Over-the-Air Computation Enabled Semi-Asynchronous Wireless Federated Learning

Zijian Zheng, *Student Member, IEEE,* Yansha Deng, *Senior Member, IEEE,* Wenqiang Yi, *Member, IEEE,* Hyundong Shin, *Fellow, IEEE* and Arumugam Nallanathan, *Fellow, IEEE*

***Abstract*—The emerging field of federated learning (FL) holds significant promise for advancing edge intelligence while preserving data privacy. However, as FL systems scale or become more heterogeneous, challenges such as spectrum scarcity and the straggler problem arise. To address these issues, this paper proposes SA-AirFed, a semi-asynchronous FL architecture compatible with Over-the-Air Computation (AirComp). We develop an efficient scheduling scheme that meets AirComp's requirements and analyze the factors affecting convergence under the Lipschitz-Smooth condition. Building on insights from the convergence analysis, we design an adaptive algorithm that mitigates staleness from semi-asynchronous aggregation and noise from AirComp by dynamically adjusting aggregation weights, formulated as a convex quadratic programming problem. Experimental results on MNIST and CIFAR-10 demonstrate that SA-AirFed significantly reduces wall-clock training time while achieving greater robustness compared to baseline models.**

***Index Terms*—Semi-asynchronous federated learning, over-the-air computation, aggregation optimization.**

I. Introduction

Unlike traditional communication systems, the new generation of wireless networks, including 5G and 6G, incorporates edge intelligence as a fundamental component, enabling support for diverse applications such as virtual reality, augmented reality, connected vehicles, and smart Internet of Things (IoTs). Meanwhile, federated learning (FL) [2] offers a practical framework for implementing edge intelligence while safeguarding user privacy.

FL can be classified into synchronous (sync), asynchronous (async), and semi-asynchronous (semi-async) modes based on the parallelism exhibited by clients during training tasks [3]. As illustrated in Fig. 1, the sync mode was initially explored by early research [4] and has since been widely studied in areas such as communication architecture [5], resource allocation [6], and convergence speed [7]. In sync FL, all selected clients receive the latest model parameters simultaneously and begin training. However, due to variations in computation time, faster clients must remain idle, waiting for slower ones to finish, a phenomenon known as the straggler problem[1]. This issue decreases time utilization efficiency and slows down convergence.

An earlier version of this paper was presented in part at the 2023 IEEE Global Communications Conference [1].

[1]In computer science literature, the word "straggler" always refers to a device with low computation capacity, while in communication literature, it is usually used to describe device with poor channel condition. In this paper, we consider more about the parallelism. So we define the "straggler" the device which cannot update their local model in-time because of both communication and computation latency.

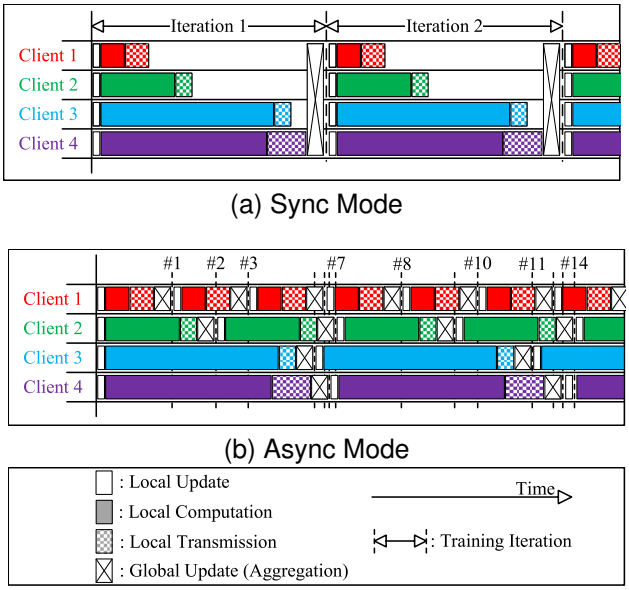


Fig. 1: Workflow of both the sync and async FL.

To address the straggler problem, the async mode has been proposed, introducing a scheduling mechanism that eliminates the need for waiting. This concept can be traced back to the Hogwild! algorithm [8], which was designed to accelerate stochastic gradient descent in distributed machine learning. Compared to the common distributed machine learning, FL always relies on wireless communication systems rather than data bus between servers to transmit training data, leading to more severe communication bottlenecks. In async mode [9], clients upload their parameters immediately after completing computations, allowing the next training round to begin without delay. However, in async mode, the parameter server (PS) often deals with outdated model parameters, a challenge known as the staleness problem. These stale updates contain outdated gradients, which will reduce the effectiveness of the model optimization. Figure 1 illustrates the differences between sync and async modes. In the sync mode example, clients $2-4$ are stragglers, causing delays for others. In contrast, in the async mode, updates from clients $2-4$ always suffer from staleness because they rely on gradients derived from models trained several iterations earlier.

In the realm of async FL, several methods have been developed to address the issue of staleness. In FedAsync [9], a memorization-based aggregation technique is used to blend the current model with the newly updated model.

This approach has been shown to be effective and is widely adopted in other async FL schemes. FedBuff [10] extends the memorization aggregation by utilizing a buffer array to store multiple updated models over time, averaging them for aggregation. ASO-Fed [11] modifies the objective function on the client side, making the updates more robust against staleness. It is important to note that these techniques cannot entirely eliminate the occurrence of staleness; instead, they aim to mitigate its impact through well-designed aggregation strategies.

Choosing between sync and async modes involves a trade-off between aggregation accuracy and time utilization efficiency. The sync mode provides more accurate gradients but results in time wasted waiting for stragglers, leading to longer iteration times. In contrast, the async mode improves time utilization but suffers from stale gradients, requiring more iterations to reach convergence. To balance the strengths and weaknesses of both approaches, a hybrid approach between sync and async, known as the semi-asynchronous (semi-async) mode, has been explored.

The concept of semi-async learning is analogous to the Stale Synchronous Parallel (SSP) algorithm used in distributed machine learning, which imposes a maximum staleness limit in model aggregation [12]. When the staleness of all clients remains below this limit, they operate in async mode. However, when a client's staleness reaches the set threshold, faster clients must wait for the slowest client until the staleness condition is resolved. Several semi-async FL frameworks have been developed based on this design principle. In SAFA [13], clients are categorized into up-to-date clients and tolerable clients. Up-to-date clients provide accurate updates to enhance computational efficiency, while tolerable clients operate asynchronously to improve time efficiency. In PORT [14], a staleness bound similar to SSP is used as a hyper-parameter to balance the level of asynchrony. And a push-pull mechanism is implemented to temporarily halt the computation of slower clients, prompting them to upload their data sooner and maintain system parallelism. FedAT [15] employs a clustered aggregation technique in which clients are grouped into tiers: clients within the same tier operate synchronously, while clients across different tiers work asynchronously. In this case, the staleness bound is naturally defined by the tier number. Building on this, TTFed [16] shifts the trigger mechanism for intra-tier aggregation from an event-based approach to time-slot polling. This change alleviates communication system strain and simplifies resource allocation.

It is important to note that all the FL frameworks mentioned above are built on digital communication, which demands spectrum resources depending on the scale of the system. However, communication has been identified as a critical bottleneck in scalable FL systems [17]. When considering the communication time required for clients to upload their models as part of the overall training cost, it becomes clear that, although async and semi-async FL improve computational efficiency, some of these gains may be offset by their communication disadvantages. Compared to the sync mode, async and semi-async modes allow clients to perform more model computation iterations, which also significantly increases the frequency of data upload. Additionally, the randomness in when clients transmit data introduces challenges for resource allocation in designing communication protocols. The resulting increase in communication overhead not only diminishes the low-latency advantage of async and semi-async modes but also limits their scalability, hindering their practical applicability.

To address the communication bottleneck in FL and improve spectrum utilization efficiency, over-the-air computation (AirComp) has been proposed as an alternative to digital communication for FL model transmission [5], [18]. In AirComp communication, clients use analog amplitude modulation. When all clients transmit on the same frequency band simultaneously, their modulated symbols collide in the wireless channel. Although this collision prevents the recovery of individual client symbols, their aggregated signal can be naturally obtained by the receiver. As a result, no matter how many clients a FL system has, the required bandwidth keeps the same. In recent years, numerous studies have focused on the application of AirComp in FL scenarios, aiming to optimize the impact of heterogeneity on training and mitigate the bottleneck effect. For instance, [19] proposes a method for training multiple AirComp FL tasks in parallel within a cellular communication system and employed a greedy algorithm to optimize device selection, thereby reducing the adverse impact of slow devices and poor-channel devices on the system. Meanwhile, [20] relaxes the power alignment requirement in AirComp and introduced the concept of Soft Straggler Alignment, which alleviates the bottleneck effect caused by the device with the worst channel conditions. Besides, both [19] and [20] investigate the utilization of MIMO beamforming to enhance transmission. While [21] leverages the statistical correlations of heterogeneous data to design a precoding scheme, achieving lower transmission errors and reduced computational complexity.

The aforementioned studies have addressed many heterogeneity-induced issues in AirComp, while there are still severe heterogeneity challenges from the FL task aspect when applying AirComp to FL scenarios. A straightforward approach would be to incorporate async or semi-async techniques into AirComp FL. However, their compatibility presents some difficulties. Since AirComp relies on the superposition of signal, clients must transmit data under a strict synchronization mechanism to ensure correct aggregation, which is incompatible with most async or semi-async workflows. Additionally, AirComp introduces noise into the model, posing a greater risk of divergence for async and semi-async FL together with the staleness issue. Many existing studies have investigated the async problem in AirComp-based FL. For instance, [22] explores two methods for estimating transmitted signals under imperfect clock synchronization, while [23] proposes a Bayesian approach to estimate misaligned AirComp transmissions. [24] proposes a digital AirComp scheme which has better compatibility with existing wireless communication systems compared with analog AirComp based on the Unsourced Massive Access protocol. In these studies, the transmission time differences among different clients are typically within a

few symbol durations, approximately on the order of tens of microseconds. These differences are caused by imperfect clock synchronization. The system still operates in a sync FL manner. However, to make AirComp compatible with async or semi-async FL, the transmission time differences among clients often span several training iterations, ranging from several minutes to tens of minutes, which makes our study on AirComp FL with semi-async aggregation fundamentally different from these works in both focus and research methodology.

To address the aforementioned problems, we propose a novel semi-async FL framework which supports AirComp. Our key contributions are as follows:

- We design a working schedule for the proposed framework, which meets the requirements of AirComp.
- We analyze the factors which may influence the convergence speed of the loss function under the Lipschitz-Smooth (L-Smooth) assumption.
- Based on the conclusions of the convergence analysis, we find the power control and model aggregation scheme for the framework. To get the optimal solution, we build an optimization problem and simplify it into a convex quadratic programming (QP) which can be solved within a complexity slightly higher than cubic time.
- Experiments show that our framework can significantly shorten the wall-clock time of training and effectively counteract staleness and noise.

The rest of this paper is organized as follows. Section II introduces the system model. The convergence analysis of SA-AirFed is given in Section III. The strategy optimization is demonstrated in Section IV. For clarity, we summarize the details to implement SA-AirFed in Algorithm 1. Section V presents the numerical results, and Section VI concludes this paper. The main notations used in this paper are summarized in Table I.

TABLE I: Notation Summary

Notation	Definition
$\mathcal{S}_{\text{tot}}, S_{\text{tot}}$	Set of all clients, size of $\mathcal{S}_{\text{tot}}$
$\mathcal{S}_m, S_m$	Set of clients in tier m, size of $\mathcal{S}_m$
$\mathcal{D}_u, D_u$	Local dataset of client u, size of $\mathcal{D}_u$
$\mathcal{D}_m, D_m$	Combined dataset of all clients in tier m, size of $\mathcal{D}_m$
$\mathbf{w}_{\text{G}}^{(k)}, \mathbf{w}_u^{(k)}$	Global and local parameters of client u at time slot k
$\mathbf{g}_{\text{G}}^{(k)}, \mathbf{g}_u^{(k)}$	Global and local gradient of client u derived from the parameters of time slot k
$F(\mathbf{w}), f(\mathbf{w})$	Global and sample-wise empirical risk of model characterized by parameter $\mathbf{w}$
d	Number of trainable parameters of the model
$\mathbf{x}_u^{(k)}$	Message to send by client u at time slot k, equals the delayed gradient after client-side aggregation weighting
$\mathbf{y}^{(k)}$	Message received by server at time slot k from AirComp, equals the result of client-side aggregation including noise and channel mismatch error
$\mathbf{u}^{(k)}$	Update vector which is used as the estimated gradient in k-th time slot to update the global model
$\alpha_m^{(k)}, \beta_u^{(k)}$	Inter-tier aggregation weight of tier m and intra-tier aggregation weight of client u, at time slot k
M, N	Total number of ties, max order of historical gradient used in server aggregation referring to the highest number of past gradient iterations
$s_\tau^{(k)}$	Server-side aggregation weight of τ-th order historical gradient at time slot k
$\mathbf{A}^{(k)}, \mathbf{B}_u^{(k)}$	Rx decoder matrix and Tx encoder matrix on client u for AirComp, at time slot k
$\varepsilon^{(k)}, \varepsilon_{\max}$	The squared Euclidean distance between the ideal transmission with perfect channel alignment and truncation and the real transmission, at time slot k and the maximum value across all time slots
$\mathbf{n}^{(k)}$	Noise received by server at time slot k
η	Learning rate
ΔT	Duration of each time slot

II. System Model

We consider a FL system, where clients from set $\mathcal{S}_{\text{tot}} = \{u_1, u_2, \cdots\}$ train a global model collaboratively with the help of a PS. Each client u_j keeps its own training data denoted by $(\mathbf{x}_i, y_i) \in \mathcal{D}_u$ with D_u samples. The whole available dataset for the FL system is the combination of all clients' datasets with D_{tot} samples, denoted by $\mathcal{D}_{\text{tot}} = \mathcal{D}_1 \cup \cdots \cup \mathcal{D}_{S_{\text{tot}}}$, where the numbers in the subscript represent the index of tiers or clients while "tot" represents the collective of all clients.

The goal of an FL system is to optimize the global model on its trainable parameters to achieve the minimum empirical risk:

$$\min_{\mathbf{w}} \quad F(\mathbf{w}) = \sum_{u \in \mathcal{S}_{\text{tot}}} \sum_{(\mathbf{x}_i, y_i) \in \mathcal{D}_u} \frac{1}{D_{\text{tot}}} f(\mathbf{w}; \mathbf{x}_i, y_i), \tag{1}$$

where $f(\cdot)$ is the sample-wise risk function, evaluating the model quality characterized by parameter $\mathbf{w}$ when tested with input $\mathbf{x}_i$ and the correct answer y_i. The expression in (1) indicates that the global loss function in FL is the average of the sample-wise loss functions across all samples in the datasets owned by all clients. It can also be expressed in the following form in some FL researches:

$$F(\mathbf{w}) = \sum_{u \in \mathcal{S}_{\text{tot}}} \frac{D_u}{D_{\text{tot}}} F_u(\mathbf{w}), \tag{2}$$

where $F_u(\mathbf{w})$ is the local empirical risk of client u and is defined as

$$F_u(\mathbf{w}) = \sum_{(\mathbf{x}_i, y_i) \in \mathcal{D}_u} \frac{1}{D_u} f(\mathbf{w}; \mathbf{x}_i, y_i). \tag{3}$$

A. Semi-Async Over-the-Air Federated Learning

Our research builds upon and extends the work of TTFed [16], a time-triggered semi-async FL approach. Before explaining the time-triggered scheme in detail, we present several definitions for clarification.

Definition 1 (Time Slot) We use "time slot" to indicate the iterations during model training, indexed by the superscript $.^{(k)}$ in our notation. We choose the name "time slot" instead of "iteration" or "round" to emphasize its correspondence with wall-clock time. The change of index k is driven solely by time rather than the model update events in conventional event-triggered settings, given by:

$$k = \left\lceil \frac{t}{\Delta T} \right\rceil, \tag{4}$$

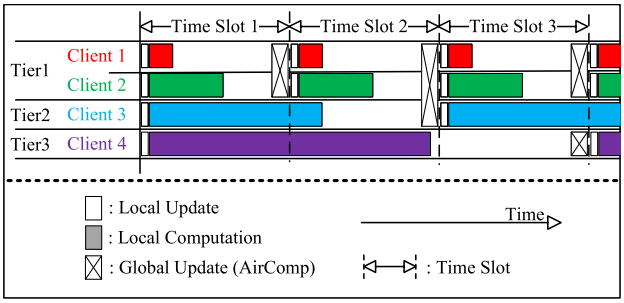


Fig. 2: Workflow of SA-AirFed.

where ΔT is a hyper-parameter called time slot duration.

Definition 2 (Available Client) Due to variations in computing capacity and dataset sizes among clients, their computation latency can differ significantly, leading to both busy and idle clients, with the latter having already completed their tasks. Clients that are able to update their computation results within a given time slot are referred to as available clients. These clients must have their computations completed and have sufficient time remaining to transmit data. The indicator $\mathbb{1}_u^{(k)} \in \{1, 0\}$ is used to represent this state, where client u is considered available in time slot k if the indicator is 1.

Definition 3 (Tier) We define a tier as a subset of clients with similar computation and communication latency. Formally, $\mathcal{S}_m$ represents the set of clients that require m time slots to complete a model update, i.e.:

$$\mathcal{S}_m = \left\{ u \;\middle|\; \mathbb{1}_{\mathrm{L},u}^{(\tau)} = 1;\ \tau = m, 2m, \cdots \right\},\ m = 1, 2, \cdots, M. \quad (5)$$

When the time slot index k is divisible by the tier number m, all clients within that tier are considered to be in an available state. We refer to the m-th tier as the available tier, which is denoted by:

$$\mathbb{1}_m^{(k)} = 1. \quad (6)$$

The working principle of TTFed is described as follows. A time-based trigger mechanism for aggregation is established by dividing the entire training process into multiple time slots. For client u, the decision to participate in the k-th aggregation depends on whether it can complete its computation and communication before the end of time slot k. Clients are naturally classified into tiers based on their participation patterns, which are determined by the ratio of their latency to the system's time slot duration. Clients within the same tier use sync aggregation, allowing AirComp to enhance communication efficiency, while aggregation across different tiers remains async, reducing the idle time caused by waiting for stragglers.

Our SA-AirFed retains most of the characteristics of TTFed, with the main modification being the alignment of the communication sessions. In TTFed, clients transmit their updated data immediately after completing computation. However, in our design, to make AirComp practical, the communication session must be aligned with the end of each time slot. This ensures that all available clients transmit data simultaneously. The working flow of SA-AirFed is illustrated in Fig. 2. Each time slot has three sessions:

1) Global Broadcast: At the beginning of each time slot, the PS uses the gradient descent (GD) algorithm to update the global model parameters, $\mathbf{w}_{\mathrm{G}}^{(k)}$, with the update vector obtained by aggregating the previously local gradients, denoted as $\mathbf{u}^{(k)}$. The learning rate η is a hyperparameter that determines the step size for adjusting the model parameters. The update expression for $\mathbf{w}_{\mathrm{G}}^{(k)}$ is as follows:

$$\mathbf{w}_{\mathrm{G}}^{(k)} = \mathbf{w}_{\mathrm{G}}^{(k-1)} - \eta \cdot \mathbf{u}^{(k)}. \quad (7)$$

After the update, the PS broadcasts the updated global model to the entire system. The newly received global model is used for gradient computation in the subsequent work sessions by clients marked as available in the previous time slot. For other clients still occupied with computation, this broadcast is ignored to enable parallelism.

2) Local Computation: In a typical FL training scenario, gradients are estimated by applying the global model to local data. However, in SA-AirFed, gradients from different tiers correspond to global model versions from different time slots. Based on the definition of tier numbers, the timeliness of each client's uploaded gradient can be tracked. Specifically, for an available client u in time slot k, its computed gradient is based on the global model from time slot $k-m$, where m is the tier number of client u, i.e.:

$$\mathbf{g}_u^{(k-m)} = \sum_{(\mathbf{x}_i, y_i) \in \mathcal{D}_u} \frac{1}{D_u} \nabla f\left(\mathbf{w}_{\mathrm{G}}^{(k-m)}; \mathbf{x}_i, y_i\right). \quad (8)$$

3) Communication: Clients that have completed gradient computation transmit their gradients simultaneously over the same frequency band via AirComp, which ensures that the required spectrum resources remain constant regardless of the number of clients. In contrast, in digital orthogonal communication, achieving sufficiently real-time transmission requires allocating more spectrum to accommodate a larger number of clients. As a result, AirComp enhances the scalability of the system. A power control algorithm is employed to equalize the received signal strength from each client and normalize the channel coefficient. The power control in AirComp generally consists of two components, which are applied on both the transmitter side (client) and the receiver side (PS). To maintain generality, we represent these components as linear mappings, denoted by the matrices $\mathbf{B}_u^{(k)}$ and $\mathbf{A}^{(k)}$. The matrix $\mathbf{B}_u^{(k)}$ is client-specific and used to compensate for channel heterogeneity, ensuring that the amplitudes of the transmitted signals are aligned at the receiver. The matrix $\mathbf{A}^{(k)}$, which remains the same for all clients, is used on the receiver side to amplify the received signal while ensuring that the transmit power on the client side satisfies the power constraint.

The received signal $\mathbf{y}^{(k)}$ is the aggregated update collected from available clients, denoted as

$$\mathbf{y}^{(k)} = \mathbf{A}^{(k)} \left(\sum_{m=1}^{M} \mathbb{1}_m^{(k)} \sum_{u \in \mathcal{S}_m} \mathbf{H}_u^{(k)} \mathbf{B}_u^{(k)} \mathbf{x}_u^{(k)} + \mathbf{n}^{(k)} \right), \quad (9)$$

where $\mathbf{H}_u^{(k)}$ represents the channel matrix between the u-th client and the receiver at the k-th time slot, $\mathbf{n}^{(k)}$ denotes the channel noise, and $\mathbf{x}_u^{(k)}$ is the normalized data transmitted by client u during the k-th time slot.

In our model, we consider OFDM-based SISO transmission. Due to the orthogonality between subcarriers, the channel matrix $\mathbf{H}_u^{(k)}$ is a $d \times d$ diagonal matrix, where d represents the size of the transmitted model gradients. The diagonal elements of $\mathbf{H}_u^{(k)}$ correspond to the channel coefficients of each symbol, which include both path loss and Rayleigh fading. The matrices $\mathbf{A}^{(k)}$ and $\mathbf{B}_u^{(k)}$ have the same shape as the channel matrix $\mathbf{H}_u^{(k)}$. Since $\mathbf{H}_u^{(k)}$ is a diagonal matrix, the pre-equalization algorithm ensures that the resulting matrices $\mathbf{A}^{(k)}$ and $\mathbf{B}_u^{(k)}$ are also diagonal. The diagonal elements of $\mathbf{A}^{(k)}$ and $\mathbf{B}_u^{(k)}$ represent the gain coefficients applied at the receiver and transmitter, respectively, to achieve power alignment. The additive noise $\mathbf{n}^{(k)}$ which is a vector with d elements is assumed to be Gaussian white noise with a power of σ_A.

B. Client Side Aggregation

At the end of a time slot, the available clients will weigh the gradients and upload them to the PS through AirComp. We use $\mathbf{x}_u^{(k)}$ to denote the weighted normalized gradients of u-th client at the end of k-th time slot, which is further expressed by

$$\mathbf{x}_u^{(k)} = \alpha_m^{(k)} \beta_u^{(k)} \frac{\mathbf{g}_u^{(k-m)}}{\left\| \mathbf{g}_u^{(k-m)} \right\|}, \tag{10}$$

where $\alpha_m^{(k)}$ represents the inter-tier aggregation weight, and $\beta_u^{(k)}$ denotes the intra-tier aggregation weight. For clients within the same tier, their inter-tier aggregation weights are identical, while intra-tier aggregation weights may vary across clients. In the studies of FL, the weight β is always set as the ratio Together, these two weights determine the overall gradient aggregation. The inter-tier aggregation weights help balance heterogeneity between different tiers and mitigate the staleness problem, while the intra-tier aggregation weights address heterogeneity among clients within the same tier.

In machine learning and FL scenarios, $\beta_u^{(k)}$ is often set as the ratio of the data volume on a device to the total dataset size, which aligns with the weight values in the empirical risk function. In SA-AirFed, since clients are divided into tiers, we define it as $\beta_u^{(k)} = D_u / D_m$, which is the data volume ratio on the tier containing the specific client.

Inter-tier aggregation weights are computed at the server and then transmitted to available clients via digital communication. This ensures that clients can apply these weights before performing AirComp. The adaptive aggregation algorithm for calculating these weights will be introduced in the subsequent section.

C. Server Side Aggregation

Compared to classic sync FL, the async and semi-async FL often require an additional aggregation step due to the modified pattern of client participation. The PS updates the global model using both the gradients currently uploaded by clients and the gradients received previously. This need arises from two key issues common in async and semi-async FL. First, each aggregation may involve only a small subset of users, leading to high variance in gradient estimation. Second, the gradients uploaded by users are often based on an outdated model, which also known as staleness.

Most async and semi-async FL methods use a recursive aggregation approach, which can be expressed as

$$\mathbf{u}^{(k)} = s^{(k)} \mathbf{y}^{(k)} + \left(1 - s^{(k)}\right) \mathbf{u}^{(k-1)}, \tag{11}$$

where $\mathbf{u}^{(k)}$ is the overall aggregated gradient used to update the global modal, $\mathbf{y}^{(k)}$ is the received data from clients by the PS involving local gradients, and $s^{(k)}$ is the aggregation weight.

This recursive aggregation method requires a buffer with only one vector element to store historical gradient and involves only a single parameter, $s^{(k)}$. It may simplify the algorithm's implementation and the tuning of hyperparameters. However, we find that this method faces problems when the staleness issue and noise become severe.

When expanding the recursive expression, it becomes clear that its value at any given moment is influenced by the values from all previous moments, i.e.,

$$\mathbf{u}^{(k)} = \prod_{\tau=0}^{k} \left(1 - s^{(k-\tau)}\right) \mathbf{u}^{(0)} + \sum_{\tau=1}^{k} s^{(\tau)} \prod_{l=0}^{\tau-1} \left(1 - s^{(k-l)}\right) \mathbf{y}^{(\tau)}. \tag{12}$$

If $\mathbf{y}^{(k)}$ does not contain any components detrimental to model convergence, this may not pose a problem. However, in the context of SA-AirFed, $\mathbf{y}^{(k)}$ is influenced not only by transmission errors and noise from AirComp but also by outdated local gradients. Therefore, although recursive aggregation only requires a single parameter, $s^{(k)}$, the actual impact of $s^{(k)}$ at each time step is very complex, which may not facilitate the algorithm when considering performance analysis and optimization. As shown in equation (12), $s^{(k)}$ and its historical values interact multiplicatively, leading to a series of complex expressions that complicate the problem.

In our SA-AirFed framework, we implement a memorized aggregation method with a finite memory length. We use N vector buffers to store historical information and apply $N+1$ coefficients to aggregate the data in these buffers with the newly received gradients, which is represented as

$$\mathbf{u}^{(k)} = \sum_{\tau=0}^{N} s_\tau^{(k)} \mathbf{y}^{(k-\tau)}, \tag{13}$$

where $s_\tau^{(k)}$ is the aggregation weight.

In our approach, we always set the value of N equal to the number of tiers M, since in SA-AirFed, the interval between two uploads from any tiers never exceeds M time slots. A buffer of length N is sufficient to store the most recent gradients from all clients. Additionally, the coefficients $s_\tau^{(k)}$ are adjustable for each tier and free from the constraints in (12), making the aggregation more flexible. In the following section, we will perform a joint optimization of the inter-tier aggregation coefficient $\alpha_m^{(k)}$ and the server-side aggregation coefficient $s_\tau^{(k)}$ by analyzing the model's convergence process.

D. Latency and Energy Model

In this section, we formulate the time and power cost model, including the computation, communication latency, and transmission power constraint.

1) *Computation Latency:* We adopt floating point operations (FLOPs) to measure computational task requirements. Denote the operational speed of the computation device as f_c. Let C denote the FLOPs the federated learning model requires, and D_u denote the dataset size on the client u. Thus, the computational latency of the client u is given by

$$\tau_u^{\mathrm{cp}} = \frac{CD_u}{f_c}. \tag{14}$$

2) *Communication Latency:* In order to ensure compatibility between AirComp and existing communication systems, we adopt the orthogonal frequency division multiplexing (OFDM) with total bandwidth B and N_M orthogonal sub-carriers. To simplify our expression, we ignore the time cost of the cyclic prefix. It should be noted that AirComp utilizes analog amplitude modulation rather than digital modulation like QAM or QPSK. We use each subcarrier to transmit one dimension of the gradient, thus the duration of each OFDM symbol is given by the reciprocal of sub-channel bandwidth, which is denoted as N_M/B. To transmit model parameters or gradients with a total length of q, $\lceil q/N_M \rceil$ OFDM symbols are required. Consequently, the communication latency for each AirComp transmission is given by

$$\tau^{\mathrm{cm}} = \left\lceil \frac{q}{N_M} \right\rceil \cdot \frac{N_M}{B}. \tag{15}$$

It is important to note that, apart from clients using AirComp for uploading gradients, all other data is transmitted via conventional digital communication. The data transmitted through digital communication includes: the updated global model parameters $\mathbf{w}_{\mathrm{G}}^{(k)}$, which is a one-dimensional vector of length d which equal to the number of trainable model parameters, and the inter-tier aggregation coefficients $\alpha_1^{(k)}, \alpha_2^{(k)}, \cdots, \alpha_m^{(k)}, \cdots, \alpha_M^{(k)}$ required for client-side aggregation, which consist of M scalar values used in the computation of (10). Since both of these parameters are transmitted from the server to the clients and all clients receive the same information, the server can utilize broadcasting to complete the transmission. This can be achieved by leveraging the idle AirComp bandwidth, ensuring that the communication process does not become a bottleneck as the number of clients increases. Besides, the channel estimation for pre-equalization can be finished parallel when the clients are computing. As a result, in our communication model, we neglect the time required for this part of the transmission, which is common in AirComp-related research [25].

3) *Communication Power:* We use the square of the L2-norm to represent the transmission power of AirComp, given by

$$P_u^{(k)} = \frac{\left\| \mathbf{B}_u^{(k)} \right\|_2^2}{\tau^{\mathrm{cm}}}. \tag{16}$$

In the model training, we aim to ensure that the total communication energy of all users within the system is bounded. To simplify the energy control algorithm and reduce unnecessary communication overhead, we use the average energy to constrain the total energy, which is a sufficient condition for the total communication energy to be bounded:

$$\sum_{u \in \mathcal{S}_{\mathrm{tot}}} \frac{1}{S_{\mathrm{tot}}} P_u^{(k)} \leq P_0. \tag{17}$$

E. Problem Formulation

To accelerate the convergence rate of SA-AirFed, we formulate an optimization problem to jointly minimize the global empirical risk, we write the objective function as

$$\text{(P1)} \ \min_{\mathbf{w}} \ F(\mathbf{w}) = \sum_{u \in \mathcal{S}_{\mathrm{tot}}} \sum_{(\mathbf{x}_i, y_i) \in \mathcal{D}_u} \frac{1}{D_{\mathrm{tot}}} f(\mathbf{w}; \mathbf{x}_i, y_i) \tag{18a}$$

$$s.t., \sum_{u \in \mathcal{S}_{\mathrm{tot}}} \frac{1}{S_{\mathrm{tot}}} P_u^{(k)} \leq P_0 \tag{18b}$$

$$\sum_{m=1}^{M} \mathbb{1}_m^k \alpha_m^{(k)} = 1 \tag{18c}$$

$$\alpha_m^{(k)} \geq 0 \tag{18d}$$

$$\sum_{\tau=1}^{n} s_\tau^{(k)} = 1 \tag{18e}$$

$$s_\tau^{(k)} \geq 0, \tag{18f}$$

where the optimization variable $\mathbf{w}$ represents the global model parameters. Equation (18b) specifies the average power constraints for wireless transmission, while equations (18c) and (18d) describe the affine constraints on inter-tier aggregation weights on the client side. Similarly, equations (18e) and (18f) define the affine constraints for server-side aggregation weights. These affine constraints are introduced to maintain the stability of the model parameter scale during training, while also simplifying the convergence analysis.

III. Convergence Analysis

To solve (P1), we conduct a convergence analysis on the SA-AirFed to better understand the impact of staleness, noise, and transmission error on the training process.

Before the convergence analysis, we make the following assumptions:

Assumption 1 (L-Smooth) We assume the global risk function is differentiable and its gradient is Lipschitz continuing with a non-negative constant L, i.e., $\forall \mathbf{w}_1, \mathbf{w}_2 \in \mathbb{R}^n$ where n is the size of the model parameter, $\exists L \geq 0$, it satisfies

$$\left\| \nabla F_{\mathrm{G}}(\mathbf{w}_1) - \nabla F_{\mathrm{G}}(\mathbf{w}_2) \right\| \leq L \left\| \mathbf{w}_1 - \mathbf{w}_2 \right\|. \tag{19}$$

Assumption 2 (Bounded Gradient Dissimilarity) We adopted a non-negative constant ζ to indicate the Euclidean distance at the k-th time slot between the global gradient and the local gradient on the client u, i.e.,

$$\left\| \mathbf{g}_u^{(k)} - \mathbf{g}_{\mathrm{G}}^{(k)} \right\| \leq \zeta, \qquad \left\| \mathbf{g}_u^{(k)} \right\| \leq \xi \left\| \mathbf{g}_{\mathrm{G}}^{(k)} \right\|, \tag{20}$$

where $\mathbf{g}_{\mathrm{G}}^{(k)}$ is the gradient of global model at k-th time slot. Similar assumptions are commonly found in machine learning and FL research, as seen in studies like [26], [27].

Assumption 3 (Bounded Gradient Change) Within a limited period $l \leq 4M$, the norm of gradient change can be bounded by a linear mapping, i.e.,

$$\left\|\mathbf{g}_{\mathrm{G}}^{(k-l)}\right\| \leq \gamma \left\|\mathbf{g}_{\mathrm{G}}^{(k-1)}\right\|, \quad \forall l \in \mathbb{Z} \cap [1, 4M]. \tag{21}$$

Similar assumptions are applied in other works on async and semi-async, like [15], [16].

Theorem 1 (Convergence Rate Bound of SA-AirFed) In the SA-AirFed algorithm, the upper bound for the update of the global empirical risk function between two adjacent time slots is as follows:

$$\begin{aligned}
&\mathbb{E}\left[F\left(\mathbf{w}_{\mathrm{G}}^{(k)}\right) - F\left(\mathbf{w}_{\mathrm{G}}^{(k-1)}\right)\right] \\
&\quad\leq \frac{\eta\left[6L^2(2M-1)^2\xi^2\gamma^2\eta^2 + L\eta - 2\right]}{2}\left\|\mathbf{g}_{\mathrm{G}}^{(k-1)}\right\|^2 + \frac{3\eta}{2}\zeta^2 \\
&\quad+ \frac{3\eta}{2}\sum_{\tau=0}^{N} s_\tau^{(k)}\varepsilon^{(k-\tau)} \\
&\quad+ 3\eta^3 L^2 \mathcal{A}_{\tau,m}^{(k)}\left[(\tau+m-1)\sum_{l=1}^{\tau+m-1}\sum_{p=0}^{N} s_\tau^{k-l}\varepsilon^{(k-l-p)}\right],
\end{aligned} \tag{22}$$

where $\varepsilon^{(k)}$ is the squared Euclidean distance between the ideal aggregation result and the real AirComp result $\mathbf{y}^{(k-\tau)}$ including channel misalignment error and noise in time slot k.

Proof: See Appendix B.

From Theorem 1, we note that when the following condition is satisfied, the factor of $\left\|\mathbf{g}_{\mathrm{G}}^{(k-1)}\right\|^2$ is negative, guaranteed by the property of quadratic function:

$$0 < \eta < \frac{\sqrt{1 + 48(2M-1)^2\xi^2\gamma^2} - 1}{12L(2M-1)^2\xi^2\gamma^2}. \tag{23}$$

Thus, we have

$$\begin{aligned}
&\mathbb{E}\left[F\left(\mathbf{w}_{\mathrm{G}}^{(K)}\right) - F\left(\mathbf{w}_{\mathrm{G}}^{(0)}\right)\right] \\
&\quad= \sum_{k=1}^{K}\mathbb{E}\left[F\left(\mathbf{w}_{\mathrm{G}}^{(k)}\right) - F\left(\mathbf{w}_{\mathrm{G}}^{(k-1)}\right)\right] \\
&\quad\leq \frac{\eta\left[6L^2(2M-1)^2\xi^2\gamma^2\eta^2 + L\eta - 2\right]}{2}\sum_{k=1}^{K}\left\|\mathbf{g}_{\mathrm{G}}^{(k-1)}\right\|^2 + \frac{3\eta K}{2}\zeta^2 \\
&\quad+ \frac{3\eta}{2}\sum_{\tau=0}^{N} s_\tau^{(k)}\varepsilon^{(k-\tau)} \\
&\quad+ 3\eta^3 L^2 \mathcal{A}_{\tau,m}^{(k)}\left[(\tau+m-1)\sum_{l=1}^{\tau+m-1}\sum_{p=0}^{N} s_\tau^{k-l}\varepsilon^{(k-l-p)}\right].
\end{aligned} \tag{24}$$

To simplify the analysis, we use the maximum value of $\varepsilon_{\max}$ to replace the complex affine transformation of it, i.e.,

$$\varepsilon_{\max} = \max_k\left(\varepsilon^{(k)}\right). \tag{25}$$

According to the property of the affine mapping, we have:

$$\begin{aligned}
&\frac{3\eta}{2}\sum_{\tau=0}^{N} s_\tau^{(k)}\varepsilon^{(k-\tau)} \\
&\quad+ 3\eta^3 L^2 \mathcal{A}_{\tau,m}^{(k)}\left[(\tau+m-1)\sum_{l=1}^{\tau+m-1}\sum_{p=0}^{N} s_\tau^{k-l}\varepsilon^{(k-l-p)}\right] \\
&\quad\leq \frac{3\eta NK\left[2L^2(2M-1)^2\eta^2 + 1\right]}{2}\varepsilon_{\max}.
\end{aligned} \tag{26}$$

By moving the negative term about $\left\|\mathbf{g}_{\mathrm{G}}^{(k-1)}\right\|^2$ to the left side and removing the factor, we end up with

$$\begin{aligned}
\min_k\left\|\mathbf{g}_{\mathrm{G}}^{(k)}\right\|^2 &\leq \sum_{k=1}^{K}\frac{1}{K}\left\|\mathbf{g}_{\mathrm{G}}^{(k)}\right\|^2 \\
&\leq \underbrace{\frac{2\left[F\left(\mathbf{w}^{(0)}\right) - F^*\right]}{-\eta K\left[6L^2(2M-1)^2\xi^2\gamma^2\eta^2 + L\eta - 2\right]}}_{a(K)} \\
&\quad+ \underbrace{\frac{3N\left[2L^2(2M-1)^2\eta^2+1\right]\varepsilon_{\max} + 3\zeta^2}{-6L^2(2M-1)^2\xi^2\gamma^2\eta^2 - L\eta + 2}}_{b},
\end{aligned} \tag{27}$$

where F^* is the optimal value of the global risk function.

The results in (27) suggest that when the number of training rounds K is sufficiently large, the first term in (27), $a(K)$, is mathematically guaranteed to tend to zero due to the sufficiently large denominator, and the model's gradient will converge to the constant second term b, which is related to factors of channel misalignment error, noise, and the degree of gradient dissimilarity. In conclusion, the SA-AirFed converges at a $\mathcal{O}(1/K)$ convergence rate with a small constant learning rate condition shown in (23).

We consider b as a function of η to analyze its numerical variation $b(\eta)$. By computing its derivative, we obtain:

$$b'(\eta) = \frac{AD\eta^2 + (2AE - 2BC)\eta - BD}{\left(C\eta^2 + D\eta + E\right)^2}, \tag{28}$$

where A and B represent the quadratic coefficient and constant term of the numerator with respect to the variable η, while C, D and E correspond to the quadratic coefficient, linear coefficient, and constant term of the denominator, respectively. This can be expressed as:

$$\begin{aligned}
A &= 6NL^2(2M-1)^2\varepsilon_{\max} \\
B &= 3N\varepsilon_{\max} + 3\zeta^2 \\
C &= -6L^2(2M-1)^2\xi^2\gamma^2 \\
D &= -L \\
E &= 2.
\end{aligned} \tag{29}$$

According to the properties of quadratic functions, $b'(\eta)$has a single root in the real number domain. When positive real number η is smaller than this root, the derivative is positive; when η is larger than this root, the derivative is negative. Therefore, we find that function $b(\eta)$ first increases and then decreases in the real number domain. Considering the valid range of η defined in (23), the right endpoint of this range coincides with a singularity of the function $b(\eta)$, and its left-hand limit approaches positive infinity. Hence, $b(\eta)$ is monotonically increasing over its defined domain. As a

result, SA-AirFed inevitably encounters an error floor, with its minimum value achieved when η is close to zero, given by:

$$b_{\min} = \frac{\varepsilon_{\max} + 3\zeta^2}{2}, \tag{30}$$

where $\varepsilon_{\max}$ is related to the channel condition and ζ is related to the data heterogeneous of the non-IID datasets among clients.

It is important to note that our analysis derives an upper bound on the model gradient. To simplify the results, we apply certain scaling which will amplify the final outcome. For instance, we replace a complex affine transformation with the maximum value of the transmission error, which leads to the appearance of $\varepsilon_{\max}$ in the final result. However ideally, the model gradient is influenced by the accumulated transmission errors $\varepsilon^{(k)}$ over multiple iterations, which undergo complex summation and averaging rather than simply reflecting the worst-case error $\varepsilon_{\max}$. Therefore, our analysis does not imply the existence of a bottleneck effect in this process.

Our conclusion can be seen as a generalization of the convergence analysis for sync FL, aligning well with the well-known convergence analysis results of sync FL, such as [7]. With the number of tiers $M = 1$, SA-AirFed can degenerate into sync FL with AirComp. And with the worst-case transmission error $\varepsilon_{\max} = 0$, SA-AirFed will transform into FL with digital communication.

Important insights can be gained from (27) about how noise, misalignment error, and staled data affect the convergence rate and accuracy of SA-AirFed.

Remark 1 (Noise and Transmission Error) To simplify the results, our upper bound analysis does not explicitly examine the relationship between model convergence and noise. Instead, it focuses on how noise affects the model's accuracy after convergence. Specifically, a larger $\varepsilon_{\max}$ results in an increased term b, indicating a greater deviation from the optimal solution. Here, $\varepsilon^{(k)}$ and $\varepsilon_{\max}$ represent the squared Euclidean distance between the ideal AirComp transmission outcome and the actual result, which is influenced by imperfections in power control and noise. As these terms are directly proportional to the transmission mean square error (MSE), optimizing the transmission MSE can help mitigate the negative impact of noise and transmission errors on model convergence, ultimately reducing the value of b.

Remark 2 (Staleness) Staleness does not explicitly appear in equation (27) but is indirectly represented through a combination of factors such as the tier number M, factors related to gradient dissimilarity such ζ and ξ, and the factor γ which is related to gradient update. This suggests that staleness is a multifaceted issue influenced by several factors. A higher tier number, greater gradient dissimilarity, and more pronounced gradient changes all contribute to increased staleness. In addition to reducing the denominator of $a(K)$, thereby slowing the model's convergence rate, staleness increases the numerator and decreases the denominator of term b, negatively impacting model accuracy. Furthermore, staleness affects the convergence condition outlined in (23), increasing the risk of non-convergence.

Remark 3 (Tier Number Trade-off) The conclusions in Remark 2 indicate that a larger number of tiers M exacerbates the staleness issue, requiring more training rounds for model convergence and resulting in lower accuracy after convergence. However, a larger M also implies scheduling with smaller time slots, reducing the waiting time for fast clients after completing their computations and decreasing the wall-clock time per iteration. Therefore, the design of M involves a trade-off: if M is either too large or too small, the total wall-clock time required for training will increase.

IV. Optimal Power Control And Aggregation Algorithm

Inspired by Remark 1 2 and 3, we decompose the original problem (P1) in (18) into two sub-problems: (P2) minimizing the MSE of AirComp and (P3) minimizing staleness. In sub-problem (P2), the optimization variables are the encoder and decoder matrices $\mathbf{A}^{(k)}$ and $\mathbf{B}_u^{(k)}$, while in the second sub-problem, the variables are the aggregation coefficients $s_\tau^{(k)}, \alpha_m^{(k)}$, and $\beta_u^{(k)}$. These two problems are naturally decoupled, as they do not share any common optimization variables.

The first sub-problem can be addressed using the algorithm proposed in [28], which studied optimal encoder and decoder design for minimizing transmission MSE under a time-averaged transmission power constraint. The sub-problem can be expressed by:

$$\text{(P2)} \quad \min_{\mathbf{A}^{(k)}, \mathbf{B}_u^{(k)}} \ \varepsilon^{(k)} = \left\| \mathbf{y}^{(k)} - \left(\sum_{m=1}^{M} \mathbb{1}_m^{(k)} \sum_{u \in \mathcal{S}_m} \mathbf{x}_u^{(k)} \right) \right\|^2 \tag{31}$$
$$s.t., \qquad (18b).$$

Given the power constraint in (18b), the optimal decoder matrix $\mathbf{A}^{(k)}$ defined in (9) can be given as

$$\mathbf{A}^{(k)} = \text{diag}_i \left[\sqrt{\frac{1}{P_0}} \sum_{m=1}^{M} \mathbb{1}_m^{(k)} \sum_{u \in \mathcal{S}_m} \left(\frac{P_0 h_{u,i}^{(k)}}{\sigma_A^2 + P_0 (h_{u,i}^{(k)})^2} \right)^2 \right], \tag{32}$$

where σ_A^2 is the power of the AWGN and $h_{u,i}^{(k)}$ is the channel coefficient of the client u for the i-th symbol.

Similarly, the optimal decoder matrix $\mathbf{B}_u^{(k)}$ can be given as

$$\mathbf{B}_u^{(k)} = \text{diag}_i \left[\sqrt{\frac{P_0 \left(\frac{P_0 h_{u,i}^{(k)}}{\sigma_A^2 + P_0 (h_{u,i}^{(k)})^2} \right)^2}{\sum_{m=1}^{M} \mathbb{1}_m^{(k)} \sum_{u \in \mathcal{S}_m} \left(\frac{P_0 h_{u,i}^{(k)}}{\sigma_A^2 + P_0 (h_{u,i}^{(k)})^2} \right)^2}} \right]. \tag{33}$$

And the transmission error can be expressed as

$$\varepsilon^{*(k)} = \left\| \text{diag}_i \left[\sum_{m=1}^{M} \mathbb{1}_m^{(k)} \sum_{u \in \mathcal{S}_m} \frac{\sigma_A^2}{\sigma_A^2 + P_0 (h_{u,i}^{(k)})^2} \right] \right\|^2, \tag{34}$$

where $\varepsilon^{*(k)}$ is defined as the optimal squared Euclidean distance between the ideal transmission and AirComp result, which is used in (22) and appears as an constant in the result of convergence analysis in (27).

However, the second sub-problem presents more challenges. Although we derive a convergence upper bound for SA-AirFed in (27) and emphasized the impact of staleness on convergence

speed and accuracy, this result is not directly applicable to the design of optimization algorithms. The primary reasons for this include:

1) The convergence analysis is based on asymptotic analysis, where the focus is on the order of various variables rather than their exact coefficients. Moreover, we introduce certain unmeasurable constants, such as L from the L-smooth assumption and ζ from the bounded gradient dissimilarity assumption. Ignoring these factors may compromise the accuracy of optimization algorithms, potentially hindering improvements in model performance.
2) The convergence analysis relies on an unrolled expression to capture the effects over the entire training process, whereas real-time optimization requires a recursive expression to leverage historical information and conserve memory. For complex models, it is impractical to record the model's state at every step.
3) Although the staleness issue is well known in async and semi-async FL, there is no widely adopted metric to quantify the degree of staleness. Therefore, it is necessary to first identify an appropriate optimization objective for sub-problem (P3).

To address these three issues, we utilize the L-Smooth property and have:

$$\begin{aligned}
&\mathbb{E}\left[F\left(\mathbf{w}_{\mathrm{G}}^{(k)}\right)-F\left(\mathbf{w}_{\mathrm{G}}^{(k-1)}\right)\right]\\
&\overset{(a)}{\leq}\mathbb{E}\left(\mathbf{w}_{\mathrm{G}}^{(k)}-\mathbf{w}_{\mathrm{G}}^{(k-1)}\right)^{\mathrm{T}}\mathbf{g}_{\mathrm{G}}^{(k-1)}+\frac{L}{2}\mathbb{E}\left\|\mathbf{w}_{\mathrm{G}}^{(k)}-\mathbf{w}_{\mathrm{G}}^{(k-1)}\right\|^2\\
&\overset{(b)}{=}-\eta\cdot\mathbb{E}\left(\mathbf{u}^{(k)}\right)^{\mathrm{T}}\mathbf{g}_{\mathrm{G}}^{(k-1)}+\frac{L\eta^2}{2}\mathbb{E}\left\|\mathbf{u}^{(k)}\right\|^2\\
&\overset{(c)}{=}\underbrace{-\frac{\eta}{2}\left\|\mathbf{g}_{\mathrm{G}}^{(k-1)}\right\|^2}_{T_1}+\underbrace{\left(-\frac{\eta-L\eta^2}{2}\mathbb{E}\left\|\mathbf{u}^{(k)}\right\|^2\right)}_{T_2}\\
&\quad+\underbrace{\frac{\eta}{2}\mathbb{E}\left\|\mathbf{u}^{(k)}-\mathbf{g}_{\mathrm{G}}^{(k-1)}\right\|^2}_{T_3}.
\end{aligned}\tag{35}$$

where the explanation of step $(a),(b)$ and (c) is shown in (51), we find when η is sufficiently small, both T_1 and T_2 are negative, indicating that the loss function is decreasing. However, T_3 always remains positive, which hinders the decrease of the loss function. The further transformation in (52), (54) and (53) also shows that T_3 is not only a metric of staleness which appears in its component T_{32}, but also a combined metric of heterogeneous which can be found in T_{31} and metric of accumulated transmission error in T_{33}. Thus, we choose T_3 as the objective function of sub-problem (P3), and end up with:

$$\begin{aligned}
(\mathrm{P3})\ \min_{\mathbf{s},\vec{\alpha}}\ &\mathbb{E}\left\|\mathbf{u}^{(k)}-\mathbf{g}_{\mathrm{G}}^{(k-1)}\right\|^2 &\text{(36a)}\\
s.t.,\quad &(18\mathrm{b})-(18\mathrm{f}).
\end{aligned}$$

Based on the definition in equation (13), we can express the term inside the norm of the objective function as

$$\begin{aligned}
&\mathbf{u}^{(k)}-\mathbf{g}_{\mathrm{G}}^{(k-1)}\\
&=\sum_{\tau=0}^{n}s_\tau^{(k)}\sum_{m=1}^{M}\alpha_m^{(k-\tau)}\mathbb{1}_m^{(k-\tau)}\left(\mathbf{g}_{\mathrm{G}}^{(k-l-m)}-\mathbf{g}_{\mathrm{G}}^{(k-1)}\right)\\
&\quad+\sum_{\tau=0}^{n}s_\tau^{(k)}\sum_{m=1}^{M}\alpha_m^{(k-\tau)}\mathbb{1}_m^{(k-\tau)}\\
&\qquad\cdot\left(\sum_{u\in\mathcal{S}_m}\beta_u^{(k-l)}\mathbf{g}_u^{(k-l-m)}-\mathbf{g}_{\mathrm{G}}^{(k-l-m)}\right).
\end{aligned}\tag{37}$$

This optimization objective is still intractable to be used since it requires the ideal global gradient $\mathbf{g}_{\mathrm{G}}^{(k)}$ and also necessitates the results of the current upload $\sum_{\tau=0}^{n}s_\tau^{(k)}\sum_{m=1}^{M}\alpha_m^{(k-\tau)}\mathbb{1}_m^{(k-\tau)}\mathbf{g}_u^{(k-l-m)}$ before the client completes it. To address this issue, we employ two approximations.

We first approximate the ideal global gradient to the gradient obtained through SA-AirFed, denoted as

$$\mathbf{g}_{\mathrm{G}}^{(k-\tau)}\approx\mathbf{u}^{(k-\tau)},\ \tau<k.\tag{38}$$

Furthermore, we substitute the current result with the gradient obtained from the previous SA-AirFed iteration, represented as

$$\mathbf{u}^{(k)}\approx\mathbf{u}^{(k-1)}.\tag{39}$$

We experimentally validated the reliability of these two approximations. Moreover, the use of aggregated gradients in place of ideal gradients has been adopted in several other studies as well, such as [29], [30].

It should be noted that, to ensure the approximation's accuracy, (39) cannot be approximated multiple times using recursion. When the time slot index is k, the estimation of $\mathbf{u}^{(k)}$ can only be obtained through the true value of $\mathbf{u}^{(k-1)}$, not through the approximated value, i.e., the true value of $\mathbf{u}^{(k-2)}$. Furthermore, (38) and (39) cannot be used consecutively on the same variable, meaning (37) cannot be directly approximated to zero by continuously applying (38) and (39), as this would result in the loss of necessary information.

With the proper use of the two approximations, we can rewrite the first term of the result in (37) as

$$\begin{aligned}
&\sum_{l=0}^{N}s_l^{(k)}\sum_{m=1}^{M}\alpha_m^{(k-l)}\mathbb{1}_m^{(k-l)}\left(\mathbf{g}_{\mathrm{G}}^{(k-l-m)}-\mathbf{g}_{\mathrm{G}}^{(k-1)}\right)\\
&\overset{(a)}{\approx}\sum_{l=0}^{N}s_l^{(k)}\sum_{m=1}^{M}\alpha_m^{(k-l)}\mathbb{1}_m^{(k-l)}\left(\mathbf{u}^{(k-l-m+1)}-\mathbf{u}^{(k)}\right)\\
&\overset{(b)}{=}\sum_{p=1}^{M+N}\left(\underbrace{\sum_{l=\max(0,p-M)}^{\min(N,p-1)}s_l^{(k)}\alpha_{p-l}^{(k-l)}\mathbb{1}_{p-l}^{(k-l)}\sum_{u\in\mathcal{S}_{p-l}}\beta_u^{(k-l)}}_{a_p^{(k)}}\right)\\
&\qquad\cdot\left(\mathbf{u}^{(k-p+1)}-\mathbf{u}^{(k)}\right)\\
&\overset{(c)}{=}\left(\mathbf{a}^{(k)}\right)^{\mathrm{T}}\mathbf{U}_{-}^{(k)},
\end{aligned}\tag{40}$$

where (a) results from the approximation demonstrated in equation (38), (b) is derived by interchanging the order of the two summations, and (c) is the result after vectorization,

where $\mathbf{a}^{(k)}$ is a column vector with $M+N$ rows, and the element in the p-th row is $a_p^{(k)}$. The matrix $\mathbf{U}_-^{(k)}$ has $M+N$ rows and the number of columns is equal to the size of the model gradients, denoted as d. The p-th row of matrix $\mathbf{U}_-^{(k)}$ is $\left(\mathbf{u}^{(k-p+1)} - \mathbf{u}^{(k)}\right)$.

Similarly, the second term in equation (37) can also be expressed as

$$\begin{aligned}
&\sum_{l=0}^{N} s_l^{(k)} \sum_{m=1}^{M} \alpha_m^{(k-l)} \mathbb{1}_m^{(k-l)} \sum_{u\in\mathcal{S}_m} \beta_u^{(k-l)} \mathbf{g}_u^{(k-l-m)} - \mathbf{g}_{\mathrm{G}}^{(k-l-m)} \\
&\approx \sum_{l=0}^{N} s_l^{(k)} \sum_{m=1}^{M} \alpha_m^{(k-l)} \mathbb{1}_m^{(k-l)} \sum_{u\in\mathcal{S}_m} \beta_u^{(k-l)} \mathbf{u}^{(k-l-m+1)} \\
&\quad - \sum_{l=0}^{N} s_l^{(k)} \sum_{m=1}^{M} \frac{D_m}{D_{\mathrm{tot}}} \sum_{u\in\mathcal{S}_m} \frac{D_u}{D_m} \mathbf{u}^{(k-l-m+1)} \\
&= \sum_{p=1}^{M+N} \left[a_p^{(k)} - \underbrace{\sum_{l=\max(0,p-M)}^{\min(N,p-1)} \frac{D_{p-l}}{D_{\mathrm{tot}}}}_{b_p^{(k)}} \right] \mathbf{u}^{(k-p+1)} \\
&= \left(\mathbf{a}^{(k)} - \mathbf{b}^{(k)}\right)^{\mathrm{T}} \mathbf{U}^{(k)},
\end{aligned} \tag{41}$$

where $\mathbf{b}^{(k)}$ is a vector similar to $\mathbf{a}^{(k)}$, with each row element denoted as $b_p^{(k)}$. The matrix $\mathbf{U}^{(k)}$ and the matrix $\mathbf{U}_-^{(k)}$ have the same shape, whose p-th row is $\mathbf{u}^{(k-p+1)}$.

Thus, the objective function can be simplified as

$$\left\| \mathbf{u}^{(k)} - \mathbf{g}_{\mathrm{G}}^{(k-1)} \right\|^2 = \left\| \mathbf{a}^{(k)} \left(\mathbf{U}_-^{(k)} + \mathbf{U} \right) - \mathbf{b}^{(k)} \mathbf{U}^{(k)} \right\|^2, \tag{42}$$

where both $\mathbf{U}^{(k)}$ and $\mathbf{U}_-^{(k)}$ can be obtained by replacing all rows whose content is $\mathbf{u}^{(k)}$ into $\mathbf{u}^{(k-1)}$ according to the second approximation, and $\mathbf{b}^{(k)}$ can be derived by collecting the sizes of each client's dataset.

Based on the definition of the vector $\mathbf{a}^{(k)}$ in equation (40), we can also express the vector $\mathbf{a}^{(k)}$ as the product of a vector and a matrix. We define the matrix $\mathbf{R}^{(k)}$, with $M+N$ rows and N columns, as

$$\mathbf{R}^{(k)} = \begin{pmatrix}
0 & 0 & \cdots & 0 \\
\alpha_1^{(k-1)}\mathbb{1}_1^{(k-1)} & 0 & \cdots & \vdots \\
\alpha_2^{(k-1)}\mathbb{1}_2^{(k-1)} & \alpha_1^{(k-2)}\mathbb{1}_1^{(k-2)} & \cdots & \vdots \\
\vdots & \alpha_2^{(k-2)}\mathbb{1}_2^{(k-2)} & \cdots & 0 \\
\alpha_M^{(k-1)}\mathbb{1}_M^{(k-1)} & \vdots & \cdots & \alpha_1^{(k-N)}\mathbb{1}_1^{(k-N)} \\
0 & \alpha_M^{(k-2)}\mathbb{1}_M^{(k-2)} & \cdots & \alpha_2^{(k-N)}\mathbb{1}_2^{(k-N)} \\
\vdots & \vdots & \ddots & \vdots \\
0 & 0 & \cdots & \alpha_M^{(k-N)}\mathbb{1}_M^{(k-N)}
\end{pmatrix}. \tag{43}$$

Thus, $\mathbf{a}^{(k)}$ can be given by

$$\mathbf{a}^{(k)} = s_0^{(k)} \mathbf{r}^{(k)} + \mathbf{R}^{(k)} \mathbf{s}_{0-}^{(k)}, \tag{44}$$

where $s_0^{(k)}$ is the coefficient corresponding to $\mathbf{u}^{(k)}$ in the server-side aggregation process, and $\mathbf{s}_{0-}^{(k)}$ is a vector of length n, consisting of other weights for server-side aggregation, excluding $s_0^{(k)}$, and $\mathbf{r}^{(k)}$ is a vector similar to a column in matrix $\mathbf{R}^{(k)}$, containing inter-tier aggregation coefficients and an indicator, given by

$$\mathbf{r}^{(k)} = \begin{pmatrix} \alpha_1^{(k)}\mathbb{1}_1^{(k)} & \alpha_2^{(k)}\mathbb{1}_2^{(k)} & \cdots \alpha_M^{(k)}\mathbb{1}_M^{(k)} & 0 & \cdots & 0 \end{pmatrix}^{\mathrm{T}}. \tag{45}$$

With the aforementioned simplifications and vectorization, the objection can be simplified as:

$$\begin{aligned}
&f(s_0^{(k)}, \mathbf{r}^{(k)}, \mathbf{s}_{0-}^{(k)}) \\
&= \left\| \left(s_0^{(k)} \mathbf{r}^{(k)} + \mathbf{R}^{(k)} \mathbf{s}_{0-}^{(k)} \right)^{\mathrm{T}} \left(\mathbf{U}_-^{(k)} + \mathbf{U}^{(k)} \right) - \mathbf{b}^{(k)} \mathbf{U}^{(k)} \right\|^2 \\
&= \frac{1}{2} \underbrace{\begin{pmatrix} s_0^{(k)}\mathbf{r}^{(\mathrm{k})} \\ \mathbf{s}_{0-}^{(k)} \end{pmatrix}^{\mathrm{T}}}_{\mathbf{x'}^{\mathrm{T}}} \underbrace{\begin{pmatrix} \mathbf{A}^{\mathrm{T}}\mathbf{A} & \mathbf{A}^{\mathrm{T}}\mathbf{B} \\ \mathbf{B}^{\mathrm{T}}\mathbf{A} & \mathbf{B}^{\mathrm{T}}\mathbf{B} \end{pmatrix}}_{\mathbf{Q}} \underbrace{\begin{pmatrix} s_0^{(k)}\mathbf{r}^{(\mathrm{k})} \\ \mathbf{s}_{0-}^{(k)} \end{pmatrix}}_{\mathbf{x'}} \\
&\quad + \underbrace{\begin{bmatrix} 2\mathbf{C}^{\mathrm{T}}\mathbf{A} \\ 2\mathbf{C}^{\mathrm{T}}\mathbf{B} \end{bmatrix}^{\mathrm{T}}}_{\mathbf{c}^{\mathrm{T}}} \underbrace{\begin{pmatrix} s_0^{(k)}\mathbf{r}^{(\mathrm{k})} \\ \mathbf{s}_{0-}^{(k)} \end{pmatrix}}_{\mathbf{x'}},
\end{aligned} \tag{46}$$

where $\mathbf{A} = \mathbf{U}_-^{(k)} + \mathbf{U}^{(k)}$, $\mathbf{B} = \left(\mathbf{R}^{(k)}\right)^{\mathrm{T}} \left(\mathbf{U}_-^{(k)} + \mathbf{U}^{(k)} \right)$ and $\mathbf{C} = -\mathbf{b}^{(k)}\mathbf{U}^{(k)}$. Thus, the problem can be converted into an convex QP issue:

$$\begin{aligned}
(\mathrm{P4}) \ \min_{\mathbf{x'}} \quad & \mathbf{x'}^{\mathrm{T}} \mathbf{Q} \mathbf{x'} + \mathbf{c}^{\mathrm{T}} \mathbf{x'} && \text{(47a)} \\
s.t., \quad & (18\mathrm{b}) - (18\mathrm{f}).
\end{aligned}$$

Problem (P4) can be conveniently solved using numerical optimization algorithms such as interior point methods, with optimization toolkits like Matlab's CVX toolbox [31], CVXPY python packag [32] or Gurobi software [33].

The time complexity of solving this problem depends on the convex optimization algorithm chosen. Taking the interior point method as an example, the time complexity can be expressed as $\mathcal{O}\left(n^{\tau+3} \log \frac{\mu_0}{\varepsilon}\right)$ [34], where the value of τ depends on the problem type and algorithm. A common value for parameter τ in QP is 0.5. Parameter μ_0 indicates the distance between the initial point and the optimal solution, ε is the tolerance of the numerical solution and $n = M+N$ shows the scale of the problem.

Complexity analysis indicates that the algorithm is slightly more complex than cubic time algorithms. However, the matrix $\mathbf{Q}$ has a block structure, and due to the use of methods such as L1 regularization to avoid overfitting, which leads to sparsity in gradients or model parameters, the algorithm does not require cubic time to solve the optimization problem in real cases. Moreover, the aggregation algorithm runs on the PS and can be performed synchronously while clients compute gradients. In our experiments, the algorithm successfully operated on a VGG11 [35] with more than 130M parameters, and it still converges faster considering the runtime of the optimization algorithm.

Through problem (P4), the inter-tier weights $\alpha_m^{(k)}$, located on the client side, and the weights $s_\tau^{(k)}$, located on the server side, can both be solved. They are included in the optimization variable $\mathbf{x'}$. For more detailed procedures, such as how each parameter is initialized, can be found in the pseudocode as Algorithm 1.

Algorithm 1: Implementation of SA-AirFed

Data: client to server: $\mathbf{x}_u^{(k)}{}_{d\times 1}$
Data: server to client: $\mathbf{w}_{\mathrm{G}}^{(k)}{}_{d\times 1}, \alpha_m^{(k)}{}_{1\times 1}, \mathbf{H}_{d\times S_{\mathrm{tot}}}^{(k)}$
Server initialization:
Set the historical data $\mathbf{R}^{(0)}, \mathbf{U}^{(0)}, \mathbf{U}_{-}^{(0)} = \mathbf{O}$;
for $u \in \mathcal{S}_{\mathrm{tot}}$ **do**
  Collect dataset size D_u;
  Collect latency τ^{cm} and τ_u^{cm};
  Allocate client u into tier $\mathcal{S}_m$, where $m = \left\lceil \frac{\tau^{\mathrm{cm}}+\tau_u^{\mathrm{cm}}}{\Delta T} \right\rceil$;
  Set initial client-side weighting $\alpha_m^{(0)} = \frac{1}{M}$, $\beta_u = \frac{D_u}{D_m}$;
end
Server side:
for $k = 0, 1, \ldots, K$ **do**
  Collect $\mathbf{y}^{(k)}$ from all available clients modeled as (9);
  Derive the optimal weight $s_\tau^{(k)}$ and $\alpha_m^{(k)}$ according to (47);
  Implement the server-side aggregation to get $\mathbf{u}^k$ according to (13);
  Update global model according to (7);
  Update the coefficients in (47) including $\mathbf{Q}$ and $\mathbf{c}$ based on $\mathbf{R}^{(k)}, \mathbf{U}^{(k)}$ and $\mathbf{U}_{-}^{(k)}$;
  Broadcast the latest model $\mathbf{w}_{\mathrm{G}}^{(k)}$ and optimal client-side inter-tier aggregation weight $\alpha_m^{(k)}$;
end
Client side:
if *at the end of a time slot* **then**
  Update the latest inter-tier aggregation weight $\alpha_m^{(k)}$;
  if *client u completes computation* **then**
    Implement client-side weighting according to (10);
    Implement AirComp transmission modeled as (9);
    Update the local model $\mathbf{w}_u^{(k)} \leftarrow \mathbf{w}_{\mathrm{G}}^{(k)}$;
  end
end

V. Numerical Results

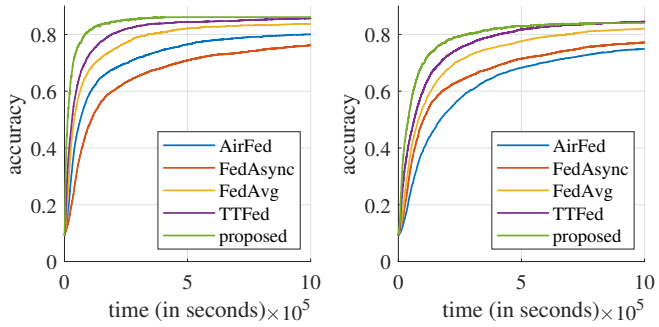
In our simulation, we consider a cellular network with a radius of $R = 500\mathrm{m}$, comprising a total of 100 users uniformly distributed around the base station. We consider additive white Gaussian noise (AWGN), path loss, and Rayleigh fading. The noise power spectral density is $N_0 = -174\mathrm{dBm/Hz}$, the path loss factor is $\alpha = 3.76$ [36], and the Rayleigh fading following the Rayleigh distribution has a scale parameter of $\sigma = 1$. The averaged transmit power constraint is $P_{\max} = 20\mathrm{dBm}$.

For performance comparison, we consider four FL frameworks including both sync, async, semi-async and AirComp settings, namely AirFed [5], FedAsync [9], FedAvg [4], and TTFed [16]. All models are optimized using SGD with a mini-batch size of 12 and a learning rate of 0.01. To prevent overfitting, we set the weight decay coefficient to 0.0001. After evaluation, we chose not to use momentum. To ensure stable training on the CIFAR-10 dataset, we applied L2 norm gradient clipping with a threshold of 1.

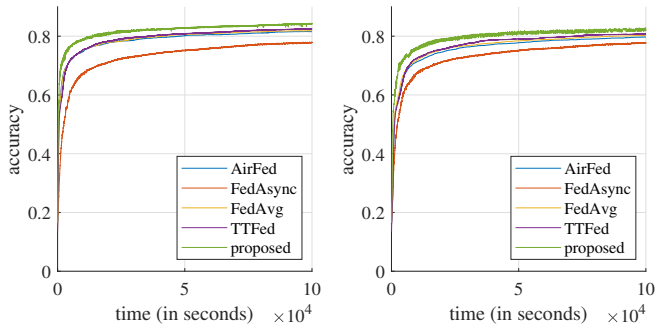
We consider two different FL scenarios to provide a more comprehensive demonstration of SA-AirFed's performance. One scenario involves the use of IoT devices, which utilize weaker computational power and limited communication resources to accomplish simpler tasks. The other scenario employs smartphones, utilizing stronger computational capabilities and more abundant communication resources to complete more complex tasks:

1) *IoT Scenario*: We assume that each IoT device runs a multi-layer perceptron with a single hidden layer containing 512 neurons, and the computational resources required to run the model for a single sample is 812534 FLOPS [37]. We presume that these IoT devices use embedded processors or microcontrollers to execute the computation task, with an average computational capacity of 100 MFLOPs/s [38]. The computational capacity of the PS is 100 GFLOPs/s. All clients share a total wireless bandwidth of $B = 1.4\mathrm{MHz}$, which is the same as the bandwidth used by the LTE Cat M1 standard. We use MNIST dataset [39], which contains 10 classes of $0-9$ hand-written digit images to train and evaluate the model. For tier-based semi-async FL, including our proposed SA-AirFed and the benchmark TTFed, we set the total number of tiers M to 8. This choice is a trade-off that prevents excessive staleness from hindering model convergence while also avoiding prolonged waiting times for stragglers, which would reduce efficiency. Alternative choices for the number of tiers will be explored in the subsequent experiments to analyse their impact on training speed and model accuracy.
2) *Smartphone Scenario*: We assume that each smartphone client runs a convolutional neural network. In our experiments, we employ VGG11 [35], which is a widely used model for image classification tasks. The computational resources required to run the model is 7.6 GFLOPs [37]. We presume that the clients execute the model using GPUs or DSPs that are specifically optimized for AI, with an average computational capacity of 100 GFLOPs/s [38]. The computational capacity of the PS is the same as that of the clients. All clients share a total wireless bandwidth of $B = 20\mathrm{MHz}$, which corresponds to the maximum system bandwidth of a single LTE cell without carrier aggregation. We employ the CIFAR-10 [40] dataset for training and validation, which contains 60000 colour images of 32×32 resolution, divided into 10 different categories. Similar to the IoT scenarios, we set the total number of tiers M for SA-AirFed and the benchmark TTFed to 5 as a balanced choice.

To study the impact of non-IID data on model convergence, we used a Dirichlet distribution to describe the number of samples of each class contained in the dataset of each model. The parameter θ can describe the degree of non-IID. When $\theta = 0$, each user will only have samples from one class. As

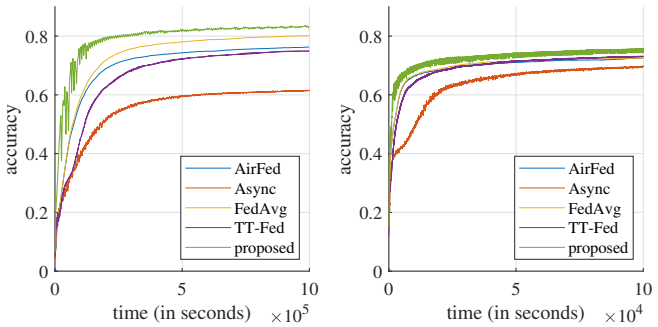


(a) MINST, $\phi = 0.50$ (b) MINST, $\phi = 0.75$



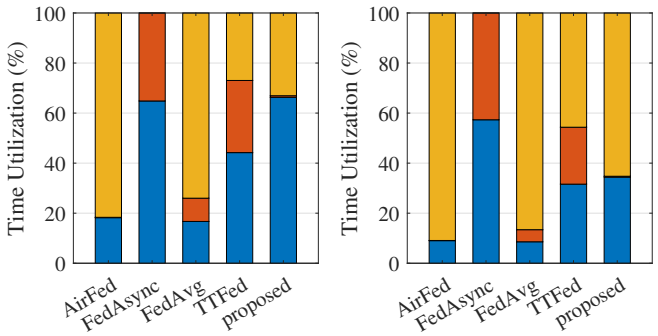
(c) CIFAR10, $\phi = 0.50$ (d) CIFAR10, $\phi = 0.75$

Fig. 3: The accuracy curve measured by wall-clock time during training with $\theta = 100$.



(a) MINST, $\phi = 0.5$ (b) CIFAR10, $\phi = 0.5$

Fig. 4: The accuracy curve measured by wall-clock time during training with $\theta = 10$.



(a) MINST, $\phi = 0.50$ (b) MINST, $\phi = 0.50$

(c) CIFAR10, $\phi = 0.50$ (d) CIFAR10, $\phi = 0.75$

Fig. 5: The time occupancy situation of all users during training.

$\theta \to \infty$, the number of samples from different classes in a user's dataset follows a uniform distribution. We use a long-tail distribution, precisely the Zipf distribution, to simulate the total number of samples obtained by different users. We use ϕ to describe the degree of tailing in the data distribution. When $\phi = 0$, all users have an equal total number of samples; as $\phi \to \infty$, the first client will process all the samples.

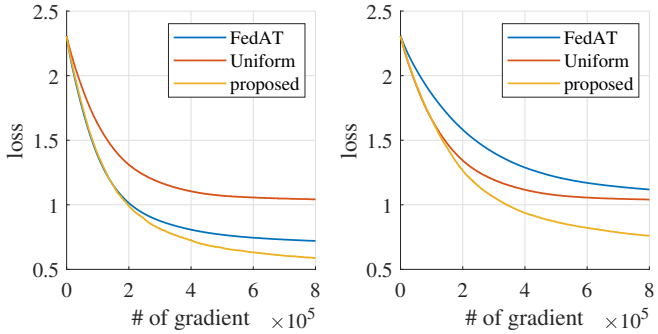
A. Predictive Performance Comparison

Fig. 3 and 4 illustrates a performance comparison between our SA-AirFed algorithm and the benchmarks under different data distribution tailing conditions. Although our algorithm does not achieve the fastest convergence rate when measured by iterations, it is the fastest when assessed using wall-clock time. This outcome aligns with the design principles of async and semi-async FL. Compared to sync FL, async and semi-async modes face the staleness problem, where each model update may not yield as accurate a gradient estimation as sync settings. However, async and semi-async modes can converge over shorter iteration intervals, making them more efficient in wall-clock measurement, which may be more concerned in practical applications.

The comparison between Fig. 3 and 4 illustrates the model performance under different levels of data distribution heterogeneity. Specifically, Fig. 3 presents the training performance under mild non-IID conditions, while Fig. 4 shows the performance under more severe non-IID conditions. All schemes experience performance degradation in the presence of more severe non-IID data, with async FL exhibiting the most significant decline.

B. Evaluation of Time Efficiency

Fig. 5 plots the time occupancy situation of our SA-AirFed and benchmarks, which explains why our algorithm can achieve higher efficiency. In the figure, "comp" represents computation time, "comm" represents communication time, and "wait" represents the time fast clients remain idle while waiting for slow clients. The tier numbers for both SA-AirFed and TTFed are unified to ensure fairness. Compared to sync AirComp, the proportion of time our SA-AirFed algo-



(a) MINST, $\phi = 0.50$ (b) MINST, $\phi = 0.75$

Fig. 6: The accuracy curve measured by the number of gradient computation. The "loss" metric is the average cross-entropy loss of the model on the test dataset.

rithm spends on gradient computation significantly increases, benefiting from the semi-async design. In comparison with TTFed, the communication time of our algorithm is reduced considerably, benefiting from the higher spectral efficiency of AirComp. It is not difficult to observe that as the tailing of the data distribution worsens, i.e., as ϕ increases, the time utilization rate of both SA-AirFed and TTFed is decreasing. This is because the data distribution not only affects the diversity of local models but also impacts the computation latency of clients. When the tailing becomes more severe, meaning the gap between users with the most data and those with the least data increases, more tiers need to be divided to ensure that the data from fast clients is collected in a timely manner. However, more tiers also mean a more severe staleness problem, which can also prevent model convergence.

C. Evaluation of Adaptive Aggregation

In Fig. 6, we test the performance of our SA-AirFed under different aggregation weighting algorithms. In the figure, the curve "FedAT" uses the aggregation method adopted in the benchmark [15], [16], and curve "Uniform" is obtained by uniform weighting. We use the number of gradient computations of all clients as the measure to compare the computation efficiency, similar to the approach used in FedAsync [9]. In our tests, the performance of adaptive weighting consistently surpassed that of the benchmark algorithm and uniform weighting.

D. Impact of Time Slot Duration

Fig. 7 shows the convergence characteristics of SA-AirFed under different tier numbers M. We find that there is an optimal value for the choice of tier number, which is aligned with our analysis in Remark 3. As shown in Fig. 7, when the user data Zipf distribution parameters are 0.50 and 0.75, the tier numbers that achieve the fastest convergence are 7 and 12, respectively. When the tier number is sufficiently small, SA-AirFed is closer to sync FL, and in this case, the stragglers problem becomes the main factor affecting efficiency. Conversely, when the tier number is sufficiently

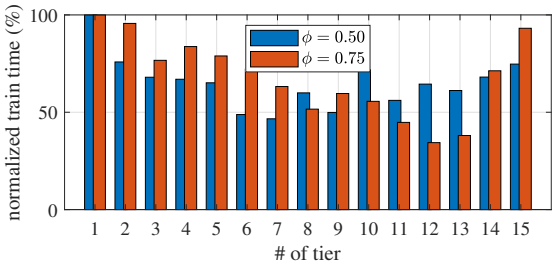


Fig. 7: Training time of SA-AirFed under different time slot duration.

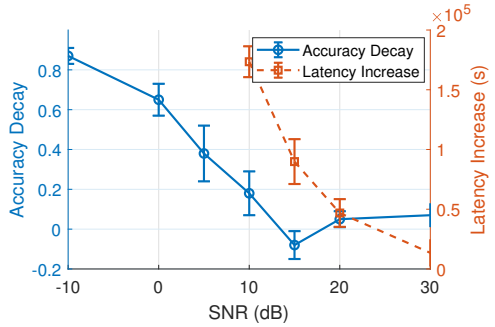


Fig. 8: Performance degradation of SA-AirFed under different noise condition. We conduct 20 tests with the same test condition and use their average performance as the results. The error bar demonstrates the standard deviation.

large, SA-AirFed is closer to totally async FL, and the main factor affecting efficiency is the staleness problem. We also found that the longer the tail of the user data distribution, the longer the training time of the model, and the larger the optimal tier number.

E. Impact of Noise Condition

Similar to other FL frameworks employing AirComp, SA-AirFed is more sensitive to noise due to the absence of error correction mechanisms in communication. We evaluate model convergence under various signal-to-noise ratio (SNR) conditions in the IoT scenario, with the results presented in Fig. 8. Compared with the baseline using error-free transmission, AirComp introduces negligible degradation in model accuracy under high SNR conditions. However, when the SNR drops below 10 dB, the impact of noise becomes significant, hindering model convergence. Notably, our experiments show that at an SNR of 15 dB, noise slightly improves model performance, increasing the top-1 accuracy by approximately 8%. This effect may be attributed to the noise during communication acting similarly to gradient noise injection, thereby enhancing the model's generalization capability.

In addition, we measure the training time required for model convergence under different SNR conditions. Specifically, we record the time needed for the model loss to reach 110% of its minimum value. Compared with error-free transmission, the presence of noise in the gradient requires more iterations for convergence, resulting in longer training time as SNR decreases. When the SNR falls below 10 dB, no stable decrease

# of clients	Dataset	AirFed	FedAsync	FedAvg	TTFed	Proposed
100	MNIST	80.06	76.24	83.68	85.68	86.10
	CIFAR-10	81.53	77.76	81.98	82.49	84.13
	CIFAR-100	41.98	21.33	64.82	61.28	61.54
	ImageNet	26.64	10.10	57.38	46.97	52.81
200	MNIST	77.95	65.01	82.52	85.41	85.34
	CIFAR-10	78.84	71.08	81.10	80.18	81.52
	CIFAR-100	50.24	43.38	61.37	55.31	60.04
	ImageNet	21.92	00.98	52.65	41.71	51.87

TABLE II: Comparison of test accuracy percentage of different methods across different datasets and number of clients.

in the loss function can be observed, and thus convergence time is not reported for these cases.

F. Scalability Test

To evaluate the scalability of SA-AirFed, we conduct experiments on more complex tasks and with a larger number of clients. Specifically, we use the CIFAR-100 dataset, which consists of 60000 32×32 color images divided into 100 classes, and the ImageNet dataset, which contains approximately 1.2 million color images across 1000 categories. The experiments are conducted under the same conditions as those used in the Smartphone Scenario. To reduce the runtime, we downsample the ImageNet images from their original resolutions (typically above 200×200) to 128×128 and randomly selected 600000 images as the training set. The experimental results are summarized in Tab. II.

In all tested scenarios, SA-AirFed successfully achieves convergence and yielded optimal or near-optimal performance compared to the benchmark methods. Notably, as task complexity increased—for example, when training an ImageNet classifier with 200 clients—SA-AirFed achieves accuracy comparable to that of FedAvg, demonstrating its capability in mitigating staleness and handling data heterogeneity.

VI. Conclusion

In this work, we designed a novel FL architecture, namely SA-AirFed, which supports semi-async aggregation and AirComp simultaneously. It introduces a time-triggered mechanism that ensures clients with similar latency can undergo sync aggregation, thus accommodating AirComp, while clients with significant latency differences perform async aggregation, thereby improving time utilization. To enhance the efficiency of SA-AirFed, we studied its convergence boundaries and transformed an optimization problem based on these findings into a tractable convex QP. Experiments under both IoT devices and smartphones scenarios show that our architecture significantly improves the time utilization of client devices and possesses the fastest convergence rate.

Appendix A
Proof of Lemmas

Lemma 1 The squared norm of the sum of vectors is bounded by the sum of the squared norms

$$\left\|\sum_{i=1}^{n}\mathbf{x}_i\right\|^2 \leq n\sum_{i=1}^{n}\|\mathbf{x}_i\|^2. \tag{48}$$

Proof: We expand the left side of (48) by inner product and use Cauchy-Schwarz inequality.

$$\begin{aligned}\left\|\sum_{i=1}^{n}\mathbf{x}_i\right\|^2 &= \sum_{i=1}^{n}\sum_{j=1}^{n}\left(\mathbf{x}_i\right)^{\mathrm{T}}\mathbf{x}_j \\ &\leq \sum_{i=1}^{n}\sum_{j=1}^{n}\|\mathbf{x}_i\|\cdot\|\mathbf{x}_j\|.\end{aligned} \tag{49}$$

Based on the inequality of arithmetic and geometric means (AM-GM), we have

$$\begin{aligned}\sum_{i=1}^{n}\sum_{j=1}^{n}\|\mathbf{x}_i\|\cdot\|\mathbf{x}_j\| &= \sum_{i=1}^{n}\sum_{j=1}^{n}\sqrt{\|\mathbf{x}_i\|^2\cdot\|\mathbf{x}_j\|^2} \\ &\leq \sum_{i=1}^{n}\sum_{j=1}^{n}\frac{\|\mathbf{x}_i\|^2+\|\mathbf{x}_j\|^2}{2} \\ &= n\sum_{i=1}^{n}\|\mathbf{x}_i\|^2.\end{aligned} \tag{50}$$

Substituting (50) back into (49) completes the proof. ■

Appendix B
Proof of Theorem 1

We start the convergence analysis by considering the change of global risk between two adjacent time slots,

$$\begin{aligned}&\mathbb{E}\left[F\left(\mathbf{w}_{\mathrm{G}}^{(k)}\right)-F\left(\mathbf{w}_{\mathrm{G}}^{(k-1)}\right)\right] \\ &\overset{(a)}{\leq}\mathbb{E}\left(\mathbf{w}_{\mathrm{G}}^{(k)}-\mathbf{w}_{\mathrm{G}}^{(k-1)}\right)^{\mathrm{T}}\mathbf{g}_{\mathrm{G}}^{(k-1)}+\frac{L}{2}\mathbb{E}\left\|\mathbf{w}_{\mathrm{G}}^{(k)}-\mathbf{w}_{\mathrm{G}}^{(k-1)}\right\|^2 \\ &\overset{(b)}{=}-\eta\cdot\mathbb{E}\left(\mathbf{u}^{(k)}\right)^{\mathrm{T}}\mathbf{g}_{\mathrm{G}}^{(k-1)}+\frac{L\eta^2}{2}\mathbb{E}\left\|\mathbf{u}^{(k)}\right\|^2 \\ &\overset{(c)}{=}\underbrace{-\frac{\eta}{2}\left\|\mathbf{g}_{\mathrm{G}}^{(k-1)}\right\|^2}_{T_1}+\underbrace{\left(-\frac{\eta-L\eta^2}{2}\mathbb{E}\left\|\mathbf{u}^{(k)}\right\|^2\right)}_{T_2} \\ &\quad+\underbrace{\frac{\eta}{2}\mathbb{E}\left\|\mathbf{u}^{(k)}-\mathbf{g}_{\mathrm{G}}^{(k-1)}\right\|^2}_{T_3}.\end{aligned} \tag{51}$$

Here $\mathbb{E}$ represents the expectation on channel randomness and $\mathbf{g}_{\mathrm{G}}^{(k)}$ is the ideal model gradient derived from all data samples with all clients participated at k-th time slot. The inequality (a) holds due to the commonly used corollary of L-smooth. Equality (b) is from the definition of model update in (7), and (c) comes from the expansion of the squared L2-norm of the difference of vectors. We end up with three terms, namely T_1, T_2 and T_3. When the learning rate η is

$$
\begin{aligned}
T_3 &\overset{(a)}{=} \frac{\eta}{2}\mathbb{E}\left\|\sum_{\tau=0}^{N} s_\tau^{(k)}\mathbf{A}^{(k-\tau)}\left(\sum_{m=1}^{M}\mathbb{1}_m^{(k-\tau)}\alpha_m^{(k-\tau)}\sum_{u\in\mathcal{S}_m}\mathbf{H}_u^{(k)}\mathbf{B}_u^{(k-\tau)}\beta_u^{(k-\tau-m)}\mathbf{g}_u^{(k-\tau-m)}+\mathbf{n}^{(k-\tau)}\right)-\mathbf{g}_{\mathrm{G}}^{(k-1)}\right\|^2 \\
&\overset{(b)}{=} \frac{\eta}{2}\mathbb{E}\Bigg\|\underbrace{\sum_{\tau=0}^{N} s_\tau^{(k)}\sum_{m=1}^{M}\mathbb{1}_m^{(k-\tau)}\alpha_m^{(k-\tau)}\sum_{u\in\mathcal{S}_m}\beta_u^{(k-\tau-m)}}_{\mathcal{A}_{\tau,m,u}^{(k)}(\cdot)}\underbrace{\left(\mathbf{A}^{(k-\tau)}\mathbf{H}_u^{(k)}\mathbf{B}_u^{(k-\tau)}-\mathbf{I}\right)\left(\mathbf{g}_u^{(k-\tau-m)}-\mathbf{g}_{\mathrm{G}}^{(k-1)}\right)}_{\mathbf{e}_u^{(k-\tau-m)}} \\
&\qquad +\underbrace{\sum_{\tau=0}^{N} s_\tau^{(k)}\sum_{m=1}^{M}\mathbb{1}_m^{(k-\tau)}\alpha_m^{(k-\tau)}\sum_{u\in\mathcal{S}_m}\beta_u^{(k-\tau-m)}}_{\mathcal{A}_{\tau,m,u}^{(k)}(\cdot)}\mathbf{g}_u^{(k-\tau-m)}-\mathbf{g}_{\mathrm{G}}^{(k-1)}+\sum_{\tau=0}^{N} s_\tau^{(k)}\mathbf{A}^{(k-\tau)}\mathbf{n}^{(k-\tau)}\Bigg\|^2 \\
&\overset{(c)}{=} \frac{\eta}{2}\mathbb{E}\left\|\mathcal{A}_{\tau,m,u}^{(k)}\left(\mathbf{g}_u^{(k-\tau-m)}-\mathbf{g}_{\mathrm{G}}^{(k-1)}+\mathbf{e}_u^{(k-\tau-m)}\right)+\sum_{\tau=0}^{N} s_\tau^{(k)}\mathbf{A}^{(k-\tau)}\mathbf{n}^{(k-\tau)}\right\|^2 \\
&\overset{(d)}{=} \frac{\eta}{2}\mathbb{E}\left\|\mathcal{A}_{\tau,m,u}^{(k)}\left[\left(\mathbf{g}_u^{(k-\tau-m)}-\mathbf{g}_{\mathrm{G}}^{(k-\tau-m)}\right)+\left(\mathbf{g}_{\mathrm{G}}^{(k-\tau-m)}-\mathbf{g}_{\mathrm{G}}^{(k-1)}\right)+\mathbf{e}_u^{(k-\tau-m)}\right]+\sum_{\tau=0}^{N} s_\tau^{(k)}\mathbf{A}^{(k-\tau)}\mathbf{n}^{(k-\tau)}\right\|^2 \\
&\overset{(e)}{\leq} \frac{3\eta}{2}\left\|\mathcal{A}_{\tau,m,u}^{(k)}\left(\mathbf{g}_u^{(k-\tau-m)}-\mathbf{g}_{\mathrm{G}}^{(k-\tau-m)}\right)\right\|^2+\frac{3\eta}{2}\left\|\mathcal{A}_{\tau,m,u}^{(k)}\left(\mathbf{g}_{\mathrm{G}}^{(k-\tau-m)}-\mathbf{g}_{\mathrm{G}}^{(k-1)}\right)\right\|^2 \\
&\qquad +\frac{3\eta}{2}\mathbb{E}\left\|\mathcal{A}_{\tau,m,u}^{(k)}\mathbf{e}_u^{(k-\tau-m)}+\sum_{\tau=0}^{N} s_\tau^{(k)}\mathbf{A}^{(k-\tau)}\mathbf{n}^{(k-\tau)}\right\|^2 \\
&\overset{(f)}{\leq} \underbrace{\frac{3\eta}{2}\mathcal{A}_{\tau,m,u}^{(k)}\left(\left\|\mathbf{g}_u^{(k-\tau-m)}-\mathbf{g}_{\mathrm{G}}^{(k-\tau-m)}\right\|^2\right)}_{T_{31}}+\underbrace{\frac{3\eta}{2}\mathcal{A}_{\tau,m,u}^{(k)}\left(\left\|\mathbf{g}_{\mathrm{G}}^{(k-\tau-m)}-\mathbf{g}_{\mathrm{G}}^{(k-1)}\right\|^2\right)}_{T_{32}} \\
&\qquad +\underbrace{\frac{3\eta}{2}\mathbb{E}\left\|\mathcal{A}_{\tau,m,u}^{(k)}\mathbf{e}_u^{(k-\tau-m)}+\sum_{\tau=0}^{N} s_\tau^{(k)}\mathbf{A}^{(k-\tau)}\mathbf{n}^{(k-\tau)}\right\|^2}_{T_{33}}
\end{aligned}
\tag{52}
$$

sufficiently small, i.e. $\eta < 1/L$, both T_1 and T_2 are negative, indicating that the global risk function would decrease as the training progresses. The third term T_3, however, is always non-negative, representing the hindrance to model convergence caused by staleness, non-iid data and channel randomness.

We further derive the term T_3 in (51) by expending the truncated update vector $\mathbf{u}^{(k)}$ in (52), where (a) stems from the definition of the estimated update vector $\mathbf{u}^{(k)}$, encompassing two key processes: client-side aggregation and server-side aggregation. These processes are denoted by (10) and (13) respectively. Inequality (b), we delineate the influence of imperfect power control by both adding and subtracting the identity matrix $\mathbf{I}$ to and from the product of matrices $\mathbf{A}^{(k-\tau)}$ and $\mathbf{B}_u^{(k-\tau)}$. The error attributed to imperfect channel alignment and truncation is defined as $\mathbf{e}^{(k-\tau-m)}$, and the intricate interplay involving client-side and server-side aggregation is encapsulated by the function $\mathcal{A}_{\tau,m,u}^{(k)}(\cdot)$, with τ, m, and u serving as the indices of summation. We exploit the affine nature of the function $\mathcal{A}_{\tau,m,u}^{(k)}(\cdot)$, leading to (c). In (d), we dissect the discrepancy between $\mathbf{g}_u^{(k-\tau-m)}$ and $\mathbf{g}_{\mathrm{G}}^{(k-1)}$ into two components: one illustrating the disparity between the local and global gradients at the identical time frame, and the other capturing the variations of the global gradient across different time slots. The derivation of inequality (e) is based on the lemma 1, as illustrated in (48). And (f) is derived according to the the affine property of $\mathcal{A}_{\tau,m,u}^{(k)}(\cdot)$ and Jensen's inequality of the convex function $\|\cdot\|^2$.

For T_{32}, we have

$$
\begin{aligned}
T_{32} &\overset{(a)}{\leq} \frac{3\eta L^2}{2}\mathcal{A}_{\tau,m}^{(k)}\left(\mathbb{E}\left\|\mathbf{w}_{\mathrm{G}}^{(k-\tau-m)}-\mathbf{w}_{\mathrm{G}}^{(k-1)}\right\|^2\right) \\
&\overset{(b)}{=} \frac{3\eta L^2}{2}\mathcal{A}_{\tau,m}^{(k)}\left(\mathbb{E}\left\|\sum_{l=1}^{\tau+m-1}\left(\mathbf{w}_{\mathrm{G}}^{(k-l)}-\mathbf{w}_{\mathrm{G}}^{(k-l-1)}\right)\right\|^2\right) \\
&\overset{(c)}{=} \frac{3\eta^3 L^2}{2}\mathcal{A}_{\tau,m}^{(k)}\left(\mathbb{E}\left\|\sum_{l=1}^{\tau+m-1}\mathbf{u}^{(k-l)}\right\|^2\right) \\
&\overset{(d)}{\leq} \frac{3\eta^3 L^2}{2}\mathcal{A}_{\tau,m}^{(k)}\left[(\tau+m-1)\sum_{l=1}^{\tau+m-1}\mathbb{E}\left\|\mathbf{u}^{(k-l)}\right\|^2\right].
\end{aligned}
\tag{53}
$$

The equality (a) follows Assumption 1 (L-Smooth) as in (19), (b) is derived by adding and subtracting $\mathbf{w}_{\mathrm{G}}^{(l)}$ when l is between $k-\tau-m-1$ and $k-2$, (c) is due to the definition of update vector of server-side aggregation, denoted in (13), (d) comes from (48) in Lemma 1.

According to the Assumption 2 demonstrated in (20), we have

$$
T_{31} \leq \frac{3\eta}{2}\mathcal{A}_{\tau,m,u}^{(k)}\left(\zeta^2\right) = \frac{3\eta\zeta^2}{2}. \tag{54}
$$

In T_{33}, we expand the overall aggregation note $\mathcal{A}_{\tau,m,u}^{(k)}$ into server-side aggregation $\sum_{\tau=1}^{N} s_\tau^{(k)}$ and client-side aggregation

$\mathcal{A}_{m,u}^{(k)}$ and utilizing Jensen's inequality, ultimately obtaining

$$T_{33} = \frac{3\eta}{2}\mathbb{E}\left\|\sum_{\tau=0}^{N} s_\tau^{(k)}\left[\mathcal{A}_{m,u}^{(k-\tau)}\left(\mathbf{e}_u^{(k-\tau-m)}\right)+\mathbf{A}^{(k-\tau)}\mathbf{n}^{(k-\tau)}\right]\right\|^2 \leq \frac{3\eta}{2}\sum_{\tau=0}^{N} s_\tau^{(k)} \underbrace{\mathbb{E}\left\|\mathcal{A}_{m,u}^{(k-\tau)}\left(\mathbf{e}_u^{(k-\tau-m)}\right)+\mathbf{A}^{(k-\tau)}\mathbf{n}^{(k-\tau)}\right\|^2}_{\varepsilon^{(k-\tau)}}, \tag{55}$$

where $\varepsilon^{(k-\tau)}$ equals the squared Euclidean distance between the result of AirComp under perfect channel alignment and truncation conditions and the actual AirComp result $\mathbf{y}^{(k-\tau)}$ including channel misalignment error and noise.

We further simplified the relationship between the update vector $\mathbf{u}^{(k-l)}$ and the gradient $\mathbf{g}_{\mathrm{G}}^{(k-1)}$ by introducing Assumption 3, resulting in

$$\begin{aligned}
&\mathbb{E}\left\|\mathbf{u}^{(k-l)}\right\|^2 \\
&\overset{(a)}{\leq} 2\left\|\mathcal{A}_{\tau,m,u}^{(k-l)}\mathbf{g}_u^{(k-l-\tau-m)}\right\|^2 + 2\sum_{\tau=0}^{N} s_\tau^{(k-l)}\varepsilon^{(k-l-\tau)} \\
&\overset{(b)}{\leq} 2\mathcal{A}_{\tau,m,u}^{(k-l)}\left\|\mathbf{g}_u^{(k-l-\tau-m)}\right\|^2 + 2\sum_{\tau=0}^{N} s_\tau^{(k-l)}\varepsilon^{(k-l-\tau)} \\
&\overset{(c)}{\leq} 2\xi^2\mathcal{A}_{\tau,m}^{(k-l)}\left\|\mathbf{g}_{\mathrm{G}}^{(k-l-\tau-m)}\right\|^2 + 2\sum_{\tau=0}^{N} s_\tau^{(k-l)}\varepsilon^{(k-l-\tau)} \\
&\overset{(d)}{\leq} 2\xi^2\gamma^2\left\|\mathbf{g}_{\mathrm{G}}^{(k-1)}\right\|^2 + 2\sum_{\tau=0}^{N} s_\tau^{(k-l)}\varepsilon^{(k-l-\tau)},
\end{aligned} \tag{56}$$

where (a) comes from Lemma 1, (b) is based on the Jensen's inequality, (c) and (d) is derived from the Assumption 2 and 3.

Substituting (54), (53) and (55) into (52) and (51), and utilizing (56) we have

$$\begin{aligned}
&\mathbb{E}\left[F\left(\mathbf{w}_{\mathrm{G}}^{(k)}\right)-F\left(\mathbf{w}_{\mathrm{G}}^{(k-1)}\right)\right] \\
&\leq \frac{\eta\left[6L^2(2M-1)^2\xi^2\gamma^2\eta^2+L\eta-2\right]}{2}\left\|\mathbf{g}_{\mathrm{G}}^{(k-1)}\right\|^2 + \frac{3\eta}{2}\zeta^2 \\
&+ \frac{3\eta}{2}\sum_{\tau=0}^{N} s_\tau^{(k)}\varepsilon^{(k-\tau)} \\
&+ 3\eta^3L^2\mathcal{A}_{\tau,m}^{(k)}\left[(\tau+m-1)\sum_{l=1}^{\tau+m-1}\sum_{p=0}^{N} s_\tau^{k-l}\varepsilon^{(k-l-p)}\right].
\end{aligned} \tag{57}$$

■

REFERENCES

[1] Z. Zheng, Y. Deng, X. Liu, and A. Nallanathan, "Asynchronous federated learning via over-the-air computation," in *IEEE Glob. Commun. Conf.*, 2023, pp. 1345–1350.

[2] C. Zhang, Y. Xie, H. Bai, B. Yu, W. Li, and Y. Gao, "A survey on federated learning," *Knowl.-Based Syst.*, vol. 216, p. 106775, 2021.

[3] C. Xu, Y. Qu, Y. Xiang, and L. Gao, "Asynchronous federated learning on heterogeneous devices: a survey," *arXiv:2109.04269*, 2023.

[4] H. B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. y Arcas, "Communication-efficient learning of deep networks from decentralized data," *arXiv:1602.05629*, 2023.

[5] G. Zhu, Y. Wang, and K. Huang, "Broadband analog aggregation for low-latency federated edge learning," *IEEE Trans. Wireless Commun.*, vol. 19, no. 1, pp. 491–506, 2020.

[6] C. T. Dinh *et al.*, "Federated learning over wireless networks: Convergence analysis and resource allocation," *IEEE/ACM Trans. Netw.*, vol. 29, no. 1, pp. 398–409, 2020.

[7] X. Li, K. Huang, W. Yang, S. Wang, and Z. Zhang, "On the convergence of fedavg on non-iid data," *arXiv:1907.02189*, 2020.

[8] B. Recht, C. Re, S. Wright, and F. Niu, "Hogwild!: a lock-free approach to parallelizing stochastic gradient descent," in *Adv. Neural Inf. Process. Syst.*, vol. 24, 2011.

[9] C. Xie, S. Koyejo, and I. Gupta, "Asynchronous federated optimization," *arXiv:1903.03934*, 2020.

[10] J. Nguyen *et al.*, "Federated learning with buffered asynchronous aggregation," in *Int. Conf. Artif. Intell. Stat.*, 2022, pp. 3581–3607.

[11] Y. Chen, Y. Ning, M. Slawski, and H. Rangwala, "Asynchronous online federated learning for edge devices with non-iid data," in *IEEE Int. Conf. Big Data*, 2020, pp. 15–24.

[12] Q. Ho *et al.*, "More effective distributed ml via a stale synchronous parallel parameter server," in *Adv. Neural Inf. Process. Syst.*, vol. 26, 2013.

[13] W. Wu, L. He, W. Lin, R. Mao, C. Maple, and S. Jarvis, "Safa: a semi-asynchronous protocol for fast federated learning with low overhead," *IEEE Trans. Comput.*, vol. 70, no. 5, pp. 655–668, 2021.

[14] N. Su and B. Li, "How asynchronous can federated learning be?" in *IEEE/ACM Int. Symp. Qual. Serv.*, 2022, pp. 1–11.

[15] Z. Chai, Y. Chen, A. Anwar, L. Zhao, Y. Cheng, and H. Rangwala, "Fedat: A high-performance and communication-efficient federated learning system with asynchronous tiers," in *Proc. Int. Conf. High Perform. Comput. Netw. Storage Anal.*, 2021, pp. 1–16.

[16] X. Zhou, Y. Deng, H. Xia, S. Wu, and M. Bennis, "Time-triggered federated learning over wireless networks," *IEEE Trans. Wireless Commun.*, vol. 21, no. 12, pp. 11 066–11 079, 2022.

[17] J. Konečný, H. B. McMahan, F. X. Yu, P. Richtárik, A. T. Suresh, and D. Bacon, "Federated learning: strategies for improving communication efficiency," *arXiv:1610.05492*, 2017.

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

[19] G. Shi, S. Guo, J. Ye, N. Saeed, and S. Dang, "Multiple parallel federated learning via over-the-air computation," *IEEE Open J. Commun. Soc.*, vol. 3, pp. 1252–1264, 2022.

[20] C. Zhong, H. Yang, and X. Yuan, "Over-the-air federated multi-task learning over mimo multiple access channels," *IEEE Trans. Wirel. Commun.*, vol. 22, no. 6, pp. 3853–3868, 2023.

[21] A. Nakai-Kasai and T. Wadayama, "Precoder design for correlated data aggregation via over-the-air computation in sensor networks," in *2022 IEEE Globecom Workshops (GC Wkshps)*, 2022, pp. 1096–1101.

[22] Y. Shao, D. Gündüz, and S. C. Liew, "Federated edge learning with misaligned over-the-air computation," *IEEE Trans. Wireless Commun.*, vol. 21, no. 6, pp. 3951–3964, 2022.

[23] Y. Shao, D. Gündüz, and S. C. Liew, "Bayesian over-the-air computation," *IEEE J. Sel. Areas Commun.*, vol. 41, no. 3, pp. 589–606, 2023.

[24] L. Qiao, Z. Gao, M. Boloursaz Mashhadi, and D. Gündüz, "Massive digital over-the-air computation for communication-efficient federated edge learning," *IEEE J. Sel. Areas Commun.*, vol. 42, no. 11, pp. 3078–3094, 2024.

[25] G. Zhu, Y. Du, D. Gündüz, and K. Huang, "One-bit over-the-air aggregation for communication-efficient federated edge learning: Design and convergence analysis," *IEEE Trans. Wirel. Commun.ons*, vol. 20, no. 3, pp. 2120–2135, 2021.

[26] D. Data and S. Diggavi, "Byzantine-resilient sgd in high dimensions on heterogeneous data," in *IEEE Int. Symp. Inf. Theory*, 2021, pp. 2310–2315.

[27] A. Koloskova, S. U. Stich, and M. Jaggi, "Sharper convergence guarantees for asynchronous sgd for distributed and federated learning," vol. 35, 2022, pp. 17 202–17 215.

[28] X. Zang, W. Liu, Y. Li, and B. Vucetic, "Over-the-air computation systems: optimal design with sum-power constraint," *IEEE Wirel. Commun. Lett.*, vol. 9, no. 9, pp. 1524–1528, 2020.

[29] R. Balakrishnan, T. Li, T. Zhou, N. Himayat, V. Smith, and J. Bilmes, "Diverse client selection for federated learning via submodular maximization," in *Int. Conf. on Learn. Representations*, 2022.

[30] X. Cao, J. Jia, Z. Zhang, and N. Z. Gong, "Fedrecover: recovering from poisoning attacks in federated learning using historical information," in *IEEE Symp. on Secur. and Privacy*, 2023, pp. 1366–1383.

[31] CVX Research Inc., "CVX: matlab software for disciplined convex programming, version 2.0," http://cvxr.com/cvx, Aug. 2012.

[32] S. Diamond and S. Boyd, "CVXPY: a python-embedded modeling language for convex optimization," *J. Mach. Learn. Res.*, vol. 17, no. 83, pp. 1–5, 2016.

[33] Gurobi Optimization, LLC, "Gurobi optimizer reference manual," 2023.

[34] F. A. Potra and S. J. Wright, "Interior-point methods," *J. Comput. Appl. Math.*, vol. 124, no. 1-2, pp. 281–302, 2000.
[35] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *arXiv:1409.1556*, 2015.
[36] W. Shi, S. Zhou, Z. Niu, M. Jiang, and L. Geng, "Joint device scheduling and resource allocation for latency constrained wireless federated learning," *IEEE Trans. Wireless Commun.*, vol. 20, no. 1, pp. 453–467, 2021.
[37] P. Molchanov, S. Tyree, T. Karras, T. Aila, and J. Kautz, "Pruning convolutional neural networks for resource efficient inference," *arXiv:1611.06440*, 2016.
[38] C. Feng, Y. Wang, Q. Chen, Y. Ding, G. Strbac, and C. Kang, "Smart grid encounters edge computing: Opportunities and applications," *Adv. Appl. Energy*, vol. 1, p. 100006, 2021.
[39] L. Deng, "The mnist database of handwritten digit images for machine learning research," *IEEE Signal Process. Mag.*, vol. 29, no. 6, pp. 141–142, 2012.
[40] A. Krizhevsky and G. Hinton, "Learning multiple layers of features from tiny images," 2009.