Asynchronous Federated Learning via Over-the-air Computation

Zijian Zheng*, Yansha Deng†, Xiaonan Liu*, Arumugam Nallanathan*
* School of Electronic Engineering and Computer Science, Queen Mary University of London, London, UK
† Department of Engineering, King's College London, London, UK
E-mail: z.zheng@qmul.ac.uk, yansha.deng@kcl.ac.uk, x.l.liu@qmul.ac.uk, a.nallanathan@qmul.ac.uk

***Abstract*—The emerging field of federated learning (FL) provides great potential for edge intelligence while protecting data privacy. However, as the system grows in scale or becomes more heterogeneous, new challenges, such as the spectrum shortage and stragglers issues, arise. These issues can potentially be addressed by over-the-air computation (AirComp) and asynchronous FL, respectively, however, their combination is difficult due to their conflicting requirements. In this paper, we propose a novel asynchronous FL with AirComp in a time-triggered manner (async-AirFed). The conventional async aggregation requests the historical data to be used for model updates, which can cause the accumulation of channel noise and interference when AirComp is applied. To address this issue, we propose a simple but effective truncation method which retains a limited length of historical data. Convergence analysis presents that our proposed async-AirFed converges on non-convex optimality function with sub-linear rate. Simulation results show that our proposed scheme achieves more than 34% faster convergence than the benchmarks, by achieving an accuracy of 85%, which also improves the time utilization efficiency and reduces the impact of staleness and the channel.**

***Index Terms*—Asynchronous federated learning, over-the-air computation, error accumulation.**

I. Introduction

Unlike traditional communication systems, the new generation of wireless communication systems, including 5G and 6G, have introduced edge intelligence as essential components of the system, which are able to support a variety of applications such as virtual reality (VR), augmented reality (AR), connected vehicles, and smart internet-of-things (IoT). Meanwhile, federated learning (FL) [1] provides an effective implementation for edge intelligence in protecting user privacy.

According to the communication methods in model training, FL can be classified into digital approach [2], and over-the-air computation (AirComp) approach [3]. The digital approach provides a higher accuracy by employing error-free communication while facing problems when there are too many participants or scarce spectrum resource. The AirComp approach significantly improves spectral efficiency by sharing the multiple-access channel (MAC) while suffering accuracy loss from the channel noise and interference.

According to the concurrency of participants, FL can be divided into synchronous approach and asynchronous approach [4]. In the sync-FL approach, all selected devices must update their local models concurrently. Since the fast devices have to wait for the slowest one, the overall speed is constrained, thereby resulting in stragglers problem. The async-FL approach offers a solution by enabling aggregation from a subset of participants. However, it may lead to a decrease in model accuracy due to outdated updates. Also, much more communication rounds are required compared to the synchronous setting, bringing heavier burden on wireless transmission. In conclusion, the sync-FL can use the latest data for each update but has lower time utilization efficiency; the async-FL needs to solve the problem of outdated updates but can significantly improve time utilization efficiency.

The time-triggered FL (TT-Fed) [5] provides a generalized form between synchronous and total asynchronous settings, allowing to find a tradeoff between training and communication efficiencies. It triggers aggregation based on time rather than events, which reduces the number of communication rounds required for model training. However, like other digital transmission based FL systems, the communication bottleneck will limit the scale of TT-Fed. To achieve digital transmission, the system must allocate dedicated bandwidth for each user. When the total bandwidth is limited but the number of clients is too large, clients may suffer from longer communication latency, resulting in a decrease in learning efficiency. Also, the uncertainty on the latency of digital communication also makes the resource allocation and the optimization on time slot duration complex and inefficient.

To ensure the scalability of FL system, the combination of asynchronous aggregation and AirComp can be a potential solution to solve both stragglers problem and spectrum shortage simultaneously. However, the characteristics of MAC require a simultaneous transmission among all participants, making it inherently incompatible with async-FL. To the best of our knowledge, no prior research has been conducted to introduce AirComp to async-FL. Based on TT-Fed, we find that its time-aligned property makes AirComp feasible in asynchronous aggregation, which further improves communication efficiency and enhances the scalability of FL.

In this paper, we propose an over-the-air federated learning

This work was supported in part by UKRI under the UK government's Horizon Europe funding guarantee (grant number 10061781), as part of the European Commission-funded collaborative project VERGE, under SNS JU program (grant number 101096034). This work is also a contribution by Project REASON, a UK Government funded project under the FONRC sponsored by the DSIT.

architecture in an asynchronous manner (async-AirFed). To the best of our knowledge, this is the first paper consider AirComp in async-FL setting. First, we analyze how the staleness, channel noise and interference introduced into the model change over time. We conclude that gradient aggregation is better than model aggregation for async-AirFed and propose a simple truncation method to stop the the error accumulation. We further study the convergence rate of async-AirFed under non-convex assumption. Finally, we evaluate the convergence speed of async-AirFed compared to three stage-of-the-art benchmarks.

II. System Model

We consider a federated learning system, where clients $u_1, u_2, \cdots$ from set $\mathcal{S}_{\text{tot}}$ train a global model collaboratively with the help of a parameter server. Each client u 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 $\mathcal{D}_{\text{tot}} = \mathcal{D}_1 \cup \cdots \cup \mathcal{D}_{S_{\text{tot}}}$. Unlike the conventional machine learning scenarios, raw training data is kept on clients and not allowed to be transmitted in FL.

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. Stochastic gradient descent (SGD) is used to solve this optimization problem.

A. Time-Triggered Async Over-the-Air Federated Learning

Our research builds upon and extends the work of X. Zhou [5], who proposed a generalized approach between sync-FL and full async-FL, namely, TT-Fed. Before explaining the time-triggered scheme in detail, we present several definitions for clarification according to [5].

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 updates of FL model in conventional event-triggered settings, given by

$$k = \left\lceil \frac{t}{\Delta T} \right\rceil, \tag{2}$$

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

Definition 2 (Available Client) Due to differences in computing capacity and dataset size among clients, their latency for computations may vary dramatically, resulting in the existence of both busy clients and idle ones with their computation accomplished. We define the idle clients at the end of time slot k as the available clients. They are timely enough to participate in the upcoming global aggregation. The indicator $\mathbb{1}_{\text{L},u}^{(k)} \in \{1, 0\}$ is used to label this state, where u is an available client in the time slot k when the indicator is 1.

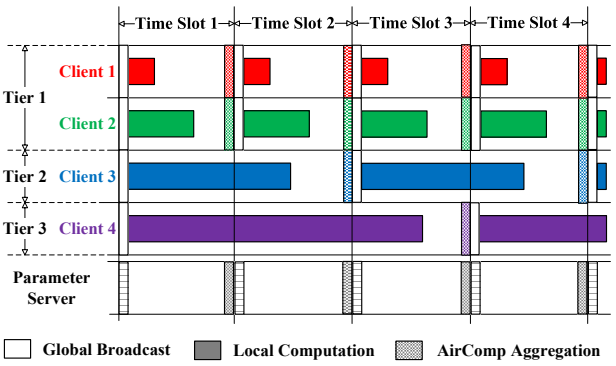

Fig. 1. The work-flow of async-AirFed.

Definition 3 (Tier) We denote tier as a subset of clients with similar computation and communication latency. Formally, $\mathcal{S}_m$ includes the clients who need m time slots for model update, i.e.,

$$\mathcal{S}_m = \left\{ u \;\middle|\; \mathbb{1}_{\text{L},u}^{(\tau)} = 1;\ \tau = m, 2m, \cdots \right\},\ m = 1, 2, \cdots, M. \tag{3}$$

To indicate the states of a tier, we integrate the client-wise indicators into tier-wise:

$$\mathbb{1}_{\text{I},m}^{(k)} = \prod_{u \in \mathcal{S}_m} \mathbb{1}_{\text{L},u}^{(k)}. \tag{4}$$

The working principle of TT-Fed is described below. By dividing the entire training process into numerous time slots, a trigger mechanism of aggregation that solely depends on time is established. For client u, its decision to participate in the k-th aggregation depends on whether it can complete the computation and communication before the end of the k-th time slot. The clients are naturally classified into tiers according to their participant pattern determined by the relative numeral of there latency and the system time slot duration. The clients within a tier adopt the sync aggregation, making the AirComp possible to improve communication efficiency. The aggregation between different tiers keeps asynchronous, reducing the waste on time to wait for stragglers.

Our async-AirFed retains most of the characteristics of TT-Fed, except for modifying the alignment of communication session. In TT-Fed, the clients transmit their update data immediately once the computation is completed. To enable AirComp in async-AirFed, the end time of the communication session needs to be aligned with the end time of the time slot, to ensure that all available clients will send data simultaneously. The working flow of async-AirFed is shown in Fig. 1. Each time slot has three sessions:

1) Global Broadcast: At the beginning of each time slot, the parameter server broadcasts the latest global model to the

whole system. For those clients who have completed the aggregation in the last time slot, the newly received global model is used for gradient computation in the consequent work sessions. For the other clients who are still busy with computation, this broadcast will be ignored to achieve parallelism.

2) Local Computation: The gradient in a typical FL training scenario is estimated by applying the global model to local data. Due to async updates, gradients from different tiers correspond to global models at different time slots. According to the definition of tier number and global broadcast session, the timeliness of each uploaded gradient by each client can be traced back. That is, for the available client u in the time slot k, its computed gradient is based on the global model at the time slot $k-m$, where m is the tier number of client the u, i.e:

$$\begin{aligned}\mathbf{x}_{\mathrm{L},u}^{(k)} &= \mathbf{g}_{\mathrm{L},u}^{(k-m)} \\ &= \sum_{(\mathbf{x}_i,y_i)\in\mathcal{D}_u} \frac{1}{D_u}\nabla f\left(\mathbf{w}_{\mathrm{G}}^{(k-m)};\mathbf{x}_i,y_i\right).\end{aligned} \tag{5}$$

3) Over-the-Air Aggregation: The global aggregation essentially occurs simultaneously, allowing AirComp leveraged to improve communication efficiency. For simplicity, we split it into two virtual sub-processes: intra-tier aggregation and extra-tier aggregation. The extra-tier aggregation refers to the aggregation of different tiers, while the intra-tier aggregation refers to the aggregation of different clients within the same tier. These two types of aggregation processes are defined by two sets of affine coefficients, which are required to be multiplied with the uploaded gradient before over-the-air computation.

Pre-distortion is employed to mitigate channel interference. It generally comprises two parts at the transmitter (client) and receiver (parameter server). To maintain generality, we represent both of these parts as linear mappings, namely, matrices $\mathbf{B}_{\mathrm{L},u}^{(k)}$ and $\mathbf{A}^{(k)}$. The matrix $\mathbf{B}_{\mathrm{L},u}^{(k)}$ varies across clients, which is used to eliminate the heterogeneous across channels, thereby aligning the amplitude of transmitted signals at the receiver. The matrix $\mathbf{A}^{(k)}$ keeps the same for all clients and is used to amplify the received signal, ensuring the transmit power at the transmitter side satisfies the power constraint.

Based on the above definitions, the process of over-the-air aggregation is expressed as

$$\begin{aligned}\mathbf{g}_{\mathrm{A}}^{(k)} = \mathbf{A}^{(k)} \sum_{m=1}^{M} \alpha_m^{(k)} \mathbb{1}_{\mathrm{I},m}^{(k)} \sum_{u\in\mathcal{S}_m} \beta_u^{(k)} \mathbf{H}_{\mathrm{L},u}^{(k)} \mathbf{B}_{\mathrm{L},u}^{(k)} \mathbf{g}_{\mathrm{L},u}^{(k-m)} \\ + \mathbf{A}^{(k)}\mathbf{n}^{(k)},\end{aligned} \tag{6}$$

where $\mathbf{H}_{\mathrm{L},u}^{(k)}$ is the uplink channel matrix between the parameter server and client u and $\mathbf{n}^{(k)}$ is the additive white Gaussian noise (AWGN).

According to the object of FL defined in (1), we choose the normalized dataset size as the intra-tier aggregation weight, which is denoted as

$$\beta_u^{(k)} = \frac{D_u}{\sum_{v\in\mathcal{S}_m} D_v}. \tag{7}$$

The extra-tier aggregation weight is generated using the same heuristic weighting scheme in [5] and [6]. The extra-tier weight in this scheme is the normalized update frequency across different tiers, which is written as

$$\alpha_m^{(k)} = \frac{\left\lfloor \frac{k}{M+1-m} \right\rfloor}{\sum_{m=1}^{M} \left\lfloor \frac{k}{m} \right\rfloor}. \tag{8}$$

Based on the aggregation process, we further define the formula for model updates as

$$\mathbf{w}_{\mathrm{G}}^{(k)} = \mathbf{w}_{\mathrm{G}}^{(k-1)} - \eta^{(k)}\mathbf{u}^{(k)}, \tag{9}$$

where $\eta^{(k)}$ is the learning rate and $\xi^{(k)}$ is given by

$$\xi^{(k)} = \sum_{m=1}^{M} \alpha_m^{(k)} \mathbb{1}_{\mathrm{I},m}^{(k)}, \tag{10}$$

and $\mathbf{u}^{(k)}$ is the vector indicating the update as

$$\mathbf{u}^{(k)} = \mathbf{g}_{\mathrm{A}}^{(k)} + \left(1-\xi^{(k)}\right)\mathbf{u}^{(k-1)}. \tag{11}$$

B. Latency and Energy Model

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

1) *Computation Latency:* We adopt floating point operations (FLOPs) as a measure of computational task requirements. Denote the floating point operations per second (FLOPS) of CPU as f_c. Let C denote the FLOPs required by the federated learning model, and D_u denote the size of dataset on the client u. Thus, the computational latency of the client u is given by

$$\tau_u^{\mathrm{cp}} = \frac{CD_u}{f_c}. \tag{12}$$

2) *Communication Latency:* In order to ensure compatibility between AirComp and existing communication systems, we adopt the orthogonal frequency division multiplexing (OFDM) modulation with total bandwidth B and N_M orthogonal sub-carriers. To simplify our expression, we ignore the time cost of cyclic prefix (CP). 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

$$\tau^{\mathrm{cm}} = \left\lceil \frac{q}{N_M} \right\rceil \cdot \frac{N_M}{B}. \tag{13}$$

3) *Communication Power:* To simplify the analysis, we use the square of the L2-norm to represent the energy required for data transmission through AirComp. The power required to implement pre-distortion and transmit $\mathbf{x}_{\mathrm{L},u}^{(k)}$ on the client u at the time slot k is given by

$$P_u^{(k)} = \frac{\left\| \mathbf{B}_{\mathrm{L},u}^{(k)} \mathbf{x}_{\mathrm{L},u}^{(k)} \right\|_2^2}{\tau^{\mathrm{cm}}}. \tag{14}$$

The communication power of the whole system is subject to the average power constraint among all clients, which is written as

$$\sum_{u\in\mathcal{S}_{\text{tot}}} \frac{1}{S_{\text{tot}}} P_u^{(k)} \leq P_0. \tag{15}$$

III. Analysis on Noise, Interference and Staleness

Although async aggregation and AirComp bring improvements to parallelism and spectral efficiency for FL, their brute-force combination will lead to new issues. First, due to the unique estimation mechanism in async-FL, the channel noise and interference introduced by AirComp may accumulate over time. In addition, the time-triggered mechanism required by AirComp makes it more difficult to analyze staleness. Currently, most research on combating staleness is based on event-triggered designs, which may be ineffective for this problem.

A. Model Aggregation vs. Gradient Aggregation

There are two variations of aggregation method in FL: model aggregation and gradient aggregation. Both exist in either AirComp or async-FL.In the fundamental FL, such as the FedAvg [1], the choice between them does not affect the convergence performance. However, we find they are different when channel noise and interference, or staleness exist. To compare their differences, we expand (9) and rewrite the corresponding global model update formula for the model aggregation as

$$\begin{aligned}\mathbf{w}_{\text{G,grad}}^{(k)} = \mathbf{w}_{\text{G}}^{(k-1)} &\\ - \eta^{(k)} \sum_{m=1}^{M} \alpha_m^{(k)} \mathbb{1}_{\text{I},m}^{(k)} \sum_{u\in\mathcal{S}_m} \beta_u^{(k)} \tilde{\mathbf{H}}_{\text{L},u}^{(k)} \mathbf{g}_{\text{L},u}^{(k-m)} &\\ - \eta^{(k)} \left(1-\xi^{(k)}\right) \mathbf{u}^{(k-1)} &\\ - \eta^{(k)} \mathbf{A}^{(k)} \mathbf{n}^{(k)}, &\end{aligned} \tag{16}$$

$$\begin{aligned}\mathbf{w}_{\text{G,mod}}^{(k)} = \sum_{m=1}^{M} \alpha_m^{(k)} \mathbb{1}_{\text{I},m}^{(k)} \sum_{u\in\mathcal{S}_m} \beta_u^{(k)} \tilde{\mathbf{H}}_{\text{L},u}^{(k)} \mathbf{w}_{\text{G}}^{(k-m)} &\\ + \left(1-\xi^{(k)}\right) \mathbf{w}_{\text{G}}^{(k-1)} &\\ - \eta^{(k)} \sum_{m=1}^{M} \alpha_m^{(k)} \mathbb{1}_{\text{I},m}^{(k)} \sum_{u\in\mathcal{S}_m} \beta_{u,m}^{(k)} \tilde{\mathbf{H}}_{\text{L},u}^{(k)} \mathbf{g}_{\text{L},u}^{(k-m)} &\\ + \mathbf{A}^{(k)} \mathbf{n}^{(k)}, &\end{aligned} \tag{17}$$

where $\tilde{\mathbf{H}}_{\text{L},u}^{(k)} = \mathbf{A}^{(k)} \mathbf{H}_{\text{L},u}^{(k)} \mathbf{B}_{\text{L},u}^{(k)}$ represents the residual equivalent channel after pre-distortion.

According to (16) and (17) , the model parameters in each time slot consist of three parts: the starting point (the component including model parameters), the update vector (the component including gradient), and the noise term (the component including AWGN). By comparing each of them, we have the following insights:

- **The gradient aggregation has less staleness and interference on starting point.** As shown in the first row of (16), the starting point of gradient aggregation in k-th time slot is the global parameter from the $(k-1)$-th time slot, while the starting point of model aggregation is a complex combination from $(k-M)$-th to $(k-1)$-th time slot, illustrated in the first and second row in (17). It's obvious that the complex combination in model aggregation includes both outdated data from previous time slots (staleness) and residual channel effects which cannot be eliminated by imperfect pre-distortion (channel interference). Due to the natural property of AirComp and async aggregation, the staleness and the channel interference are inevitable, while the gradient aggregation could reduce their impact by preventing parameters from being contaminated again.
- **The gradient aggregation is more robust under noise.** For most cases, the learning rate $\eta^{(k)}$ is much smaller than 1, making the noise term in the gradient greatly shrunk. Considering that AirComp uses uncoded analog transmission, this advantage is particularly important for model accuracy in long-distance or strict power-limited scenarios.

B. Truncation Mechanism

To maximize data diversity, the asynchronous aggregation needs to use previous data to replace absent data from unavailable tiers. As shown in (11), $\mathbf{u}^{(k-1)}$ is combined with the updated gradient $\mathbf{g}_{\text{A}}^{(k)}$ to derive the update vector $\mathbf{u}^{(k)}$. This approach works well when the transmission is error-free, while it has error accumulation problem when channel noise and interference appear.

We solve (11) based on the first-order variable coefficient difference equation and end up with the following:

$$\begin{aligned}\mathbf{u}^{(k)} = \mathbf{p}^{(k)} + \sum_{l=1}^{k-1} \prod_{\tau=k}^{k-l+1} \left(1-\xi^{(\tau)}\right) \mathbf{p}^{(k-l)} &\\ + \mathbf{e}^{(k)} + \sum_{l=1}^{k-1} \prod_{\tau=k}^{k-l+1} \left(1-\xi^{(\tau)}\right) \mathbf{e}^{(k-l)} &\\ + \prod_{\tau=1}^{k} \left(1-\xi^{(k)}\right) \mathbf{u}^{(0)}, &\end{aligned} \tag{18}$$

where $\mathbf{p}^{(k)} = \sum_{m=1}^{M} \alpha_m^{(k)} \mathbb{1}_{\text{I},m}^{(k)} \sum_{u\in\mathcal{S}_m} \beta_u^{(k)} \mathbf{g}_{\text{L},u}^{(k-m)}$ represents the aggregated gradient from available tiers and $\mathbf{e}^{(k)} = \mathbf{A}^{(k)}\mathbf{n}^{(k)} + \sum_{m=1}^{M} \alpha_m^{(k)} \mathbb{1}_{\text{I},m}^{(k)} \sum_{u\in\mathcal{S}_m} \beta_u^{(k)} \left(\tilde{\mathbf{H}}_{\text{L},u}^{(k)} - \mathbf{I}\right) \mathbf{g}_{\text{L},u}^{(k-m)}$ represents the channel noise and interference caused by imperfect pre-distortion.

The second row shows how errors from channel accumulated over time. To eliminate this influence, we introduce a truncation mechanism. The core idea of truncation comes from a simple fact: the reason why we need previous information is to include the gradient from unavailable tiers in each model update. Given that the slowest tier in async-AirFed requires M time slots, we only need to retain at most Mth-order memory when $\mathbf{u}^{(k)}$ iterates. Besides, for scenarios with poor channels,

the trade-off between data diversity and communication robustness may need to prioritize the communication. To address this, we set the memory length of $\mathbf{u}^{(k)}$ as a hyper-parameter, called "truncation order", and denoted as n_t.

The modified update vector $\mathbf{u}_t^{(k)}$ requires an additional truncation operation. In this mechanism, an n_t length buffer is required to store the received data $\mathbf{g}_{\mathrm{A}}$ and the applied weight $\left(1-\xi^{(\tau)}\right)$ from $(k-1)$-th to $(k-n_t-1)$-th time slot. To truncate the accumulated error, the historical received data beyond the given truncation order will be removed after applying its corresponding weight, which is given by

$$\mathbf{u}_t^{(k)} = \mathbf{g}_{\mathrm{A}}^{(k)} + \left(1-\xi^{(k)}\right)\mathbf{u}_t^{(k-1)} - \prod_{\tau=k}^{k-n_t}\left(1-\xi^{(\tau)}\right)\mathbf{g}_{\mathrm{A}}^{(k-n_t-1)}. \tag{19}$$

Theorem 1 (Truncated Update with n_t Orders) Under the truncated update vector formula in (19), $n_t \geq 1$, and $\mathbf{u}_t^{(\tau)} = \mathbf{0}, \quad \forall \tau = -n_t+1, \cdots, -1, 0$, the non-recursive expression is given by

$$\mathbf{u}_t^{(k)} = \mathbf{g}_{\mathrm{A}}^{(k)} + \sum_{l=1}^{n_t}\prod_{\tau=k}^{k-l+1}\left(1-\xi^{(\tau)}\right)\mathbf{g}_{\mathrm{A}}^{(k-l)}. \tag{20}$$

Proof. The proof is simple with Mathematical Induction.

IV. Convergence Analysis

In this section, we analyze the convergence behaviour of the proposed async-AirFed. We first introduce the following assumptions:

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

$$\left\|\nabla F_{\mathrm{G}}\left(\mathbf{w}_1\right) - \nabla F_{\mathrm{G}}\left(\mathbf{w}_2\right)\right\|_2 \leq L\left\|\mathbf{w}_1 - \mathbf{w}_2\right\|_2. \tag{21}$$

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.,

$$\left\|\mathbf{g}_{\mathrm{L},u}^{(k)} - \mathbf{g}_{\mathrm{G}}^{(k)}\right\|_2 \leq \zeta. \tag{22}$$

Assumption 3 (Bounded Gradient Aggregation) At each time slot, the received gradient via async-AirFed aggregation has a limited norm and is linearly bounded by the ideal global gradient, i.e,

$$\left\|\mathbf{g}_{\mathrm{A}}^{(k)}\right\|_2 \leq \chi\left\|\mathbf{g}_{\mathrm{G}}^{(k)}\right\|_2. \tag{23}$$

Assumption 4 (Bounded Global Gradient Staleness) For each time slot, the norm of the delayed gradient within the max-staleness steps is linearly bounded by the norm of the latest gradient, which is denoted as

$$\left\|\mathbf{g}_{\mathrm{G}}^{(k-m)}\right\|_2 \leq \beta\left\|\mathbf{g}_{\mathrm{G}}^{(k-1)}\right\|_2, \quad \forall m \in \mathbb{Z} \cap [1, M]. \tag{24}$$

Theorem 2 (Constant Learning Rate) Under Assumption 1-4, the norm of model gradient in async-AirFed are bounded by

$$\min_{k\in\{1,\cdots,K\}}\left\|\mathbf{g}_{\mathrm{G}}^{(k)}\right\|^2 \leq \frac{2r_0}{\eta K} + L\eta\cdot n_{avg} + 3\zeta + 3e_{avg}, \tag{25}$$

where $r_0 = F_{\mathrm{G}}\left(\mathbf{w}_{\mathrm{G}}^{(0)}\right) - F_{\mathrm{G}}\left(\mathbf{w}_{\mathrm{G}}^{*}\right)$, n_{avg} denotes the time-averaged noise after the pre-distortion at receiver side, e_{avg} denotes the averaged transmission mean square error (MSE) caused by imperfect pre-distortion and η is the constant learning rate set by

$$\eta \leq \frac{\sqrt{1+2\beta\mathcal{P}(M)}-1}{L\beta\mathcal{P}(M)}, \tag{26}$$

where $\mathcal{P}(M)$ is a polynomial of tier number M.

Several insights can be obtained from **Theorem 2** on how the model would converge with async-AirFed. By choosing small enough learning rate, which is given by (26), the magnitude of gradient is ensured to diminish as training iterations increase. But the descent of the model gradient will encounter a floor caused by the noise and interference in the channel, as well as the non-IID data among clients. The impact of channel noise is mitigated by the learning rate less than 1, as the aforementioned comparison between model aggregation and gradient aggregation.

V. Numerical Results

We consider a wireless network with 200 clients and a parameter server. The clients are uniformly distributed inside a circle of radius $R_{\max} = 400\mathrm{m}$. The channel coefficients consist of path loss with $\alpha = 3.76$ [5] and small-scale fading that follows Rayleigh distribution with $\sigma = 1$. The total bandwidth is $B = 10\mathrm{MHz}$, and the noise power spectrum density is $N_0 = -174\mathrm{dBm/Hz}$. The average power constraint is $P_0 = 20\mathrm{dBm}$.

We build a multilayer perceptron (MLP) with 128 hidden units, ReLU activation function, and a $\mathrm{softmax}$ output layer. The number of FLOPs requested for computing one data sample on this model is $C = 203106$ [7]. The model is trained and then evaluated on the MNIST dataset. To study the convergence properties under severe heterogeneity, we assume that the number of data samples held by clients follows a heavy-tailed Zipf distribution with an exponent of 0.5. For performance comparison, we consider three benchmarks:

1) **FedAsync** [8]: Each client is assigned an orthogonal dedicated channel. Once a client completes its computation, it immediately updates the global model.
2) **TT-Fed** [5]: The communication scheme is similar to FedAsync, while the aggregation occurs only at the end of each time slot.

   We use channel capacity to estimate the time required for digital communication in the above two methods, assuming that model parameters/gradients are 16 bits.

3) **AirFed** [9]: The global model uses synchronous aggregation, while the parameter server receives updates via AirComp.

We use the pre-distortion algorithm proposed in [10] for AirFed and our proposed async-AirFed, which minimizes the communication MSE under instantaneous average power constraints.

Fig. 2 plots the testing accuracy during training of our proposed async-AirFed and other benchmarks. The tier number for async-AirFed and TT-Fed is set to be $M = 10$, namely, $\Delta T = 0.1 \cdot \max_u (\tau_u^{\text{cp}} + \tau_u^{\text{cm}})$. Gradient aggregation is selected instead of model aggregation, and the accumulated error truncation is deployed to async-AirFed to deal with channel noise and interference. The truncation order is set to be $n_t = 4$. For a fair comparison, all schemes use the same constant learning rate $\eta = 0.05$. The computational speed of each client's CPU is assumed to be $f_c = 1\text{GFLOPS}$.

Due to noise and interference from AirComp and staleness from async aggregation, the training curve of async-AirFed shows strong fluctuations. However, it still has the highest convergence rate compared to other benchmarks. It is observed that although AirFed exists straggers issue, and is not proposed to solve this problem, it still has a higher convergence speed than the other two async aggregation algorithms in digital communication due to the shorter communication latency. TT-Fed is more suitable for scenarios with significant heterogeneity in sample classification [5], such as vertical federated learning. Therefore, in our experiments which is a horizontal FL setting, the convergence speed of TT-Fed is slower than FedAsync.

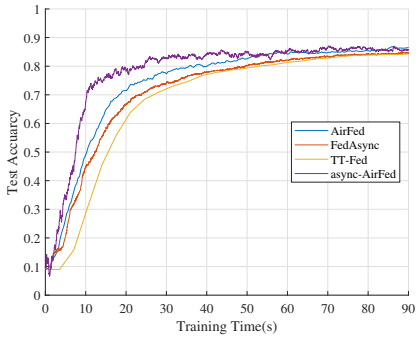

Fig. 2. Testing accuracy during training our proposed async-AirFed and other three benchmarks.

Fig. 3 plots the average time utilization among all clients during the whole training process under the four schemes, which further explains the reason why the four schemes have different convergence speed. The labels "comp", "comm", and "wait" in the legend represent the computation time, the communication time, and the idle time waiting for stragglers, respectively. Compared to AirFed that also utilizes AirComp, our async-AirFed reduces the time to wait for stragglers from 84% to 9%. As a cost, the communication time increases from 3% to 16% because of higher aggregation frequency required by async-AirFed. The maximum ratio of the increase in

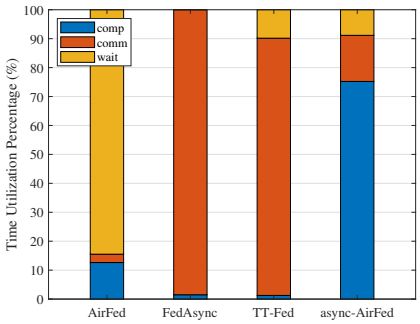

Fig. 3. Time utilization during model training.

communication time is less than the total tiers number M, with the supremum reached when almost all clients are assigned to the first tier. Compared to FedAsync and TT-Fed which utilize asynchronous aggregation as well, the communication time of our async-AirFed is reduced by 82% and 73% respectively, therefore more time can be assigned to execute computation tasks.

VI. Conclusion

In this paper, we proposed an asynchronous FL algorithm that integrated with AirComp, namely async-AirFed. We further analyzed the challenges brought by the combination of async and AirComp and compared the impact of model aggregation and gradient aggregation on convergence. We proposed a simple truncation method to eliminate the accumulation of errors. Our convergence analysis proved that async-AirFed converges on non-convex optimality function with sub-linear rate. Finally, our simulation results demonstrated that async-AirFed could achieve more than 34% faster convergence than benchmarks by improving the average time utilization efficiency among all clients.

References

[1] H. B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. Y. Arcas, "Communication-efficient learning of deep networks from decentralized data," in *Proc. AISTATS*, Apr. 2016, pp. 1273–1282.

[2] C. Zhang, Y. Xie, H. Bai, B. Yu, W. Li, and Y. Gao, "A survey on federated learning," in *Knowl.-Based Syst.*, vol. 216, Mar. 2021, Art. no. 106775.

[3] A. Şahin and R. Yang, "A survey on over-the-air computation," in *arXiv:2210.11350*, 2022.

[4] C. Xu, Y. Qu, X. Yong, and L. Gao, "Asynchronous federated learning on heterogeneous devices: A survey," *arXiv:2109.04269*, 2021.

[5] X. Zhou, Y. Deng, H. Xia, S. Wu and M. Bennis, "Time-triggered federated learning over wireless networks," in *IEEE Trans. Commun.*, vol. 21, no. 12, pp. 11066-11079, Dec. 2022.

[6] Z. Chai, Y. Chen, L. Zhao, Y. Cheng, and H. Rangwala, "FedAT: A communication-efficient federated learning method with asynchronous tiers under non-IID data," *arXiv:2010.05958*, 2020.

[7] P. Molchanov, S. Tyree, T. Karras, T. Aila, and J. Kautz, "Pruning convolutional neural networks for resource efficient inference," in *Proc. ICLR*, Apr. 2017, pp. 1–17

[8] C. Xie, S. Koyejo, and I. Gupta, "Asynchronous federated optimization," *arXiv:1903.03934*, 2019.

[9] 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, Jan. 2020.

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