DySTop: Dynamic Staleness Control and Topology Construction for Asynchronous Decentralized Federated Learning

Yizhou Shi, Qianpiao Ma, Yan Xu, Junlong Zhou, *Member, IEEE*, Ming Hu, *Member, IEEE*, Yunming Liao, Hongli Xu, *Member, IEEE*

***Abstract*—Federated Learning (FL) has emerged as a potential distributed learning paradigm that enables model training on edge devices (*i.e.*, workers) while preserving data privacy. However, its reliance on a centralized server leads to limited scalability. Decentralized federated learning (DFL) eliminates the dependency on a centralized server by enabling peer-to-peer model exchange. Existing DFL mechanisms mainly employ synchronous communication, which may result in training inefficiencies under heterogeneous and dynamic edge environments. Although a few recent asynchronous DFL (ADFL) mechanisms have been proposed to address these issues, they typically yield stale model aggregation and frequent model transmission, leading to degraded training performance on non-IID data and high communication overhead. To overcome these issues, we present DySTop, an innovative mechanism that jointly optimizes dynamic staleness control and topology construction in ADFL. In each round, multiple workers are activated, and a subset of their neighbors is selected to transmit models for aggregation, followed by local training. We provide a rigorous convergence analysis for DySTop, theoretically revealing the quantitative relationships between the convergence bound and key factors such as maximum staleness, activating frequency, and data distribution among workers. From the insights of the analysis, we propose a worker activation algorithm (WAA) for staleness control and a phase-aware topology construction algorithm (PTCA) to reduce communication overhead and handle data non-IID. Extensive evaluations through both large-scale simulations and real-world testbed experiments demonstrate that our DySTop reduces completion time by 46.7% and the communication resource consumption by 48.3% compared to state-of-the-art solutions, while maintaining the same model accuracy.**

***Index Terms*—Decentralized federated learning, edge computing, asynchronous, staleness control, topology construction.**

I. Introduction

RECENT advances in the Internet of Things (IoT) promote continuous generation of vast quantities of data generated from physical environments [1]–[3]. Generally, this data is sent to a remote cloud for processing, raising concerns about potential privacy leakage and considerable latency. In response, *edge computing* has evolved as a viable solution by distributing computation capacity to edge devices (*i.e.*, workers) for accelerated local data processing. Furthermore, it promotes the adoption of federated learning (FL), which coordinates workers to perform distributed machine learning [4] while preserving data privacy.

Traditional FL, also known as Centralized FL (CFL), comprises a set of workers and a centralized parameter server (PS) [4]–[6]. These workers collaboratively train a global model by performing local updates on their own datasets and subsequently transmitting their model parameters to the PS. The PS next aggregates these uploaded models and distributes the aggregated model back to the workers for the subsequent training round. This procedure usually maintains multiple training rounds until model convergence. Though CFL is a mature and widely used scheme, it comes with notable limitations. First, its reliance on a star topology inherently restricts system scalability. Second, frequent communication between the PS and workers generates a substantial traffic workload and creates a communication bottleneck at the PS, posing a significant risk of a single point failure.

Decentralized federated learning (DFL) has emerged as a solution to overcome the limitations of CFL [7]–[12]. In DFL, each worker performs local model updating and transmits the updated models to its neighbors through peer-to-peer (P2P) communications. Since DFL does not rely on a centralized PS, it can achieve high scalability and construct network topology more flexibly. When a worker in the system fails, it is only necessary for the affected workers to re-establish P2P connections with other workers. Moreover, each worker communicates with its neighbors, effectively distributing the communication overhead that was originally borne solely by the PS in CFL. Therefore, DFL is particularly suitable for emerging IoT scenarios such as vehicular networks and UAV clusters [17]. These environments naturally form decentralized P2P networks and are often unsuitable for maintaining long-term communication with a centralized remote server. However, achieving high efficiency in DFL presents unique challenges that need to be addressed.

- **Edge Heterogeneity:** Each worker interacts with heterogeneous neighbors with varying computational and communication capabilities, data sizes, and physical locations [18]. Therefore, the coordination required for training in DFL is more complex compared with CFL.
- **Edge Dynamic:** The network topologies may change over time due to intermittent connections among workers

Corresponding authors: Qianpiao Ma, Junlong Zhou.

Yizhou Shi, Qianpiao Ma, Yan Xu and Junlong Zhou are with the School of Computer Science and Engineering, Nanjing University of Science and Technology, Nanjing, Jiangsu 210094, China (e-mail: yizhou_shi@njust.edu.cn, maqianpiao@njust.edu.cn, 084621120@njucm.edu.cn, jlzhou@njust.edu.cn).

Ming Hu is with the School of Computing and Information Systems, Singapore Management University, Singapore (e-mail: hu.ming.work@gmail.com).

Yunming Liao and Hongli Xu are with the School of Computer Science and Technology, University of Science and Technology of China, Hefei, Anhui 230027, China (e-mail: ymliao98@mail.ustc.edu.cn, xuhongli@ustc.edu.cn).

TABLE I: Comparison of different DFL mechanisms.

DFL Mechanisms		Handling Edge Heterogeneity	Handling Edge Dynamic	Communication Consumption	Handling Non-IID	Handling Staleness
Synchronous	BrainTorrent [7]	Poor	Poor	High	Poor	-
	GossipFL [8]	Poor	Poor	High	Poor	-
	D-Cliques [9]	Poor	Poor	Medium	Good	-
	MATCHA [10]	Poor	Medium	Low	Poor	-
	L2PL [11]	Poor	Medium	Low	Poor	-
	FedHP [12]	Poor	Medium	Low	Good	-
Asynchronous	HADFL [13]	Good	Poor	High	Poor	Poor
	AsyNG [14]	Good	Good	Medium	Good	Poor
	AsyDFL [15]	Good	Good	Medium	Good	Poor
	SA-ADFL [16]	Good	Good	High	Poor	Medium
	DySTop (Ours)	Good	Good	Low	Good	Good

caused by their mobility [19]. Furthermore, a worker with poor communication conditions may lead to disconnections for its neighbors.

- **Communication Resource Constraint:** On one hand, since the absence of the PS, DFL typically requires more training rounds to converge than CFL. On the other hand, in DFL, each worker sends its model to multiple workers rather than a single PS, resulting in more communication resource consumption compared to CFL [8].
- **Data Non-IID:** The data collected on workers is often not a uniform sample of the overall distribution, leading to data non-independent-and-identically-distributed (non-IID) [4], which degrades FL training. Even more critically, unlike the PS in CFL, which trains the global model over all data, each worker in DFL may obtain models trained over more biased data [14].

There are two communication schemes for DFL: synchronous and asynchronous. Most of the existing DFL researches adopt synchronous communication [7]–[12], where each worker waits for all its neighbors to finish local model updating and aggregates them with its own local model. However, due to edge heterogeneity, workers are compelled to wait for the slowest worker to complete its local updating and model transferring, known as the *straggler problem*. Moreover, edge dynamics may render synchronous DFL impractical, as dynamic network conditions and potential disconnections can lead to significant idle time while workers wait for model transmissions from all neighbors in each round.

To handle heterogeneity and dynamics in edge networks, the asynchronous DFL (ADFL) scheme [13]–[16] further eliminates the synchronization barrier in synchronous DFL. In ADFL, each worker is allowed to exchange models with its neighbors at any time and aggregate models it has received so far upon finishing previous local updating. Since a worker in ADFL does not need to wait for all its neighbors to transmit their latest local models, ADFL effectively addresses edge heterogeneity and edge dynamics. However, the existence of stragglers still causes different relative frequencies of workers to perform asynchronous updating, which leads to *staleness* concern [20]. Consequently, workers may aggregate the out-of-date models from their neighbors, causing unsatisfactory training performance [21]. To this end, our previous work [16] SA-ADFL realizes dynamic staleness control for ADFL by appropriately determining a worker to perform model aggregation and transferring in each round. However, in SA-ADFL, the determined worker sends its model to all neighbors within its communication range, resulting in significant communication and lacking fine-grained handling of data non-IID.

To address this, we propose DySTop, a novel mechanism that performs **Dy**namic **S**taleness Control and **Top**ology Construction for ADFL. In each round, DySTop dynamically activates multiple workers, each selecting a subset of its neighbors and pulling their models for aggregation. We provide a rigorous convergence analysis for DySTop, revealing the significance of both staleness control and topology construction in achieving satisfactory training performance. Guided by the analysis, we design a worker activation algorithm to achieve more effective staleness control than SA-ADFL, and a phase-aware topology construction algorithm considering both communication efficiency and data non-IID. The summarization of different DFL mechanisms are shown in Table I.

Our main contributions to this paper are as follows:

- We propose DySTop, a dynamic staleness control and topology construction mechanism for ADFL, and provide its convergence analysis, theoretically revealing how the convergence bound depends on factors such as maximum staleness, activating frequency, and data distribution.
- Based on the convergence analysis, we formulate a training time minimization problem for DySTop given staleness and communication resource constraints, and transform the problem into multiple per-round subproblems by Lyapunov optimization.
- To solve the subproblems, we design a worker activation algorithm (WAA) for staleness control and a phase-aware topology construction algorithm (PTCA) for reducing communication overhead and handling data non-IID.
- We conduct both large-scale simulations and real-world testbed experiments to evaluate the performance of our proposed mechanism and algorithms. Experimental results show that our algorithms can reduce completion time by 46.7% and the communication resource consumption by 48.3% under the same accuracy requirement compared with the state-of-the-art mechanisms.

The rest of this paper is organized as follows. Section II reviews the related works. Section III introduces the DySTop mechanism and formulates the problem. Section IV provides

the convergence analysis. The algorithms for DySTop are proposed in Section V. The experimental results of the simulations and testbeds are shown in Sections VI and VII, respectively. Section VIII concludes this paper.

II. Related Works

The communication schemes of DFL determine how workers perform local training and exchange models, and can be categorized as either synchronous or asynchronous. Most existing DFL studies adopt synchronous communication schemes, which are reviewed in Section II-A. More recently, a few DFL studies have explored asynchronous schemes, as reviewed in Section II-B. Although asynchronous schemes can improve resource utilization and reduce idle time, they introduce model staleness. The techniques for mitigating staleness in asynchronous FL are reviewed in Section II-C.

A. Communication Schemes of Synchronous DFL

In synchronous DFL, each worker waits for all its neighbors to finish local model updating and performs model aggregation. Therefore, it is important to construct an efficient topology for model parameter transmission in DFL [22]. BrainTorrent [7] establishes a fully-connected DFL, requiring each worker to share its local model with all others in each round. However, it incurs heavy communication costs due to the large volume of data transmissions. Partially connected topologies reduce the communication overhead by enabling each worker to exchange its local model only with its neighboring workers. For example, GossipFL [8] creates a highly sparsified topology, allowing each worker to transmit its local model to only one peer in each round, realizing efficient communication. D-Cliques [9] proposes a new topology where workers are grouped into connected cliques based on data distribution, which helps reduce gradient bias. Yet, previous works tend to construct a fixed topology and ignore the edge dynamic. Therefore, some works focus on re-constructing the topology at regular intervals. For example, MATCHA [10] employs matching decomposition to break down the base topology into disjoint subgraphs, and samples a subset of these subgraphs to construct a sparse topology in each round. L2PL [11] proposes a reinforcement learning-based method to dynamically construct an optimal sparse topology for each round. FedHP [12] controls local updating frequency and network topology adaptively to improve the overall learning efficiency.

However, the above synchronous DFL mechanisms share a common drawback: they force each worker to await the completion of all other workers, *i.e.*, the synchronization point, before the subsequent round can begin. In edge heterogeneity environments, this process results in slow convergence, since the overall efficiency is dictated by the slowest worker.

B. Communication Schemes of Asynchronous DFL

A few studies have explored asynchronous communication schemes for DFL, allowing workers to exchange models whenever they finish local updating, instead of waiting for the synchronization points. For example, HADFL [13] facilitates decentralized training on heterogeneous workers in an asynchronous manner, with probabilistic participation in model synchronization and aggregation in each round. AsyNG [14] and AsyDFL [15] develop a joint optimization framework integrating gradient pushing and neighbor selection to achieve a communication-training trade-off in ADFL.

However, these ADFL studies overlook the staleness problem inherent in asynchronous communication. This oversight results in highly stale models, which generate significant deviation from current gradient trajectories and intensify the detrimental effects of non-IID data.

C. Deal with Staleness in Asynchronous FL

As previously stated, staleness significantly impacts the performance of asynchronous FL. Many asynchronous CFL studies have focused on mitigating the negative effects of staleness. For example, FedAsync [20] relieves the impact of staleness on training by assigning smaller aggregation weights to stale models. FedSA [21] employs a semi-asynchronous mechanism where partial workers participate in each global updating to control staleness. SAFA [23] tackles staleness by simply discarding models that become excessively stale during the training process. SC-AFL [24] restricts the staleness degree within a certain bound by dynamically adjusting the aggregated strategy in each round. ASO-Fed [25] deploys dynamic learning rates for workers based on their participation frequencies in global updating. Air-FedGA [26], [27] assigns workers into groups to mitigate staleness. The authors in [28] and [29] compensate for the gradients of stale models by leveraging the approximate Taylor expansion.

However, the above staleness control methods primarily focus on asynchronous CFL scenarios, whereas staleness control in ADFL is more complex and remains largely unexplored. Our previous work SA-ADFL [16] controls staleness for ADFL by determining a worker to perform model updating and send its model to all neighbors in each round. However, it incurs high communication overhead and lacks fine-grained handling of data non-IID. In this paper, we propose DySTop, a mechanism that dynamically activates multiple workers, each selecting a subset of its neighbors and pulling their models for aggregation, thereby reducing communication overhead and effectively addressing data non-IID.

III. System Overview and Problem Formulation

A. Preliminary for ADFL

We consider an asynchronous decentralized federated learning system, where a set of workers $\mathcal{V} = \{v_1, v_2, \cdots, v_N\}$ is deployed to train local models. Given the inherent edge dynamics that induce topological changes, we characterize the time-varying network topology through a directed graph $\mathcal{G}_t = (\mathcal{V}_t, \mathcal{E}_t)$ per round t. The vertex set $\mathcal{V}_t$ tracks available workers, and the edge set $\mathcal{E}_t$ contains the links between devices, with $e(v_i, v_j) \in \mathcal{E}_t$ specifying the transfer path from v_i to v_j. For worker v_i, we define $\mathcal{N}_t^i = \{v_j | e(v_j, v_i) \in \mathcal{E}_t, v_j \in \mathcal{V}_t\} \cup \{v_i\}$ as its in-neighbors set that consists of all workers with an in-coming link to v_i include itself.

Correspondingly, the out-neighbors set of v_i at round t is defined as $\hat{\mathcal{N}}_t^i = \{v_j | e(v_i, v_j) \in \mathcal{E}_t, v_j \in \mathcal{V}_t\} \cup \{v_i\}$.

Each worker $v_i \in \mathcal{V}$ conducts local training on its local dataset $\mathcal{D}_i = \bigcup_{k=1}^{|\mathcal{D}_i|}(\mathbf{X}_i^k, y_i^k)$ with size $D_i = |\mathcal{D}_i|$, where $\mathbf{X}_i^k$ denotes the feature vector of k-th sample and y_i^k represents the corresponding label. Let $\mathbf{w}$ be the model parameter. The local loss function of worker v_i is given by

$$F_i(\mathbf{w}) \triangleq \mathbb{E}_{\xi_i \in \mathcal{D}_i} f(\mathbf{w}; \xi_i), \tag{1}$$

where $f(\mathbf{w}; \xi_i)$ computes the loss on mini-batch over mini-batch ξ_i drawn from local dataset $\mathcal{D}_i$. $f(\cdot;\cdot)$ accommodates various loss formulations, *e.g.*, cross-entropy, focal or hinge loss. Define $\alpha_i = \frac{D_i}{D}$ as the relative data size of v_i to the total, the global loss function is defined as

$$F(\mathbf{w}) \triangleq \sum_{v_i \in \mathcal{V}} \frac{D_i}{D} F_i(\mathbf{w}) = \sum_{v_i \in \mathcal{V}} \alpha_i F_i(\mathbf{w}). \tag{2}$$

The objective is to find the optimal parameter vector $\mathbf{w}^*$ so as to minimize $F(\mathbf{w})$, *i.e.*, $\mathbf{w}^* = \text{argmin}_{\mathbf{w}} F(\mathbf{w})$.

Inspired by [30], we divide the ADFL process into multiple rounds, and we introduce $t \in \mathcal{R} = \{1, 2, \cdots, T\}$ to indicate the index of a round. At each round t, the local model of worker v_i is represented as $\mathbf{w}_t^i$. Due to the asynchronous updating mechanism and heterogeneous computational capabilities across workers, the local model updates often span multiple rounds. Let τ_t^i denote the interval (called the *staleness*) between the latest starting-to-training round of v_i and the current round t. The local model at round t actually satisfies

$$\mathbf{w}_t^i = \mathbf{w}_{t-\tau_t^i}^i. \tag{3}$$

B. System overview of DySTop

We propose a dynamic staleness control and topology construction mechanism for ADFL, called DySTop, which is formally described in Alg. 1. DySTop mainly involves two key roles, *i.e.*, a set of workers and a coordinator. Periodically, the coordinator acquires worker-specific status data such as available bandwidth, model staleness, and data distribution, which it then uses to determine worker activation and topology construction strategies. Now we introduce the process of DySTop on both the worker side and the coordinator side.

On the worker side (Lines 3-10), each worker maintains two threads: an updating thread (Lines 4-7) and a pushing thread (Lines 8-10). For updating thread, if worker v_i receives an EXECUTE message from the coordinator at round t and has completed its local training of the current round, it sends PULL message to its in-neighbor workers $v_j, \forall v_j \in \mathcal{N}_t^i$, then each in-neighbors v_j transmits its local model $\mathbf{w}_t^j$ to v_i. Next, v_i aggregates the pulled local models by

$$\hat{\mathbf{w}}_t^i = \sum\nolimits_{v_j \in \mathcal{N}_t^i} \sigma_t^{i,j} \cdot \mathbf{w}_t^j = \sum\nolimits_{v_j \in \mathcal{N}_t^i} \sigma_t^{i,j} \cdot \mathbf{w}_{t-\tau_t^j}^j. \tag{4}$$

where $\sigma_t^{i,j}$ is the aggregated weight of v_j on the worker v_i at round t, calculated as $\sigma_t^{i,j} = D_j / \sum_{v_{j'} \in \mathcal{N}_t^i} D_{j'}$. After model aggregation, the worker v_i performs local training as

$$\mathbf{w}_{t+1}^i = \hat{\mathbf{w}}_t^i - \eta \nabla F_i(\hat{\mathbf{w}}_t^i; \xi_t^i). \tag{5}$$

where η is the learning rate, and $\nabla F_i(\hat{\mathbf{w}}_t^i; \xi_t^i)$ denotes the stochastic gradient over mini-batch ξ_t^i. Meanwhile, at any time receiving PULL message from the out-neighbor $v_j \in \hat{\mathcal{N}}_t^i$, the pushing thread of worker v_i pushes its up-to-date local model, which is formulated in Eq. (3).

Algorithm 1 Dynamic Staleness Control and Topology Construction for ADFL (DySTop)

1: **for** $t = 1$ to T **do**
2: **Processing at Each Device** v_i**:**
3: ***Updating Thread:***
4: **if** Receive EXECUTE message **and** finish local training **then**
5: Obtain $\mathbf{w}_t^j$ of each device $v_j \in \mathcal{N}_t^i$ by sending PULL message to its in-neighbors;
6: Aggregate local models to obtain $\hat{\mathbf{w}}_t^i$ by Eq. (4);
7: Update local model $\mathbf{w}_{t+1}^i$ by Eq. (5);
8: ***Pushing Thread:***
9: **if** Receive PULL message from $v_j \in \hat{\mathcal{N}}_t^i$ **then**
10: Push local model $\mathbf{w}_{t-\tau_t^i}^i$ to v_j;
11: **Processing at the Coordinator:**
12: Determine active set $\mathcal{A}_t$ according to Alg. 2 to pull models and perform aggregation;
13: Contruct network topology $\mathcal{G}_t$ according to Alg. 3;
14: Send EXECUTE message to $v_i \in \mathcal{A}_t$;
15: **for each** $v_i \in \mathcal{V}$ **do**
16: Update staleness τ_{t+1}^i by Eq. (6);

On the coordinator side (Lines 11-16), at the beginning of each round t, the coordinator determines a subset of workers $\mathcal{A}_t \subseteq \mathcal{V}$ (we call the active set) for aggregation at round t, which will be elaborated in Alg. 2. Each worker v_i is assigned a binary indicator $a_t^i \in \{0, 1\}$. That is, $a_t^i = 1$ if v_i is activated for model aggregation at round t, and $a_t^i = 0$ otherwise. Then, the coordinator constructs the network topology $\mathcal{G}_t$, which will be elaborated in Alg. 3. Next, the coordinator sends EXECUTE message to $v_i \in \mathcal{A}_t$, and updates staleness of each worker $v_i \in \mathcal{V}$ as

$$\tau_{t+1}^i = \left(\tau_t^i + 1\right)\left(1 - a_t^i\right). \tag{6}$$

That is, for worker $v_i \in \mathcal{A}_t$, the next round staleness τ_{t+1}^i is reset to 0, while the staleness of all other workers increases. To control the staleness of workers, we define a maximal bound τ_{bound}, which all workers must satisfy $\tau_t^i \leq \tau_{\text{bound}}$. Unlike prior work [21], [23], we do not treat τ_{bound} as a hard constraint. Therefore, DySTop does not interrupt the local training of stale workers or drop stale models whose staleness exceeds τ_{bound}. Instead, we serve τ_{bound} as a soft reference parameter to incentivize workers with higher staleness to be prioritized for scheduling. This mechanism will be detailed in Section V.

C. Problem Formulation

For DySTop, one problem is how to determine a proper active set $\mathcal{A}_t$ at each round t, resulting in the process of model training and transmission. Let H_t denote the duration of round t. h_i represents the local training time required for worker v_i. Then the time consumed by v_i for local training at round t is calculated as

$$h_t^{i,\text{cmp}} = \max\left\{h_i - \sum_{k=t-\tau_t^i}^{t-1} H_k, 0\right\}. \tag{7}$$

Let $h_t^{i,j,\mathrm{com}}$ denote the time consumed by the transmission of the model from v_j to v_i at round t. Thus, the sum of training and transmission time of the activated worker v_i at round t is

$$H_t^i = h_t^{i,\mathrm{cmp}} + \max_{v_j \in \mathcal{N}_t^i} \left\{ h_t^{i,j,\mathrm{com}} \right\}. \tag{8}$$

Therefore, the duration of round t is calculated as

$$H_t = \max_{v_i \in \mathcal{A}_t} \left\{ H_t^i \right\}. \tag{9}$$

The other problem is how to construct a network topology $\mathcal{G}_t$ for model transmission given communication resource constraints. Specifically, let $c_t^{i,j} \in \{0,1\}$ indicate whether the directed link from v_j to v_i is connected for model transmission at round t. $c_t^{i,j} = 1$ if the link is connected, and otherwise $c_t^{i,j} = 0$. Let b denote the bandwidth resource consumption for transmitting one local model. Since both pulling models and pushing models consume bandwidth, the bandwidth consumption of worker v_i at round t is

$$B_t^i = \left(\sum\nolimits_{v_j \in \mathcal{N}_t^i} c_t^{i,j} + \sum\nolimits_{v_k \in \hat{\mathcal{N}}_t^i} c_t^{k,i} \right) \cdot b. \tag{10}$$

Unlike centralized FL, which always maintains a global model in the PS, DFL has no real global model due to its fully decentralized setting. Therefore, we define the global weighted model at round t as

$$\mathbf{w}_t = \sum_{v_i \in \mathcal{V}} \frac{D_i}{D} \mathbf{w}_t^i = \sum_{v_i \in \mathcal{V}} \alpha_i \mathbf{w}_t^i. \tag{11}$$

Let $\boldsymbol{a} = \{\boldsymbol{a}_t | t \in [1,T]\}$ and $\boldsymbol{c} = \{\boldsymbol{c}_t | t \in [1,T]\}$ denote the worker activation and topology construction strategies employed throughout the T rounds of training process, respectively, where $\boldsymbol{a}_t = \{a_t^i | v_i \in \mathcal{V}\}$ and $\boldsymbol{c}_t = \{c_t^{i,j} | v_i, v_j \in \mathcal{V}\}$. Our problem is formulated as

$$\textbf{(P1)}: \min_{\boldsymbol{a},\boldsymbol{c}} \sum_{t=1}^{T} H_t \tag{12a}$$

$$\textbf{s.t.} \quad F(\mathbf{w}_T) \le \epsilon, \tag{12b}$$

$$\tau_t^i \le \tau_{\text{bound}}, \qquad \forall v_i \in \mathcal{V}, t \in \mathcal{R}, \tag{12c}$$

$$B_t^i \le \hat{B}_t^i, \qquad \forall v_i \in \mathcal{V}, t \in \mathcal{R}, \tag{12d}$$

$$a_t^i \in \{0,1\}, \qquad \forall v_i \in \mathcal{V}, t \in \mathcal{R}, \tag{12e}$$

$$c_t^{i,j} \in \{0,1\}, \qquad \forall v_i, v_j \in \mathcal{V}, t \in \mathcal{R}. \tag{12f}$$

The inequality (12b) ensures the convergence of global loss function after T rounds, where ϵ is the convergence threshold. The set of inequalities (12c) represents that the staleness of each worker cannot exceed the maximal bound τ_{bound}. The set of inequalities (12d) represents that bandwidth allocation for each worker should not exceed its bandwidth resource budget $\hat{B}_t^i$, which is time-varying considering the dynamic network condition. Our goal is to jointly optimize the worker activation strategy $\boldsymbol{a}$ and topology construction strategy $\boldsymbol{c}$ to minimize the overall training duration in DySTop, *i.e.*, $\min_{\boldsymbol{a},\boldsymbol{c}} \sum_{t=1}^{T} H_t$.

IV. Convergence Analysis

A. Assumptions

We use the following assumptions that are classical to the analysis of the loss functions $F_i(\mathbf{w}), \forall v_i \in \mathcal{V}$ [15] [5] [31].

Assumption 1 (Smoothness): *$F_i(\mathbf{w})$ is L-smooth, i.e.,* $\forall \mathbf{w}_1, \mathbf{w}_2$, $F_i(\mathbf{w}_2) - F_i(\mathbf{w}_1) \le \langle \nabla F_i(\mathbf{w}_1), \mathbf{w}_2 - \mathbf{w}_1 \rangle + \frac{L}{2} \|\mathbf{w}_2 - \mathbf{w}_1\|^2$.

Assumption 2 (Convexity): *$F_i(\mathbf{w})$ is μ-strongly convex, i.e.,* $\forall \mathbf{w}_1, \mathbf{w}_2$, $F_i(\mathbf{w}_2) - F_i(\mathbf{w}_1) \ge \langle \nabla F_i(\mathbf{w}_1), \mathbf{w}_2 - \mathbf{w}_1 \rangle + \frac{\mu}{2} \|\mathbf{w}_2 - \mathbf{w}_1\|^2$.

Note that Assumption 2 holds for those models with convex loss functions, like support vector machines and linear regression. Our mechanism can also work for models (*e.g.*, CNN) with non-convex loss functions, which is shown in Section VI.

B. Analysis of Convergence Bounds

To describe the data distribution among workers, we give the following definitions, which are widely adopted in the existing works [5] [31] [32].

Definition 1 (Gradient Divergence): *For $\forall v_i, \mathbf{w}$, we define ξ_i as the upper bound of the gradient divergence between the dataset of worker v_i and the globe dataset, i.e.,*

$$\|\nabla F(\mathbf{w}) - \nabla F_i(\mathbf{w})\| \le \xi_i$$

Definition 2 (Local Gradient): *$\mathbf{w}_i^*$ is the local optimal parameter vector on worker v_i, then we define the expectation of its random gradients on v_i is*

$$g_i^* = \mathbb{E}[\|\nabla F_i(\mathbf{w}_i^*; \xi_i)\|^2]$$

where ξ_i is a arbitrary sample on worker v_i.

For simplicity, we denote the minimum of the global loss function F as F^*, short for $F(\mathbf{w}^*)$.

1) Convergence of Local Updating in One Round: We begin by presenting the following Lemma 1, which details the convergence of the local update in one round.

Lemma 1 : *Taking $\eta < \frac{\mu}{2L^2}$, by the local updating of Eq.* (5)*, it holds that*

$$\mathbb{E}[F(\mathbf{w}_{t+1}^i)] - F^* \le \rho(\mathbb{E}[F(\hat{\mathbf{w}}_t^i)] - F^*) + \delta_i,$$

where $\rho = 1 - \mu\eta$, and $\delta_i = \frac{\eta}{2}\xi_i + L\eta^2 g_i^*$.

Proof. See Appendix A. □

2) A key Lemma for Analysis: To simplify the expression, we define three sequences of matrices $\mathbf{X}_t$, $\mathbf{Y}_t^j$ and $\mathbf{Z}_t$ for $t \ge 1$, where $\mathbf{X}_t = \mathrm{diag}(x_t^1, x_t^2, ..., x_t^N)$ and $\mathbf{Y}_t^j = \mathrm{diag}(y_t^{1,j}, y_t^{2,j}, ..., y_t^{N,j})$, $j \in [1,N]$ are $N \times N$ diagonal matrices. $\mathbf{Z}_t = [z_t^1, z_t^2, ..., z_t^N]^\top$ is a N-dimensional vector. In particular, if worker v_i is not activated by the coordinator at round t, *i.e.*, $v_i \notin \mathcal{A}_t$, then the i-th elements of $\mathbf{X}_t$, $\mathbf{Y}_t^j$ and $\mathbf{Z}_t$ are given by $x_t^i = 1$, $y_t^{i,j} = 0$ and $z_t^i = 0$, respectively. x_t^i and $y_t^{i,j}$ satisfy $\theta_t^i = x_t^i + \sum_{v_j \in \mathcal{N}_t^i} y_t^{i,j} \le 1$. We first state a key lemma for our statement. For convenience, let $\omega_t^j = t - \tau_t^j - 1 \ge 0$, $\tau_{\max} = \max_{j,t}\{\tau_t^j\}$ and $\theta_{\max} = \max_{i,t}\{\theta_t^i\}$.

Lemma 2 : *Let $\mathbf{Q}_t$ be a sequence of matrices for $t \ge 0$. If $\mathbf{Q}_t \le \mathbf{X}_t \mathbf{Q}_{t-1} + \sum_{v_j \in \mathcal{N}_t^i} \mathbf{Y}_t^j \mathbf{Q}(\omega_t^j) + \mathbf{Z}_t$, then we have*

$$\mathbf{Q}_T \le \prod_{t=0}^{T} \mathbf{P}_t \mathbf{Q}_0 + \sum_{t=0}^{T} \boldsymbol{\Delta}_t,$$

where $\mathbf{P}_t = diag(p_t^1, p_t^2, ..., p_t^N)$ is a $N \times N$ diagonal matrix and each scalar component satisfies

$$p_t^i = \begin{cases} \theta_{\max} \frac{1}{1+\tau_{max}}, & \text{if } v_i \in \mathcal{A}_t \\ 1, & \text{otherwise} \end{cases}.$$

$\mathbf{\Delta}_t$ *is a* M*-dimensional vector satisfies the following recursive relation:*

$$\mathbf{\Delta}_t = \begin{cases} [0,0,...,0]^\top, t=0 \\ (\mathbf{X}_t + \sum_{v_j \in \mathcal{N}_t^i} \mathbf{Y}_t^j - \mathbf{E}) \sum_{r=0}^{t-1} \mathbf{\Delta}_r + \mathbf{Z}_t, t \geq 1. \end{cases}$$

$\mathbf{E}$ *denotes the* $N \times N$ *identity matrix, whose elements on the main diagonal are* 1 *and all off-diagonal elements are* 0*.*

Proof. See Appendix B. □

3) Convergence Bound of DySTop: We next analyze DySTop's convergence bound with Lemma 1 and Lemma 2 in the following Theorem 1. For clarity, we represent the weight vector of all workers as $\mathbf{A} = [\alpha_1, ...\alpha_N]$, where each v_i's weight is $\alpha_i = \frac{D_i}{D}$, denoting the relative data size to the total. Since a global model does not exist in a decentralized topology, we consider the weighted sum of of all local models, that is $\mathbf{w}_T = \sum_{v_i \in \mathcal{V}} \frac{D_i}{D} \mathbf{w}_T^i = \sum_{v_i \in \mathcal{V}} \alpha_i \mathbf{w}_T^i$.

Theorem 1 : $\mathbf{w}_0$ *is the initial model on each worker. After inter-cluster aggregation Eq.* (4) *is performed* T *times, the weighted model* $\mathbf{w}_T$ *satisfies*

$$\mathbb{E}[F(\mathbf{w}_T)] - F^* \leq \sum_{v_i \in \mathcal{V}} \alpha_i \rho^{\frac{\psi_i T}{1+\tau_{\max}}} (F(\mathbf{w}_0) - F^*) + \mathbf{A} \sum_{t=0}^{T} \mathbf{\Delta}_t,$$

where ψ_i *denotes the activating frequency of worker* v_i*, and* $\mathbf{\Delta}_t$ *meets the following recursive relation:*

$$\mathbf{\Delta}_t = \begin{cases} [0,0,...,0]^\top, t=0 \\ \mathbf{W}_t \sum_{r=0}^{t-1} \mathbf{\Delta}_r + \mathbf{Z}_t, t \geq 1. \end{cases} \tag{13}$$

where $\mathbf{W}_t = diag(w_t^1, w_t^2, ..., w_t^N)$ *is a* $N \times N$ *diagonal matrix and the scalar components satisfy*

$$w_t^i = \begin{cases} \rho, & \text{if } v_i \in \mathcal{A}_t \\ 1, & \text{otherwise} \end{cases}.$$

and $\mathbf{Z}_t = [z_t^1, z_t^2, ..., z_t^N]^\top$ *is a vector and the scalar components satisfy*

$$z_t^i = \begin{cases} \sum_{v_j \in \mathcal{N}_t^i} \sigma_t^{i,j} \delta_j, & \text{if } v_i \in \mathcal{A}_t \\ 0, & \text{otherwise} \end{cases}.$$

Proof. See Appendix C. □

C. Discussions

We denote the convergence bound of DySTop in Theorem 1 as $Bound_T = \sum_{v_i \in \mathcal{V}} \alpha_i \rho^{\frac{\psi_i T}{1+\tau_{\max}}} (F(\mathbf{w}_0) - F^*) + \mathbf{A} \sum_{t=0}^{T} \mathbf{\Delta}_t$. From this, we can derive several meaningful corollaries.

Corollary 1 : *The convergence bound* $Bound_T$ *decreases as the upper bound of staleness* $\tau_{\max}$ *decreases.*

Corollary 2 : *The convergence bound* $Bound_T$ *decreases as the activating frequency* ψ_i *of each worker* v_i *increases.*

Corollary 3 : *According to Eqs.* (29) *and* (13)*,* $\mathbf{\Delta}_t$ *is partially influenced by the data distribution. As the degree of data non-IID among workers increases, the value of* ξ_i *for each worker* v_i *rises, leading to a high convergence bound* $Bound_T$*.*

Corollary 1 indicates that convergence performance can be improved by controlling the maximum staleness during training, ensuring it does not exceed a certain upper bound, *i.e.*, $\tau_{\max} \leq \tau_{\text{bound}}$. Corollary 2 suggests that increasing the number of activated workers $|\mathcal{A}_t|$ in each round can also improve convergence performance. However, this does not necessarily lead to a short convergence time, since each round's completion time depends on the model aggregations and training of all activated workers. Therefore, activating more workers may prolong the duration of each round. Corollaries 1 and 2 highlight the importance of carefully determining the worker set $|\mathcal{A}_t|$ to control staleness and accelerate convergence. From Corollary 3, when the data distribution among workers is IID, *i.e.*, $\xi_i = 0$ for $\forall v_i \in \mathcal{V}$, the convergence performance of DySTop can be further improved. In practice, however, data is often non-IID for corollary workers. To address this, one can design the training topology and select neighbors for each activated worker such that the union of a worker and its neighbors' datasets approximates an IID distribution, thereby enhancing training performance [9].

Corollaries 1-3 theoretically reveal the significance of both staleness control and topology construction in achieving better training performance, which will be elaborated in Section V.

V. Algorithm Description

A. Algorithm Overview

Tackling the problem **P1** is a complex endeavor due to three main challenges below:

- First, **P1** is a round-coupled problem. The staleness constraints of workers are coupled with worker activation strategies across rounds.
- Second, based on Corollaries 1 and 2, there is a trade-off between staleness control and convergence rate with different activated workers, but it's hard to quantify the correlation between them.
- Third, based on Corollary 3, a lower degree of data non-IID leads to faster convergence. However, communication constraint forces us to select only a subset of neighbors, and improper selection can conversely exacerbate the degree of data non-IID.

To overcome these challenges, we first employ Lyapunov optimization to decouple the challenging round-coupled optimization problem into deterministic per-round subproblems. This prevents cross-round decision coupling from compromising global optimality. We further solve each per-round subproblem in two steps. First, we devise a worker activation algorithm (WAA) to determine the active set $\mathcal{A}_t$ in each round, where we utilize Lyapunov drift-plus-penalty function to quantify the trade-off between staleness control and convergence rate. After that, we design a phase-aware topology construction algorithm (PTCA) to dynamically adapt the topology $\mathcal{G}_t$ in each round. We set individual strategies for different training phases to reduce communication overhead and non-IID degree.

Fig. 1 shows a four-worker example to demonstrate the workflow of DySTop. There is a network topology with 4 workers ($\mathcal{V} = \{v_1, v_2, v_3, v_4\}$). At a certain round t, the coordinator first collects the information of workers (*e.g.*, current model staleness and estimated training time of the workers) and performs WAA to determine the active set

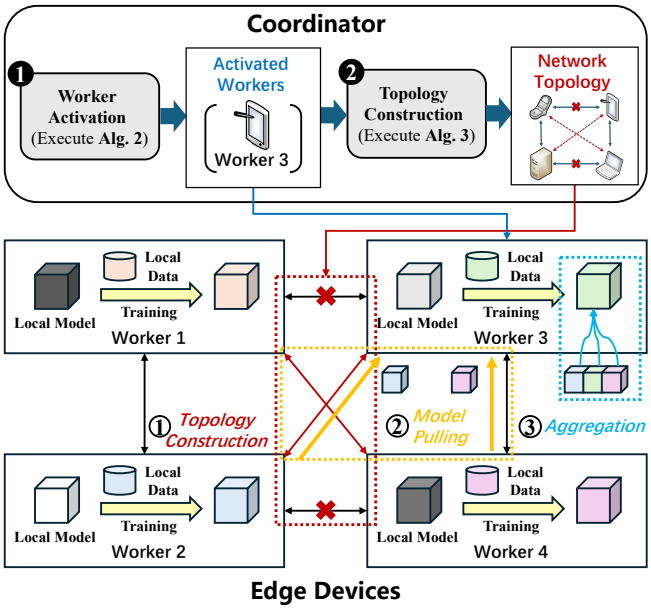


Fig. 1: Workflow Illustration of DySTop

$\mathcal{A}_t = \{v_3\}$. Next, the coordinator performs PTCA to construct the topology and notifies workers to adapt their neighbors. In our example, the directed communications of v_1-v_3 and v_2-v_4 are removed. Instead, directed communications v_2-v_3 and v_1-v_4 are added. Then v_3 pulls models from neighbors v_1 and v_4 to perform aggregation with its own local model. This workflow is executed in all rounds to solve the decoupled subproblems.

B. Lyapunov Optimization Based Problem Transformation

The core idea of Lyapunov optimization is transforming the staleness constraint into a queue stability problem. Following this idea, For each worker v_i, we create a staleness queue q_t^i to track the difference between its cumulative staleness up to round t and the given upper bound of staleness τ_{bound}. q_t^i is updated by the following recurrence relation:

$$q_{t+1}^i = \max\{q_t^i + \tau_t^i - \tau_{\text{bound}}, 0\}. \tag{14}$$

where q_0^i is initialized as 0. By employing Lyapunov optimization theory, we can obtain Theorem 2 about **P1** as follows,

Theorem 2 : *Approximating* **P1** *involves solving subproblem* **P2** *at each round* $t, \forall t \in \mathcal{R}$

$$\textbf{(P2)}: \min_{\boldsymbol{a}_t, \boldsymbol{c}_t} \sum_{v_i \in \mathcal{V}} q_t^i(\tau_t^i - \tau_{\text{bound}}) + V \cdot \hat{H}_t \tag{15}$$

$$\textbf{\textit{s.t.}} \quad \text{(12b)} - \text{(12f)},$$

where Eq. (15) *is a drift-plus-penalty function.* $\hat{H}_t$ *is computed as* $\frac{H_t}{H_{\text{Sync}}}$*, and* H_{Sync} *is the training time under synchronous updating, which is determined by the slowest worker.* V *serves as a trade-off factor, balancing the staleness queue stabilization against the minimization of training duration. The expected training duration obtained by solving* **P2** *has a minimum gap*

$$\sum_{t=1}^{T} \mathbb{E}\left[\hat{H}_t | \Theta_t\right] \leq \frac{T \cdot \Gamma}{V} + \hat{H}^*, \tag{16}$$

where $\hat{H}^*$ *represents the normalized training time under the optimal strategies, and* Γ *is a positive constant.*

Proof. See Appendix D. □

The optimization objective of **P2** is to independently determine the worker activation strategy $\boldsymbol{a}_t$ and the topology construction strategy $\boldsymbol{c}_t$ for each round, aiming to minimize the weighted sum of the workers' current staleness deviation and round duration.

C. Worker Activation Algorithm (WAA)

To address **P2**, we first propose the WAA to determine the worker activation strategy $\boldsymbol{a}_t$ at each round, as described in Alg. 2. The drift-plus-penalty function Eq. (15) provides the quantification of the trade-off between staleness control and single-round duration. In addition, Eq. (9) inspires us to prioritize workers with smaller H_t^i, which can help control staleness while reducing single-round duration H_t. Consequently, the target of WAA is to minimize Eq. (15) by activating an appropriate number of workers with smaller H_t^i. At the beginning of each round t, we initialize the active set $\mathcal{A}_t$ as empty, and set all $a_t^i \in \boldsymbol{a}_t$ to 0 (Line 1). Then, we sort workers in $\mathcal{V}$ in the ascending order of the sum of training time and transmission time H_t^i, and the set of sorted workers is denoted as $\mathcal{V}_{\text{sort}}$ (Line 2). Next, we sequentially add workers from $\mathcal{V}_{\text{sort}}$ to $\mathcal{A}_t$. For each progressively larger active set $\mathcal{A}_t$, up to the maximum worker count, we pre-update each worker's staleness and compute the corresponding value of Eq. (15), denoted as $S_{\mathcal{A}_t}$ (Lines 3-5). The minimum value $S_{\min}$ and the corresponding set $\mathcal{A}_t^*$ are recorded (Lines 6-8). Finally, we set each a_t^i whose worker is in $\mathcal{A}_t^*$ to 1 (Lines 9-10) and return the worker activation strategy $\boldsymbol{a}_t$ (Line 11).

It is worth noting that the computational complexity of WAA is dominated by the sorting operation in Line 2, reducing the decision overhead to $O(N \log N)$. This ensures the coordinator can make rapid scheduling decisions. To validate this scalability, we measured the execution time of WAA across varying numbers of workers. The results in Fig. 2 indicate that the running time remains in the millisecond range even with a large number of workers (*e.g.*, 1000), and as the number of workers increases, the growth in execution time aligns consistently with the theoretical $O(N \log N)$ curve.

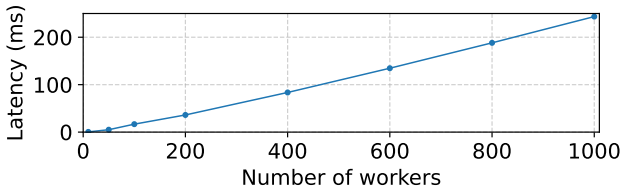


Fig. 2: Latencies of performing WAA with different numbers of workers.

D. Phase-aware Topology Construction Algorithm (PTCA)

We develop the PTCA to construct the network topology by selecting neighbors for each activated worker $v_i \in \mathcal{A}_t$. PTCA can adaptively partition the overall training process into

Algorithm 2 Worker Activation Algorithm (WAA)

Input: Staleness degree $\tau_t^i, \forall v_i \in \mathcal{V}$, value of virtual queue $q_t^i, \forall v_i \in \mathcal{V}$, sum of training time and transmission time $H_t^i, \forall v_i \in \mathcal{V}$

Output: The worker activation strategy $\boldsymbol{a}_t = \{a_t^i | v_i \in \mathcal{V}\}$ at round t

Initialize: $\mathcal{A}_t = \varnothing$, $a_t^i = 0, \forall a_t^i \in \boldsymbol{a}_t$, $S_{\min} = +\infty$;
$\mathcal{V}_{\text{sort}} \leftarrow$ sort $\mathcal{V}$ in the ascending order of H_t^i;
for $K = 1$ to N **do**
  $\mathcal{A}_t = \mathcal{A}_t \cup \mathcal{V}_{\text{sort}}[K]$;
  $S_{\mathcal{A}_t} \leftarrow$ calculate Eq. (15) for given $\mathcal{A}_t$;
  if $S_{\mathcal{A}_t} < S_{\min}$ **then**
    $S_{\min} = S_{\mathcal{A}_t}$;
    $\mathcal{A}_t^* = \mathcal{A}_t$;
for each $v_i \in \mathcal{A}_t^*$ **do**
  $a_t^i = 1$;
Return: $\boldsymbol{a}_t = \{a_t^i | v_i \in \mathcal{V}\}$.

two phases based on the model convergence status. Then, for each phase, we independently design the strategy for selecting neighbors based on different objectives.

1) Phase 1: Deal with Non-IID Data: In early training phases, non-IID data significantly slows down model convergence [33]. Motivated by [9], we pair workers with significantly different data distributions as neighbors. This can help their combined datasets better represent all classes, in turn mitigating model divergence (an example is shown in Fig. 3).

Earth Mover's Distance (EMD) [34] is a general measure to quantify the difference of two datasets' distributions. We denote the EMD between v_i and v_j's datasets $\mathcal{D}_i$ and $\mathcal{D}_j$ as

$$\text{EMD}(\mathcal{D}_i, \mathcal{D}_j) = \sum_{c_k \in \mathcal{C}} \| \frac{D_i^k}{D_i} - \frac{D_j^k}{D_j} \|. \tag{17}$$

where c_k is the k-th class of the global dataset $\mathcal{D}$ and the set $\mathcal{C}$ consists of classes of data in $\mathcal{D}$. D_i^k represents the size of the k-th class data in v_i's dataset $\mathcal{D}_i$. In addition, considering the worker mobility and worker-to-worker distance's influence on model transmission, we also take physical distance into account. Let $\text{Dist}(v_i, v_j)$ denote the physical distance between workers v_i and v_j. We define the priority for v_i to select v_j as a neighbor as follows

$$p_1(v_i, v_j) = \frac{\text{EMD}(\mathcal{D}_i, \mathcal{D}_j)}{\text{EMD}_{\max}} + \left(1 - \frac{\text{Dist}(v_i, v_j)}{\text{Dist}_{\max}}\right). \tag{18}$$

where $\text{EMD}_{\max}$ and $\text{Dist}_{\max}$ indicate the maximum EMD and distance among workers.

2) Phase 2: Diverse Neighbors and Staleness Control: As the training progresses, the impact of staleness increases [12], [24]. At the same time, diverse neighbor selections play a crucial role in enhancing accuracy [35]. Therefore, we design the following priority for v_i to select v_j as a neighbor

$$p_2(v_i, v_j) = \left(1 - \frac{\text{Pull}(v_i, v_j)}{t}\right) \frac{1}{1 + |\tau_i - \tau_j|}. \tag{19}$$

where $\text{Pull}(v_i, v_j)$ records the times of v_i pulling models from v_j. The term $\left(1 - \frac{\text{Pull}(v_i, v_j)}{t}\right)$ encourages the selection of workers with fewer pull requests, and the term $\frac{1}{1+|\tau_i - \tau_j|}$ ensures that the staleness gap between workers is not too large.

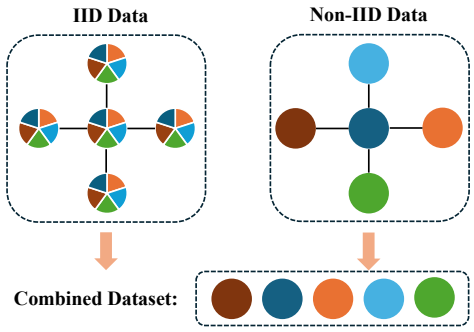


Fig. 3: A five-worker example of IID and non-IID scenarios, where each color represents a different class of the dataset. In the IID scenario (left), all classes are in equal proportions on all workers, while in the non-IID scenario (right), each worker has only one class. However, both scenarios' combined datasets of all workers represent the same distribution.

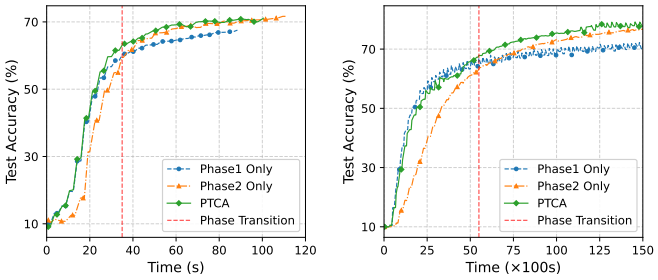


Fig. 4: Comparison of different priority-based topology construction strategies on non-IID datasets. *Left*: FMNIST; *Right*: CIFAR-10.

3) Algorithm design for PTCA: Inspired by [36], we introduce the relative training loss change rate $\Delta\bar{F}_t(K)$ to automatically trigger phase transitions, which is defined as follows,

$$\Delta\bar{F}_t(K) = \frac{1}{K} \sum_{k=1}^{K} \frac{\bar{F}_{t-k-1} - \bar{F}_{t-k}}{\bar{F}_{t-k-1}}, \tag{20}$$

where $\bar{F}_t$ is the average of the latest losses from all workers up to round t. $\Delta\bar{F}_t(K)$ represents the average loss change over the past K rounds. When $\Delta\bar{F}_t(K)$ is larger than the threshold κ, the model is actively learning features, and data heterogeneity is the primary challenge. The algorithm executes Phase 1 to align distributions. When $\Delta\bar{F}_t(K) < \kappa$, the training stabilizes, and the impact of stale models increases. The algorithm automatically transitions to Phase 2 to impose tighter staleness constraints and refine convergence.

The goal of our proposed PTCA is to determine the topology construction strategy $\boldsymbol{c}_t$ for each round t, which is formally described in Alg. 3. Let C_t^i denote the workers within v_i's communication range include itself at round t. To maximize bandwidth utilization subject to each worker's bandwidth budget, Alg. 3 iteratively selects the highest-priority neighbor for each activated worker $v_i \in \mathcal{A}_t$. This process continues until the bandwidth of workers in C_t^i is fully consumed. Specifically, we first initialize each worker's in-neighbor and out-neighbor sets as empty sets, and set all $c_t^{i,j} \in \boldsymbol{c}_t$ to 0. B_{tmp} is maintained

to track the cumulative bandwidth consumption of all workers (Line 1). Then, we calculate $\Delta\bar{F}_t(K)$ to determine the current phase, and sort C_t^i in descending order of corresponding priority for each activated worker (Lines 2-6). Next, for each iteration (Lines 7-22), each worker $v_i \in \mathcal{A}_t$ first confirms sufficient bandwidth resources for model pulling (Lines 9-10), then selects the top-ranked worker from sorted C_t^i with available bandwidth for model transmission, and updates neighbor sets and bandwidth consumption (Lines 11-18). PTCA terminates iteration and returns $\boldsymbol{c}_t$ when the total bandwidth consumption remains unchanged between two consecutive iterations (Lines 19-23).

To verify the validation of our phase-aware strategy, we compare our PTCA with three settings: 1) Phase 1-Only (PTCA that only uses $p_1(v_i, v_j)$), 2) Phase 2-Only (PTCA that only uses $p_2(v_i, v_j)$), 3) PTCA. Note that we set κ as 1×10^{-3}. We experiment with 100 workers to train a CNN model on FMNIST and ResNet-18 on CIFAR-10 with non-IID data. We use a red vertical dashed line to represent the phase transition point in PTCA, and Fig. 4 shows that PTCA can adaptively switch to Phase 2 when the convergence speed slows down in different scenarios. The results of the experiment demonstrate that Phase 1-Only trains faster initially but converges to lower accuracy, while Phase 2-Only trains slower initially but achieves higher accuracy. The entire PTCA method achieves both faster convergence and higher ultimate accuracy compared to both the above two settings.

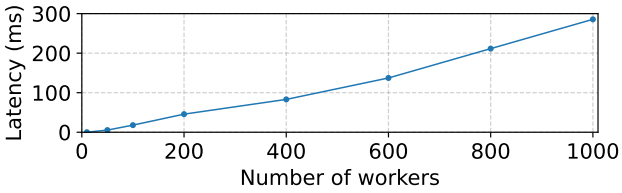


Fig. 5: Latencies of performing PTCA with different numbers of workers.

4) Discussion on the feasibility of PTCA at scale: The complexity of PTCA is approximately $O(N^2 \log N)$, stemming from neighbor sorting and iterative link establishment under bandwidth constraints. Scalability tests (as shown in Fig. 5) indicate that the execution time of PTCA grows consistently with this theoretical curve as the number of workers increases, but still remains in the millisecond range, ensuring the real-time topology construction per round. Moreover, although computing pairwise EMD between all workers is computationally intensive, it is executed as a pre-processing offline step. Consequently, this overhead does not affect the per-round efficiency of the actual training process.

VI. Simulation Experiment

In this section, we simulate a large-scale DFL in mobile edge network to verify the effectiveness of our proposed mechanism and algorithms.

A. System Setup

1) Environment: We perform our simulations on a workstation with one Intel(R) Xeon(R) CPU and 4 NVIDIA GeForce RTX 3090Ti GPUs with 24GB of video memory. The system environment is CUDA v12.1, and cuDNN v8.9.2, We use PyTorch to simulate a large-scale federated learning system with 100 heterogeneous workers distributed randomly across a 100m×100m region. Specifically, the model training time is calculated by $H_i = \zeta_i \frac{D_i}{|\xi_i|}$, where ζ_i is the training time of one mini-batch on worker v_i. We measure the actual training time for one batch 100 times on our experimental equipment. We then multiply this by a coefficient drawn from a normal distribution to simulate the varying computational capabilities of heterogeneous devices. For model transmission, we use Shannon Formula to simulate the transmission rate $r_t^{i,j} = b \cdot \log_2\left(1 + \frac{p_j \cdot g_t^{i,j}}{\gamma^2}\right)$. Referring to [37], [38], the channel gain $g_t^{i,j}$ is exponentially distributed with mean $G_0 \cdot \text{Dist}(v_i, v_j)^{-4}$, where $G_0 = -43$dB is the path-loss constant at a reference distance of 1 m. The transmit power p_j is set to range between 10dBm and 20dBm, then we multiply it by a fluctuation sampled from a normal distribution for each worker's transmit power. The noise power is set as $\gamma = 10^{-13}$ W and the bandwidth is set as $b = 1$MHz. To simulate the dynamic network topology caused by worker mobility, we adopt a random walk model [39]. At the end of each round, each worker updates its location by moving a random step (uniformly sampled from $[0, 2]$m) in a random

Algorithm 3 Phase-aware Topology Construction Algorithm (PTCA)

Input: Active set $\mathcal{A}_t$ and lists of workers in v_i's communication range $C_t^i, v_i \in \mathcal{V}$
Output: The topology construction strategy $\boldsymbol{c}_t = \{c_t^{i,j} | v_i, v_j \in \mathcal{V}\}$ at round t

1: **Initialize:** $\mathcal{N}_t^i = \hat{\mathcal{N}}_t^i = \varnothing$, $B_t^i = 0$, $\forall v_i \in \mathcal{V}$, $B_{\text{tmp}} = 0$, $c_t^{i,j} = 0, \forall c_t^{i,j} \in \boldsymbol{c}_t$;
2: Calculate $\Delta\bar{F}_t(K)$ by Eq. (20);
3: **if** $\Delta\bar{F}_t(K) > \kappa$ **then**
4: Sort C_t^i in the descending order of $p_1(v_i, v_j), \forall v_i \in \mathcal{A}_t$;
5: **else**
6: Sort C_t^i in the descending order of $p_2(v_i, v_j), \forall v_i \in \mathcal{A}_t$;
7: **while** true **do**
8: **for each** $v_i \in \mathcal{A}_t$ **do**
9: **if** $B_t^i + b > \hat{B}_t^i$ **then**
10: **continue**;
11: **while** $len(C_t^i) > 0$ **do**
12: **if** $B_t^{C_t^i[0]} + b > \hat{B}_t^{C_t^i[0]}$ **then**
13: $C_t^i = C_t^i - C_t^i[0]$;
14: **else**
15: $c_t^{i,C_t^i[0]} = 1$, $\mathcal{N}_t^i \cup \{C_t^i[0]\}$, $\hat{\mathcal{N}}_t^{C_t^i[0]} \cup \{v_i\}$;
16: $B_t^i = B_t^i + b$, $B_t^{C_t^i[0]} = B_t^{C_t^i[0]} + b$;
17: $C_t^i = C_t^i - C_t^i[0]$;
18: **break**;
19: **if** $B_{\text{tmp}} - \sum_{v_i \in \mathcal{V}} B_t^i = 0$ **then**
20: **break**;
21: **else**
22: $B_{\text{tmp}} = \sum_{v_i \in \mathcal{V}} B_t^i$
23: **Return:** $\boldsymbol{c}_t = \{c_t^i | v_i, v_j \in \mathcal{V}\}$.

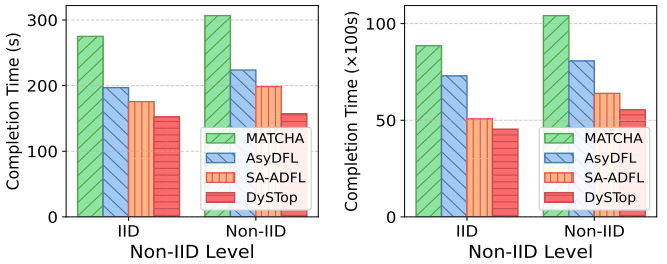


Fig. 6: Completion time varies with different non-IID levels on the two datasets. *Left*: FMNIST; *Right*: CIFAR-10.

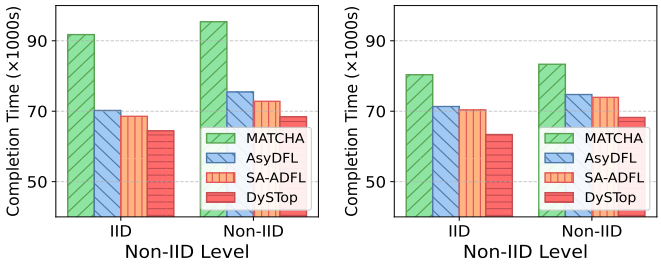


Fig. 7: Completion time varies with different non-IID levels on the two datasets. *Left*: CIFAR-100; *Right*: ImageNet-100.

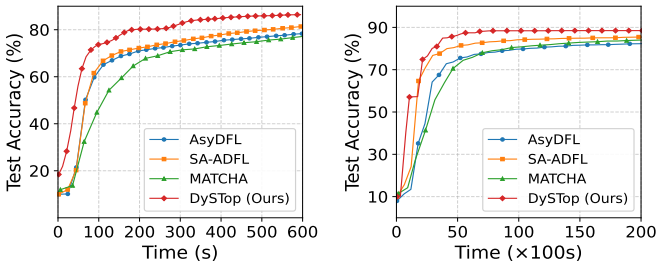


Fig. 8: Test Accuracy vs. Time on the two datasets (IID). *Left*: FMNIST; *Right*: CIFAR-10.

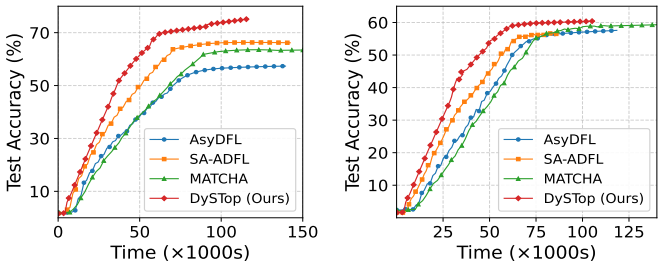


Fig. 9: Test Accuracy vs. Time on the two datasets (IID). *Left*: CIFAR-100; *Right*: ImageNet-100.

direction. Each worker's communication range is set to 20m, and it automatically disconnects from its original neighbors once they are out of range.

2) Datasets and Models: We select four classical benchmark datasets: FMNIST [40], CIFAR-10, CIFAR-100 [41], and ImageNet-100 (a subset of ImageNet [42]). FMNIST and CIFAR-10 are used for the 10-class classification tasks. CIFAR-100 and ImageNet-100 are utilized for the more challenging 100-class classification tasks. Following [43], we use the Dirichlet distribution $Dir(\phi)$ to simulate data heterogeneity. The parameter ϕ determines the level of distribution skew, with smaller values indicating higher heterogeneity. In our experiments, we set $\phi = 0.5$ to represent non-IID conditions, and use $\phi = 1.0$ to approximate an IID setting. We also choose four classical models, CNN [44], ResNet-18, ResNet-101 [45], and GoogleNet [46], which are customized to adapt to the corresponding datasets. The CNN is composed of two 5×5 convolution layers, two 2×2 max pooling layers, two fully-connected layers, and one softmax layer for output. ResNet-18 and ResNet-101 utilize residual blocks to facilitate deeper architectures, with the latter extending the depth to 101 layers for enhanced feature representation. GoogLeNet introduces additional architectural complexity through Inception modules, which parallelize multiple convolutional branches to capture multi-scale spatial features.

3) DFL Training settings: To replicate a realistic decentralized scenario, the parameters of each model are initialized independently on their respective workers, rather than being uniformly distributed by the coordinator. We ensure a fair comparison across all investigated DFL methods by employing a consistent SGD optimizer with a learning rate of 0.01 and a momentum of 0.9. The local training is conducted with a batch size of 32 and runs for 2 local epochs. The entire DFL training process lasts for 400 rounds.

4) Benchmarks: We compare our DySTop mechanism with the following DFL benchmarks.

- MATCHA [10]: A synchronous decentralized federated learning mechanism that divides the base topology into disjoint subgraphs and parallelizes inter-subgraphs communication. Considering the synchronous mechanism and sparse topology, we regard MATCHA as a lower bound of communication overhead.
- AsyDFL [15]: An asynchronous decentralized federated learning mechanism that incorporates neighbor selection and model push to handle non-IID data and edge heterogeneity. However, it lacks staleness control techniques.
- SA-ADFL [16]: An asynchronous decentralized federated learning mechanism that dynamically controls staleness, where each worker pushes its model to all neighbors.

5) Performance Metrics: We employ four key metrics to measure the performance of our proposed algorithms and the benchmark. 1) *Test Accuracy* is calculated as the ratio of correctly predicted data points by the model to the total number of data points. We record the test accuracy across all workers every 5 rounds and use the average of these recorded values as our evaluation metric. 2) *Communication Overhead* is calculated as the total bandwidth consumed by all workers to reach a certain accuracy. 3) *Completion time* is adopted to measure the time when convergence.

B. Evaluation Results

1) Performance Comparison with Benchmarks: Figs. 6-7 presents the completion time varying with different non-IID levels for training on FMINST, CIFAR-10, CIFAR-100 and ImageNet-100. As shown in these two figures, the completion time by the four algorithms on the four datasets increases with increasing non-IID level. For example, the completion time of DySTop takes 152.20s and 156.89s, increasing by 3.08%,

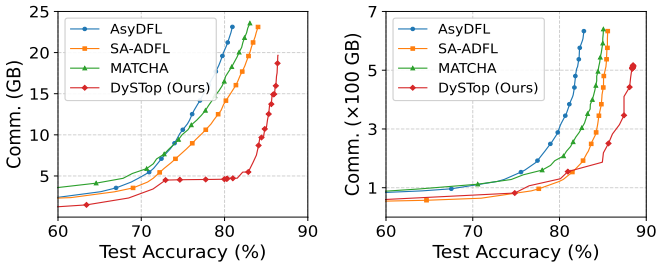


Fig. 10: Communication Overhead vs. Test Accuracy on the two datasets (IID). *Left*: FMNIST; *Right*: CIFAR-10.

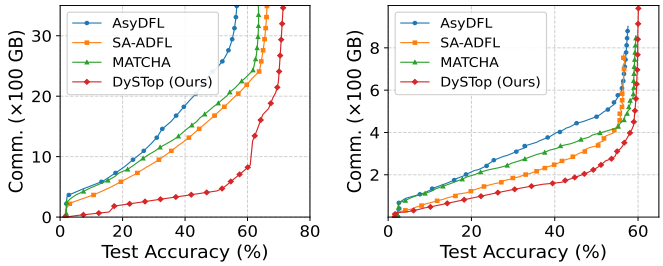


Fig. 11: Training Loss vs. Time on the two datasets (IID). *Left*: CIFAR-100; *Right*: ImageNet-100.

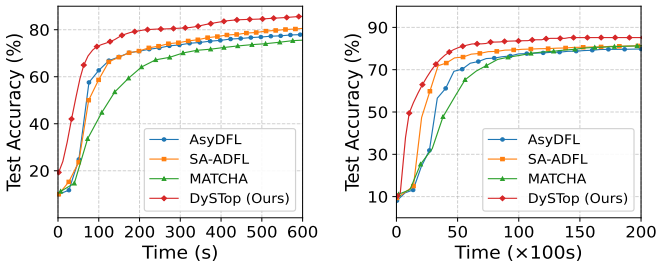


Fig. 12: Test Accuracy vs. Time on the two datasets (Non-IID). *Left*: FMNIST; *Right*: CIFAR-10.

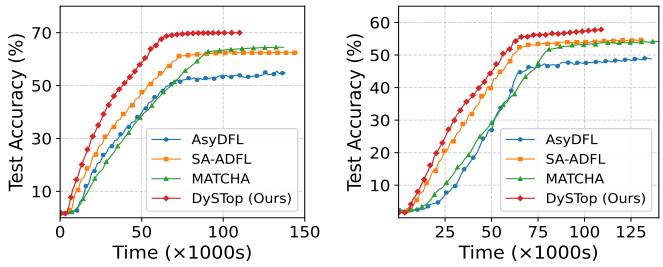


Fig. 13: Test Accuracy vs. Time on the two datasets (Non-IID). *Left*: CIFAR-100; *Right*: ImageNet-100.

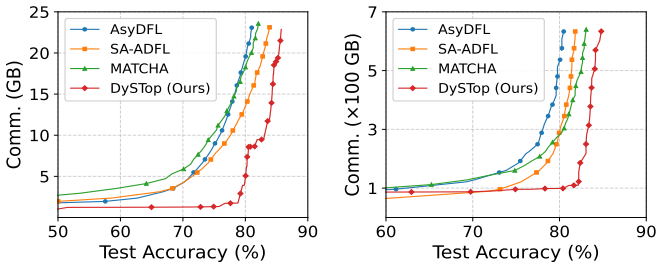


Fig. 14: Communication Overhead vs. Test Accuracy on the two datasets (Non-IID). *Left*: FMNIST; *Right*: CIFAR-10.

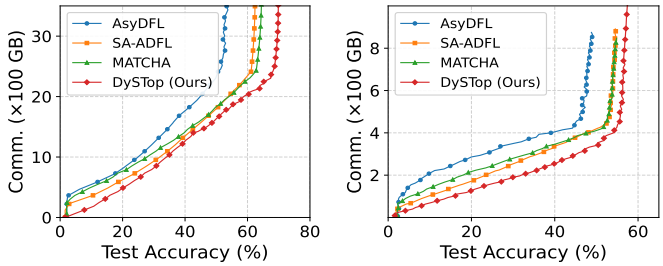


Fig. 15: Communication Overhead vs. Test Accuracy on the two datasets (Non-IID). *Left*: CIFAR-100; *Right*: ImageNet-100.

for training CNN on FMNIST, whereas AsyDFL, SA-ADFL, and MATCHA show larger increases of 13.78%, 13.14%, and 11.49%. The experiment results demonstrate that DySTop can better deal with non-IID data, since DySTop dynamically constructs a topology considering non-IID data among workers. Meanwhile, DySTop can always take the least completion time in comparison with the other algorithms. For example, in the non-IID settings, the completion times of DySTop, AsyDFL, SA-ADFL, and MATCHA are 68,417s, 75,466s, 72,826s and 95,402s, respectively, for training ResNet-101 on CIFAR-100. DySTop reduces the completion time by 9.34%, 6.05% and 28.29% compared with AsyDFL, SA-ADFL, and MATCHA, respectively.

Figs. 8-15 illustrate the performance of DySTop versus benchmarks in terms of test accuracy and communication overhead on both IID and non-IID data settings. Firstly, as the non-IID level increases, the performance of all algorithms decreases. For example, by Fig. 8 and Fig. 12, for training ResNet-18 on CIFAR-10, the model accuracy of DySTop reaches 88.47% in the IID scenario but drops slightly to 85.18% under the non-IID settings after 20,000s. Secondly, DySTop can achieve a faster convergence rate compared to AsyDFL, SA-ADFL, and MATCHA. For instance, as shown in Fig. 9, when taking 60,000s for training GoogLeNet on ImageNet-100, DySTop achieves the test accuracy of 58.51%, outperforming AsyDFL (48.43%), SA-ADFL (52.32%), and MATCHA (45.30%). Thirdly, DySTop consumes less communication overhead than asynchronous algorithms to achieve the required accuracy. For example, when training GoogleNet on ImageNet-100 (as shown in Fig. 15), the communication overhead of DySTop is 37.51% and 22.05% less than that of AsyDFL and SA-ADFL when reaching 40% accuracy. This is because DySTop effectively constructs a sparse topology, reducing the communication resources consumption while ensuring accuracy. Also, when comparing with the synchronous algorithm (MATCHA), DySTop still demonstrates higher communication resource efficiency, reducing 25.26% communication overhead. Although the synchronous algorithm has less communication frequency compared to asynchronous ones, its stability is severely challenged by worker mobility, while DySTop is designed to cope better with the dynamic movement of workers. Consequently, DySTop can achieve the best performance across all non-IID levels in non-convex scenarios.

2) The Impact of Staleness Constraint: Fig. 16 shows average staleness degree varying with different settings of staleness bound τ_{bound} in DySTop on the two datasets. As

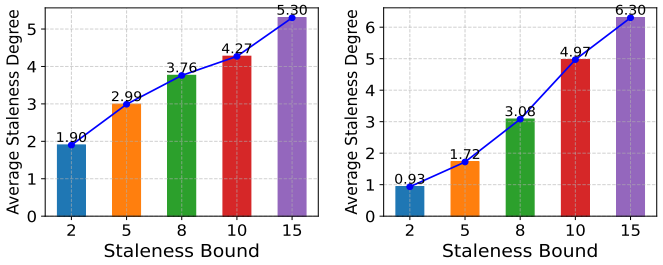


Fig. 16: Average staleness degree varies with different settings of staleness bound in DySTop on the two datasets. *Left*: CIFAR-10; *Right*: CIFAR-100.

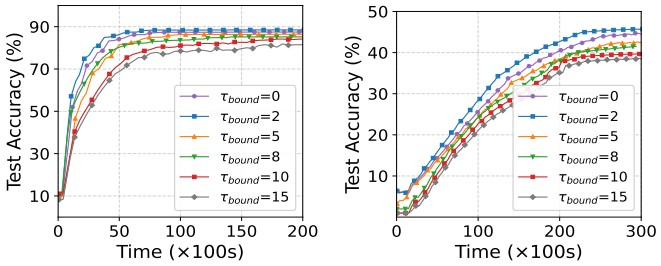


Fig. 17: Test Accuracy vs. Time on the two datasets with various values of τ_{bound}. *Left*: CIFAR-10; *Right*: CIFAR-100.

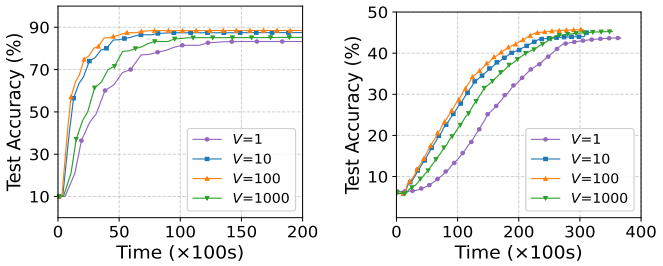


Fig. 18: Test Accuracy vs. Time on the two datasets with various values of V. *Left*: CIFAR-10; *Right*: CIFAR-100.

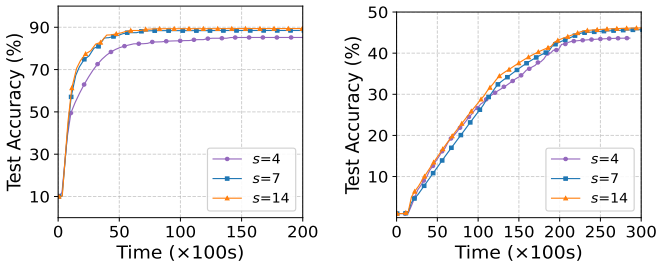


Fig. 19: Test Accuracy vs. Time on the two datasets with various values of s. *Left*: CIFAR-10; *Right*: CIFAR-100.

shown in this figure, DySTop can always control the staleness below τ_{bound}. That is, at the difference of staleness bounds of 2, 5, 8, 10, and 15, DySTop can control average staleness degrees as 1.90, 2.99, 3.76, 4.27, 5.30 on CIFAR-10 and 0.93, 1.72, 3.08, 4.97, 6.30 on CIFAR-100. Fig. 17 illustrates the test accuracies with respect to the training time of various values of $\tau_{\text{bound}} = \{0, 2, 5, 8, 10, 15\}$. Among these settings, $\tau_{\text{bound}} = 2$ yields the highest accuracy, *e.g.*, 45.69% when training a ResNet-18 on CIFAR-100. When staleness is too small (*e.g.*, $\tau_{\text{bound}} = 0$), the training process tends to become synchronous, leading to idle computation resources and reduced accuracy (*e.g.*, 43.46%). On the other hand, excessive staleness (*e.g.*, $\tau_{\text{bound}} = 15$) may cause excessive staleness and gradient divergence, which also decreases achieved accuracy (*e.g.*, 38.56%).

3) The Impact of Control Parameter: Fig. 18 illustrates the test accuracy with respect to the training time of the control parameter V. A larger value of V places more importance on reducing training time, while a smaller value concentrates on staleness control. Based on the results, we observe that the value of V influences the convergence rate. For instance, after 25,000s of training ResNet-18 on the CIFAR-100, the accuracy reaches to 45.29% at $V = 100$, compared to the accuracies of 39.78%, 43.68%, 43.46%, when $V = 1, 10, 1000$. Furthermore, the consumed training time of DySTop to achieve 40% accuracy is 25,770s, 18,843s, 17,514s, and 21,590s when $V = 1, 10, 100, 1000$. Note that similar results are also observed on CIFAR-10. The results show that $V = 100$ achieves the fastest convergence rate across various FL training tasks. This is because the Lyapunov penalty term is a dimensionless metric representing the relative training acceleration compared to synchronous updates. Therefore, a fixed control parameter appropriately tuned is effective across different scenarios.

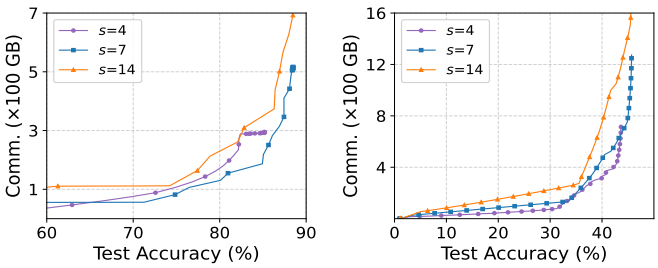


Fig. 20: Communication overhead vs. Test Accuracy on the two datasets with various numbers of neighbors. *Left*: CIFAR-10; *Right*: CIFAR-100.

4) The Impact of Neighbors Number: To further investigate the impact of varying neighbor numbers on worker training performance. We select different sample sizes s as the number of in-neighbors of each worker. The values of s are set as $\lceil \log_2 N \rceil = 7$, $\lceil \frac{\log_2 N}{2} \rceil = 4$ and $\lceil 2 \log_2 N \rceil = 14$. Fig. 19 shows the test accuracy with respect to the training time for varying values of sample size s. As shown in this figure, DySTop achieves model accuracies of 85.17%, 88.48%, and 89.37% on CIFAR-10, and 43.63%, 45.65%, and 46.03% on CIFAR-100, when $s = 4, 7, 14$, respectively. This indicates that a larger s value promotes the model convergence due to a denser network topology. However, this phenomenon exhibits diminishing returns, as the improvement in model accuracy slows with increasing values of s. In Fig. 20, we compare the communication overhead required for different sample sizes s to achieve a target test accuracy. As shown in the figure, pulling models from a larger number of neighboring workers increases communication overhead. For example, to achieve 80% and 40% accuracy respectively, DySTop consumes bandwidth of 179GB, 130GB, and 261GB for training on CIFAR-10, and 373GB, 474GB, and 788GB for training

TABLE II: Performance of DySTop under different phase transition thresholds κ on CIFAR-10 and CIFAR-100.

Threshold (κ)	**CIFAR-10**			**CIFAR-100**		
	Trans. Round	**Con. Time (s)**	**Acc. (%)**	**Trans. Round**	**Con. Time (s)**	**Acc. (%)**
1×10^{-2}	41	8,240	88.01	176	24,120	44.95
$\mathbf{1 \times 10^{-3}}$	**91**	**6,811**	**88.51**	**286**	**21,877**	**45.47**
1×10^{-4}	136	10,185	87.92	361	27,350	44.80

on CIFAR-100, when $s = 4, 7, 14$, respectively. This indicates that a larger s inevitably leads to a growth in communication overhead. Figs. 19-20 jointly illustrate that an appropriate number of neighbors can significantly reduce communication overhead while maintaining model accuracy.

5) The impact of threshold κ: In PTCA, we use a threshold κ to control when PTCA transitions from Phase 1 to Phase 2. A smaller κ allows PTCA to switch phases later, and vice versa. We explore the impact of the value of κ. Table II records the phase transition rounds, model convergence times, and final accuracy under different κ values. A higher threshold ($\kappa = 1 \times 10^{-2}$) switches to Phase 2 too early (*e.g.*, at round 41 for CIFAR-10). This shortens the rapid learning period of Phase 1, which slows down the overall convergence (increasing time to 8,240s). On the other hand, a lower threshold ($\kappa = 1 \times 10^{-4}$) delays the transition too much (*e.g.*, to round 361 for CIFAR-100). Staying in Phase 1 for too long also prolongs the convergence time (increasing time to 27,350s). $\kappa = 1 \times 10^{-3}$ provides the best balance, achieving the highest accuracy (88.51% for CIFAR-10 and 45.47% for CIFAR-100) with the shortest completion time. It is worth noting that the accuracy variation imposed by varying the value of κ is slight.

VII. System Implementation

In this section, we present a system implementation and deploy several algorithms on this real testbed platform.

A. Testbed Setup

Similar to [47], we implement a real testbed platform comprising one coordinator and 15 workers (labeled from v_1 to v_{15}). Fig. 21 illustrates a snapshot of the hardware devices in our testbed. Specifically, we set 4 Jetson Nano devices, 4 Jetson Orin NX devices, 3 Jetson Orin Nano devices, 3 Jetson Orin devices and 1 Jetson Xavier AGX device as heterogeneous workers. A Dell T640 Server is set as a coordinator. Detailed heterogeneous configurations are shown in Table III. Devices above are connected via wireless connection by a router, which is wired to the coordinator. We use mpi4p package to achieve the model and information transmission among workers and coordinators.

The testbed experiments are conducted with two lightweight models designed for resource-constrained devices (*i.e.*, SquezeeNet [48], MobileNet-V2 [49]) and two datasets (SVHN [50] and CIFAR-100 [41]). SquezeeNet is trained on SVHN, including 73,257 street view house numbers training images and 26,032 test images. To adapt SqueezeNet for the SVHN dataset, we adjust initial convolutional layer to have 96 output channels (with a 3×3 kernel, stride 1, and padding 1) and replace the original classifier. The new classifier now

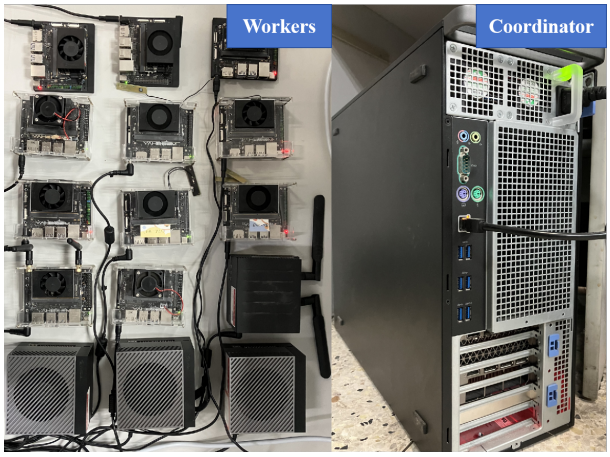


Fig. 21: A snapshot of the implemented testbed.

TABLE III: Real Testbed Platform Settings

Type	**Device**	**Component**	**Num.**
Worker	Jetson Nano	128-core Maxwell GPU	4
	Jetson Orin Nano	1024-core Ampere GPU	3
	Jetson Orin NX	1024-core Ampere GPU	4
	Jetson Orin	2048-core Ampere GPU	3
	Jetson Xavier AGX	512-core Volta GPU	1
Coordinator	Dell T640 Server	Intel Xeon Silver 4210R CPU	1

comprises a dropout layer, a 1×1 convolutional layer with 10 output channels (for the digit classes), a ReLU activation, and an adaptive average pooling layer to reduce feature maps to 1×1. These changes allow SqueezeNet to effectively process SVHN's 3-channel images and classify them into 10 distinct digit categories. MobileNet-V2 is trained on CIFAR-100, consisting of 60,000 images for training and 10,000 for the test across 100 classes. For the MobileNetV2 model, we modify its initial convolutional layer and adjust the stride of certain feature blocks to 1 to preserve more spatial information. Additionally, we reconfigure the final convolutional layer within the feature module and replace the classifier's output layer to match the 100 CIFAR-100 classes. Similar to simulation experiments, we use the Dirichlet distribution to partition the datasets, setting the non-IID levels to 0.5 and 1.0.

Additionally, to verify the superiority of DySTop on our real testbed platform, we use the same benchmarks and metrics as those in Section VI.

B. Testing Results

Fig. 22 presents the completion time to achieve certain accuracies varying with different non-IID levels on the two datasets SVHN and CIFAR-100, respectively. As shown in

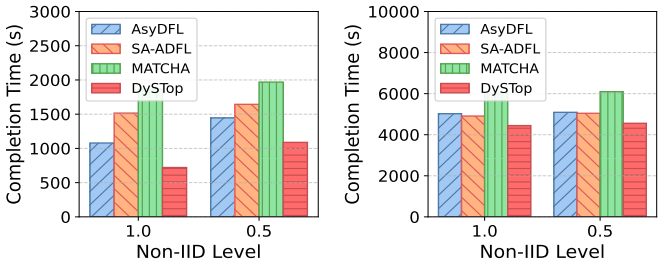


Fig. 22: Completion time varies with different non-IID levels on the two datasets. *Left*: SVHN; *Right*: CIFAR-100.

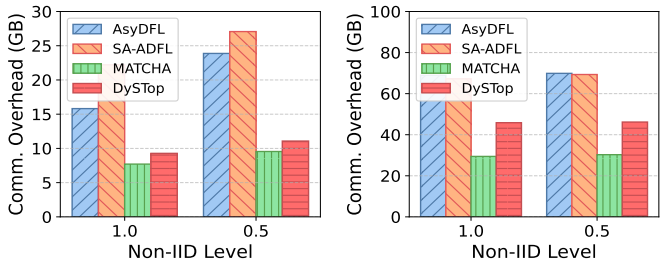


Fig. 23: Communication overhead varies with different non-IID levels on the two datasets. *Left*: SVHN; *Right*: CIFAR-100.

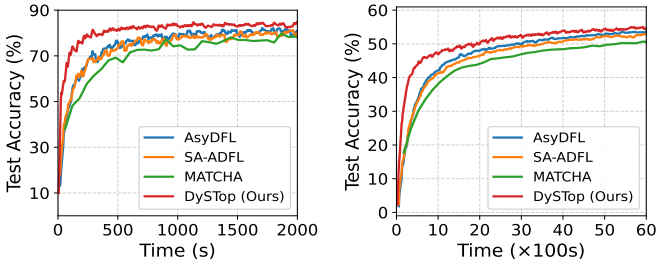


Fig. 24: Test Accuracy vs. Time on the two datasets (IID). *Left*: SVHN; *Right*: CIFAR-100.

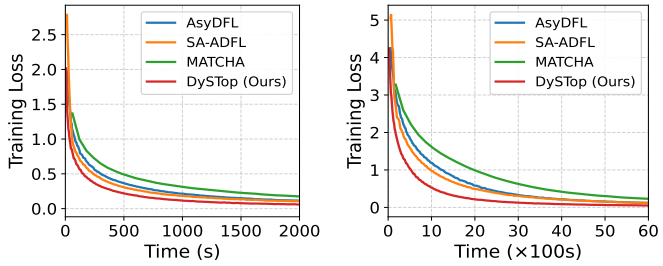


Fig. 25: Training Loss vs. Time on the two datasets (IID). *Left*: SVHN; *Right*: CIFAR-100.

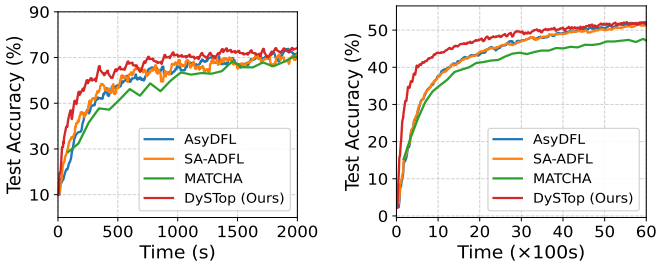


Fig. 26: Test Accuracy vs. Time on the two datasets (Non-IID). *Left*: SVHN; *Right*: CIFAR-100.

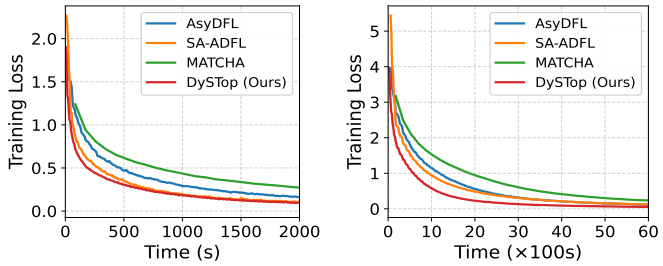


Fig. 27: Training Loss vs. Time on the two datasets (Non-IID). *Left*: SVHN; *Right*: CIFAR-100.

the figure, DySTop can always take the least completion time compared with the other algorithms. For example, for training SqueezeNet on SVHN in IID scenario, the completion time to achieve 80% accuracy of AsyDFL, SA-ADFL, MATCHA and DySTop is 1079s, 1516s, 1865s, and 721s, respectively. DySTop reduces the completion time by 33.17%, 52.44% and 61.34% compared with AsyDFL, SA-ADFL and MATCHA. In addition, when training SqueezeNet on SVHN in non-IID scenario, the completion time to achieve 70% accuracy of AsyDFL, SA-ADFL, MATCHA and DySTop is 1445s, 1644s, 1969s and 1088s, respectively. DySTop reduces the completion time of AsyDFL, SA-ADFL and MATCHA by 24.71%, 33.82% and 44.73%. The experiment results demonstrate that DySTop can better adapt to both the IID and non-IID scenarios with less completion time for model convergence.

Fig. 23 compares the communication overhead by different algorithms to achieve certain accuracies. As shown in the figure, the DySTop consumes the least communication overhead compared with the other two asynchronous algorithms. Furthermore, for training MobileNet-V2 on IID CIFAR-100 to 50% accuracy, the DySTop significantly reduces the communication overhead by 33.57% and 31.82%, compared with AsyDFL and SA-ADFL. When training MobileNet-V2 on non-IID CIFAR-100, the communication overhead of DySTop is 33.92% and 33.38% less than that of AsyDFL and SA-ADFL. From Figs. 22 and 23, although asynchronous DFL approaches inherently have a much higher communication frequency, DySTop still achieves faster model convergence with tolerable communication overhead.

Figs. 24-27 show training process for IID and non-IID scenarios, respectively. DySTop can always achieve a faster convergence rate in both IID and non-IID settings, compared with AsyDFL, SA-ADFL and MATCHA. For example, as shown in Fig. 24, when training SqueezeNet on SVHN with the IID setting, after 2000s of training, DySTop achieves a stable test accuracy of 84.49%, outperforming AsyDFL (82.01%), SA-ADFL (80.69%) and MATCHA (79.37%). For the non-IID scenario shown in Fig. 26, after 6000s training in MobileNet-V2 on CIFAR-100, DySTop achieves a stable test accuracy of 52.18%, outperforming AsyDFL (51.84%), SA-ADFL (51.21%) and MATCHA (46.61%). The experiment results demonstrate that DySTop can still outperform under a real test-bed platform.

Remark 1 : *Compared to simulations, testbed experiments show different relative performance among mechanisms. MATCHA performs worse in the testbed mainly due to greater device heterogeneity and fewer workers, which intensify the*

straggler effect and delay model aggregation, thus reducing training efficiency.

VIII. Conclusion

In this paper, we have proposed DySTop, an innovative asynchronous decentralized federated learning (ADFL) mechanism designed to overcome the limitations of the existing synchronous DFL and asynchronous DFL mechanisms. In each round, DySTop dynamically activates multiple workers, each selecting a subset of its neighbors and pulling their models for aggregation. We have proved the convergence of DySTop, quantitatively revealing how key factors, such as the maximum staleness, worker activation frequency, and data distribution, affect the convergence bound of ADFL. Guided by the analysis, we have designed a worker activation algorithm to control staleness, and a phase-aware topology construction algorithm to reduce communication overhead and handle non-IID data. Extensive experiments on both simulation and testbeds have demonstrated the superiority of DySTop.

References

[1] Y. Hu, Q. Jia, Y. Yao, Y. Lee, M. Lee, C. Wang, X. Zhou, R. Xie, and F. R. Yu, "Industrial internet of things intelligence empowering smart manufacturing: a literature review," *IEEE Internet of Things Journal*, vol. 11, no. 11, pp. 19 143–19 167, 2024.

[2] J. Zhou, Y. Shen, L. Li, C. Zhuo, and M. Chen, "Swarm intelligence-based task scheduling for enhancing security for IoT devices," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 42, no. 6, pp. 1756–1769, 2023.

[3] X. Hou, J. Zhou, L. Li, P. Cong, Z. Wu, and M. Chen, "Latency and reliability-aware dynamic task offloading and scheduling for energy-harvesting systems in mobile edge computing," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, pp. 1–1, 2025, early Access.

[4] B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. Arcas, "Communication-efficient learning of deep networks from decentralized data," in *Proceedings of IEEE International Conference on Artificial Intelligence and Statistics*, 2017, pp. 1273–1282.

[5] S. Wang, T. Tuor, T. Salonidis, K. K. Leung, C. Makaya, T. He, and K. Chan, "Adaptive federated learning in resource constrained edge computing systems," *IEEE Journal on Selected Areas in Communications*, vol. 37, no. 6, pp. 1205–1221, 2019.

[6] X. Li, K. Huang, W. Yang, S. Wang, and Z. Zhang, "On the convergence of fedavg on non-iid data," in *Proceedings of International Conference on Learning Representations*, 2020, pp. 1–26.

[7] A. G. Roy, S. Siddiqui, S. Pölsterl, N. Navab, and C. Wachinger, "Braintorrent: A peer-to-peer environment for decentralized federated learning," 2019. [Online]. Available: https://arxiv.org/abs/1905.06731

[8] Z. Tang, S. Shi, B. Li, and X. Chu, "Gossipfl: A decentralized federated learning framework with sparsified and adaptive communication," *IEEE Transactions on Parallel and Distributed Systems*, vol. 34, no. 3, pp. 909–922, 2022.

[9] A. Bellet, A.-M. Kermarrec, and E. Lavoie, "D-cliques: Compensating for data heterogeneity with topology in decentralized federated learning," in *Proceedings of International Symposium on Reliable Distributed Systems*, 2022, pp. 1–11.

[10] J. Wang, A. K. Sahu, Z. Yang, G. Joshi, and S. Kar, "Matcha: Speeding up decentralized sgd via matching decomposition sampling," in *Proceedings of Indian Control Conference*, 2019, pp. 299–300.

[11] H. Xu, M. Chen, Z. Meng, Y. Xu, L. Wang, and C. Qiao, "Decentralized machine learning through experience-driven method in edge networks," *IEEE Journal on Selected Areas in Communications*, vol. 40, no. 2, pp. 515–531, 2022.

[12] Y. Liao, Y. Xu, H. Xu, L. Wang, and C. Qian, "Adaptive configuration for heterogeneous participants in decentralized federated learning," in *Proceedings of IEEE International Conference on Computer Communications*, 2023, pp. 1–10.

[13] J. Cao, Z. Lian, W. Liu, Z. Zhu, and C. Ji, "Hadfl: Heterogeneity-aware decentralized federated learning framework," in *Proceedings of ACM/IEEE Design Automation Conference*, 2021, pp. 1–6.

[14] M. Chen, Y. Xu, H. Xu, and L. Huang, "Enhancing decentralized federated learning for non-iid data on heterogeneous devices," in *Proceedings of IEEE International Conference on Data Engineering*, 2023, pp. 2289–2302.

[15] Y. Liao, Y. Xu, H. Xu, M. Chen, L. Wang, and C. Qiao, "Asynchronous decentralized federated learning for heterogeneous devices," *IEEE/ACM Transactions on Networking*, vol. 32, no. 5, pp. 4535–4550, 2024.

[16] Q. Ma, J. Liu, Q. Jia, X. Zhou, Y. Hu, and R. Xie, "Dynamic staleness control for asynchronous federated learning in decentralized topology," in *Proceedings of International Conference on Wireless Artificial Intelligent Computing Systems and Applications*, 2024, pp. 99–117.

[17] A. Asheralieva, D. Niyato, and X. Wei, "Dynamic distributed model compression for efficient decentralized federated learning and incentive provisioning in edge computing networks," *IEEE Transactions on Mobile Computing*, vol. 24, no. 7, pp. 6293–6314, 2025.

[18] Y. Cui, K. Cao, J. Zhou, and T. Wei, "Optimizing training efficiency and cost of hierarchical federated learning in heterogeneous mobile-edge cloud computing," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 42, no. 5, pp. 1518–1531, 2023.

[19] L. Fu, N. Cheng, X. Wang, R. Sun, N. Lu, Z. Su, and C. Li, "Dta-rl: Dynamic topology adaptive reinforcement learning approach for task offloading in mobile edge computing," in *Proceedings of IEEE Global Communications Conference*, 2024, pp. 3334–3339.

[20] C. Xie, S. Koyejo, and I. Gupta, "Asynchronous federated optimization," 2020. [Online]. Available: https://arxiv.org/abs/1903.03934

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

[22] L. Palmieri, C. Boldrini, L. Valerio, A. Passarella, and M. Conti, "Impact of network topology on the performance of decentralized federated learning," *Computer Networks*, vol. 253, p. 110681, 2024.

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

[24] S. Sun, Z. Zhang, Q. Pan, M. Liu, Y. Wang, T. He, Y. Chen, and Z. Wu, "Staleness-controlled asynchronous federated learning: Accuracy and efficiency tradeoff," *IEEE Transactions on Mobile Computing*, vol. 23, no. 12, pp. 12 621–12 634, 2024.

[25] Y. Chen, Y. Ning, M. Slawski, and H. Rangwala, "Asynchronous online federated learning for edge devices with non-iid data," in *Proceedings of IEEE International Conference on Big Data*, 2020, pp. 15–24.

[26] Q. Ma, J. Zhou, X. Hou, J. Liu, H. Xu, J. Miao, and Q. Jia, "Air-fedga: A grouping asynchronous federated learning mechanism exploiting over-the-air computation," in *Proceedings of IEEE International Parallel and Distributed Processing Symposium*, 2025, pp. 1–12.

[27] Q. Ma, X. Song, J. Zhou, H. Wang, Y. Liao, J. Liu, and H. Xu, "Asynchronous federated learning over non-iid data via over-the-air computation," *IEEE Transactions on Networking*, vol. 34, pp. 2165–2180, 2026.

[28] S. Zheng, Q. Meng, T. Wang, W. Chen, N. Yu, Z.-M. Ma, and T.-Y. Liu, "Asynchronous stochastic gradient descent with delay compensation," in *Proceedings of International Conference on Machine Learning*, 2017, pp. 4120–4129.

[29] H. Zhu, J. Kuang, M. Yang, and H. Qian, "Client selection with staleness compensation in asynchronous federated learning," *IEEE Transactions on Vehicular Technology*, vol. 72, no. 3, pp. 4124–4129, 2022.

[30] J. Tsitsiklis, D. Bertsekas, and M. Athans, "Distributed asynchronous deterministic and stochastic gradient optimization algorithms," *IEEE Transactions on Automatic Control*, vol. 31, no. 9, pp. 803–812, 1986.

[31] Q. Ma, Y. Xu, H. Xu, J. Liu, and L. Huang, "Feduc: A unified clustering approach for hierarchical federated learning," *IEEE Transactions on Mobile Computing*, vol. 23, no. 10, pp. 9737–9756, 2024.

[32] L. Nguyen, P. H. Nguyen, M. Dijk, P. Richtárik, K. Scheinberg, and M. Takác, "Sgd and hogwild! convergence without the bounded gradients assumption," in *Proceedings of International Conference on Machine Learning*, 2018, pp. 3750–3758.

[33] X. Wang, Y. Wang, M. Yang, F. Li, X. Wu, L. Fan, and S. He, "Fedsiamda: Dual-aggregated federated learning via siamese network for non-iid data," *IEEE Transactions on Mobile Computing*, vol. 24, no. 2, pp. 985–998, 2025.

[34] Y. Rubner, C. Tomasi, and L. J. Guibas, "The earth mover's distance as a metric for image retrieval," *International journal of computer vision*, vol. 40, pp. 99–121, 2000.

[35] M. De Vos, S. Farhadkhani, R. Guerraoui, A.-M. Kermarrec, R. Pires, and R. Sharma, "Epidemic learning: Boosting decentralized learning

with randomized communication," *Advances in Neural Information Processing Systems*, vol. 36, pp. 36 132–36 164, 2023.
[36] J. Liu, H. Xu, L. Wang, Y. Xu, C. Qian, J. Huang, and H. Huang, "Adaptive asynchronous federated learning in resource-constrained edge computing," *IEEE Transactions on Mobile Computing*, vol. 22, no. 2, pp. 674–690, 2023.
[37] F. Wang, S. Cai, and V. K. N. Lau, "Decentralized dnn task partitioning and offloading control in mec systems with energy harvesting devices," *IEEE Journal of Selected Topics in Signal Processing*, vol. 17, no. 1, pp. 173–188, 2023.
[38] Z. Wang, Z. Zhang, Y. Tian, Q. Yang, H. Shan, W. Wang, and T. Q. S. Quek, "Asynchronous federated learning over wireless communication networks," *IEEE Transactions on Wireless Communications*, vol. 21, no. 9, pp. 6961–6978, 2022.
[39] H. Hu, Q. Wang, R. Q. Hu, and H. Zhu, "Mobility-aware offloading and resource allocation in a mec-enabled iot network with energy harvesting," *IEEE Internet of Things Journal*, vol. 8, no. 24, pp. 17 541–17 556, 2021.
[40] Y. LeCun, L. Bottou, Y. Bengio, P. Haffner *et al.*, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.
[41] A. Krizhevsky, "Learning multiple layers of features from tiny images," pp. 32–33, 2009. [Online]. Available: https://www.cs.toronto.edu/~kriz/learning-features-2009-TR.pdf
[42] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A. C. Berg, and L. Fei-Fei, "Imagenet large scale visual recognition challenge," *International Journal of Computer Vision*, vol. 115, no. 3, pp. 211–252, 2015.
[43] T. Lin, L. Kong, S. U. Stich, and M. Jaggi, "Ensemble distillation for robust model fusion in federated learning," in *Proceedings of International Conference on Neural Information Processing Systems*, 2020, pp. 2351 – 2363.
[44] S. Shalev-Shwartz and S. Ben-David, *Understanding Machine Learning: From Theory to Algorithms*. USA: Cambridge University Press, 2014.
[45] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778.
[46] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, "Going deeper with convolutions," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2015, pp. 1–9.
[47] J. Ke, J. Zhou, D. Meng, Y. Zeng, Y. Shi, X. Qu, and S. Guo, "Jcsrc: Joint client selection and resource configuration for energy-efficient multi-task federated learning," *IEEE Transactions on Computers*, vol. 74, no. 12, pp. 4094–4108, 2025.
[48] F. N. Iandola, S. Han, M. W. Moskewicz, K. Ashraf, W. J. Dally, and K. Keutzer, "Squeezenet: Alexnet-level accuracy with 50x fewer parameters and <0.5mb model size," 2016. [Online]. Available: https://arxiv.org/abs/1602.07360
[49] A. G. Howard, M. Zhu, B. Chen, D. Kalenichenko, W. Wang, T. Weyand, M. Andreetto, and H. Adam, "Mobilenets: Efficient convolutional neural networks for mobile vision applications," 2017. [Online]. Available: https://arxiv.org/abs/1704.04861
[50] I. J. Goodfellow, Y. Bulatov, J. Ibarz, S. Arnoud, and V. Shet, "Multi-digit number recognition from street view imagery using deep convolutional neural networks," 2014. [Online]. Available: https://arxiv.org/abs/1312.6082
[51] 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 *Proceedings of International Conference on Neural Information Processing Systems*, 2017, p. 5336–5346.

Yizhou Shi received the B.S. degree in Cyberspace Security from the Nanjing University of Science and Technology, Nanjing, China, in 2024. He is currently pursuing the Ph.D. degree with the Nanjing University of Science and Technology, Nanjing, China. His research interests are in the area of edge computing and federated learning.

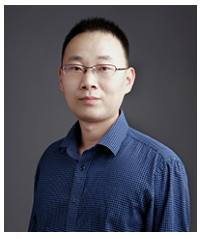
Qianpiao Ma received the B.S. degree in computer science and the Ph.D. degree in computer software and theory from the University of Science and Technology of China, Hefei, China, in 2014 and 2022, respectively. He is currently an Associate Professor at the School of Computer Science and Engineering, Nanjing University of Science and Technology, Nanjing, China. He has published more than 20 papers in famous journals and conferences, including IEEE JSAC, IEEE TMC, INFOCOM and IPDPS. His primary research interests include federated learning, mobile-edge computing, and distributed machine learning.

Yan Xu received the B.S. degree in Computer Science and Technology from the Nanjing University of Chinese Medicine, Nanjing, China, in 2025. She is currently pursuing the Ph.D. degree with the Nanjing University of Science and Technology, Nanjing, China. Her research interests are in the area of edge computing and federated learning.

Junlong Zhou is an Associate Professor at the School of Computer Science and Engineering, Nanjing University of Science and Technology, China. His research interests include edge computing, cloud computing, and embedded systems, where he has published 120 papers, including more than 40 in premier IEEE/ACM Transactions. He has been a Subject Area Editor for Journal of Systems Architecture and an Associate Editor for Journal of Circuits, Systems, and Computers and IET Cyber-Physical Systems: Theory & Applications. He received the Best Paper Award from IEEE iThings2020, CPSCom2022, and ICITES2024.

Ming Hu received his B.E. and Ph.D. degrees from the Software Engineering Institute, East China Normal University, Shanghai, China, in 2017 and 2022, respectively. He is currently a research scientist at Singapore Management University (SMU). Previously, he was a research fellow at Nanyang Technological University (NTU), Singapore. His research interests include the area of design automation of cyber-physical systems, federated learning, trustworthy AI, and software engineering.

Yunming Liao received B.S. degree in 2020 from the University of Science and Technology of China. He is currently pursuing his Ph.D. degree in the School of Computer Science and Technology, University of Science and Technology of China. His research interests include mobile edge computing, federated learning, and edge intelligence.



Hongli Xu (Member, IEEE) received the B.S. degree in computer science and the Ph.D. degree in computer software and theory from the University of Science and Technology of China, China, in 2002 and 2007, respectively. He is currently a Professor with the School of Computer Science and Technology, University of Science and Technology of China. He has published more than 100 papers in famous journals and conferences, including IEEE/ACM TNET, IEEE TMC, IEEE TPDS, INFOCOM, and ICNP. He has held more than 30 patents. His main research interests include software-defined networks, edge computing, and the Internet of Things. He was awarded the Outstanding Youth Science Foundation of NSFC in 2018. He has won the best paper award or the best paper candidate in several famous conferences.