using the MNIST, FEMNIST [9] and CIFAR10 [10] datasets, and the experimental results demonstrate that P-FedAvg can significantly outperform the original FedAvg and other competitive baselines.

The rest of the paper is organized as follows. The related work is discussed in Sec. II. The P-FedAvg algorithm is designed in Sec. III. Its convergence rates are derived in Sec. IV. Its communication cost is analyzed in Sec. V. The influence of the overlay topology and the algorithm to optimize the mixing matrix are presented in Sec. VI. The experiments are reported in Sec. VII before our paper is concluded in Sec. VIII.

[1]Our initial work has been published in IEEE INFOCOM 2021. The current version has significantly extended our initial work by completing the proof of theorems, conducting analysis of communication cost, revising case study and extending experiments.

II. Related Work

In this section, we briefly review the literature related to our study, including federated learning, decentralized parallel learning, and communications in FL.

A. Federated Learning

The FL framework was originally proposed by [5] to preserve privacy for users without uploading data samples. However, no theoretical results on the effectiveness of FedAvg were provided. Later, Stich [11] and Yu *et al.* [12] theoretically derived the convergence rates of FedAvg for convex and non-convex loss functions respectively, by assuming iid samples and full participation mode. However, it is more common that samples on FL clients are non-iid. In light of this, the convergence rate of FedAvg with non-iid samples was studied in [13]–[16]. Since clients need to communicate with the PS via the Internet, a number of works studied how to improve the convergence rate of FedAvg with heterogeneous and limited resources [17]–[19]. For most previous works, they commonly assumed that there existed a single centralized PS, which was responsible for the communication with all clients. The bottleneck caused by the single PS in FL has been largely overlooked by these works.

B. Decentralized Parallel Learning

To overcome the shortcoming of a single PS, decentralized parallel learning (*e.g.*, distributed SGD) has been extensively studied in recent years. In the distributed SGD framework, multiple workers execute computation and communication tasks independently. Lian *et al.* [7] proposed a synchronized decentralized parallel SGD algorithm, in which all workers computed local gradients in parallel and then exchanged gradients with neighbors synchronously, and later proposed an asynchronous distributed SGD algorithm that can achieve a faster convergence rate by assuming iid samples in [20]. Wang and Joshi [21] extended the analysis of the above distributed SGD algorithm by allowing each client to execute multiple iterations before synchronization. The convergence rate analysis with non-iid samples was provided in [22]. Koloskova *et al.* [23] investigated the relation between the convergence rate of decentralized SGD algorithms and the connectivity of the graph formed by workers. The above algorithms are applicable for a computing cluster in which workers can communicate with each other efficiently. However, due to the unreliable client-to-client communication, the communication overhead is too high if all FL clients communicate with each other via the Internet.

C. Communications in FL

Vanhaesebrouck *et al.* [6] pointed out that it is almost impossible to deploy an always-on reliable PS in practice. Communication congestion occurs frequently if there is a

single PS in FL [7]. Some previous works studied how to reduce communication traffic of the PS in FL. Li *et al.* [24] proposed a gradient sparsification technique to reduce the size of transmitted data. Reisizadeh *et al.* [25] synthetically adopted methods of quantized message-passing, periodic averaging and partial participation to address the communications and scalability challenges in FL. Liu *et al.* [26] proposed a hierarchical FL, in which a three-level tree was formed among PSes and clients. However, such kind of architecture was not flexible and the root node was still a communication bottleneck. Lian *et al.* [7], [20] showed that the decentralized learning algorithms can outperform the centralized algorithms when communication conditions were poor. Wang *et al.* [27] develops a network-aware distributed learning methodology to improve network resource utilization across a network of fog devices.

Different from these works, our contribution lies in proposing a flexible PFL framework that can distribute the communication traffic among multiple PSes. In addition, theoretical analysis is provided to guarantee fast convergence of PFL.

III. Design of Parallel FL Algorithm

In this section, we first introduce the system model of PFL and then describe the design of parallel learning algorithm (namely, *P-FedAvg*) for PFL.

A. System Model of PFL

In a parallel FL system, suppose that there are M PSes. For PS i, it covers a set of clients denoted by $\mathcal{N}_i$ with cardinality N_i. Meanwhile, $\mathcal{N} = \cup_{i=1}^{M}\mathcal{N}_i$ and $\mathcal{N}_i \cap \mathcal{N}_{i'} = \phi$. The M PSes are connected, and the formed topology can be captured by a matrix $\mathbf{L}$. The dimension of $\mathbf{L}$ is $M \times M$. If PS i and PS i' are connected, we have $L_{ii'} = 1$. [2] Otherwise $L_{ii'} = 0$. Note that we only consider an undirected scenario which implies that $L_{ii'} = L_{i'i}$.

Data samples are distributed on clients. For client j, it owns a set of samples, $\mathcal{D}_j$. The objective of these clients is to train a common machine learning model which can be expressed by a loss function $\tilde{f}(\mathbf{x})$. Here, $\mathbf{x}$ with dimension d is the set of parameters to be determined by the learning algorithm. Formally, $\tilde{f}(\mathbf{x})$ can be defined by samples on clients as follows:

$$\tilde{f}(\mathbf{x}) = \sum_{i=1}^{M} \sum_{j \in \mathcal{N}_i} p_j \tilde{f}_j(\mathbf{x}). \tag{1}$$

Here, $\tilde{f}_j$ is the portion of the loss function contributed by client j, $\tilde{f}_j : \mathbb{R}^d \rightarrow \mathbb{R}, \forall j \in \mathcal{N}$, $p_j = \frac{D_j}{\sum_{i=1}^{M}\sum_{j \in \mathcal{N}_i} D_j}$, D_j is the cardinality of $\mathcal{D}_j$. It can be defined as $\tilde{f}_j(\mathbf{x}) = E_{\xi_j \in \mathcal{D}_j}\tilde{f}(\mathbf{x}; \xi_j), \forall j \in \mathcal{N}$. $\tilde{f}(\mathbf{x}; \xi_j)$ is the loss function defined with a single sample ξ_j. The learning objective is to find $\mathbf{x}^*$ that can achieve the minimized loss function, *i.e.*, $\tilde{f}^* := \tilde{f}(\mathbf{x}^*) = min\ \tilde{f}(\mathbf{x})$. If $f_j(\mathbf{x}) = MN_i p_j \tilde{f}_j(\mathbf{x})$, $\tilde{f}(\mathbf{x})$ can be re-defined by

$$f(\mathbf{x}) = \sum_{i=1}^{M} \frac{1}{M} \sum_{j \in \mathcal{N}_i} \frac{1}{N_i} f_j(\mathbf{x}). \tag{2}$$

Assuming that M, N_i and p_j are all constant in the process of training, minimizing f and $\tilde{f}$ are equivalent. Note that we will use f_j as the loss function contributed by client j in the following.

As described in Fig. 1, the parallel FL works as follows:

1) Each PS i distributes the set of model parameters [3] $\mathbf{x}$ to a number of selected clients in $\mathcal{N}_i$ to start a round of global iteration.
2) Each client conducts E rounds of local iterations and returns the updated $\mathbf{x}$ to its PS.
3) Each PS i aggregates model parameters returned from its clients, and then mixes the model parameters with its neighbor PSes.
4) If the termination conditions are not satisfied, go back to Step 1.

B. P-FedAvg Algorithm

We first define $\mathbf{x}_j$ as the parameter vector maintained by client j. The intermediate parameter vector $\mathbf{v}_i = \frac{1}{K_i}\sum_{j \in \mathcal{K}_i} \mathbf{x}_j$ for PS i represents the model parameters obtained by aggregating parameters from clients in $\mathcal{K}_i$ with cardinality K_i. $\mathbf{x}_i$ is the parameter vector after exchanging intermediate parameter vectors between PSes. We further define the mixing vector $\mathbf{w}_i = (w_{i1}, w_{i2}, \ldots, w_{iM})^T$ where $w_{ii'} \geq 0$ with $1 \leq i' \leq M$ and $\sum_{i'=1}^{M} w_{ii'} = 1$. $\mathbf{w}_i$ represents the weights for PS i to exchange model parameters with its neighbors. In other words, if PS i exchanges its model parameters with its neighbors, $\mathbf{x}_i$ is updated as $\mathbf{x}_i = \mathbf{V}\mathbf{w}_i$ where $\mathbf{V} = (\mathbf{v}_1, \ldots, \mathbf{v}_M)$. Similarly, we define the mixing matrix $\mathbf{W}$ as $\mathbf{W} = (\mathbf{w}_1, \ldots, \mathbf{w}_M)$. $\mathbf{W}$ can be determined based on the topology matrix $\mathbf{L}$, which will be further discussed later.

Let $\nabla f(\mathbf{x}, \mathcal{B})$ denote the gradient of the loss function obtained with the sample batch set $\mathcal{B}$ with cardinality B. Specifically, it is $\nabla f(\mathbf{x}, \mathcal{B}) = \frac{1}{B}\sum_{\forall \xi \in \mathcal{B}} \nabla f(\mathbf{x}, \xi)$ where $\nabla f(\mathbf{x}, \xi)$ is the gradient of the loss function obtained with a particular sample ξ. Let η denote the learning rate. By extending the FedAvg algorithm proposed in the seminal work [5], we design the Parallel FedAvg (P-FedAvg) algorithm in Alg. 1.

In Alg. 1, each client executes E rounds of local iterations before its parameters are returned to PS. r is the index for global iterations while t_r is the index for local iterations. $rE + t_r$ is the index for the total number of iterations including both global and local iterations. $\mathcal{K}_i(t)$ implies that the selected clients are different in round r for a fixed cardinality K_i.

In comparison with the original FedAvg algorithm, each PS in P-FedAvg conducts FedAvg in parallel. Note that, to make these PSes reach a consensus, in each round of global iteration, each PS not only aggregates parameters from its covered clients (*i.e.*, line 8 in Alg. 1), but also exchanges model parameters with its neighbors (*i.e.*, line 10 in Alg. 1).

[2] In our study, $L_{ii} = 1$.

[3] $\mathbf{x}$ can be randomly generated for the first round of global iteration.

Algorithm 1 Parallel Federated Averaging (P-FedAvg).

Input: learning rate η_t, mixing matrix $\mathbf{W}$, initialized parameters $\mathbf{x}_0$, number of iterations of each round E
Output: model parameter $\mathbf{x}$

Server($\mathbf{x}_0$):
Initialize model parameter $\mathbf{x}$ with $\mathbf{x}_0$
for each round $r = 1, 2, \ldots, \frac{T}{E}$ **do**
  for each PS $i = 1..M$ parallel **do**
    $\mathcal{K}_i$ = A random set of $\mathcal{N}_i$ selected by PS i
    for each client $j \in \mathcal{K}_i(r)$ parallel **do**
      $\mathbf{x}_j$ = Client($\mathbf{x}_i$)
    end for
    $\mathbf{v}_i = \frac{1}{K_i} \sum_{j \in \mathcal{K}_i(t)} \mathbf{x}_j$
    Exchange model parameters with neighbor PSes
    $\mathbf{x}_i = \mathbf{V}\mathbf{w}_i$
  end for
end for

Client(x):
for t_r=1,2..,E **do**
  $\mathcal{B}(t_r)$ = A batch of samples selected from local data
  $\mathbf{x} = \mathbf{x} - \eta_{rE+t_r} \frac{1}{|\mathcal{B}|} \sum_{\forall \xi \in \mathcal{B}(t_r)} \nabla f(\mathbf{x}, \xi)$
end for
return $\mathbf{x}$

C. Communication Topology and Mixing Matrix

Intuitively, to make the P-FedAvg algorithm converge as fast as possible, PSes should form a fully connected graph. However, it is well known that the communication cost via the Internet is expensive. This implies that it is impractical to construct a fully connected graph for M PSes. Thus, it is more reasonable to assume that M PSes form a connected graph with a limited number of edges.

In this work, we assume that the topology formed by PSes is fixed which is determined by system operators. Given the topology matrix $\mathbf{L}$, we have $w_{ii'} > 0$ only if $L_{ii'} = 1$. Next the problem is to determine the mixing matrix $\mathbf{W}$ based on L so that the P-FedAvg algorithm can converge with the fastest rate.

We defer the determination of $\mathbf{W}$ after the analysis of the convergence rate. Here, we temporarily assume that $\mathbf{W}$ is known so that we can derive the converge rate of P-FedAvg first. Note that $\mathbf{W}$ is a doubly stochastic symmetric matrix with $\mathbf{W}\mathbf{1} = \mathbf{1}$, $\mathbf{1}^T\mathbf{W} = \mathbf{1}^T$ and $\mathbf{W} = \mathbf{W}^T$.

IV. Convergence Analysis

In this section, we provide the convergence analysis of the P-FedAvg algorithm under non-iid data distribution. Our analysis will show that the converge rate is determined by the total number of conducted iterations, the number of PSes and the mixing matrix $\mathbf{W}$.

A. Notation and Definition

Let t denote the index of the total number of conducted iterations, which implies that $r = \lfloor \frac{t}{E} \rfloor$ and $t_r = t \ mod \ E$. To facilitate our analysis, we define a number of variables as follows. Based on $\mathbf{v}_i(t)$ and $\mathbf{x}_i(t)$, we define

$$\bar{\mathbf{v}}(t) = \frac{1}{M} \sum_{i=1}^{M} \mathbf{v}_i(t), \tag{3}$$

$$\bar{\mathbf{x}}(t) = \frac{1}{M} \sum_{i=1}^{M} \mathbf{x}_i(t) = \sum_{i=1}^{M} \sum_{j \in \mathcal{K}_i(t)} \frac{1}{MK_i} \mathbf{x}_j(t). \tag{4}$$

In fact, $\bar{\mathbf{v}}(t)$ and $\bar{\mathbf{x}}(t)$ are virtual variables representing the global consensus obtained by aggregating parameters across all PSes. We use them to prove that each $\mathbf{x}_i(t)$ will converge to $\bar{\mathbf{x}}(t)$ finally. For simplicity, we define the parameter matrix after iteration t as

$$\mathbf{X}(t) = [\mathbf{x}_1(t), \mathbf{x}_2(t), \ldots, \mathbf{x}_M(t)]. \tag{5}$$

Similarly, the virtual parameter matrix is defined as

$$\bar{\mathbf{X}}(t) = [\bar{\mathbf{x}}(t), \bar{\mathbf{x}}(t), \ldots, \bar{\mathbf{x}}(t)] = \mathbf{X}(t) \frac{1}{M} \mathbf{1}\mathbf{1}^T. \tag{6}$$

Here $\mathbf{1}\mathbf{1}^T$ represents a $M \times M$ matrix with all elements of value 1.

To derive the evaluation of $\mathbf{X}(t)$, we define the gradient of PS i as $\nabla f_i(\mathbf{x}_i(t)) = \frac{1}{K_i} \sum_{j \in \mathcal{K}_i(t)} \nabla f(\mathbf{x}_j(t), \mathcal{B}_j)$ where $\nabla f(\mathbf{x}_j(t), \mathcal{B}_j)$ is the gradient obtained with batch samples of client j. The gradient matrix is denoted by

$$\partial f(\mathbf{X}(t)) = [\nabla f_1(\mathbf{x}_1(t)), \nabla f_2(\mathbf{x}_2(t)), \ldots, \nabla f_M(\mathbf{x}_M(t))]. \tag{7}$$

Let η_t be the learning rate at iteration t. From a logical view, when $t \bmod E \neq 0$, the update of client j's parameters would be $\mathbf{x}_j(t) = \mathbf{x}_j(t-1) - \eta_t \nabla f(\mathbf{x}_j(t), \mathcal{B}_j)$. Otherwise if $t \bmod E = 0$, the parameters will be aggregated. In summary, the relationship between $\mathbf{x}$ and $\mathbf{v}$ can be expressed as:

$$\mathbf{V}(t) = \mathbf{X}(t-1) - \eta_t \partial f(\mathbf{X}(t)), \tag{8}$$

$$\mathbf{X}(t) = \begin{cases} \mathbf{V}(t) & t \bmod E \neq 0, \\ \mathbf{V}(t)\mathbf{W} & t \bmod E = 0, \end{cases} \tag{9}$$

where $\mathbf{W}$ is the mixing matrix and $\mathbf{V}$ is the intermediate computation before PSes exchange parameters with each other.

B. Assumptions

Similar to previous work [7], [13], [23], we make a few necessary assumptions to facilitate our analysis.

Assumption 1: (L-smoothness) The loss function, $f_j : \mathbb{R}^d \to \mathbb{R}, j \in \mathcal{N}$, is differentiable and there exists a constant L such that for all $\mathbf{x}, \mathbf{x}' \in \mathbb{R}^d$: $\| \nabla f_j(\mathbf{x}, \mathcal{D}_j) - \nabla f_j(\mathbf{x}', \mathcal{D}_j) \| \leq L \| \mathbf{x} - \mathbf{x}' \|$.

Assumption 2: (μ-convex) The loss function, $f_j : \mathbb{R}^d \to \mathbb{R}, \forall j \in \mathcal{N}$, is μ-convex which means that there exists a constant μ such that for all $\mathbf{x}, \mathbf{x}' \in \mathbb{R}^d$: $f_j(\mathbf{x}) - f_j(\mathbf{x}') + \frac{\mu}{2} \| \mathbf{x} - \mathbf{x}' \|^2 \leq \langle \nabla f_j(\mathbf{x}, \mathcal{D}_j), (\mathbf{x} - \mathbf{x}') \rangle$.

Assumption 3: (Bounding Stochastic Gradient) For client j, $\xi_j(t)$ represents the data sampled from the local data of the client j at time t. The stochastic gradients are uniformly bounded by G: $\mathbb{E} \| \nabla f_j(\mathbf{x}_j(t), \xi_j(t)) \|^2 \leq G^2, \forall j \in \mathcal{N}$.

TABLE I
SUMMARY OF MAIN NOTATIONS

M	The number of servers
$\mathcal{N}_i$	The client set covered by server i
N_i	The cardinality of client set $\mathcal{N}_i$
$\mathcal{K}_i$	The selected client set covered by server i
K_i	The cardinality of selected client set $\mathcal{K}_i$
$\mathcal{D}_j$	The dataset owned by client j
D_j	The cardinality of dataset $\mathcal{D}_j$
p_j	The percentage of dataset owned by client j
$\mathcal{B}_j$	The batch sampled from dataset $\mathcal{D}_j$
$\mathbf{x}_j$	The parameter vector maintained by client j
$\mathbf{v}_i$	The intermediate parameter vector obtained by aggregating parameters from client set $\mathcal{K}_i$
$\mathbf{x}_i$	The parameter vector obtained by aggregating parameters from client set $\mathcal{K}_i$
$\bar{\mathbf{x}}$	The average of all $\mathbf{x}_i$
$\bar{\mathbf{v}}$	The average of all $\mathbf{v}_i$
$f(\mathbf{x})$	Loss function
$f_j(\mathbf{x})$	Loss function for client j
$\nabla f(\mathbf{x}_j(t), \mathcal{B}_j(t))$	The gradient obtained with batch samples of client j
$\nabla f(\mathbf{x}_i(t))$	The gradient obtained by averaging gradients of all selected clients
f^*	The minimized loss function
$\mathbf{W}$	Mixing matrix
E	The number of local iterations in one global iteration

Assumption 4: (Bounding the variance) For client j, given a parameter vector $\mathbf{x}_j$ and a sample ξ_j randomly collected from $\mathcal{D}_j$, the variance of stochastic gradients is bounded by: $\mathbb{E} \parallel \nabla f_j(\mathbf{x}_j; \xi_j) - \nabla f_j(\mathbf{x}_j, \mathcal{D}_j) \parallel^2 \leq \sigma_j^2, \forall j \in \mathcal{N}$.

Assumption 5: (Expected Consensus Rate) The mixing matrix, $\mathbf{W}$, is a symmetric stochastic matrix, i.e., $\mathbf{W}^T = \mathbf{W}$, $\mathbf{1}^T\mathbf{W} = \mathbf{1}^T$ *and* $\mathbf{W1} = \mathbf{1}$. *There exists a constant* $0 < p \leq 1$ *such that* $\forall \mathbf{X} \in \mathbb{R}^{d\times M}$, $\parallel \mathbf{XW} - \mathbf{X}\frac{1}{M}\mathbf{1}\mathbf{1}^T \parallel_F^2 \leq (1-p) \parallel \mathbf{X} - \mathbf{X}\frac{1}{M}\mathbf{1}\mathbf{1}^T \parallel_F^2$.

Here, $\mathbf{X}\frac{1}{M}\mathbf{1}\mathbf{1}^T$ is the global average parameter matrix across all PSes, while $\mathbf{XW}$ is one step of iteration by exchanging parameters between PSes. Assumption 5 guarantees that the gap between $\mathbf{X}$ and $\mathbf{X}\frac{1}{M}\mathbf{1}\mathbf{1}^T$ becomes smaller through one step iteration, and finally $\mathbf{X}$ converges to $\mathbf{X}\frac{1}{M}\mathbf{1}\mathbf{1}^T$.

According to [28], $1-p$ is the spectral radius of the matrix $\mathbf{W} - \frac{1}{M}\mathbf{1}\mathbf{1}^T$. p is a critical parameter that will affect the convergence rate significantly. $\mathbf{W}$ should be determined to maximize p. More details will be discussed in the next section.

1) Non-iid Data Assumption: It is well known that the sample distribution in FL is non-iid. Let f^* and $f_j^*, \forall j \in \mathcal{N}$ denote the minimum values of f and f_j respectively.

Assumption 6: (Non-iid degree) There exists a constant Γ that quantifies the degree of non-iid as follows: $\Gamma = f^* - \sum_{i=1}^{M}\sum_{j\in\mathcal{N}_i}\frac{1}{MN_i}f_j^*$, *where* $\sum_{i=1}^{M}\sum_{j\in\mathcal{N}_i} q_j = 1$.

The non-iid degree Γ is originally proposed in [13] to measure the distance between the minimum value of the global loss function and the average minimum value of local loss functions and has been widely used in existing studies (e.g., [29]–[31]). If the sample distribution is iid, it implies $f^* = f_j^*, \forall j$ and Γ goes to zero as the number of samples grows. Otherwise, $\Gamma > 0$ captures the degree of heterogeneity if the sample distribution is heterogeneous on clients.

C. Convergence Rates

In this section, we present the convergence rate analysis of the P-FedAvg algorithm under both full and partial participation modes. In addition, we will discuss the implications of the convergence rates by comparing them with that of the centralized mode presented in the work [13].

1) Full Participation Mode: For the full participation mode, each server i involves all clients in $\mathcal{N}_i$ as the selected clients in the set $\mathcal{K}_i$ for each round of global iteration. In this scenario, by leveraging the assumptions listed in the last subsection, we can derive the convergence rate of P-FedAvg as below.

Theorem 1: In the full participation mode, if Assumptions 1 to 6 hold and let $\eta_t = \frac{4}{\mu(a+t)}, a = max\{16k, 2E\}, k = \frac{L}{\mu}$, *and the P-FedAvg algorithm will cease after T iterations, then*

$$f(\mathbf{x}_{avg}(T)) - f^* \leq \frac{\mu a^3}{T^3} \parallel \bar{\mathbf{x}}(0) - \mathbf{x}^* \parallel^2 + \frac{8(2a+T)}{\mu T^2}\Big(8E^2G^2 + 6L\Gamma + \frac{64E^2G^2}{p}(\frac{2}{p}+1) + \sum_{j\in\mathcal{N}} q_j^2\frac{\sigma_j^2}{B_j}\Big), \tag{10}$$

where $\mathbf{x}_{avg}(T) = \frac{1}{S_T}\sum_{t=0}^{T} s_t\bar{\mathbf{x}}(t)$, $S_T = \sum_{t=0}^{T} s_t, s_t = (a+t)^2$ *and* $q_j = \frac{1}{MN_i}$ *if* $j \in \mathcal{N}_i$.

In Theorem 1, we prove the convergence of $\mathbf{x}_{avg}(T)$ which is defined as the average of $\bar{\mathbf{x}}(t)$ over T iterations. Meanwhile Assumption 5 guarantees that $\mathbf{x}_i(t)$ will converge to $\bar{\mathbf{x}}(t)$ as long as $\bar{\mathbf{x}}(t)$ can converge. q_j is the weight of client j. Through comparing the convergence rate derived in Theorem 1 and the convergence rate of the centralized scenario derived in the work [13], we can observe that:

- The asymptotic convergence rate of P-FedAvg is $O\left(\frac{1}{T}\left(E^2G^2 + L\Gamma + \frac{E^2G^2}{p^2} + \frac{E^2G^2}{p} + \sum_{j\in\mathcal{N}} q_j^2\frac{\sigma_j^2}{B_j}\right)\right)$. In comparison, the asymptotic convergence rate of the FedAvg algorithm is $O\left(\frac{1}{T}\left(E^2G^2 + L\Gamma + \sum_{j\in\mathcal{N}} q_j^2\frac{\sigma_j^2}{B_j}\right)\right)$ according to [13]. Apparently, the decentralized organization of PSes will bring extra cost to the algorithm.
- For P-FedAvg, p is a crucial parameter that heavily determines the convergence rate. If p can be regarded as a constant, the convergence rate of P-FedAvg is asymptotically the same as that of FedAvg. p is affected by the topology matrix $\mathbf{L}$ and the mixing matrix $\mathbf{W}$. We will further investigate the algorithm to maximize p in Sec. VI.

For a centralized FL, the communication expense for supporting all users is very large. Although P-FedAvg has a slower theoretical convergence rate, it could reduce the large communication expense through decentralized PSes.

2) Partial Participation Mode: In practice, it is more common that only a portion of clients can be engaged in each round of global iterations due to the limited communication resource. Thus, we can further assume that each PS i randomly selects K_i clients with replacement from the set $\mathcal{N}_i$ for a round of global iteration.

Theorem 2: In the partial participation mode, if Assumptions 1 to 6 hold and let $\eta_t = \frac{4}{\mu(a+t)}, a = max\{16k, 2E\}$, $k = \frac{L}{\mu}$ *and the P-FedAvg algorithm will cease after T*

iterations, then

$$f(\mathbf{x}_{avg}(T)) - f^* \leq \frac{\mu a^3}{T^3} \parallel \bar{\mathbf{x}}(0) - \mathbf{x}^* \parallel + \frac{8(2a+T)}{\mu T^2}\Big(8E^2G^2 + 6L\Gamma + \frac{64E^2G^2}{p}(\frac{2}{p}+1) + \sum_{j\in\mathcal{N}} q_j^2 \frac{\sigma_j^2}{B_j} + \sum_{i=1}^{M} \frac{4E^2G^2}{M^2K_i}\Big), \quad (11)$$

where $\mathbf{x}_{avg}(T) = \frac{1}{S_T}\sum_{t=0}^{T} s_t \bar{\mathbf{x}}(t)$, $S_T = \sum_{t=0}^{T} s_t$, $s_t = (a+t)^2$ *and* $q_j = \frac{1}{MN_i}$ *if* $j \in \mathcal{N}_i$.

The convergence rate in Theorem 2 is very similar to that in Theorem 1. Through comparing the convergence rate under partial participation in Theorem 2 with that of the centralized FedAvg in [13], we can observe that:

- For simplicity, let $\Lambda = E^2G^2 + L\Gamma + \sum_{j\in\mathcal{N}} q_j^2 \frac{\sigma_j^2}{B_j}$. The asymptotic convergence rate of P-FedAvg is $O\left(\frac{1}{T}\left(\Lambda + \frac{E^2G^2}{p^2} + \frac{E^2G^2}{p} + \sum_{i=1}^{M}\frac{E^2G^2}{M^2K_i}\right)\right)$. In comparison, the asymptotic convergence rate of the FedAvg algorithm is $O\left(\frac{1}{T}\left(\Lambda + \frac{E^2G^2}{K}\right)\right)$ according to [13].
- By temporarily ignoring the impact of p, the convergence rate of P-FedAvg heavily depends on the value of K_i's. If $\sum_{i=1}^{M} K_i = K$, it is easy to verify that $\sum_{i=1}^{M}\frac{E^2G^2}{M^2K_i} \geq \frac{E^2G^2}{K}$. Thus the centralized algorithm can asymptotically achieve a better convergence rate when selecting the same number of clients.
- However, with M decentralized PSes, it is reasonable to assume that $\sum_{i=1}^{M} K_i \gg K$. By abusing notations a little bit, we assume K is the number of participating clients in each global iteration in FedAvg. Therefore, in practice P-FedAvg can possibly achieve a much better convergence rate through involving more participating clients.

Other than the potential benefit to improve the convergence rate under the partial participation mode, it is more important that P-FedAvg can distribute the communication traffic from a single centralized PS to a number of decentralized PSes. Hence, P-FedAvg is more efficient in communications and robust because the collapse of a single PS will not affect the entire system significantly.

D. Proof Outline

Due to limited space, we can only briefly sketch the proof outline of Theorem 1 and Theorem 2 in our paper. [4]

1) Proof Outline of Theorem 1: Since Assumption 5 has guaranteed that $x_i(t)$ will converge to $\bar{\mathbf{x}}(t)$, we only need to prove that $\bar{\mathbf{x}}(t)$ will converge to $\mathbf{x}^*$ in P-FedAvg. We first show the per step convergence as follows.

Lemma 1: (Per step speed of P-FedAvg) If Assumptions 1- 6 hold, and $\eta_t = \frac{4}{\mu(a+t)}, a = max\{16k, 2E\}, k = \frac{L}{\mu}$, *then*

$$\mathbb{E}_{\mathcal{B}_1(t)\ldots\mathcal{B}_N(t)} \parallel \bar{\mathbf{x}}(t+1) - \mathbf{x}^* \parallel \leq (1-\mu\eta_t) \parallel \bar{\mathbf{x}}(t) - \mathbf{x}^* \parallel^2 + 2\sum_{i=1}^{M}(\frac{1}{M} \parallel \bar{\mathbf{x}}(t) - \mathbf{x}_i(t) \parallel^2 + \sum_{j\in\mathcal{N}_i}\frac{1}{MN_i} \parallel \mathbf{x}_i(t) - \mathbf{x}_j(t) \parallel^2) - \frac{3}{4}\eta_t(f(\bar{\mathbf{x}}(t)) - f^*) + \eta_t^2(6L\Gamma + \sum_{i=1}^{M}\sum_{j\in\mathcal{N}_i}\frac{\sigma_j^2}{M^2N_i^2B_j}) \quad (12)$$

Here $\mathcal{B}_1(t), \ldots, \mathcal{B}_N(t)$ represent sample batches randomly selected by clients for the training at iteration t. It is clear from (12) that we need to further bound the terms $\sum_{i=1}^{M}\frac{1}{M} \parallel \bar{\mathbf{x}}(t) - \bar{\mathbf{x}}_i(t) \parallel^2$ and $\sum_{i=1}^{M}\sum_{j\in\mathcal{N}_i}\frac{1}{MN_i} \parallel \mathbf{x}_j(t) - \mathbf{x}_i(t) \parallel^2$).

Lemma 2: (Bounding the divergence among PSes) If Assumption 3 and 5 hold, and $\eta_t = \frac{4}{\mu(a+t)}, a = max\{16k, 2E\}, k = \frac{L}{\mu}$, *then*

$$\parallel \mathbf{X}(t) - \bar{\mathbf{X}}(t) \parallel_F^2 \leq \frac{32}{p}(1+\frac{2}{p})E^2MG^2\eta_t^2. \quad (13)$$

Lemma 3: (Bounding the divergence between PSes and clients) If Assumption 3 holds and $\eta_t = \frac{4}{\mu(a+t)}, a = max\{16k, 2E\}, k = \frac{L}{\mu}$, *then*

$$\sum_{j\in\mathcal{N}_i}\frac{1}{MN_i} \parallel \mathbf{x}_i(t) - \mathbf{x}_j(t) \parallel^2 \leq \frac{4}{M}E^2\eta_t^2G^2. \quad (14)$$

We can substitute (13) and (14) into (12) to refine the bound in Lemma 1. Then, we can finally prove the convergence rate in Theorem 1 with the refined bound.

Let $e_t = \frac{3}{4}(f(\bar{\mathbf{x}}(t)) - f^*)$ denote the gap between the loss function at iteration t and the minimum loss function. We further define $A = 8E^2G^2(\frac{4+p}{p})^2 + 6L\Gamma + \sum_{i=1}^{M}\sum_{j\in\mathcal{N}_i}\frac{\sigma_j^2}{M^2N_i^2B_j}$ and $r_t = \parallel \bar{\mathbf{x}}(t) - \mathbf{x}^* \parallel^2$. Through (12), (13) and (14), we can prove the following essential inequality: $r_{t+1} \leq (1-\mu\eta_t)r_t - \eta_t e_t + \eta_t^2 A$. Based on this inequality, we could recursively derive that $\frac{1}{S_T}\sum_{t=0}^{T-1} s_t e_t \leq \frac{\mu a^3}{4S_T}r_0 + \frac{2T(2a+T)}{\mu S_T}A$, where S_T and s_t are defined in Theorem 1. Then, through rearranging the equation, we can obtain the convergence rate in Theorem 1.

2) Proof Outline of Theorem 2: To prove the convergence rate in Theorem 2, we use a trick by assuming that all clients will be activated by PSes to update. But in the aggregation step, PSes just aggregate those selected clients. Although it is different from the reality, the evolution of parameters on PSes is identical to that in the original P-FedAvg algorithm.

We divide the measure of the convergence rate into two parts: i) the convergence rate of all clients and ii) the deviation between selected clients and all clients. The proof of the first part is almost the same as the proof of Theorem 1, and thus we focus on bounding the second part. To derive the bound, we first show that the expectation of the average parameters from selected clients is equal to that of all clients for every global iteration.

Lemma 4: (Unbiased sampling scheme) If t *is divisible by* E *and PSes randomly select clients with replacement, then*

$$E_{\mathcal{K}(t)}(\bar{\mathbf{x}}(t)) = \bar{\mathbf{v}}(t), \quad (15)$$

where $\mathcal{K}(t) = \cup_{i=1}^{M}\mathcal{K}_i(t)$ *represents all selected clients.*

Then, we further prove that the parameter variance between selected clients and all clients.

[4]The detailed proof is provided in submitted supplementary files.

TABLE II
THE COMMNICATION TRAFFIC OF P-FEDAVG AND FEDAVG IN A COMMUNICATION ROUND

	FedAvg	**P-FedAvg**
Total	$2Ks$	$2Ks + \|L - I\|_0 s$
Peak	$2Ks$	$\max_{1<i<=M} 2(K_i + \|L_i\|_0 - 1)s$

Lemma 5: (Bounding the variance of $\bar{x}(t)$) If Assumption 3 holds and $\eta_t = \frac{4}{\mu(a+t)}, a = max\{16k, 2E\}, k = \frac{L}{\mu}$, then

$$\| \bar{\mathbf{x}}(t) - \bar{\mathbf{v}}(t) \|^2 \leq \sum_{i=1}^{M} \frac{4E^2G^2\eta_t^2}{M^2K_i}. \tag{16}$$

The proof of Theorem 2 can be completed by expanding $E_{\mathcal{B}_1(t)...\mathcal{B}_n(t)} \| \bar{\mathbf{x}}(t) - \mathbf{x}^* \|^2$ as $E \| \bar{\mathbf{v}}(t) - \mathbf{x}^* \|^2 + E \| \bar{\mathbf{x}}(t) - \bar{\mathbf{v}}(t) \|^2 + 2 E < \bar{\mathbf{x}}(t) - \bar{\mathbf{v}}(t), \bar{\mathbf{v}}(t) - \mathbf{x}^* >$. Here $< \mathbf{x}, \mathbf{x}' >$ is the inner product of vectors $\mathbf{x}$ and $\mathbf{x}'$.

The first term $E \| \bar{\mathbf{v}}(t) - \mathbf{x}^* \|^2$ can be bounded with the similar method used to prove Theorem 1. According to Lemma 4, we have $2E < \bar{\mathbf{x}}(t) - \bar{\mathbf{v}}(t), \bar{\mathbf{v}}(t) - \mathbf{x}^* >= 0$. According to Lemma 5, the term $E \| \bar{\mathbf{x}}(t) - \bar{\mathbf{v}}(t) \|^2$ is bounded by $\sum_{i=1}^{M} \frac{4E^2G^2\eta_t^2}{M^2K_i}$. Finally, by putting these bounds together, we can derive the convergence rate in Theorem 2.

V. COMMUNICATION COST ANALYSIS

The convergence rate analysis presented in the last section can only evaluate the learning performance in terms of the number of iterations. However, the cost to conduct each global iteration is very different between PFL and FL from the communication perspective. In this section, we compare the communication cost in terms of communication traffic and communication time between PFL and FL to demonstrate the advantages of P-FedAvg.

A. Communication Traffic Analysis

Firstly, we compare the total communication traffic of each global iteration. Let s denote the model size. For FL, both the uplink and downlink communication traffic is Ks in each global iteration since the PS needs to exchange model parameters with K participating clients. Hence, the total incurred communication traffic size is $2Ks$ in each global iteration by FL. In contrast, PS i only communicates with K_i participating clients, and thus the incurred communication traffic between PS i and its clients is $2K_is$. Meanwhile, each PS needs to mix its model parameters with neighbors which incurs $2(K_i + \|L_i\|_0 - 1)s$ communication traffic in each global iteration by PS i. Here L_i denotes the neighbor PSes of PS i.

The comparison of communication traffic is presented in Table II. We let $\sum_{i=1}^{M} K_i = K$ to conduct a fair comparison. It is interesting to note that P-FedAvg incurs additional $\|L - I\|_0 s$ communication traffic. However, the peak communication traffic of P-FedAvg is $\max_{1<i<=M} 2(K_i + \|L_i\|_0 - 1)s$, which is much less than that of FedAvg as long as K participating clients are scattered in different PSes. It is not difficult to interpret this result. If clients from a particular PS dominate participating clients in P-FedAvg, the communication of this PS will become the bottleneck of the PFL system such that PFL cannot outperform FL.

B. Communication Time Analysis

To compare PFL and FL in a finer granularity, we take the topology influence into account by comparing the communication time cost.

We employ two classical network models (namely, *edge-capacitated* and *node-capacitated* network models) to analyze the communication time cost of FL and PFL. The difference between them lies in the bottleneck of the communication network. Edge-capacitated and node-capacitated network models put restrictions on the capacity of overlay links and terminal nodes, respectively [32]. Peer-to-Peer (P2P) Network is a representative node-capacitated network. Typically, multicast network is edge-capacitated because data replication is performed on network nodes (e.g., routers).

P-FedAvg runs in a synchronous manner. Each PS halts until receiving model parameters of adjacent PSes before entering the next global iteration. We characterize the communication time cost between PSes as follows. For any two adjacent PSes i and i' in the overlay topology, the *communication time cost* is defined as the time interval between the time point of starting a global iteration for PS i and the time point of receiving parameters from PS i by PS i'. The time interval consists of two parts: 1) *transmission delay*, which is determined by the model size and the communication capacity of links between PSes and clients; 2) *computation delay* accounting for E local iterations in clients. [5]

Let $d_{ii'}$ denote the consumed time for communication between two PSes i and i' in a global iteration. Then,

$$d_{ii'} = \max_{j \in \mathcal{K}_i} \{T_{ij} + EZ_j\} + T_{ii'}, \tag{17}$$

where T_{ij} (or $T_{ii'}$) represents the transmission delay between PS i and the participating client j (or PS i'), and Z_j denotes the computation delay of one local iteration by client j. We assume that the computation time cost per local iteration by client j is a random variable obeying $Z_j \sim F_Z$.

We model T_{ij} and $T_{ii'}$ as random variables to account for the random communication capacity between PSes and clients. Let b_i denote the capacity of PS i to communicate with its clients and b_j denote the uplink capacity of client j, then the time cost of transmitting a model with size s taken by PS i and client j can be defined as $Y_i^1 = \frac{s}{b_i}$ and $Y_j^2 = \frac{s}{b_j}$, respectively. Without loss of generality, we assume that $Y_i^1 \sim F_{Y^1}$ and $Y_j^2 \sim F_{Y^2}$ are random variables. Then, T_{ij} is a random variable which is expressed as

$$T_{ij} = K_iY_i^1 + Y_j^2.$$

Assume that each PS equally allocates the communication capacity among its clients. Let B_i denote the communication capacity between PS i and its neighbor PSes, which is inherently random as well. Similarly, the PS equally allocates the

[5]To simplify our analysis, we ignore the time cost of the simple aggregation operation in PSes.

communication capacity among its neighbor PSes. Then $T_{ii'}$ is defined as

$$T_{ii'} = s \Big/ \min\left\{ \frac{B_i}{\|L_i\|_0 - 1}, \frac{B_{i'}}{\|L_{i'}\|_0 - 1}, A_{ii'} \right\}, \quad (18)$$

which is a random variable assumed to obey $T_{ii'} \sim F_T$. Here s is the model size. $\|L_i\|_0$ is the number of i's neighbours in matrix L and $A_{ii'}$ is the available bandwidth along the link between PS i and i'. For node-capacitated networks, the communication bottleneck lies in $\frac{B_i}{\|L_i\|_0 - 1}$ or $\frac{B_{i'}}{\|L_{i'}\|_0 - 1}$, while for edge-capacitated networks, the communication bottleneck lies in $A_{ii'}$.

Based on $d_{ii'}$, the delay matrix is defined as $\mathbf{D} = (d_{ii'})$. The communication time cost is determined by the largest $d_{ii'}$.

Definition 5.1: Given the overlay network $\mathbf{L}$ *associated with the delay matrix* $\mathbf{D} = (d_{ii'})$*, the time cost of each global iteration is* $\alpha = \max_{\forall (i,i'): L_{ii'}=1} d_{ii'}$.

Since the communication and computation are independently conducted on different clients and PSes, we can naturally assume that Y_i^1's, Y_j^2's, Z_j's and $T_{ii'}$'s are iid (independent and identically distributed) random variables. The expected time cost of each global iteration, *i.e.* α, can be computed as follows.

Theorem 3: Let all the participating clients set $\mathcal{K} = \bigcup_{1 \leqslant i \leqslant M} \mathcal{K}_i$ *and PSes communicate in the overlay network* $\mathbf{L}$*, we have*

$$\begin{aligned} \mathbb{E}[\alpha] \leq & \max_{1 \leqslant i \leqslant M} K_i \mathbb{E}\left[\mathrm{Y}^1_{(M)}\right] + \mathbb{E}\left[\mathrm{Y}^2_{(K)}\right] \\ & + E\mathbb{E}\left[\mathrm{Z}_{(K)}\right] + \mathbb{E}\left[\mathrm{T}_{(\|L-I\|_0)}\right], \end{aligned} \quad (19)$$

where $\mathrm{Y}^1_{(M)} = \max_{1 \leqslant i \leqslant M} Y_i^1$, $\mathrm{Y}^2_{(K)} = \max_{j \in \mathcal{K}} Y_j^2$, $\mathrm{Z}_{(K)} = \max_{j \in \mathcal{K}} Z_j$ *and* $\mathrm{T}_{(\|L-I\|_0)} = \max_{\forall (i,i'): L_{ii'}=1} T_{ii'}$ *denoting the highest order statistic of* M, K, K *and* $\|L - I\|_0$ *iid random variables, respectively [33].*

The proof of Theorem 3 is straightforward. Since max is a convex function, the expectation of α can be bounded by the sum of expectations of T_{ij}, $T_{ii'}$ and EZ_j. T_{ij} measures the time cost to communicate with client j by PS i, and thus $T_{ij} = K_i Y_i^1 + Y_j^2$. The communication time cost of $T_{ii'}$ can also be analyzed under two cases: *edge-capacitated* and *node-capacitated* network models. Computation cost is the product of local iterations E and Z_j.

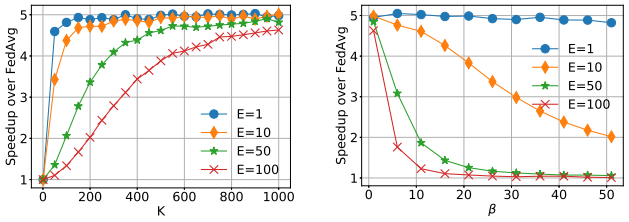
To explicitly demonstrate the benefit brought by P-FedAvg, we define the speed-up metric S_M of P-FedAvg over FedAvg with M PSes as the expected time cost of each global iteration of FedAvg over P-FedAvg.

$$S_M = \frac{K\mathbb{E}\left[\mathrm{Y}^1_{(1)}\right] + \mathbb{E}\left[\mathrm{Y}^2_{(K)}\right] + E\mathbb{E}\left[\mathrm{Z}_{(K)}\right]}{\max\limits_{1 \leqslant i \leqslant M} K_i \mathbb{E}\left[\mathrm{Y}^1_{(M)}\right] + \mathbb{E}\left[\mathrm{Y}^2_{(K)}\right] + E\mathbb{E}\left[\mathrm{Z}_{(K)}\right] + \mathbb{E}\left[\mathrm{T}_{(\|L-I\|_0)}\right]}. \quad (20)$$

Here each PS coordinates K_i clients with $K = \sum_{i=1}^{M} K_i$.

From Theorem 3 and Eq. (20), we have the following insights:

1) From Eq. (20), we can observe that P-FedAvg can significantly speed up FL if communication capacity of the PS is the bottleneck. The advantage of P-FedAvg lies in reducing the term $K\mathbb{E}\left[\mathrm{Y}^1_{(1)}\right]$ in FedAvg to the term $\max\limits_{1 \leqslant i \leqslant M} K_i \mathbb{E}\left[\mathrm{Y}^1_{(M)}\right]$ in P-FedAvg. If $K\mathbb{E}\left[\mathrm{Y}^1_{(1)}\right]$ dominates the time cost of FedAvg, P-FedAvg can speed up FL by a factor of $\frac{K}{\max K_i}$. However, if the bottleneck lies in clients, *e.g.*, the time cost is dominated by the computation cost $E\mathbb{E}\left[\mathrm{Z}_{(K)}\right]$, P-FedAvg cannot really speed up FL.
2) If participating clients are dominated by clients from a particular PS, P-FedAvg cannot really accelerate the convergence since $\max K_i \approx K$ and P-FedAvg consumes additional $\mathbb{E}\left[\mathrm{T}_{(\|L-I\|_0)}\right]$ time to mix parameters with other PSes.
3) It is worth mentioning the difference between edge-capacitated networks and node-capacitated networks. For edge-capacitated networks, adding more links in $\mathbf{L}$ is highly likely to reduce the communication time cost $\mathbb{E}\left[\mathrm{T}_{(\|L-I\|_0)}\right]$. In contrast, for node-capacitated networks, the communication time cost could be inflated if adding more links since both $\frac{B_i}{\|L_i\|_0 - 1}$ and $\frac{B_{i'}}{\|L_{i'}\|_0 - 1}$ in $T_{ii'}$ can be reduced with denser connections. This result indicates the complication to optimize PFL in practice. From Theorems 1 and 2, the convergence rate is higher if the connection is denser because p is larger. Whereas, the convergence rate analysis in Theorems 1 and 2 ignores the communication cost, which however increases with more links in the overlay network $\mathbf{L}$.



(a) Variable participating clients (b) 1,000 participating clients

Fig. 2. The speed-up of P-FedAvg by distributing K participating clients among $M = 5$ PSes with different local iterations E on clients. β refers to the ratio of computation to communication time cost.

C. Runtime Analysis

According to our analysis, P-FedAvg can significantly outperform FedAvg when communication capacity is the system bottleneck. To further illustrate this point, we investigate how the speed-up metric S_M defined in Eq. (20) is affected by system parameters. We numerically plot the change of S_M (computed according to Eq. (20)) in two scenarios in Fig. 2. Here, we set Y_i^1, Y_j^2, $T_{ii'}$, $Z_j \sim Exponential(1)$ with mean value equal to 1. To observe the influence of computation capacity relative to communication capacity, we define $\beta = \mathbb{E}[Z_j] / \mathbb{E}\left[Y_i^1\right]$. We set $M = 5$ PSes, and each of them coordinates $\frac{K}{M}$ clients where K ranges from 50 to 1,000.

In Fig. 2(a), the value of S_M becomes higher as the number of clients in the system increases. The reason is that the

communication traffic becomes heavier with the increase of client population. In Fig. 2(b), we tune the computation capacity such that the ratio β of computation to communication time cost ranges from 1 to 50. S_M gradually diminishes with the increase of β. The reason is that the computation time cost becomes the bottleneck of the system with the increase of β, while P-FedAvg can effectively speed up FL when communication is the system bottleneck. It is worth noting that S_M is also affected by the parameter E. If E is smaller, it implies that clients need to communicate more frequently with the PS, and thus P-FedAvg can better speed up FL. But no matter what the value of E is, we can always find cases in which P-FedAvg can significantly speed up FL.

VI. Impacts of PS Overlay Topology and Mixing Matrix

According to the results derived in the previous section, p can heavily affect the convergence rates for both full and partial participation modes. In this section, we proceed to discuss the algorithm to maximize p given a topology matrix $\mathbf{L}$. Then, we conduct a case study with five specific scenarios to illustrate how p is determined by $\mathbf{L}$.

A. Optimization of Mixing Matrix

Recall that $\mathbf{L}$ describes the overlay topology formed by PSes with $Lii' = 1$ if PS i is connected with PS i' while $\mathbf{W}$ is the mixing matrix which is constructed based on $\mathbf{L}$. In other words, we can assign $w_{ii'} > 0$ only if $L_{ii'=1}$. Otherwise, $w_{ii'}$ must be 0. On the other hand, $\mathbf{W}$ is a set of important parameters in the P-FedAvg algorithm because $\mathbf{W}$ determines the value of p.

In fact, maximizing p for a general case is a non-convex optimization problem. Fortunately, we study an undirected network topology, which implies that $\mathbf{L}$ is a symmetric matrix. Therefore, we let $\mathbf{W}$ be symmetric matrix as well to simplify our analysis. At the same time, $\mathbf{W}$ must be a stochastic matrix to make sure that PSes can reach a consensus after a certain number of iterations.

Theorem 4: If $\mathbf{W}$ *is a symmetric stochastic matrix,* i.e., $\mathbf{W}^T = \mathbf{W}$, $\mathbf{1}^T\mathbf{W} = \mathbf{1}^T$ *and* $\mathbf{W1} = \mathbf{1}$, *we can let* $1 - p = \| \mathbf{W} - \frac{\mathbf{1}\mathbf{1}^T}{M} \|_2^2$ *such that Assumption 5 holds.*

Proof: We prove this theorem briefly by assuming that $\mathbf{X}$ is an arbitrary $M \times M$ matrix to further gain the insights to construct efficient networks. Then

$$\begin{aligned}
\| \mathbf{XW} - \mathbf{X}\frac{1}{M}\mathbf{1}\mathbf{1}^T \|_F^2 &= \| (\mathbf{X} - \mathbf{X}\frac{1}{M}\mathbf{1}\mathbf{1}^T)\mathbf{W} \|_F^2, \\
&= \| (\mathbf{X} - \mathbf{X}\frac{1}{M}\mathbf{1}\mathbf{1}^T)(\mathbf{W} - \frac{1}{M}\mathbf{1}\mathbf{1}^T) \|_F^2, \\
&\leq \| \mathbf{X} - \mathbf{X}\frac{1}{M}\mathbf{1}\mathbf{1}^T \|_F^2 \| \mathbf{W} - \frac{1}{M}\mathbf{1}\mathbf{1}^T \|_2^2, \\
&= (1-p) \| \mathbf{X} - \mathbf{X}\frac{1}{M}\mathbf{1}\mathbf{1}^T \|_F^2 .
\end{aligned}$$

□

From Theorem 4, we can conclude that maximizing p is equivalent to minimizing $\| \mathbf{W} - \frac{\mathbf{1}\mathbf{1}^T}{M} \|_2$. The optimization problem can be formally defined as follows:

$$\begin{aligned}
\min_{\mathbf{W}} \ & \| \mathbf{W} - \frac{\mathbf{1}\mathbf{1}^T}{M} \|_2, \\
\text{s.t.} \quad & \mathbf{W} = \mathbf{W}^T, \\
& \mathbf{W1} = \mathbf{1}, \\
& w_{ii'} \leq L_{ii'}, \forall i, i' \in [1 \dots M].
\end{aligned}$$

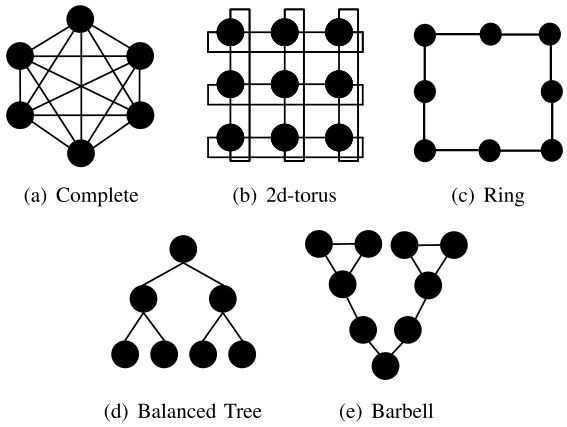


Fig. 3. The figure to show the topologies of complete, ring, 2d-torus, balanced tree and barbell.

Note that $\| \mathbf{W} - \frac{\mathbf{1}\mathbf{1}^T}{M} \|_2 \leq \lambda$ iff $(\mathbf{W} - \frac{\mathbf{1}\mathbf{1}^T}{M})^T(\mathbf{W} - \frac{\mathbf{1}\mathbf{1}^T}{M}) \preceq \lambda^2 I, \forall \lambda \geq 0$. Thus, the above optimization problem could be further converted to a semi-definite programming (SDP) problem which is formally defined as follows:

$$\begin{aligned}
\min \ & \lambda, \\
\text{s.t.} \quad & \mathbf{W} = \mathbf{W}^T, \\
& -\lambda I \preceq (\mathbf{W} - \frac{\mathbf{1}\mathbf{1}^T}{M}) \preceq \lambda I, \\
& \mathbf{W1} = \mathbf{1}, \\
& w_{ii'} \leq L_{ii'}, \forall i, i' \in [1 \dots M]. \qquad (21)
\end{aligned}$$

The SDP problem is similar to the Markov chain fastest mixing problem [34], [35] and gossip matrix fastest convergence problem [28], [36], which can be solved by the interior point method [37]. It is worth mentioning that a similar result was derived in [38], but the proposed approach based on Laplacian matrix is based on the assumption that the weights contributed by one's neighbours are equal, which is not required in our analysis.

B. Case Study

To answer how to construct efficient overlay networks to connect PSes, we study five specific cases.

Figures 3(a)-3(e) show five topologies formed by PSes. Each dot represents an PS and each edge indicates that two connected PSes can communicate with each other. The five cases are very popular and have been extensively investigated by previous works. Therefore, we use them for our case study.

- *Complete*: Complete topology is a special case of mesh topology, which provides redundant data paths between nodes and is often used in large backbone networks to ensure reliability. In Fig. 3(a), nodes are fully connected.
- *2d-Torus*: Torus topology is also a commonly used topology in parallel computer systems [39]. Nodes are arranged in a high-dimensional linear array. As shown in

TABLE III
THE COMPARISON UNDER DIFFERENT PS OVERLAY TOPOLOGIES WITH M PSES AND K PARTICIPATING CLIENTS

Topology	p	Peak Traffic	Total Traffic	Communication Time	Disconnection Tolerance
Complete	1	$2(\frac{K}{M}+M-1)s$	$2\left(K+\frac{M(M-1)}{2}\right)s$	$\frac{2Ks}{Mb}+\frac{(M-1)s}{B}$	$M-1$
2d-torus	$O\left(\frac{1}{M}\right)$	$2(\frac{K}{M}+4)s$	$2(K+2M)s$	$\frac{2Ks}{Mb}+\frac{4s}{B}$	4
Balanced tree	$O\left(\frac{1}{M}\right)$	$\geq 2(\frac{K}{M}+3)s$	$2(K+M-1)s$	$\geq \frac{2Ks}{Mb}+\frac{3s}{B}$	1
Ring	$O\left(\frac{1}{M^2}\right)$	$2(\frac{K}{M}+2)s$	$2(K+M)s$	$\frac{2Ks}{Mb}+\frac{2s}{B}$	2
Barbell	$O\left(\frac{1}{M^3}\right)$	$2(\frac{K}{M}+m_1)s$	$2\left(K+m_1(m_1-1)+m_2+1\right)s$	$\frac{2Ks}{Mb}+\frac{m_1 s}{B}$	1

Fig. 3(b), PSes are connected to their nearest neighbors in each axis, and the corresponding PSes on the opposite side of the array are also connected.

- *Ring*: In the ring topology, each PS is connected with two other PSes to form a ring as shown in Fig. 3(c).
- *Balanced tree*: In the balance tree topology, PSes are arranged in a hierarchical manner, as shown in Fig. 3(d).
- *Barbell*: Barbell topology is commonly used to simulate congestion control [40] and it consists of two cliques (each with m_1 nodes) connected by a path (m_2 nodes). Fig. 3(e) shows a barbell topology with m_1=3 and m_2=3.

Recall that the number of neighbor PSes of a particular PS i is $\|L_i\|_0 - 1$. For the case study, we simply assign $w_{ii'} = \frac{1}{\max\{\|L_i\|_0-1,\|L_{i'}\|_0-1\})+1}$ if $L_{ii'} =$ and $i \neq i'$, and $w_{ii} = 1-\sum_{i'\neq i} w_{ii'}$. This assignment of $w_{ii'}$ can ensure that $\mathbf{W}$ satisfies Assumption 5 and it is easy to compute the value p for each case [41]. The computed p is listed in Table III.

To examine the pros and cons of each topology, we compare the performance of P-FedAvg in these topologies in Table III, which are also explained as follows.

1) Comparison of Convergence Rates: According to Theorems 1 and 2, the convergence rate of the P-FedAvg algorithm has two terms absent from that of FedAvg, which are $O\left(\frac{E^2G^2}{p^2}\right)$ and $O\left(\sum_{i=1}^{M}\frac{E^2G^2}{M^2K_i}\right)$. To ease our comparison, we name the former one as the *Convergence to Consensus (C2C)* and the latter one as the *Parallel Engagement (PE)* hereafter.

- *Complete*: Complete topology has the fastest convergence rate because $p = 1$. For this case, the C2C term $O\left(\frac{1}{T}E^2G^2\right)$ is irrelevant to the number of PSes M.
- *2d-Torus*: There is a tradeoff between the C2C term and the PE term. The C2C term $O\left(\frac{1}{T}M^2E^2G^2\right)$ increases with the square of the number of PSes M while the PE term $O\left(\sum_{i=1}^{M}\frac{E^2G^2}{M^2K_i}\right)$ decreases because more PSes can engage more clients into each round of iteration.
- *Ring*: It is obvious that the C2C term of the ring topology is $O\left(\frac{1}{T}M^4E^2G^2\right)$, and its convergence rate is low for a PFL system with a large number of PSes.
- *Balanced tree*: For the balanced tree topology we can also observe that the C2C term $O\left(\frac{1}{T}M^2E^2G^2\right)$ increases with the number of PSes, though its growth rate with M is much lower than that of ring.
- *Barbell*: It is notable that the C2C term of barbell is $O\left(\frac{1}{T}M^9E^2G^2\right)$ indicating that its convergence rate will be low with a large number of PSes.

2) Comparison of Communication Traffic: The communication traffic includes the communication traffic between PSes, and between each PS and its clients. Supposing that N clients are equally distributed across M PSes, the peak and total communication traffic of different topologies in P-FedAvg are listed in Table III, from which we can observe:

- The communication traffic of the complete topology is the heaviest. Its peak communication traffic and the total communication traffic are proportional to M and M^2, respectively. This result manifests that complete topology may not be the best one in practice due to its heavy communication traffic.
- Ring, 2d-torus and balanced tree have lighter communication traffic because their sparser connections between PSes. The peak communication traffic is only proportional to $O(\frac{K}{M})$ indicating that they can effectively improve the training efficiency if being deployed properly.
- For the barbell topology, its communication traffic depends on the value of m_1 where $2m_1 + m_2 = M$. Since the peak communication traffic is proportional to $O(\frac{K}{M}+m_1)$, its communication traffic is comparable to that of 2d-torus, balanced tree and ring if m_1 is small. Otherwise, it is similar to complete topology.

3) Comparison of Communication Time: We further compare the communication time cost of each topology by leveraging Theorem 3 and using a node-capacitated network model. The comparison using edge-capacitated network model can be conducted in a similar way.

We assume that clients and PSes are homogeneous in the PFL system. The communication capacity of each client (or PS) is b (or B) to simplify the comparison of the communication time cost.

From this comparison, we can find that the communication time cost of the complete topology is the highest because of its heavy communication traffic among PSes. The communication time cost of each global iteration for ring, 2d-torus and balanced tree is much lower. Again, the communication time cost of the barbell topology depends on m_1. If m_1 approaches M, it is similiar to complete topology with heavy communication time cost, otherwise if m_1 approaches 0, its communication time cost becomes low.

4) Comparison of Robustness: In reality, it is possible that a PS may crash down and cannot provide aggregation service any more or the links between a few PSes are disconnected. A system is robust if it can still work when some PSes leave the system. We define the *disconnection tolerance* as the number of links that can be removed without making the network disconnected. The disconnection tolerance of each topology is listed in Table III.

From the comparison of disconnection tolerance, we can observe that complete topology is the most robust one, in which PSes are still connected if a large portion of links are removed from the system. 2d-torus is the second-best one for tolerating the removal of links. PSes in 2d-torus are still connected if at most 4 links are removed from the system. Thus, 2d-torus is better than other topologies, which is still connected if more than 2 links are removed.

In summary, the PS overlay topology design for a PFL system should jointly consider *convergence rate, communication traffic, communication time* and *robustness*. System designers should consider how to balance these factors. Based on our case study, 2d-torus is the best topology for implementing the PFL system achieving the best overall performance. Dislike complete, 2d-torus will not incur significant communication traffic between PSes. Therefore, its communication traffic and communication time cost are comparable to other topologies. Meanwhile, its convergence rate in terms of the number of iterations and its robustness are better than other topologies.

VII. Experiments

In this section, we conduct experiments to verify our theoretical analysis.

A. Experimental Settings

1) Datasets: We use three datasets for our experiments.

- MNIST [42] dataset, which contains 69,035 handwritten digital images of 0-9, and the data is divided among 1000 clients. In order to simulate a non-iid setting, each client can only own samples of only 2 digits.
- FEMNIST dataset is the federated version [43] of EMNIST [9], which consists of 805,263 samples of digits and English characters (62 classes in total). The dataset is split among 3,550 unbalanced clients based on character writers. The different writing styles across clients guarantee the non-iid distribution of samples.
- CIFAR10 dataset [10], which consists of 60,000 color images in 10 classes. We distribute the data among 200 clients in an non-iid manner where each client owns images with only 2 classes.

2) Models: We implement a multi-layer perceptron (MLP) to perform classification tasks on the MNIST. The MLP model contains 1 hidden layer with 512 units. CNN models are implemented to classify images in FEMNIST and CIFAR10. The CNN model contains two layers followed by a pooling layer and a final dense layer. The models we adopt are the same as the one used in [43] and [42] for FEMNIST and CIFAR10, respectively. The details of model parameters are listed in Table IV.

We tune the algorithm hyperparameters through grid search on different datasets. And we set the batch size for local iterations as 32 when comparing with baseline algorithms. The number of local iterations E is set as 20 for the models. The learning rate is set as a decreasing function with the number of iterations being $\eta_t = \frac{\eta_0}{1+t/10}$.

TABLE IV

Model Details. The Average Computation Time is Obtained on NVIDIA Tesla V100 for 100 Iterations

Model	Dataset	Parameters ($\times 10^3$)	Model Size (Mbits)	Computation Time (ms)
MLP	MNIST	406	13	2.38
CNN1	FEMNIST	6,603	211	4.99
CNN2	CIFAR10	2,156	69	23.25

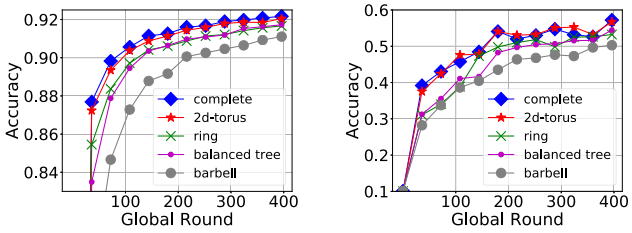
3) Baseline Algorithms: We compare our algorithm with the following baseline algorithms:

- FedAvg, which in fact is a special case of our P-FedAvg when deploying a single PS in PFL [5]. In our experiments, we set $E = 20$ and $K = 36$ in FedAvg.
- HierFAVG [44], which is a three-level FL training algorithm. The root is a single PS and the leaves are clients. The middle layer is constructed by a number of intermediate servers. In each round, intermediate servers activate K clients to perform E_1 local iterations before a partial aggregation. The root node performs a global aggregation after every E_2 partial aggregation. In our experiments, to ensure the same number of local iterations and activated clients, we use 9 intermediate servers and set $E_1 = 10$, $E_2 = 2$ and $K = 9$.
- DPSGD [7], which takes all clients as PSes. Clients can directly communicate with their neighbors to exchange parameters. In each round, each client performs 20 local iterations, and then exchanges model parameters with neighbor clients.

4) Clients and PSes: In our experiments, we implement five topologies. To conduct a fair comparison, we deploy 9 PSes for PFL and each PS activates 4 clients for each global iteration. Moreover, we consider heterogeneous computation resources among clients, dynamic network connections between PS and clients and different PS overlay network models. The settings are summarized as follows:

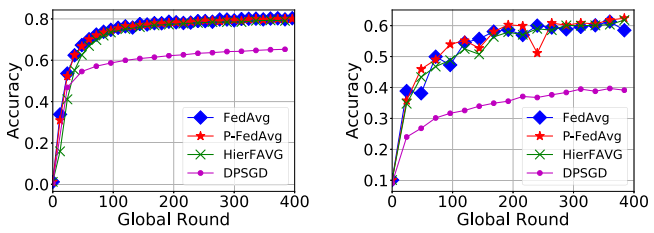
- *Heterogeneous Clients*: we use GPU benchmarks data [6] to measure the relative difference of clients' computation performance. The reference computation time is obtained by training models on NVIDIA Tesla V100 and for each client the time is randomly expanded or reduced.
- *Network Connections between PS and Clients*: We use the 4G network traces in [45] to simulate network bandwidth between PS and its clients. To communicate with multiple clients simultaneously, orthogonal frequency-division multiple access (OFDMA) is adopted and each OFDM subcarrier is 15kHz.
- *Network Connections between PSes*: We generate the overlay topology of PSes based on the method used in [46] for simulating edge-capacitated (Géant) and node-capacitated (Exodus) networks, respectively. In the edge-capacitated network, the upload and download communication capacity between each PS is 1Gbps while the average communication capacity between PSes is 100Mbps. In contrast, for the node-capacitated network, the capacity for upload and download between PSes is

[6]https://lambdalabs.com/gpu-benchmarks



(a) MNIST (b) CIFAR10

Fig. 4. The comparison of model accuracy over the number of conducted global iterations for P-FedAvg under different topologies.



(a) FEMNIST (b) CIFAR10

Fig. 5. The comparison of convergence rate versus global iterations of different algorithms.

100Mbps. The communication capacity between PSes is 820Mbps. Our settings are consistent with the measurement results reported in [47].

B. Experimental Results

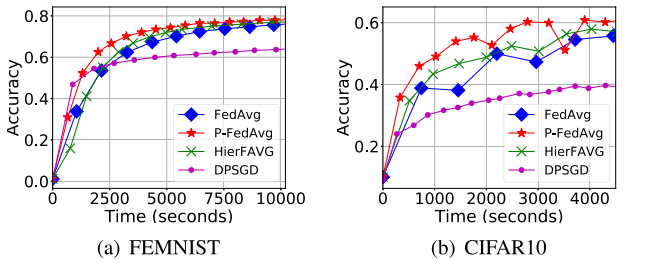
1) Evaluating Convergence Rates: In Theorems 1 and 2, we have proved that p can significantly influence the convergence rate of P-FedAvg. To verify this point, we implement the P-FedAvg algorithm in five topologies to evaluate its convergence rate in terms of the number of conducted iterations. According to Assumption 5, the values of p are 1.0, 0.84, 0.29, 0.12 and 0.08 for the complete, 2d-torus, ring, balanced tree and barbell topologies, respectively.

We conduct experiments on the MNIST and CIFAR10 datasets and the results are presented in Fig. 4. From the experiment results, we can observe that the convergence rate is different with the overlay network topology indicating that p can influence the final model accuracy of FL. It is worth noting that both complete topology and 2d-torus can achieve faster convergence than other topologies. Barbell is the slowest one and its model accuracy is much worse than that of other topologies.

We compare the convergence rate of P-FedAvg with other baselines in terms of the number of conducted iterations in Fig. 5. We use the FEMNIST and CIFAR10 datasets for this experiment, and adopt the 2d-torus and ring topology for P-FedAvg and DPSGD, respectively. As we can see from Fig. 5, the convergence rate of P-FedAvg is very close to that of FedAvg. DPSGD is the worst one due to the following reasons: 1) The data distributed on clients is non-iid and highly divergent; 2) Each client only exchanges parameters with a few number of adjacent clients (e.g., 2 in the ring topology) with no aggregation in the PS. Thus its convergence rate is very slow.

If algorithms are merely evaluated in terms of the number of iterations, FedAvg is the best one because P-FedAvg and other parallel algorithms consume additional cost to mix parameters on different PSes. However, P-FedAvg is only slightly influenced since the gap between P-FedAvg and FedAvg is very small when 2d-torus is adopted.

2) Evaluating Performance Over Time: In practice, it is more reasonable to compare the performance of different algorithms over the training time because time cost of each global iteration can be various for different algorithms. Therefore, we further compare P-FedAvg with other baselines in Fig. 6. Again, we use the FEMNIST and CIFAR10 datasets for this experiment, and adopt 2d-torus and ring topology for P-FedAvg and DPSGD, respectively. In Fig. 6, x-axis represents the consumed training time while y-axis represents the model accuracy on the test dataset.



(a) FEMNIST (b) CIFAR10

Fig. 6. The comparison of model accuracy over training time for different algorithms on FEMNIST and CIFAR10 under the node-capacitated network.

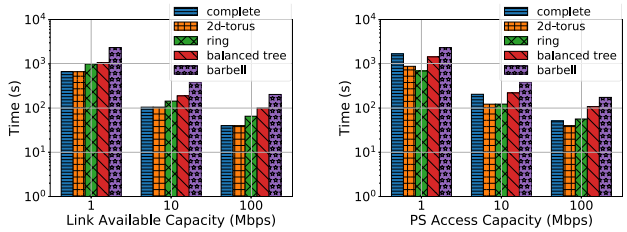
TABLE V
THE COMPARISON OF TIME COST (MILLISECOND) OF DIFFERENT ALGORITHMS IN A GLOBAL ITERATION

	FedAvg	HierFAVG		DPSGD	P-FedAvg	
		Géant	Exodus		Géant	Exodus
MNIST	6,190	3,672	4,657	2,137	3,023	3,146
FEMNIST	90,940	40,559	57,293	37,039	39,347	54,160
CIFAR10	31,144	16,034	21,206	11,992	14,372	14,655

In Fig. 6, it is observed that P-FedAvg achieves the highest model accuracy and the fastest convergence. The reason is that P-FedAvg can effectively reduce the time cost of each global iteration by distributing the communication traffic from a centralized PS to multiple parallel PSes.

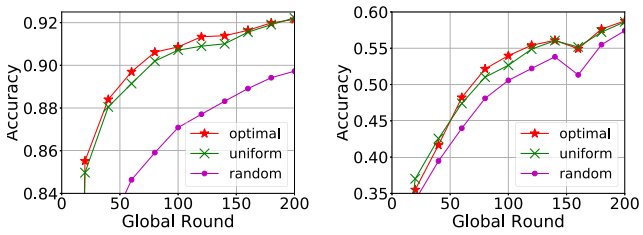
3) Evaluating Time Cost per Global Iteration: To further demonstrate the advantage of P-FedAvg, we compare the time cost of a global iteration for different algorithms in Table V. For P-FedAvg and HierFAVG, we implement two kinds of network models, namely edge-capacitated network (Géant) and node-capacitated network (Exodus).

As we can see from Table V, the time cost of DPSGD is the lowest. However, without a PS aggregating parameters, the model accuracy of DPSGD is the worst. The time cost of P-FedAvg is much lower than that of other algorithms for both datasets under both kinds of networks because of its parallel communication mode. Although its time cost is higher than that of DPSGD, its convergence rate has been guaranteed by



(a) Edge-Capacitated Model (b) Node-Capacitated Model

Fig. 7. Training time for reaching the expected test accuracy (80%) of P-FedAvg on the MNIST dataset under different network models.



(a) MNIST (b) CIFAR10

Fig. 8. Comparison of convergence rates with different mixing matrices.

our Theorems 1 and 2. and thus P-FedAvg can finally achieve the highest model accuracy.

4) Evaluating Topologies: To evaluate the performance of each topology, we conduct experiments to compare the consumed time to reach the target model accuracy (80%) on the MNIST dataset in order to explore which topology should be adopted in practice. In Fig. 7(a), we vary the average link capacity, i.e., $A(i, i')$, from 1Mbps to 100Mbps; while in Fig. 7(b), the communication capacity of each PS is varied from 1Mbps to 100Mbps.

From the results presented in Fig. 7(a) and Fig. 7(b), we can observe that 2d-torus is almost the best in all cases that can reach the target model accuracy with a low time cost, which is consistent with our insights of the case study in Sec. VI. It is worth mentioning that the complete topology performs very well in the edge-capacitated network because its p value is 1 and more links can always lower the communication time cost in the edge-capacitated network as we have analyzed in Theorem 3. However, the complete topology cannot be used in node-capacitated networks due to too heavy communication traffic. In contrast, barbell is always the worst one in all cases. The p value of barbell is the lowest. The result implies that the extremely low value of p can always severely compromise the final model accuracy.

5) Evaluating Mixing Weights: We empirically study the influence of the mixing matrix by using a randomly generated topology according to the previous work [48]. We set the sampling probability to 0.5 for each edge, and the average degree of the generated matrix is 4. We compare three mixing matrices: *optimal*, *uniform* and *random*. The optimal weights are determined by (21). For uniform (or random), each PS evenly (or randomly) allocates mixing weights among its neighbors. Given the three mixing matrices, we can compute their p values, which are 0.4036 for optimal, 0.2418 for uniform and 0.0288 for random respectively. According to our analysis, the convergence is faster if the p value is larger. Therefore, the optimal one should be the best one achieving the highest convergence rate.

We conduct experiments in Fig. 8 using the MNIST and CIFAR10 datasets with batch size of 32. The results indeed verify our analysis that the optimal one always achieves the highest convergence rates on different datasets. In contrast, the convergence rate of random is the worst one. Uniform is a good sub-optimal choice, and its convergence performance is rather close to that of optimal.

VIII. Conclusion

To overcome the bottleneck of the centralized PS in traditional FL, we proposed the PFL framework by deploying a number of decentralized PSes with a newly designed P-FedAvg algorithm. We proved the convergence rate of P-FedAvg with non-iid samples under both full and partial client participation modes. We also analyzed the communication traffic and the communication time cost in theory between FL and PFL. The influence of overlay topology formed by PSes on model accuracy was explored, and the optimal mixing weights were determined through solving a convex optimization problem to achieve the highest convergence rate. Through a case study, we illustrated that PFL can significantly accelerate the convergence rate, especially when the system scales in terms of client population is large. In the end, we conducted extensive experiments on MNIST, FEMNIST and CIFAR10 datasets to verify our analysis. In the future, we will study the influence of dynamic network topology on the convergence rate by theoretical analysis and experiments.

References

[1] Z. Zhong *et al.*, "P-FedAvg: Parallelizing federated learning with theoretical guarantees," in *Proc. IEEE Conf. Comput. Commun. (INFOCOM)*, Vancouver, BC, Canada, May 2021, pp. 1–10, doi: 10.1109/INFOCOM42981.2021.9488877.

[2] E. Bagdasaryan, A. Veit, Y. Hua, D. Estrin, and V. Shmatikov, "How to backdoor federated learning," in *Proc. 23rd Int. Conf. Artif. Intell. Statist. (AISTATS)*, Sicily, Italy, Aug. 2020, pp. 2938–2948.

[3] R. C. Geyer, T. Klein, and M. Nabi, "Differentially private federated learning: A client level perspective," 2017, *arXiv:1712.07557*.

[4] A. Bhowmick, J. Duchi, J. Freudiger, G. Kapoor, and R. Rogers, "Protection against reconstruction and its applications in private federated learning," 2018, *arXiv:1812.00984*.

[5] B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. Y. Arcas, "Communication-efficient learning of deep networks from decentralized data," in *Proc. 20th Int. Conf. Artif. Intell. Statist. (AISTATS)*, Fort Lauderdale, FL, USA, Apr. 2017, pp. 1273–1282.

[6] P. Vanhaesebrouck, A. Bellet, and M. Tommasi, "Decentralized collaborative learning of personalized models over networks," in *Proc. 20th Int. Conf. Artif. Intell. Statist. (AISTATS)*, Fort Lauderdale, FL, USA, Apr. 2017, pp. 509–517.

[7] X. Lian, C. Zhang, H. Zhang, C. Hsieh, W. Zhang, and J. Liu, "Can decentralized algorithms outperform centralized algorithms? A case study for decentralized parallel stochastic gradient descent," in *Proc. Adv. Neural Inf. Process. Syst., Annu. Conf. Neural Inf. Process. Syst.*, Long Beach, CA, USA, Dec. 2017, pp. 5330–5340.

[8] T. Nishio and R. Yonetani, "Client selection for federated learning with heterogeneous resources in mobile edge," in *Proc. ICC - IEEE Int. Conf. Commun. (ICC)*, Shanghai, China, May 2019, pp. 1–7.

[9] G. Cohen, S. Afshar, J. Tapson, and A. van Schaik, "EMNIST: Extending MNIST to handwritten letters," in *Proc. Int. Joint Conf. Neural Netw. (IJCNN)*, May 2017, pp. 2921–2926.

[10] A. Krizhevsky and G. Hinton, "Learning multiple layers of features from tiny images," Univ. Toronto, Toronto, Ontario, Tech. Rep. TR-2009, 2009.
[11] S. U. Stich, "Local SGD converges fast and communicates little," in *Proc. 7th Int. Conf. Learn. Represent. (ICLR)*, New Orleans, LA, USA, May 2019, pp. 1–17.
[12] H. Yu, S. Yang, and S. Zhu, "Parallel restarted SGD with faster convergence and less communication: Demystifying why model averaging works for deep learning," 2018, *arXiv:1807.06629*.
[13] X. Li, K. Huang, W. Yang, S. Wang, and Z. Zhang, "On the convergence of FedAvg on non-IID data," in *Proc. 8th Int. Conf. Learn. Represent. (ICLR)*, Addis Ababa, Ethiopia, Apr. 2020, pp. 1–26.
[14] T. Li, A. K. Sahu, M. Zaheer, M. Sanjabi, A. Talwalkar, and V. Smith, "Federated optimization in heterogeneous networks," in *Proc. Mach. Learn. Syst. (MLSys)*, Austin, TX, USA, Mar. 2020, pp. 429–450.
[15] A. Khaled, K. Mishchenko, and P. Richtárik, "First analysis of local GD on heterogeneous data," 2019, *arXiv:1909.04715*.
[16] Y. Huang *et al.*, "Personalized cross-silo federated learning on non-iid data," in *Proc. 35th AAAI Conf. Artif. Intell., AAAI, 33rd Conf. Innov. Appl. Artif. Intell., IAAI, 11th Symp. Educ. Adv. Artif. Intell. (EAAI)*, Feb. 2021, pp. 7865–7873. [Online]. Available: https://ojs.aaai.org/index.php/AAAI/article/view/16960
[17] Y. Sun, S. Zhou, and D. Gündüz, "Energy-aware analog aggregation for federated learning with redundant data," in *Proc. IEEE Int. Conf. Commun. (ICC)*, Dublin, Ireland, Jun. 2020, pp. 1–7, doi: 10.1109/ICC40277.2020.9148853.
[18] H. Wang, Z. Kaplan, D. Niu, and B. Li, "Optimizing federated learning on non-IID data with reinforcement learning," in *Proc. IEEE Conf. Comput. Commun. (INFOCOM)*, Toronto, ON, Canada, Jul. 2020, pp. 1698–1707.
[19] C. T. Dinh *et al.*, "Federated learning over wireless networks: Convergence analysis and resource allocation," *IEEE/ACM Trans. Netw.*, vol. 29, no. 1, pp. 398–409, Feb. 2021, doi: 10.1109/TNET.2020.3035770.
[20] X. Lian, W. Zhang, C. Zhang, and J. Liu, "Asynchronous decentralized parallel stochastic gradient descent," in *Proc. 35th Int. Conf. Mach. Learn. (ICML)*, Stockholm, Sweden, Jul. 2018, pp. 3049–3058.
[21] J. Wang and G. Joshi, "Cooperative SGD: A unified framework for the design and analysis of communication-efficient SGD algorithms," 2018, *arXiv:1808.07576*.
[22] J. Wang, A. Kumar Sahu, Z. Yang, G. Joshi, and S. Kar, "MATCHA: Speeding up decentralized SGD via matching decomposition sampling," 2019, *arXiv:1905.09435*.
[23] A. Koloskova, N. Loizou, S. Boreiri, M. Jaggi, and S. U. Stich, "A unified theory of decentralized SGD with changing topology and local updates," 2020, *arXiv:2003.10422*.
[24] S. Li, Q. Qi, J. Wang, H. Sun, Y. Li, and F. R. Yu, "GGS: General gradient sparsification for federated learning in edge computing*," in *Proc. IEEE Int. Conf. Commun. (ICC)*, Dublin, Ireland, Jun. 2020, pp. 1–7.
[25] A. Reisizadeh, A. Mokhtari, H. Hassani, A. Jadbabaie, and R. Pedarsani, "Fedpaq: A communication-efficient federated learning method with periodic averaging and quantization," in *Proc. Int. Conf. Artif. Intell. Statist.*, 2020, pp. 2021–2031.
[26] L. Liu, J. Zhang, S. H. Song, and K. B. Letaief, "Client-edge-cloud hierarchical federated learning," in *Proc. IEEE Int. Conf. Commun. (ICC)*, Dublin, Ireland, Jun. 2020, pp. 1–6.
[27] S. Wang, Y. Ruan, Y. Tu, S. Wagle, C. G. Brinton, and C. Joe-Wong, "Network-aware optimization of distributed learning for fog computing," *IEEE/ACM Trans. Netw.*, vol. 29, no. 5, pp. 2019–2032, Oct. 2021.
[28] L. Xiao and S. Boyd, "Fast linear iterations for distributed averaging," *Syst. Control Lett.*, vol. 53, no. 1, pp. 65–78, 2004.
[29] M. Salehi and E. Hossain, "Federated learning in unreliable and resource-constrained cellular wireless networks," *IEEE Trans. Commun.*, vol. 69, no. 8, pp. 5136–5151, Aug. 2021.
[30] L. Cui, X. Su, Y. Zhou, and Y. Pan, "Slashing communication traffic in federated learning by transmitting clustered model updates," *IEEE J. Sel. Areas Commun.*, vol. 39, no. 8, pp. 2572–2589, Aug. 2021.
[31] M. M. Amiri, T. M. Duman, D. Gündüz, S. R. Kulkarni, and H. V. P. Poor, "Blind federated edge learning," *IEEE Trans. Wireless Commun.*, vol. 20, no. 8, pp. 5129–5143, Aug. 2021.
[32] S. Zhang, M. Chen, Z. Li, and L. Huang, "Optimal distributed broadcasting with per-neighbor queues in acyclic overlay networks with arbitrary underlay capacity constraints," in *Proc. IEEE Int. Symp. Inf. Theory*, Istanbul, Turkey, Jul. 2013, pp. 814–818, doi: 10.1109/ISIT.2013.6620339.
[33] H. A. David and H. N. Nagaraja, *Order Statistics* (Wiley Series in Probability and Statistics), 3rd ed. Hoboken, NJ, USA: Wiley, 2003. [Online]. Available: https://doi.org/10.1002/0471722162
[34] S. Boyd, P. Diaconis, and L. Xiao, "Fastest mixing Markov chain on a graph," *SIAM Rev.*, vol. 46, no. 4, pp. 667–689, 2003.
[35] O. Cihan and M. Akar, "Fastest mixing reversible Markov chains on graphs with degree proportional stationary distributions," *IEEE Trans. Autom. Control*, vol. 60, no. 1, pp. 227–232, Jan. 2015.
[36] R. Olfati-Saber, J. A. Fax, and R. M. Murray, "Consensus and cooperation in networked multi-agent systems," *Proc. IEEE*, vol. 95, no. 1, pp. 215–233, Jan. 2007.
[37] H. Wolkowicz, R. Saigal, and L. Vandenberghe, *Handbook of Semidefinite Programming: Theory, Algorithms, and Applications*, vol. 27. New York, NY, USA: Springer, 2012.
[38] J. Wang, A. K. Sahu, Z. Yang, G. Joshi, and S. Kar, "MATCHA: Speeding up decentralized SGD via matching decomposition sampling," in *Proc. 6th Indian Control Conf. (ICC)*, Dec. 2019, pp. 299–300.
[39] I. Raicu and S. Palur, "Understanding torus network performance through simulations," in *Proc. Greater Chicago Area Syst. Res. Workshop*, 2014, pp. 1–2.
[40] S. Floyd and V. Jacobson, "Random early detection gateways for congestion avoidance," *IEEE/ACM Trans. Netw.*, vol. 1, no. 4, pp. 397–413, Aug. 1993, doi: 10.1109/90.251892.
[41] D. J. Aldous, "Lower bounds for covering times for reversible Markov chains and random walks on graphs," *J. Theor. Probab.*, vol. 2, no. 1, pp. 91–100, Jan. 1989.
[42] 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.
[43] S. Caldas *et al.*, "LEAF: A benchmark for federated settings," 2018, *arXiv:1812.01097*.
[44] M. S. H. Abad, E. Ozfatura, D. Gündüz, and O. Ercetin, "Hierarchical federated learning ACROSS heterogeneous cellular networks," in *Proc. IEEE Int. Conf. Acoust., Speech Signal Process. (ICASSP)*, Barcelona, Spain, May 2020, pp. 8866–8870.
[45] J. van der Hooft *et al.*, "HTTP/2-based adaptive streaming of HEVC video over 4G/LTE networks," *IEEE Commun. Lett.*, vol. 20, no. 11, pp. 2177–2180, Nov. 2016.
[46] R. Mahajan, N. Spring, D. Wetherall, and T. Anderson, "Inferring link weights using end-to-end measurements," in *Proc. 2nd ACM SIGCOMM Workshop Internet Meas. (IMW)*, 2002, pp. 231–236. [Online]. Available: https://www.microsoft.com/en-us/research/publication/inferring-link-weights-using-end-end-measurements/
[47] X. Zhang *et al.*, "Towards reliable (and Efficient) job executions in a practical geo-distributed data analytics system," 2018, *arXiv:1802.00245*.
[48] B. Yin, X. Lu, J. Huang, and Y. Kang, "Analysis of topology dynamics for unstructured P2P networks," *Comput. Commun.*, vol. 80, pp. 72–81, Apr. 2016.

Xuezheng Liu received the M.S. degree from Sun Yat-sen University, Guangzhou, China, in 2016, where he is currently pursuing the Ph.D. degree with the School of Computer Science and Engineering, under the supervision of Prof. D. Wu. His research interests include federated learning and edge computing.

Zhicong Zhong received the B.Eng. degree from the School of Computer Science and Engineering, Sun Yat-sen University, in 2021. He is currently pursuing the Ph.D. degree with the Department of Computer Science and Engineering, The Chinese University of Hong Kong. His research interests include high performance computing and database system for persistent memory and multicore processor.

Yipeng Zhou (Member, IEEE) received the bachelor's degree in computer science from the University of Science and Technology of China (USTC) and the M.Phil. and Ph.D. degrees from the Information Engineering (IE) Department, The Chinese University of Hong Kong (CUHK). He was a Post-Doctoral Fellow with the Institute of Network Coding (INC), CUHK, from August 2012 to August 2013. From September 2013 to September 2016, he was a Lecturer with the College of Computer Science and Software Engineering, Shenzhen University. From August 2016 to February 2018, he was a Research Fellow with the Institute for Telecommunications Research (ITR), University of South Australia. He is currently a Senior Lecturer in computer science with the School of Computing, Macquarie University. He has published more than 90 papers, including IEEE INFOCOM, ICNP, IWQoS, IEEE/ACM TRANSACTIONS ON NETWORKING, IEEE JOURNAL ON SELECTED AREAS IN COMMUNICATIONS, IEEE TRANSACTIONS ON PARALLEL AND DISTRIBUTED SYSTEMS, IEEE TRANSACTIONS ON MOBILE COMPUTING, and IEEE TRANSACTIONS ON MULTIMEDIA. His research interests lie in distributed/federated learning, privacy protection, and networking. He was a recipient of ARC DECRA in 2018.

Di Wu (Senior Member, IEEE) received the B.S. degree from the University of Science and Technology of China, Hefei, China, in 2000, the M.S. degree from the Institute of Computing Technology, Chinese Academy of Sciences, Beijing, China, in 2003, and the Ph.D. degree in computer science and engineering from The Chinese University of Hong Kong, Hong Kong, in 2007. He was a Post-Doctoral Researcher with the Department of Computer Science and Engineering, Polytechnic Institute of New York University, Brooklyn, NY, USA, from 2007 to 2009, advised by Prof. K. W. Ross. He is currently a Professor and an Associate Dean with the School of Computer Science and Engineering, Sun Yat-sen University, Guangzhou, China. His research interests include edge/cloud computing, multimedia communication, internet measurement, and network security. He was a recipient of the IEEE INFOCOM 2009 Best Paper Award and IEEE Jack Neubauer Memorial Award.

Xu Chen received the Ph.D. degree in information engineering from The Chinese University of Hong Kong in 2012. He is currently a Full Professor with Sun Yat-sen University, Guangzhou, China, and the Vice Director of the National and Local Joint Engineering Laboratory of Digital Home Interactive Applications. He was a Post-Doctoral Research Associate with Arizona State University, Tempe, USA, from 2012 to 2014; and a Humboldt Scholar Fellow with the Institute of Computer Science, University of Goettingen, Germany, from 2014 to 2016. He was a recipient of the Prestigious Humboldt Research Fellowship awarded by Alexander von Humboldt Foundation of Germany, the 2014 Hong Kong Young Scientist Runner-Up Award, the 2017 IEEE Communication Society Asia–Pacific Outstanding Young Researcher Award, the 2017 IEEE ComSoc Young Professional Best Paper Award, the Honorable Mention Award of 2010 IEEE international conference on Intelligence and Security Informatics, the Best Paper Runner-Up Award of 2014 IEEE International Conference on Computer Communications (INFOCOM), and the Best Paper Award of 2017 IEEE International Conference on Communications. He is currently an Area Editor of IEEE OPEN JOURNAL OF THE COMMUNICATIONS SOCIETY; and an Associate Editor of the IEEE TRANSACTIONS WIRELESS COMMUNICATIONS, IEEE INTERNET OF THINGS JOURNAL, and IEEE JOURNAL ON SELECTED AREAS IN COMMUNICATIONS Series on Network Softwarization and Enablers.

Min Chen (Fellow, IEEE) has been a Full Professor with the School of Computer Science and Technology, Huazhong University of Science and Technology (HUST), since February 2012. He is also the Director of the Embedded and Pervasive Computing Laboratory and the Data Engineering Institute, HUST. Before HUST, he was an Assistant Professor with the School of Computer Science and Engineering, Seoul National University. His Google Scholar Citations reached more than 33,480 with an H-index of 88. His top paper was cited 3,745 times. He was selected as a Highly-Cited Researcher from 2018 to 2021. He is an IEEE Fellow for his contributions to data-driven communication, caching, and computing. He received the IEEE Communications Society Fred W. Ellersick Prize in 2017 and the IEEE Jack Neubauer Memorial Award in 2019. He is the Founding Chair of IEEE Computer Society Special Technical Communities on Big Data. He is the Chair of IEEE Globecom 2022 eHealth Symposium.

Quan Z. Sheng (Member, IEEE) received the Ph.D. degree in computer science from the University of New South Wales (UNSW). He is currently a Full Professor and the Head of the School of Computing, Macquarie University. Before moving to Macquarie, he spent ten years at the School of Computer Science, The University of Adelaide, serving in senior leadership roles, such as the Acting Head of the school and the Deputy Head of the school. He was a Post-Doctoral Research Scientist at CSIRO ICT Centre. From 1999 to 2001, he also worked at UNSW as a Visiting Research Fellow. Prior to that, he spent six years as a Senior Software Engineer in industries. His research interests include the Web of Things, the Internet of Things, big data analytics, web science, service-oriented computing, pervasive computing, and sensor networks. He is ranked by Microsoft Academic as one of the Most Impactful Authors in Services Computing (ranked top 5 all time worldwide). He was a recipient of the AMiner Most Influential Scholar Award on IoT (2019), ARC Future Fellowship (2014), Chris Wallace Award for Outstanding Research Contribution (2012), and Microsoft Research Fellowship (2003).