Internet of Things 34 (2025) 101771

Contents lists available at ScienceDirect

Internet of Things

journal homepage: www.elsevier.com/locate/iot

Research article

FedAWT: Adaptive federated learning via dynamic epoch adjustment for heterogeneous clients in ad-hoc networks

Geonhee Kim [a], Seunghyun Park [b], Hyunhee Park [a,*]

[a] Department of Information and Communication Engineering, Myongji University, 116, Myeongji-ro, Cheoin-gu, Yongin-si, Gyeonggi-do 17058, Republic of Korea

[b] Department of Computer Engineering, Hansung University, 116, Samseongyo-ro 16-gil, Seongbuk-gu, Seoul, 02876, Republic of Korea

ARTICLE INFO

Dataset link: https://github.com/ghkim919/FedAWT

Keywords:
Federated learning
Decentralized federated learning
Heterogeneity
Internet of Things

ABSTRACT

Recent advancements in IoT devices have enabled on-device processing for tasks such as real-time inference and autonomous control. However, federated learning (FL) in such environments remains challenging due to limited computation resources, unstable connectivity, and severe data and system heterogeneity. To overcome these challenges, this paper proposes FedAWT (Federated learning with Adaptive Weighted Tree), a decentralized FL framework designed for ad-hoc IoT networks. A latency-aware system model is constructed to reflect communication delay, bandwidth constraints, and variations in device-level computational performance. FedAWT introduces two core mechanisms. The first is a dynamic epoch adjustment method that allocates training epochs according to local convergence status and training stability. The second is a minimum spanning tree based communication strategy that reduces model transfer latency. Experimental results confirm that FedAWT improves test accuracy by up to 33.86 % compared to baseline methods in heterogeneous environments. These results demonstrate the effectiveness of FedAWT in enhancing convergence stability and training efficiency under resource-constrained and decentralized environment.

1. Introduction

The Internet of Things (IoT) has witnessed a considerable surge in demand, which has led to substantial advancements in sensor and embedded system technologies. Consequently, research leveraging these technologies has attracted considerable attention [1–4]. IoT devices are evolving beyond simple data collection. They now perform complex tasks such as video analysis, autonomous control, and real-time decision making. Deep learning technology plays a key role in this shift. It enables data preprocessing and inference directly on devices. As a result, IoT systems can meet real-time processing requirements at the edge [5,6].

In such an environment, federated learning (FL) is a promising approach for ensuring both privacy and network efficiency [7,8]. FL enables distributed training by sharing only model parameters among devices. However, centralized FL (CFL) still relies on a central server for model aggregation. This centralized architecture has structural limitations, such as vulnerability to single points of failure (SPoF) and network bottlenecks. In IoT environments with limited communication infrastructure, issues like update delays, packet loss, and client dropouts are common. As an alternative, decentralized federated learning (DFL) has been proposed. DFL enables model exchange and training through direct communication between neighboring nodes [9]. While DFL alleviates the load on the central server, ad-hoc IoT networks still face physical constraints, including non-standard topologies, inter-node communication delays, and connection instability. Moreover, DFL is more sensitive to system and data heterogeneity during training.

* Corresponding author.
E-mail addresses: ghkim919@mju.ac.kr (G. Kim), sp@hansung.ac.kr (S. Park), hhpark@mju.ac.kr (H. Park).

https://doi.org/10.1016/j.iot.2025.101771
Received 11 August 2025; Received in revised form 13 September 2025; Accepted 22 September 2025
Available online 30 September 2025

To address these issues, this paper proposes FedAWT (Federated learning with Adaptive Weighted Tree), a decentralized federated learning framework applicable to ad-hoc IoT networks. FedAWT considers the communication properties, computational resources, and heterogeneity among IoT devices, and presents the following main contributions

- Define a system model that factors in the computing performance, communication latency, and bandwidth constraints of IoT devices, and construct a minimum spanning tree (MST) to minimize communication latency when sharing models, thereby generating optimal communication paths.
- By dynamically adjusting the number of training epochs based on the convergence degree of the model during training, even devices with limited resources can effectively participate in training, thereby reducing model bias caused by data heterogeneity.
- The training completion rate for the assigned epochs of the devices participating in training is calculated, and based on this, the epochs are adjusted for all participating devices with computational capacity deviations to reduce client dropouts caused by system heterogeneity.

This paper proposes a DFL framework based on ad-hoc networks, which considers both communication efficiency and training performance. It enables stable global model convergence even in the presence of system and data heterogeneity across clients. This paper is organized as follows. Section 2 reviews prior research on FL, DFL, and heterogeneity issues, and discusses their relation to the proposed method. Section 3 introduces the system model of FedAWT. Section 4 presents the MST-based communication optimization and dynamic epoch adjustment techniques. Section 5 discusses the experimental results. Finally, Section 6 concludes this paper.

2. Related works

This section describes FL and DFL and introduces existing research on reducing data and system heterogeneity.

2.1. Federated learning

Traditional deep learning typically relies on centralized training, where raw data is uploaded to a central server for model training. While this enables effective global model learning, it raises privacy concerns and scalability issues [8,10]. To overcome these limitations, Google proposed federated learning (FL) in 2016 [7]. In FL, clients train local models using private data and share only model parameters or gradients with a central server. This approach preserves privacy and reduces communication overhead. FL has been adopted in real-world applications such as predictive keyboards and recommendation systems, and is expanding into domains like IoT, autonomous vehicles, and healthcare [11–13]. FedAvg [7], the most widely used FL algorithm, updates the global model via simple averaging. Subsequent studies have proposed improvements to address client heterogeneity, non-independent and identically distributed (non-IID) data, and asynchronous updates [14–21]. In IoT environments, applying FL is more challenging due to limited computational and energy resources [22]. Many IoT devices lack dedicated processors, leading to higher energy consumption and operational delays when executing complex training tasks [23]. Furthermore, dynamic wireless connectivity can cause frequent packet losses and communication failures, making continuous server–client interaction unstable. These constraints necessitate lightweight models, energy-aware training, and decentralized FL frameworks to ensure training stability in such environments.

2.2. Decentralized federated learning

DFL enables model training without relying on a central server. Instead, clients exchange models directly with their neighbors [20]. Each client integrates its neighbors' models into its own, allowing global training to emerge from local interactions. This framework improves fault tolerance against server failures and enables training in environments with limited infrastructure [21,24]. Representative approaches include D-PSGD [20], which averages model parameters among neighbors, and GossipFL [21], which sparsifies communication by interacting with a single neighbor per round. These methods enhance communication efficiency and scalability, especially in resource-constrained scenarios. However, most existing DFL studies assume fully connected or static network topologies [20,21]. In real-world ad-hoc networks, links between nodes change dynamically, and devices may drop out due to interference or power issues. These factors require DFL systems to account for varying communication delays, energy constraints, and computational heterogeneity. To be practical, DFL frameworks must incorporate dynamic topology adaptation, resource-aware scheduling, and lightweight aggregation mechanisms that reflect real-world constraints. Recent researches deal with such dynamic environments. [25] proposed a method where unmanned aerial vehicle(UAV)s act as local aggregators, jointly optimizing the UAVs' flight paths and the transmission power of ground IoT device clients to minimize the system's total energy consumption. Additionally, [26] proposes a hierarchical approach utilizing edge servers to minimize overall learning latency through client grouping and resource allocation. [27] proposes a parameter synchronization strategy based on a 2D ring network topology. Aforementioned research focuses on improving transmission performance by addressing energy efficiency during data transmission and client selection. FedAWT focuses on optimizing the physical communication paths between devices instead of client selection. Specifically, it solves the communication problem in DFL by using an MST that minimizes communication delay per round, reflecting the network's dynamic topology and the communication performance of each node.

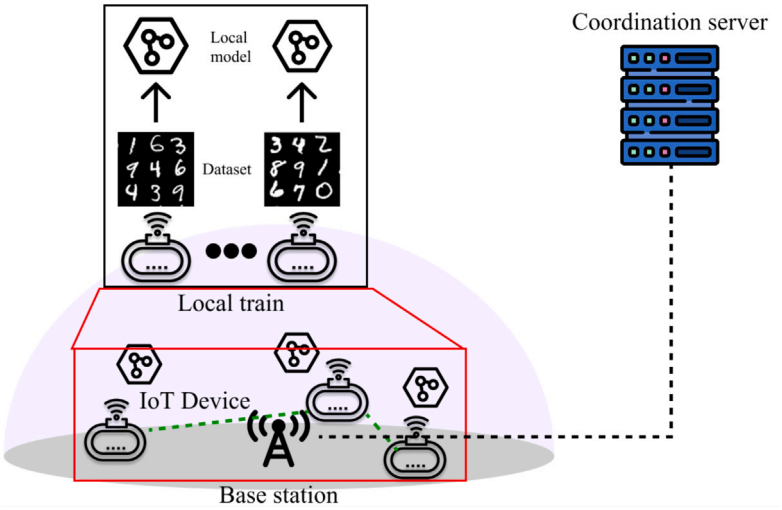

Fig. 1. System model.

2.3. Data and system heterogeneity

In FL, clients often possess locally collected data with different distributions, leading to data heterogeneity. This non-IID nature can slow convergence and cause global model bias by overrepresenting local characteristics. Methods such as SCAFFOLD [16] introduce correction terms to mitigate model drift, while FedAEB [28] and ARFL [29] propose adaptive training policies based on client energy, data quality, or convergence levels. [30] proposed a method that trains a client topology based on model similarity to identify groups of similar clients and utilizes this relationship in the global model. System heterogeneity refers to variations in device resources, including computation power, memory, battery, and network quality. These differences result in imbalanced training contributions and frequent dropouts of low-performance clients. Algorithms like FedProx [14] and FedConv [31] attempt to alleviate this by adjusting local epoch lengths or compressing model complexity.

Addressing heterogeneity is even more challenging in DFL, where global coordination is absent and clients operate with limited knowledge of the network. Inconsistent training progress, information asymmetry, and unsynchronized updates can all degrade performance Particularly in cases of system heterogeneity where each client has different model structures and sizes, the complexity of managing the state of heterogeneous models and the aggregation process increases in a DFL environment without a central server to manage models. This paper builds upon FedWT [24], which reduces data heterogeneity using peer-review-based weighted aggregation, and further extends it to account for system heterogeneity in realistic environments by dynamically adjusting local epochs based on convergence status and training completion rates.

3. System model

3.1. System model overview

This paper is based on an ad hoc IoT network with a wireless mesh topology. Fig. 1 shows the overall system structure. Each IoT device can receive support from nearby base stations to communicate with the coordination server, and communicates with neighboring IoT devices and base stations while performing training. The total number of devices is defined as U, and the set of devices is represented as $\mathcal{U} = 1, 2, \ldots, U$. Each device is denoted as $u \in \mathcal{U}$, and only some devices participate in federated learning during each communication round o. The set of participating devices is defined as $\mathcal{S}^o \subset \mathcal{U}$. The entire dataset is composed of $\mathcal{D} = D_1, D_2, \ldots, D_U$, and each device perform training based on location-based heterogeneous local data.

3.2. Latency model

Channel gain between devices is modeled based on distance, and the distance $d(i, j)$ between IoT devices $i, j \in \mathcal{U}$ is defined as the three-dimensional Euclidean distance. Given the position coordinates of the two devices as (x_i, y_i, z_i) and (x_j, y_j, z_j), the distance is computed as follows:

$$d(i,j) = \text{Euclidean}(x_i, y_i, z_i, x_j, y_j, z_j) = \sqrt{(x_i - x_j)^2 + (y_i - y_j)^2 + (z_i - z_j)^2}, \quad i \neq j \tag{1}$$

A shorter distance between devices results in a higher channel gain, thereby improving communication efficiency.

In this paper, each client performs local training in communication round o, followed by model aggregation and distribution among IoT devices. To quantitatively analyze the model transmission delay that occurs during this process, the transmission speed is defined as follows. The transmission speed $r_{i,j}^{o}$ when transmitting data from device i to device j is given by Eq. (2):

$$r_{i,j} = b_i^o B \log_2(1 + \frac{P_i G_{i,j}}{A_0}) \tag{2}$$

B denotes the total available bandwidth, and b_i^o $(0 \leq b_i^o \leq 1)$ denotes the bandwidth ratio allocated to device i. Each device is assumed to share bandwidth without interference, and IoT devices not selected in round o are not allocated any bandwidth. P_i denotes the transmission power of device i, and A_0 represents the power density of AWGN (additive white Gaussian noise). $G_{i,j}$ is the channel gain between two devices, simplified based on distance under the assumption of a LoS (Line-of-Sight) environment, and defined as in Eq. (3).

$$G_{i,j} = \frac{1}{d(i,j)} \tag{3}$$

The data size of the model parameters of IoT device i is denoted as S_i. In this system model, device-related data sent to the coordination server for MST generation is assumed to be small compared to the model size and therefore negligible. The time $t_{i,j}^{\text{trans}}$ required for device i to transmit the model to device j in the oth round is given by Eq. (4).

$$t_{i,j}^{\text{trans}} = \frac{S_i}{r_{i,j}^o} \tag{4}$$

The local training time for device i, denoted as t_i^{train}, is defined in Eq. (5). Let e_i^o be the number of local epochs for device i in communication round o, and c_i the number of CPU cycles required to process a single data sample. The subset of the dataset used by device i in round o is denoted as d_i, and f_i represents the CPU clock speed of device i.

$$t_i^{\text{train}} = \frac{e_i \cdot c_i \cdot |d_i^o|}{f_i} \tag{5}$$

3.3. Minimum spanning tree model

This section describes the MST-based transmission delay model applied during path selection for model exchange after local training. The total training time in round o, denoted as $T^{\text{train},o}$, is determined by the maximum training time among the participating devices, as defined in Eq. (6):

$$T^{\text{train},o} = \max(t_1^{\text{train},o}, \ldots, t_s^{\text{train},o}), \quad s \in S^o \tag{6}$$

After training, each IoT device transmits its model to its parent node and child nodes during peer review and aggregation. To calculate the upper bound of the total transmission delay $T^{\text{trans},o}$ during the aggregation process, a skewed tree structure is assumed. In this structure, transmissions occur sequentially at each level of the MST constructed by the participating devices. In this case, the total model exchange delay is approximated as the sum of the transmission delays $t_{i,j}^{\text{trans}}$ of all links in the tree and is expressed as follows:

$$\max_{(i,j)\in\tau} t_{i,j}^{\text{trans}} \leq T^{\text{trans},o} \leq \sum_{(i,j)\in\tau} t_{i,j}^{\text{trans}} \tag{7}$$

τ denotes the tree structure generated by the coordination server.

The total latency T^o in round o is represented by the sum of the delay time $T^{train,o}$ required for local training in round o and the latency $T^{\text{trans},o}$ required for model sharing.

$$T^o = T^{\text{train},o} + T^{\text{trans},o} \tag{8}$$

In this case, the training time is randomly determined during the client selection stage. As a result, it varies dynamically depending on the sampled clients, which makes optimization difficult. To address this, the goal is to construct an optimal spanning tree by minimizing the upper and lower bounds of the communication delay T^{trans} during MST generation. The optimal spanning tree τ^* that minimizes the total transmission delay between devices i and j is defined in Eq. (9).

$$\min T^{\text{trans}} = \tau^* = \text{argmin}_{\tau\in\mathcal{T}(C)} \sum_{i,j\in\tau} t_{i,j}^{\text{trans}} \tag{9}$$

$\mathcal{T}(C)$ denotes the set of spanning trees created in device cluster C participating in the current round.

3.4. Decentralized federated learning model

This section describes DFL models in which IoT devices engage in cooperative learning. Device u can communicate with neighboring devices and trains a local model m_u using the data samples $\mathcal{D}_u$ it possesses. In the model exchange phase, each device participating in training reviews other models using its own dataset. Each device then obtains the weight γ_u assigned to each model. γ_u is the weight assigned to the entire model, obtained using the peer review and aggregation process described in [24]. During training, the goal of all devices is to optimize the global loss function $L(M)$ using the weights γ_u of the local loss function $L(m_u)$.

$$L(M) = \frac{1}{\mathcal{U}} \sum_{u\in\mathcal{U}} \gamma_u L(m_u) \tag{10}$$

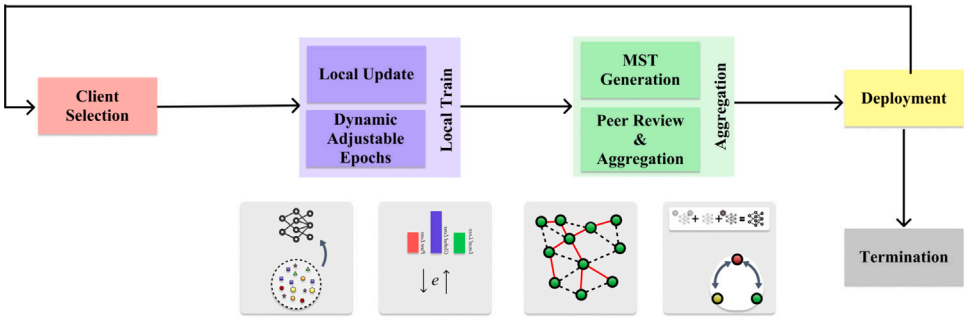

Fig. 2. Overview of the training pipeline, including client selection, local training, MST-based aggregation, and global model deployment.

4. Federated learning via dynamic epoch adjustment for heterogeneous clients in ad-hoc networks

4.1. Training overview

Data heterogeneity in FL indicates that the distribution of data held by each client is non-IID. This imbalance in distribution negatively affects the convergence speed and performance of FL. For instance, in the standard FedAvg algorithm, equal weights are assigned to all clients' local models during aggregation. This can degrade generalization performance, as the models of clients with severe data bias are equally reflected in the global model. The problem becomes more severe in environments with high data heterogeneity. Furthermore, many FL scenarios assume that all clients use the same model architecture, but in real-world environments, there is system heterogeneity due to differences in computational performance, battery capacity, and communication latency between clients [31]. In such an environment, training methods that require the same amount of computation can cause training failure or early dropout of low-performance clients, resulting in excessive reflection of the models of specific high-performance clients and exacerbating the bias of the global model.

To address this issue, this paper proposes FedAWT, which simultaneously considers data and system heterogeneity in a mesh topology environment. FedAWT comprises the following four steps: (1) local model training and dynamic epoch adjustment, (2) MST-based transmission path setting, (3) peer review based aggregation and weight adjustment, and (4) global model redistribution. An overview of the entire process is presented in Fig. 2. FedAWT improves the generalization performance of global models through efficient model transfer using MST and performance-based weight adjustment between nodes, while also reducing communication delays during aggregation. In addition, by dynamically adjusting the number of epochs according to the computing resources of IoT devices during local learning, enables participation from resource-constrained devices.

4.2. Model update and dynamic adjustable epoch

In traditional FL, the same number of epochs is applied to all clients in batches. However, in a distributed environment based on IoT devices with different performance and network environments, applying the same number of epochs may frequently cause training failures (dropouts) on devices with insufficient resources. This results in only a subset of client models being included in the aggregation, leading to aggregation bias. In extreme cases, only high-performance devices' models may be included, causing severe global model bias. Such system heterogeneity can be more severe in DFL environments operating without central server support.

To address this issue, FedAWT introduces a mechanism in which each device transmits its training completion status and local loss value to the coordination server. Using this information, the server computes the Dynamic Adjustable Factor (DAF) and Learning Stability Factor (LSF), which are then used to assign an adaptive number of training epochs to each device without additional hyper parameter tuning. This approach improves global model convergence and performance in heterogeneous environments. Additionally, to alleviate model variance between clients that may result from data heterogeneity and dynamic epoch adjustment, a proximal term is added to the local loss function to induce stable convergence.

4.2.1. Dynamic adjustable factor

DAF determines the number of training epochs to be applied in the next round based on the local loss value obtained during training. It dynamically adjusts the epoch count per client by capturing the gap between local and global convergence. DAF is defined in Eq. (11).

$$\mathrm{DAF}_i = \alpha(\widetilde{l_i^o - l_i^{\mathrm{past}}}) + (1-\alpha)(\widetilde{l_i^o - l_{\mathrm{global}}}), \quad 0 \le \alpha \le 1 \tag{11}$$

$\widetilde{(\cdot)}$ denotes the normalization operation. The first term indicates the temporal change in local loss. If $l_i^o > l_i^{\mathrm{past}}$, it implies that the local model has not yet converged, and thus more epochs are assigned. The second term measures the discrepancy between the local and global loss values. If the local model is sufficiently close to the global model, excessive training may destabilize the global model, and the number of epochs is reduced accordingly. The parameter α controls the relative influence of the two terms.

4.2.2. Learning stability factor

LSF is a penalty coefficient that adjusts the next round's epoch count e_i^{next} for each client, based on the overall training completion rate within the DFL system. It is designed to suppress excessive training failures that may result from overly aggressive epoch assignment by DAF. Each device reports its training completion status to the coordination server, which calculates the training completion rate cap^o across all devices. If this rate exceeds a predefined threshold β, no penalty is applied. Otherwise, a proportionally reduced factor is used.

$$\text{LSF}^o = \begin{cases} \frac{\text{cap}^o}{\beta} & \text{if } \frac{\text{cap}^o}{\beta} \leq 1 \\ 1 & \text{if } \frac{\text{cap}^o}{\beta} > 1 \end{cases} \tag{12}$$

4.2.3. Calculating epochs for the next participation

Finally, the next round epoch number e_i^{next} for each device i is defined in Eq. (13).

$$e_i^{\text{next}} = \max(1, \min(E, \text{LSF} \cdot \lceil \frac{e_i^o}{\sigma} + e_i^o \cdot \text{DAF}_i \rceil)) \tag{13}$$

E denotes the maximum number of epochs, and $\sigma > 0$ is a tunable scaling factor. Eq. (13) computes the adjusted epoch value by combining the LSF with the client-specific adaptive factor, aiming to prevent excessive training and ensure stable convergence. The complete epoch adjustment procedure is outlined in Algorithm 1.

Algorithm 1: Dynamic Adjustable Epoch

Input: Set of devices participating in training $\mathcal{S}$, Local loss on each device l, device index $i \in \mathcal{S}$
Output: Epochs by device

$\text{DAF}_i \leftarrow \alpha(\widetilde{l_i^o - l_i^{\text{past}}}) + (1-\alpha)(\widetilde{l_i^o - l_{\text{global}}})$ // DAF calculation by device
$\text{cap}^o \leftarrow \frac{\text{Number of Complete UAV}}{|S|}$
$\text{LSF}^o \leftarrow \frac{\text{cap}^o}{\beta}$ // LSF calculation
$e_i^{\text{next}} = \max(1, \min(E, \text{LSF} \cdot \lceil \frac{e_i^o}{\sigma} + e_i^o \cdot \text{DAF}_i \rceil))$ // Epochs to be applied when participating in the next training for each device
Transmit each e^{next} to the device.

4.3. Minimum spanning tree generation

The minimum spanning tree (MST) for model distribution is constructed based on the transmission latency between IoT devices. The MST generation algorithm consists of two main steps. In the first step, a complete graph is generated based on the latency between the devices that participated in training. After local training, each device submits its location, transmission speed, and channel status to the coordination server, which uses this information to construct the graph. In the second step, the coordination server constructs a latency based graph G based on the received information, selects a random device as the starting node, and generates an MST using the Prim's algorithm. The root node is then determined as the device that minimizes the tree depth. The final MST structure is shared with all participating devices.

The algorithm 2 summarizes the MST generation process.

Algorithm 2: MST Generation

Input: Set of participating devices $\mathcal{S}$, Transmission latency between devices $r_{i,j}^{\text{trans}}, \quad i,j \in \mathcal{S}$
Output: MST τ

$\tau \leftarrow \emptyset$ // Initial spanning tree edge set
$S_{\text{visited}} \leftarrow \{\text{Random device } u_{random} \in \mathcal{S}\}$ // Initialize visited device set
while $S_{visited} \neq \mathcal{S}$ **do**
  $(i,j) \leftarrow \arg\min_{i \in S_{\text{visited}}, j \in \mathcal{S} \setminus S_{\text{visited}}} r_{i,j}^{\text{trans}}$ // Select the smallest value among $\tau_{i,j}^t rans$
  $\tau \leftarrow \tau \cup \{(i,j)\}$ // Add edge to MST
  $S_{\text{visited}} \leftarrow S_{\text{visited}} \cup \{j\}$
return τ

After MST generation, the global model is produced and deployed through peer-review-based aggregation, as described in [24]. In FedAWT, model evaluation is performed via dynamic epoch adjustment and peer review between parent and child nodes. Upon receiving the model from its parent node, each device evaluates it using its local data to compute a loss value. It then transmits its own model along with the evaluation score to the parent. The parent node, in turn, evaluates the models of its children and performs weighted aggregation. This peer-review-based aggregation process is recursively applied from the leaf nodes up to the root node, enabling hierarchical model integration within the MST structure. This process helps alleviate the imbalance among local models caused by the non-IID nature of training data. Moreover, dynamic epoch adjustment increases the opportunity for clients with fewer computing resources to participate in training in a DFL network.

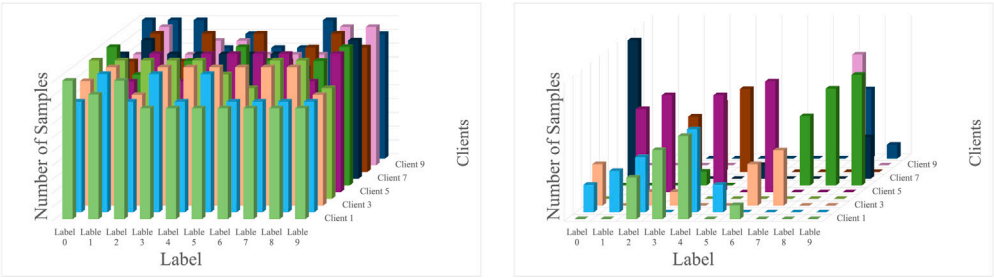

Fig. 3. Comparison between IID and non-IID settings across clients, showing balanced vs. imbalanced label distributions.

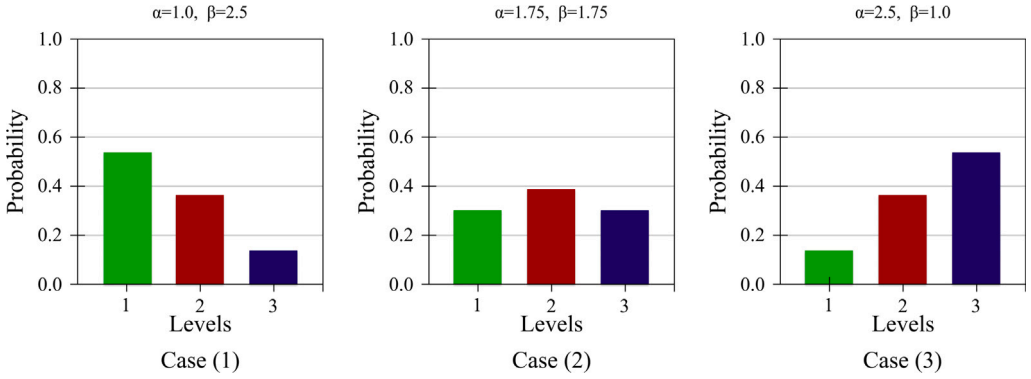

Fig. 4. Client distributions across three cases, each defined by different α and β parameters of the beta-binomial distribution to simulate varying compute resource levels.

5. Simulation

5.1. Environment configuration

Two distributed training environments were constructed for simulation, and MNIST, Fashion-MNIST, and CIFAR-10 were used as training datasets. The total number of IoT devices was set to 500, and the ratio of devices participating in training in each round was fixed at 20%. The learning rate was set to 0.001. Fig. 3 shows the data distribution scenario used in the experiment, illustrating a typical non-IID situation where the label distribution and sample size vary among clients, resulting in data distribution imbalance.

Fig. 5 shows the network topology used in this paper. Fig. 5 also shows the network structure applied in each experimental environment. Based on these combinations, we simulate training under different network conditions. Traditional CFL is based on the star topology shown in Fig. 5(a), while DFL is based on the mesh topology shown in Fig. 5(b). For data distribution, the non-IID scenario defined in Fig. 3 is used, where each UAV is randomly assigned four unique class labels, resulting in highly skewed label distributions across clients.

To configure a heterogeneous system environment, discrete resource levels (Level) are hierarchically defined based on the computational resource levels of devices. Level 1 is defined as the device with the highest computational performance, and subsequent levels are configured such that computational capability decreases progressively based on the device performance reduction ratio ρ. For example, if $\rho = 0.5$, Level 2 UAVs have 50% of the computational resources of Level 1, and Level 3 has half of that, or 25% of the computational capability. As the performance level of the device decreases, it takes more time to perform the same number of epochs, and there is a possibility that training may fail to complete depending on the number of epochs e allocated relative to the total maximum number of epochs E.

In particular, devices with lower resource levels experience a greater computational burden when the number of epochs is large, leading to an increased training failure rate. The training failure rate d_i of device i is defined as follows:

$$\text{IncompleteRate}_{(e,\ \text{level})} = \max\left(0,\ e - \rho^{\frac{e(\text{level}-1)}{E}} \cdot E\right) \tag{14}$$

Eq. (14) has a structure that exponentially reduces the scope of training that can be performed according to the resource level of the device. In particular, even in the same epoch e, the lower the device level, the greater the incompletion rate calculated due to the influence of the term $\rho^{\frac{e(\text{level}-1)}{E}}$. This formulation enables quantitative simulation of training failure probabilities for low-performance devices under high system heterogeneity. The distribution of device resource levels follows a beta binomial distribution. Fig. 4 shows the performance distribution of devices applied during training.

Table 1
Simulation parameter settings.

Parameter	Value
Number of Devices (U)	500
Sample Ratio per Round	5%
Performance Shrinkage Ratio (ρ)	0.5
Number of Total Epoch (E)	5
Model	ANN, CNN
Learning Rate	0.001
Size of the CNN Model (S_i)	2×10^7 bit
Optimizer	ADAM
Number of Total Round (R)	500
Power Spectral Density of The Gaussian Noise (A_0)	1.0×10^{-8}
Transmit Power of Client i (P_i)	2 W
Channel Bandwidth (B)	5 MHz
CPU Computation capacity (f)	[1, 5] GHZ
CPU cycles per bit (c)	20 cycles/bit

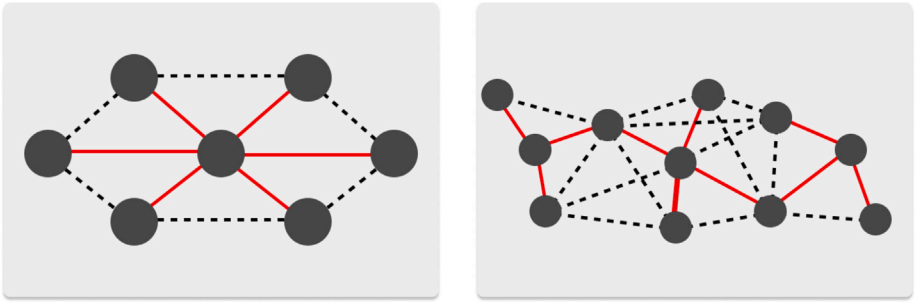

Fig. 5. Network topologies used in simulation: (a) centralized star and (b) decentralized mesh structures.

Detailed settings for the training environment are summarized in Table 1. For performance evaluation, FedAvg [7], a representative centralized FL algorithm, is applied. Additionally, three decentralized methods are considered: D-PSGD [20], D-PSGD with a proximal term, and GossipFL [21].

5.2. FedAWT evaluation and discussion

5.2.1. Performance of top K-1 accuracy

Table 2 shows the results of comparing the training accuracy of various FL methods in three scenarios based on the computational performance of clients. In a simple MNIST dataset under a Non-IID environment, the proposed FedAWT achieved up to 2.01% higher accuracy than the existing FedAvg by applying dynamic weights. This is because FedAWT effectively mitigated the impact of low-performance clients with frequent training failures by dynamically adjusting weights during model aggregation, reflecting whether clients have completed training. On the other hand, existing methods that use a fixed number of epochs show a noticeable performance degradation due to an increase in the failure rate caused by differences in client performance. This tendency is particularly significant when the proportion of devices with insufficient computing resources increases.

In the case of FedAvg, the impact of the incomplete rate is limited on relatively simple datasets such as MNIST and Fashion MNIST, resulting in slightly higher performance than FedAWT in some scenarios. For example, it shows a performance advantage of 0.26% and 1.00% in MNIST Case 2 and Fashion MNIST Case 1, respectively. However, on datasets with high image complexity like CIFAR-10, the performance of FedAvg drops sharply as the proportion of low-performing clients increases, showing significant decreases of 16.59% and 14.10% in Case 2 and Case 3, respectively. In contrast, FedAWT maintains training stability and performance even with complex datasets, achieving the highest performance with an accuracy of 55.99% on CIFAR-10 based on Case 3.

Compared to methods such as D-PSGD, Proximal Term applied to D-PSGD, and GossipFL that consider a decentralized federated learning environment, FedAWT showed excellent performance in all scenarios. In particular, based on the MNIST-CNN benchmark, an accuracy improvement of at least 5.81% to a maximum of 33.86% was observed compared to *D-PSGD*, which can be attributed to the combination of message topology-based communication optimization (MST) and a dynamic weight-based model aggregation strategy. This structure further improves the robustness and generalization ability of training compared to existing methods that simply aggregate the average of models.

Table 2
Top k-1 accuracy performance comparison for three compute resource scenarios.

		Case 1	Case 2	Case 3
MNIST (CNN)	FedAvg	99.01%	**99.10**%	96.80%
	D-PSGD	93.99%	92.79% 93.00%	
	D-PSGD + Proximal Term	95.29%	93.56%	93.36%
	GossipFL	70.75%	71.00%	64.95%
	FedAWT	**99.10%**	98.84%	**98.81**%
Fashion MNIST (CNN)	FedAvg	**88.99** %	86.10%	85.00%
	D-PSGD	76.99%	75.00%	73.00%
	D-PSGD + Proximal Term	77.23%	75.48%	76.87%
	GossipFL	55.12%	49.87%	54.11%
	FedAWT	87.99%	**88.01**%	**87.00**%
CIFAR-10 (CNN)	FedAvg	55.49%	16.59%	14.10%
	D-PSGD	40.00%	36.99%	31.99%
	D-PSGD + Proximal Term	43.99%	41.36%	38.89%
	GossipFL	18.75%	17.92%	16.80%
	FedAWT	**61.99**%	**56.99**%	**55.99%**

In summary, FedAWT demonstrates superior performance across various datasets and client computation configurations. This suggests that the core design elements of FedAWT, namely dynamic epoch adjustment, unfinished rate-based weight adjustment, and tree-based efficient communication strategy, can effectively address real-world factors such as data and system heterogeneity and network constraints. Therefore, FedAWT is a promising approach that can provide reliable FL performance even in distributed environments with limited resources and poor network conditions, such as IoT device-based ad-hoc environments.

5.2.2. Train loss

Fig. 6 shows the results of comparing the training loss of each FL method according to the client's computing performance heterogeneity scenarios (Cases 1–3). The experimental results show that FedAWT achieved the fastest convergence speed and the lowest loss value in all scenarios, demonstrating robust training performance even in environments with large computing resource deviations. FedAWT tends to show high losses temporarily because dynamic weight adjustment is performed when local models are not sufficiently trained in the early stages, resulting in large fluctuations in the aggregated weights. However, as training progresses, the loss values of each client's model do not fluctuate greatly, and the weight adjustment gradually stabilizes. Particularly in scenarios like Case 3, where clients with limited computational resources account for a high proportion, FedAWT exhibits the lowest initial loss and a stable decrease curve, achieving a final loss of 0.02423, which is significantly superior to DFL.

FedAvg shows convergence characteristics similar to FedAWT in Cases 1 and 2, but in Case 3, the initial loss increases rapidly and the final loss remains higher than FedAWT. This seems to be because the fixed epoch setting did not take into account the incomplete training of clients with low computing performance, which hindered the convergence of the entire model. D-PSGD follows a distributed structure-based averaging method, resulting in limited loss reduction and slower convergence speed compared to FedAWT. Especially in Cases 2 and 3, the convergence curve remains gradual, and the final loss value is also high. D-PSGD + Proximal Term exhibits similar initial characteristics to FedAWT in terms of convergence speed improvement, but experiences a temporary surge in loss (up to 5.55) in Case 3 and shows lower convergence stability compared to FedAWT. This shows that although Proximal Term provides some training stability, its effectiveness is limited when there are many unfinished trainings without dynamic adjustment at the client level. GossipFL has an unstable convergence curve in all scenarios, and the loss remains at a high level overall. Unlike FedAWT, which is designed based on a mesh topology, GossipFL is based on a fully connected assumption, leading to ineffective propagation and aggregation in real-world ad-hoc communication environments, resulting in poor convergence. FedAWT compensates for model incompletion in environments with large computational resource deviations through dynamic epoch adjustment and considers communication efficiency through MST-based transmission paths, showing consistent superiority in terms of training convergence speed and loss stability compared to existing centralized and decentralized federated learning methods.

5.2.3. Communication traffic and latency performance

Fig. 7 compares the average data transmission traffic (left) and the average transmission latency (right) per round in the FL environment, where devices use the MNIST dataset and CNN model to reach the target accuracy of 80%. Since all devices share their models directly with neighboring devices in each round, D-PSGD has the highest transmission latency in each round. In addition, since all nodes repeatedly propagate models, the total traffic consumption is also very high. GossipFL communicates only with a single neighbor, so the transmission latency per round is the lowest. However, due to limited connectivity and slow propagation speeds, more rounds were required to reach the target accuracy, resulting in the second-highest cumulative transmission traffic.

FedAWT utilizes an MST-based tree structure to construct aggregation paths, thereby minimizing model transmission redundancy and improving communication efficiency. Due to the nature of the tree structure, the transmission latency in a single round is relatively large, but by quickly achieving the accuracy target within a few rounds, it showed the most efficient results among DFLs in terms of overall transmission traffic. In particular, *FedAWT* achieved both fast convergence and low transmission traffic in a device-based ad-hoc environment with computational performance and topology constraints.

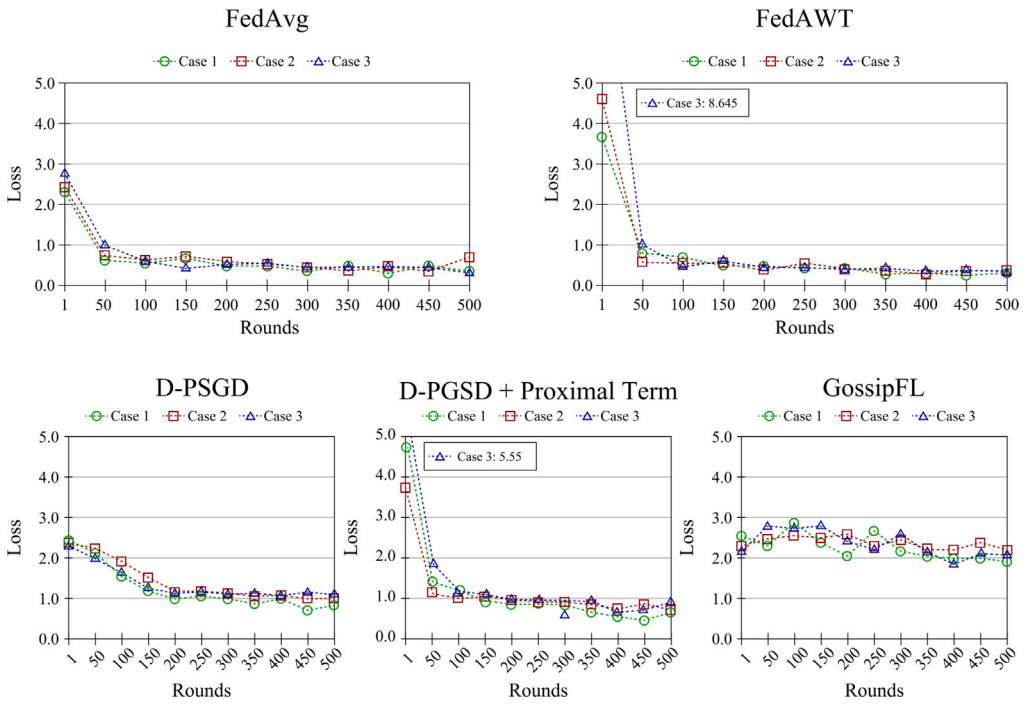

Fig. 6. Training loss curves of each FL method under varying device capability scenarios, illustrating differences in convergence behavior and stability.

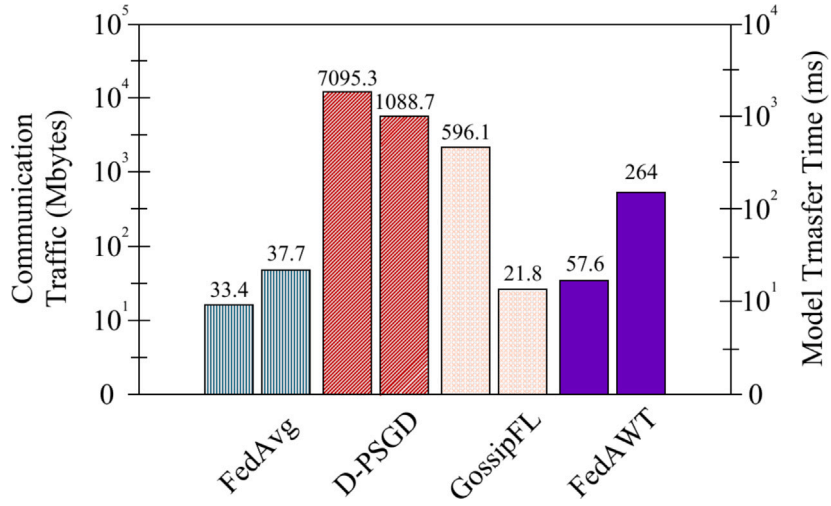

Fig. 7. Communication traffic and model transfer latency per round to reach the target accuracy across different FL methods.

5.2.4. Epoch incomplete rate

Fig. 8 shows the results of comparing the client incomplete rate (Incomplete Rate) under different computational performance heterogeneity scenarios (Case 1–3) between the fixed epoch method and FedAWT. In the fixed epoch method, the same amount of training is required regardless of the client's resource situation or training failure history, even as rounds progress, so a high incomplete rate of 20% or more is consistently maintained in all scenarios. In particular, in cases such as Case 3, where the proportion of low-performance clients is high, a dropout rate of 40% or more occurs, which adversely affects the overall model convergence performance and training efficiency. FedAWT dynamically adjusts the number of epochs based on the computational performance of clients and local loss information, resulting in a sharp decrease in the failure rate after the initial rounds and maintaining a low level of less than 10% in most rounds. This dynamic adjustment strategy contributes to increasing the overall training success rate and minimizing the waste of communication and computational resources. In particular, even in Cases 2 and 3, the failure rate

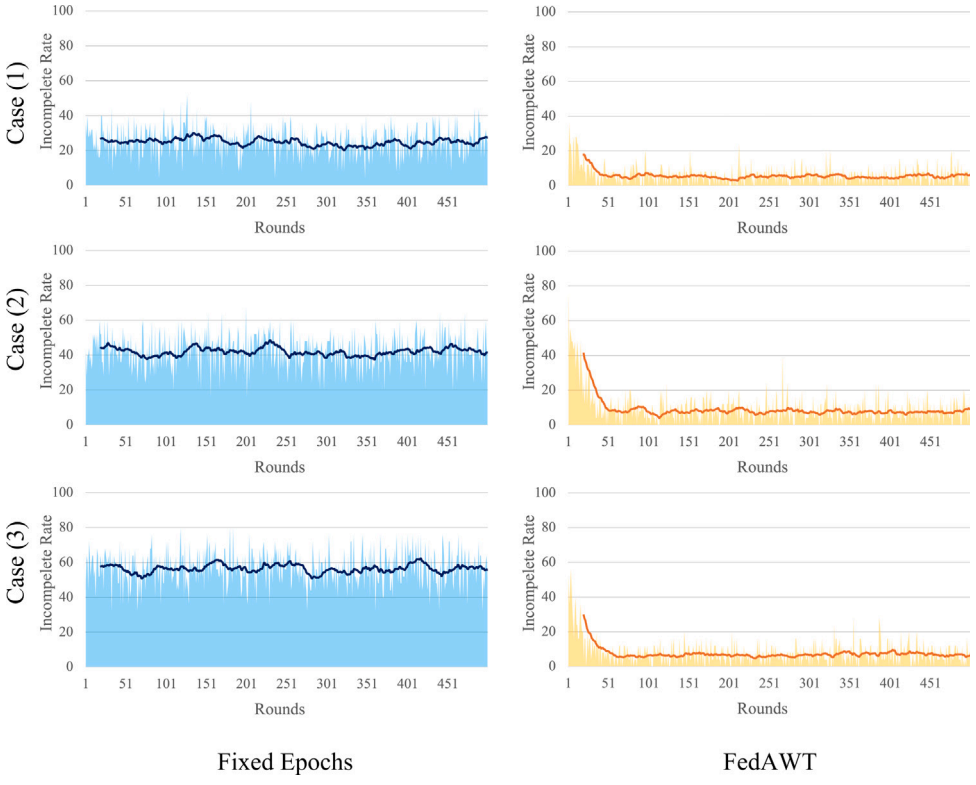

Fig. 8. Comparison of incomplete rates across rounds between fixed epoch and FedAWT under different client heterogeneity scenarios.

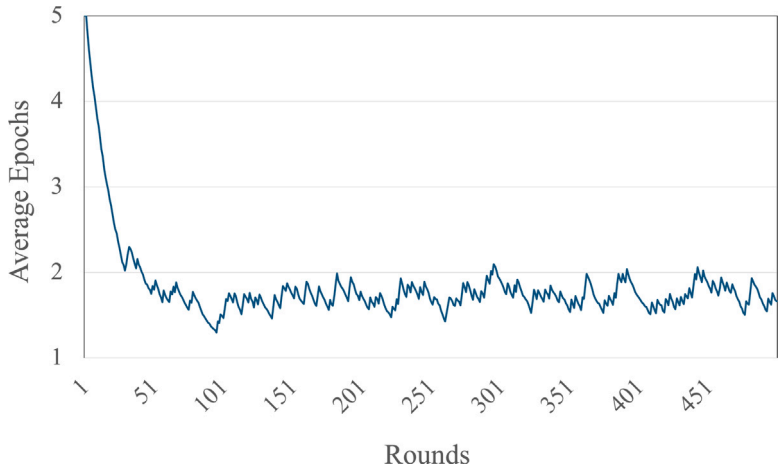

Fig. 9. Variation in the average number of epochs per round under FedAWT training.

converged to less than half that of the fixed epoch method, experimentally confirming that the training structure is robust to system heterogeneity.

However, this structure also involves certain trade-offs. Reducing the number of epochs is effective in lowering the uncompleted rate, but it may cause the local models of some low-performance clients to be reflected in the aggregation without sufficient training. This may lead to information loss in environments where the global model's expressiveness is low or data bias exists. Therefore, although FedAWT shows excellent results in terms of incomplete rate and convergence stability, it is necessary to consider setting a minimum number of epochs or additional loss-based supplementary training mechanisms to prevent overly conservative training restrictions, as shown in Fig. 9.

5.3. Analysis and discussion of simulation results

In this section, the experimental results of the proposed FedAWT framework are analyzed in comparison with existing FL methods, focusing on key factors contributing to performance differences. The practical implications, limitations, and future research directions are also discussed in the context of realistic IoT-based distributed environments. FedAWT exhibited stable training performance in environments characterized by data and system heterogeneity. Across diverse datasets (MNIST, Fashion-MNIST, CIFAR-10) and varying distributions of client computational capabilities, it achieved improved accuracy and convergence speed compared to both centralized (FedAvg) and decentralized (D-PSGD, GossipFL) baselines. By dynamically adjusting the number of local training epochs based on each device's computational resources and training status, FedAWT reduced training failures among low-performance devices and facilitated broader client participation. Although the MST-based communication strategy introduced additional latency per round, it effectively reduced cumulative communication traffic. Furthermore, the peer-review-based weight aggregation method alleviated global model bias under non-IID conditions, thereby enhancing generalization performance.

Specifically, the MST-based aggregation mechanism improves performance by extending the distance-based minimum spanning tree approach proposed in [24] to a latency-aware scheduling scheme, thereby enabling more efficient and timely model aggregation. Furthermore, the peer-review-based weighting process enhances the contribution of models with better generalization, alleviating the degradation caused by data heterogeneity during aggregation. In parallel, the dynamic epoch adjustment technique mitigates both temporal and local training instability by adaptively assigning training epochs based on local loss behavior and convergence trends. Coupled with a penalty mechanism derived from training completion rates, this strategy facilitates broader participation of low-resource devices, thereby reducing the impact of system-level heterogeneity. These two mechanisms contribute to improved global model performance by addressing the dual challenges of heterogeneous data distributions and device capabilities. These features collectively enable robust federated learning under practical constraints such as limited resources, communication variability, and data imbalance, supporting the applicability of FedAWT in real-world IoT-based environments.

While FedAWT demonstrates stable performance, several aspects remain to be refined. LSF, introduced to suppress training dropouts, effectively enhances participation from low-performance devices. However, if the threshold for training completion is overly sensitive to dropouts, the adaptive epoch control may excessively reduce training iterations, thereby limiting global model convergence. To ensure the convergence stability of the global model, the proximal term is used to prevent significant changes in the model updates of high-performance devices, thereby preventing divergence of the global model. However, it does not incorporate considerations for the minimum epoch required to directly guarantee training on low-performance devices. Additionally, since both MST construction and epoch assignment depend on global information provided by the coordination server, FedAWT is not directly applicable to fully decentralized architectures. Because FedAWT is a hybrid architecture that utilizes a coordination server for MST construction and epoch allocation to ensure stability. The overhead incurred for this parameter is negligible compared to the model size. However, this dependency may impose scalability limitations in environments with large-scale device participation or in fully serverless environments. Finally, FedAWT currently focuses on aggregation and learning during training, employing random sampling for client participation during this process. However, real-world federated learning scenarios are constrained by various environmental factors such as client idle states, battery levels, and communication conditions, making this approach unsuitable for practical application. Furthermore, due to privacy constraints in federated learning scenarios, direct use of client data is impossible. Consequently, data-based selection strategies face significant limitations when applied to real-world applications.

To address these issues, future work could consider the following approaches. To prevent cases where adaptive epoch control, such as the LSF factor, excessively restricts training, research into methods for adjusting the lower bound of dynamic epoch adjustment could be considered. Additionally, to reduce dependency on a single coordination server and enhance scalability, a framework that distributes coordination tasks — such as MST configuration and epoch allocation — through layered decentralization can be considered. We plan to conduct research on establishing a hierarchical structure by creating local aggregators capable of managing subsets of clients, thereby reducing the load on a single server and laying the groundwork for applying distributed federated learning in large-scale environments. Finally, to overcome the limitations of random sampling, research can be conducted to introduce intelligent client selection strategies. As referenced in [28], applying reinforcement learning-based selection policies allows for comprehensive consideration of non-personal information such as each client's resources and network status without infringing on privacy. Furthermore, future research will explore utilizing non-personal information that does not reveal each client's data distribution as a selection condition. These researches will optimize the participation of devices with limited resources and significantly enhance the efficiency and stability of real-world federated learning scenarios.

6. Conclusion

This paper proposes FedAWT, a DFL framework that considers data and system heterogeneity issues arising in a mesh network environment composed of heterogeneous IoT devices. FedAWT aims to improve the convergence and generalization performance of global models without bias toward high-performance clients through a dynamic epoch control method that reflects the heterogeneous computing performance of devices, a model transfer path configuration based on minimum spanning trees (MST) that minimizes communication delays, and a peer review based weight aggregation method used in FedWT. Simulation results show that FedAWT performs better than existing DFL methods even with a large number of UAVs and non-IID distribution environments, effectively mitigating communication delays and training bias. In addition, it improves the training participation rate of UAVs with limited computational resources, enabling balanced training of the entire system. Future research may consider client selection for training time optimization. Although this paper uses a method of randomly sampling clients, it is expected that learning efficiency and resource utilization can be further improved through the same method of performing reinforcement learning-based selection policies that comprehensively consider client computing resources, data quality, network status, etc., as in [28].

CRediT authorship contribution statement

Geonhee Kim: Writing – review & editing, Writing – original draft, Visualization, Validation, Supervision, Software, Resources, Project administration, Methodology, Investigation, Funding acquisition, Formal analysis, Data curation, Conceptualization. **Seunghyun Park:** Writing – review & editing, Visualization, Validation, Supervision, Software, Methodology, Investigation, Funding acquisition, Formal analysis. **Hyunhee Park:** Writing – review & editing, Visualization, Validation, Supervision, Software, Resources, Project administration, Methodology, Investigation, Funding acquisition, Formal analysis, Data curation, Conceptualization.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgments

This work was supported by the National Research Foundation of Korea (NRF) grant funded by the Korea government (MSIT) and the Ministry of Trade, Industry & Energy(MOTIE) of the Republic of Korea (No. 2022R1A2C2005705, No. RS-2024-00469138), and this research was financially supported by Hansung University for Prof. Seunghyun Park.

Data availability

Link to my data is available at https://github.com/ghkim919/FedAWT.

References

[1] W.H. Hassan, Current research on internet of things (IoT) security: A survey, Comput. Netw. 148 (2019) 283–294.
[2] M.S. Rahman, M.A. Islam, M.A. Uddin, G. Stea, A survey of blockchain-based IoT ehealthcare: Applications, research issues, and challenges, Internet Things 19 (2022) 100551.
[3] B.B. Gupta, M. Quamara, An overview of internet of things (IoT): Architectural aspects, challenges, and protocols, concurrency and computation, Pr. Exp. 32 (21) (2020) e4946.
[4] R.J. Fereira, C. Ranaweera, K. Lee, J.G. Schneider, Energy efficient resource management for real-time IoT applications, Internet Things 30 (2025) 101515.
[5] Y. Liu, J. Wang, J. Li, S. Niu, H. Song, Machine learning for the detection and identification of internet of things devices: A survey, IEEE Internet Things J. 9 (1) (2021) 298–320.
[6] L. Dutta, S. Bharali, Tinyml meets IoT: A comprehensive survey, Internet Things 16 (2021) 100461.
[7] B. McMahan, E. Moore, D. Ramage, S. Hampson, B.A. y Arcas, Communication-efficient learning of deep networks from decentralized data, in: Proceedings of the International Conference on Artificial Intelligence and Statistics, AISTATS, PMLR, 2017, pp. 1273–1282.
[8] P. Kairouz, H.B. McMahan, B. Avent, A. Bellet, M. Bennis, A.N. Bhagoji, et al., Advances and open problems in federated learning, Foundations and Trends® in Machine Learning 14 (1–2) (2021) 1–210.
[9] A. Lalitha, S. Shekhar, T. Javidi, F. Koushanfar, Fully decentralized federated learning, in: 3rd Workshop on Bayesian Deep Learning (NeurIPS), 2018.
[10] C. Zhang, Y. Xie, H. Bai, B. Yu, W. Li, Y. Gao, A survey on federated learning, Knowl.-Based Syst. 216 (2021) 106775.
[11] Q. Yang, Y. Liu, T. Chen, Y. Tong, Federated machine learning: Concept and applications, ACM Transactions on Intelligent Systems and Technology 10 (2) (2019) 12:1–12, 19.
[12] P.M. Mammen, Federated learning: Opportunities and challenges, 2021, p. 8, arXiv preprint arXiv:2101.0542.
[13] D.C. Nguyen, M. Ding, P.N. Pathirana, A. Seneviratne, J. Li, H.V. Poor, Federated learning for internet of things: A comprehensive survey, IEEE Commun. Surv. & Tutorials 23 (3) (2021) 1622–1658.
[14] T. Li, A.K. Sahu, M. Zaheer, M. Sanjabi, A. Talwalkar, V. Smith, Federated optimization in heterogeneous networks, Proc. Mach. Learn. Syst. 2 (2020) 429–450.
[15] J. Wang, Q. Liu, H. Liang, G. Joshi, H.V. Poor, Tackling the objective inconsistency problem in heterogeneous federated optimization, Adv. Neural Inf. Process. Syst. 33 (2020) 7611–7623.
[16] S.P. Karimireddy, S. Kale, M. Mohri, S. Reddi, S. Stich, A.T. Suresh, SCAFFOLD: Stochastic controlled averaging for federated learning, in: Proceedings of the International Conference on Machine Learning, ICML, PMLR, 2020, pp. 5132–5143.
[17] D.A.E. Acar, Y. Zhao, R.M. Navarro, M. Mattina, P.N. Whatmough, V. Saligrama, Federated learning based on dynamic regularization, 2021, p. 3, arXiv preprint arXiv:2111.0426.
[18] J. Park, J. Ko, FedHM: Practical federated learning for heterogeneous model deployments, ICT Express 10 (2) (2024) 387–392.
[19] J. Bang, S. Woo, J. Lee, DRL-empowered joint batch size and weighted aggregation adjustment mechanism for federated learning on non-IID data, ICT Express 10 (4) (2024) 863–870.
[20] X. Lian, C. Zhang, H. Zhang, C.J. Hsieh, W. Zhang, J. Liu, Can decentralized algorithms outperform centralized algorithms? A case study for decentralized parallel stochastic gradient descent, in: Advances in Neural Information Processing Systems, 2017, p. 30.
[21] Z. Tang, S. Shi, B. Li, X. Chu, Gossipfl: A decentralized federated learning framework with sparsified and adaptive communication, IEEE Trans. Parallel Distrib. Syst. 34 (3) (2022) 909–922.
[22] M.H. Alsharif, R. Kannadasan, W. Wei, K.S. Nisar, A.H. Abdel-Aty, A contemporary survey of recent advances in federated learning: Taxonomies, applications, and challenges, Internet Things 27 (2024) 101251.
[23] A. Imteaj, U. Thakker, S. Wang, J. Li, M.H. Amini, A survey on federated learning for resource-constrained IoT devices, IEEE Internet Things J. 9 (1) (2021) 1–24.
[24] G. Kim, J. Kim, Y. Kim, H. Kim, H. Park, Fedwt: Federated learning with minimum spanning tree-based weighted tree aggregation for UAV networks, ICT Express 11 (2) (2025) 275–280.
[25] J. Tang, J. Nie, Y. Zhang, Z. Xiong, W. Jiang, M. Guizani, Multi-UAV-assisted federated learning for energy-aware distributed edge training, IEEE Trans. Netw. Serv. Manag. 21 (1) (2023) 280–294.

[26] F.P.C. Lin, S. Hosseinalipour, N. Michelusi, C.G. Brinton, Delay-aware hierarchical federated learning, IEEE Trans. Cogn. Commun. Netw. 10 (2) (2023) 674–688.
[27] R. Zong, Y. Qin, F. Wu, Z. Tang, K. Li, Fedcs: Efficient communication scheduling in decentralized federated learning, Inf. Fusion 102 (2024) 102028.
[28] F. Zheng, Y. Sun, B. Ni, Fedaeb: Deep reinforcement learning based joint client selection and resource allocation strategy for heterogeneous federated learning, IEEE Trans. Veh. Technol. 73 (6) (2024) 8835–8846.
[29] M.P. Uddin, Y. Xiang, B. Cai, X. Lu, J. Yearwood, L. Gao, ARFL: Adaptive and robust federated learning, IEEE Trans. Mob. Comput. 23 (5) (2023) 5401–5417.
[30] M. Ma, T. Li, X. Peng, Beyond the federation: Topology-aware federated learning for generalization to unseen clients, 2024, p. 9, arXiv preprint arXiv:2407.0494.
[31] L. Shen, Q. Yang, K. Cui, Y. Zheng, X.Y. Wei, J. Liu, J. Han, FedConv: A learning-on-model paradigm for heterogeneous federated clients, in: Proceedings of the 22nd Annual International Conference on Mobile Systems, Applications and Services (MobiSys), 2024, pp. 398–411.