Async-HFL: Efficient and Robust Asynchronous Federated Learning in Hierarchical IoT Networks

Xiaofan Yu
x1yu@ucsd.edu
University of California San Diego
La Jolla, California, USA

Ludmila Cherkasova
lucy.cherkasova@gmail.com
Arm Research
San Jose, California, USA

Harsh Vardhan
hharshvardhan@ucsd.edu
University of California San Diego
La Jolla, California, USA

Quanling Zhao
quzhao@ucsd.edu
University of California San Diego
La Jolla, California, USA

Emily Ekaireb
eekaireb@ucsd.edu
University of California San Diego
La Jolla, California, USA

Xiyuan Zhang
xiyuanzh@ucsd.edu
University of California San Diego
La Jolla, California, USA

Arya Mazumdar
arya@ucsd.edu
University of California San Diego
La Jolla, California, USA

Tajana Šimunić Rosing
tajana@ucsd.edu
University of California San Diego
La Jolla, USA

ABSTRACT

Federated Learning (FL) has gained increasing interest in recent years as a distributed on-device learning paradigm. However, multiple challenges remain to be addressed for deploying FL in real-world Internet-of-Things (IoT) networks with hierarchies. Although existing works have proposed various approaches to account data heterogeneity, system heterogeneity, unexpected stragglers and scalability, none of them provides a systematic solution to address all of the challenges in a hierarchical and unreliable IoT network. In this paper, we propose an asynchronous and hierarchical framework (*Async-HFL*) for performing FL in a common three-tier IoT network architecture. In response to the largely varied networking and system processing delays, *Async-HFL* employs asynchronous aggregations at both the gateway and cloud levels thus avoids long waiting time. To fully unleash the potential of *Async-HFL* in converging speed under system heterogeneities and stragglers, we design *device selection* at the gateway level and *device-gateway association* at the cloud level. Device selection module chooses diverse and fast edge devices to trigger local training in real-time while device-gateway association module determines the efficient network topology periodically after several cloud epochs, with both modules satisfying bandwidth limitations. We evaluate *Async-HFL*'s convergence speedup using large-scale simulations based on ns-3 and a network topology from NYCMesh. Our results show that *Async-HFL* converges 1.08-1.31x faster in wall-clock time and saves up to 21.6% total communication cost compared to state-of-the-art asynchronous FL algorithms (with client selection). We further validate *Async-HFL* on a physical deployment and observe its robust convergence under unexpected stragglers.

IoTDI '23, May 9–12, 2023, San Antonio, TX, USA

ACM ISBN 979-8-4007-0037-8/23/05.
https://doi.org/10.1145/3576842.3582377

CCS CONCEPTS

• **Computer systems organization → Sensor networks**; • **Computing methodologies → Machine learning**.

KEYWORDS

Federated Learning, Hierarchical Sensor and IoT Networks, Asynchronous FL.

ACM Reference Format:
Xiaofan Yu, Ludmila Cherkasova, Harsh Vardhan, Quanling Zhao, Emily Ekaireb, Xiyuan Zhang, Arya Mazumdar, and Tajana Šimunić Rosing. 2023. Async-HFL: Efficient and Robust Asynchronous Federated Learning in Hierarchical IoT Networks. In *International Conference on Internet-of-Things Design and Implementation (IoTDI '23), May 9–12, 2023, San Antonio, TX, USA.* ACM, New York, NY, USA, 18 pages. https://doi.org/10.1145/3576842.3582377

1 INTRODUCTION

Embedding intelligence into ubiquitous IoT devices can perform more complex tasks, thus benefiting a wide range of applications including personal healthcare [9], smart cities [28], and self-driving vehicles [6]. To enable distributed learning in a large-scale network, Federated Learning (FL) has appeared as a promising paradigm. The learning procedure begins with the central server distributing the global model to selected devices. Then each device trains with gradient descent on its local dataset and sends the updated model back to the server. Finally, the central server aggregates the received models to obtain a new global model. Edge devices do not reveal the local dataset but only share the updated model. Hence FL collaboratively learns from distributed devices while preserving users' privacy. The canonical baseline in FL is Federated Averaging (FedAvg) [41] which employs synchronous global aggregation - the central server performs aggregation after the slowest device returns, thus is impeded by unacceptable long delays or stragglers. Recent contributions on semi-asynchronous FL [12, 15, 43, 51, 58] alleviate the issue by aggregating updates that arrive within a certain period and dealing with late model updates asynchronously. However, the semi-asynchronous scheme still suffers from untimely updates

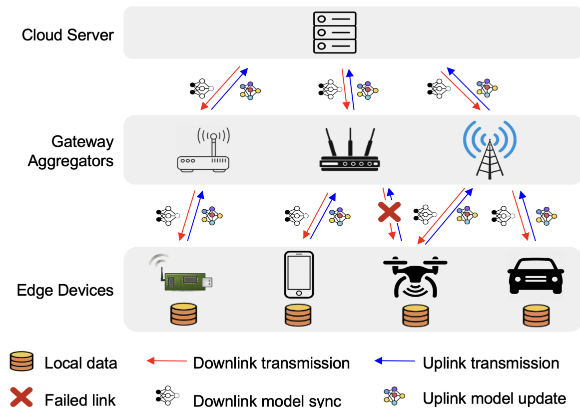

Figure 1: System architecture for a common three-tier IoT networks running FL applications with heterogeneous delays and unreliable networks.

with hard-to-tune waiting periods on heterogeneous and unreliable networks.

For real-world IoT networks, we recognize that the diverse nature of the overall system prevents an efficient and robust FL deployment. A large number of real deployments are organized in a hierarchical manner, for example, NYCMesh [2], HPWREN [1]. All of these architectures can be simplified to the three-tier structure of cloud server, gateway aggregators, and edge devices as shown in Fig. 1. The cloud layer offers powerful servers with abundant resources and effective processing capabilities. The gateway layer includes base stations and routers, acting as an intermediate hub connecting the cloud and edge devices. The bottom layer of edge devices refers to small mobile systems like sensors, smartphones, drones, etc., usually subject to limited resources and energy.

Deploying FL on heterogeneous and hierarchical IoT networks faces the following challenges:

(C1) **Heterogeneous data distribution**: the distribution of local data on edge devices can be largely different due to environmental variations or users' specifics. Non-independent and identically distributed (non-iid) data has been shown to slow down or prevent FL convergence without careful and tailored management [29, 37, 48].

(C2) **Heterogeneous system characteristics**: The edge devices are equipped with various CPU chips, memory storage and communication technologies. As a result, in Fig. 1, the computational delay on each layer and the communication delay between two layers can be largely different. Applying synchronous and semi-asynchronous FL are subject to longer waiting time.

(C3) **Unexpected stragglers (or device dropout)**: Stragglers are common in every layer of IoT networks, due to energy shortage, circuit shortage or wireless interference. Without careful management, the learning procedure might be delayed or completely hang up due to stragglers.

(C4) **Scalability**: Naïvely extending a two-tier algorithm to hierarchical networks (3 tiers and more) can lead to significant performance degradation, e.g., unconverged model, significant communication load [39]. How to preserve the positive gains while avoiding undesired degradation during scaling to hierarchical architectures remains an active research topic.

While previous works have studied how to improve FL convergence under one or two of data heterogeneity [29, 37, 48], system heterogeneity [11, 32, 36], unexpected stragglers [42], and hierarchical FL for better scalability [20, 59], none of existing work provides a *systematic* solution to address all challenges in a *hierarchical and unreliable* IoT network. Our work is the *first* end-to-end framework that uses (i) asynchronous and hierarchical FL algorithm and (ii) system management design to enhance efficiency and robustness, for handling all challenges (C1)-(C4).

In this paper, we propose *Async-HFL*, an asynchronous and hierarchical framework for performing FL in three-tier and unreliable IoT networks.

• On the algorithmic side, *Async-HFL* utilizes asynchronous aggregations at both the gateway and the cloud, i.e., the aggregation is performed immediately after receiving a new updated model. Therefore, fast edge devices do not have to wait for the slower peers and stragglers can easily catch up after downloading the latest global model. Compared to naïvely extending existing two-tier asynchronous FL to a three-tier hierarchy, *Async-HFL* stabilizes convergence and saves communication cost by adding an intermediate gateway aggregation layer.

• Moreover, on the system management side, we propose two modules to improve the performance of asynchronous algorithm under system heterogeneities and stragglers. We design *device selection module* at the gateway level and *device-gateway association module* at the cloud level. Gateway-level device selection determines which device to trigger local training in real-time, while cloud-level device-gateway association manages network topology (i.e., which device connects to which gateway) for longer-term performance. Both modules formulate and solve Integer Linear Programs to jointly consider data heterogeneity, system characteristics, and stragglers. For data heterogeneity, we define the *learning utility* metric to quantify gradient affinity and diversity of devices, inspired from online coreset selection [56]. For system heterogeneity, we monitor the latencies per gateway-device link and available connections.

• *Async-HFL* is different from previous works in that (i) *Async-HFL* considers *finer-grained* information of gradient diversity instead of just loss values (as in state-of-the-art asynchronous client selection algorithms [60]), and (ii) *Async-HFL* incorporates device selection and network topology management at various tiers, which collaboratively optimize model convergence in hierarchical and unreliable IoT networks. To minimize communication overhead, during warm up, we collect the gradients of all devices, perform Principle Component Analysis (PCA) and distribute the PCA parameters to all devices. During training, only the principle components of gradients are exchanged.

In summary, the contributions of *Async-HFL* are listed as follows:

- Recognizing the unique challenges in hierarchical and unreliable IoT networks, *Async-HFL* uses asynchronous aggregations at both the gateway and cloud levels. We formally prove the convergence of *Async-HFL* under non-iid data distribution.

- To quantify data heterogeneity in a finer manner, we propose the *learning utility* metric based on gradient diversity to guide decision making in *Async-HFL*.
- To collaboratively optimize model convergence under data heterogeneity, system heterogeneity and stragglers, *Async-HFL* incorporates distributed modules of the *gateway-level device selection* and the *cloud-level device-gateway association*. Communication overhead is reduced by exchanging compressed gradients from PCA.
- We implement and evaluate *Async-HFL*'s convergence speedup and communication saving under various network characteristics using large-scale simulations based on ns-3 [3] and NYCMesh [2]. Our results demonstrate a speedup of 1.08-1.31x in terms of *wall-clock* convergence time and total communication savings of up to 21.6% compared to state-of-the-art asynchronous FL algorithms. We further validate *Async-HFL* on a physical deployment with Raspberry Pi 4s and CPU clusters and show robust convergence under stragglers.

Relationship with other FL research: *Async-HFL* focuses on addressing system variations and potential stragglers in hierarchical IoT networks, thus is orthogonal to other FL techniques of personalization [46], pruning [34] and masking [35]. Combining *Async-HFL* with the above-mentioned techniques is feasible but is not the focus of this paper.

The rest of the paper is organized as follows: Section 2 reviews related works. Section 3 introduces background and models. Section 4 presents a motivating study. Section 5 expands on the details of *Async-HFL*. Section 6 covers the experimental setups and results. Finally, the whole paper is concluded in Section 7.

2 RELATED WORK

In this section, we review state-of-the-art works and summarize the existing frameworks in Table 1 with regard to challenges (C1)-(C4).

Table 1: Comparing *Async-HFL* and existing works.

Method	Challenges			
	(C1)	(C2)	(C3)	(C4)
Sync FL (two tier)	✓	✓	×	×
Async FL (two tier)	✓	✓	✓	×
Hier. FL (>two tier)	✓	✓	×	✓
Async-HFL (three tier)	✓	✓	✓	✓

Synchronous FL. Based on FedAvg, a large number of works have studied synchronous FL under data and system heterogeneity from both theoretical and practical perspectives [11, 29, 32, 37, 42, 48]. The client (or device) selection procedure can be carefully designed to mitigate heterogeneity by leveraging various theoretical tools [8, 30, 44, 47, 54, 55]. While most works only consider data and computational delay heterogeneity, TiFL [11], Oort [32] and PyramidFL [36] brought up the communication delay variation and implemented smart client selection to balance statistical and system utilities. Nevertheless, all above works consider FL performing in data centers, while long delays and stragglers in unreliable IoT networks can lead to unsatisfied performance with synchrony.

Asynchronous FL. In contrast to synchronous FL, the asynchronous scheme leads to faster convergence under unstable networks especially with millions of devices [26]. An increasing number of asynchronous FL works have been published in recent years, with focuses on client selection [16, 23, 27, 60, 61], weight aggregation [50, 50, 57] and transmission scheduling [33]. Semi-asynchronous mechanisms are developed to aggregate buffered updates [12, 15, 43, 51, 58]. However, how to fully utilize the asynchronous property in hierarchical and heterogeneous IoT systems have not been addressed.

Hierarchical FL. In hierarchical FL, gateways perform intermediate aggregations before sending their local models to the cloud, so that the backhaul communications between gateways and the cloud are reduced [39]. Multiple works have formulated client association and resource allocation problems to jointly optimize computation and communication efficiency in synchronous hierarchical FL [4, 5, 39, 40]. Recent efforts studied mobility-aware [20] and dynamic hierarchical aggregations for new data [59]. SHARE [18] separated the device selection and device-gateway association into two subproblems, then jointly minimized communication cost and shaped data distribution at aggregators for better global accuracy. RFL-HA [49] adopts synchronous aggregation within each subcluster and asynchronous aggregation between cluster heads (gateways) and the central cloud. All above works employ synchronous aggregation in the system, thus suffering from stragglers.

The only work that studies asynchronous and hierarchical FL is [50]. In contrast, we provide a systematic framework with additional management modules to adaptively optimize convergence under (C1)-(C4) in real-time.

3 BACKGROUND

In this section, we provide background on learning model (Section 3.1), system model (Section 3.2) and common techniques in existing two-tier asynchronous FL (Section 3.3). To help the readers get familiar with the notations, we present the list of notations in Table 2. We also depict an example deployment in Fig. 2 which will be referred to as we introduce the models.

3.1 Learning Model

Suppose each sensor device i collects data points $(x^i, y^i) \sim \mathcal{D}^i$ where $\mathcal{D}^i$ indicates the underlying data distribution at i. We assume all distributions are drawn from the same domain. Non-iid data distribution happens when $\mathcal{D}^i \neq \mathcal{D}^j, \forall i \neq j$. In practice, the underlying distribution $\mathcal{D}^i$ is unknown and we only have access to a finite number of n_i samples at each device. Note, that n_i can be different on various devices and at various time while we omit the time subscript for simplicity. We aim for the typical goal of FL: to learn a uniform model $\boldsymbol{\omega} \in \mathbb{R}^d$ to be deployed on all distributed devices. Combining with personalized models is also feasible but exceeds the scope of this paper. The loss function $\ell(\boldsymbol{\omega}; x^i, y^i)$ is defined as an error function of how well the model $\boldsymbol{\omega}$ performs with respect to sample (x^i, y^i). We settle for minimizing the empirical risk minimization problem (ERM) over all devices as follows:

$$\min_{\boldsymbol{\omega} \in \mathbb{R}^d} L_N(\boldsymbol{\omega}) = \frac{1}{N} \sum_{i=1}^{N} L^i(\boldsymbol{\omega}) = \frac{1}{N} \sum_{i=1}^{N} \frac{1}{n_i} \sum_{k=1}^{n_i} \ell(\boldsymbol{\omega}; x_k^i, y_k^i), \quad (1)$$

Table 2: List of important notations in problem formulation.

Symbol	Meaning	Symbol	Meaning
C	Central cloud	$\mathcal{G}$	Set of G gateways
$\mathcal{N}$	Set of N sensor nodes	$\mathbf{J}_t$	Feasible gateway-sensor links at time t
$\mathbf{I}_t$	Connected gateway-sensor links at time t	τ_i^C	Computational delay on device i
τ_{ji}^D, τ_{ij}^U	Downlink and uplink delays between j and i	τ_{ij}	Gateway round latency between j and i
R_{ij}	Average data rate on link i, j	R_j	Data rate of all selected devices at gateway j
B_j	Bandwidth limitation at gateway j	$\ell(\boldsymbol{\omega}; x, y)$	Loss function defined on $\boldsymbol{\omega}$ and (x, y)
$L^i(\boldsymbol{\omega})$	Empirical loss function at device i	$L_N(\boldsymbol{\omega})$	Empirical loss function at central cloud
$\mathcal{D}^i$	Data distribution at sensor node i	n_i	Number of samples at sensor node i
$\boldsymbol{\omega}_h$	Global model weights after h cloud epochs	γ	Learning rate at sensor nodes
$\boldsymbol{\omega}_{\tau,z}^j$	Gateway model at gateway j downloaded from cloud at τ and aggregated after z gateway epochs	$\boldsymbol{\omega}_{\tau,\zeta,e}^i$	Sensor model at sensor node i downloaded from gateway at ζ, which the gateway downloaded from the cloud at τ and aggregated after e device epochs
H, Z, E	Number of cloud, gateway and device epochs	ρ	Regularization weight in async FL algorithm
α, β	Exponential decay factor at cloud and gateways	$s(\cdot)$	Staleness function
u_i, η_i, v_i	Learning utility, gradients affinity and diversity metric for device i	κ, ϕ	Hyperparameter in device selection and association

Here $L_N(\boldsymbol{\omega})$ is the global loss function at the central cloud, and $L^i(\boldsymbol{\omega})$ is the loss function at sensor device i.

3.2 System Model

Network topology. In a hierarchical IoT network, suppose C denotes the central cloud. Let $[n]$ be a set of integers $\{1, ..., n\}$. We define $\mathcal{G} = [G]$ as a set of G gateways (or base stations) and $\mathcal{N} = [N]$ as a large set of N static deployed sensor devices. While all gateways should have feasible paths to the central cloud, there might be multiple gateways that are reachable from one sensor device. Reachability can be limited due to multiple reasons such as range limitation of wireless technology, failed sensor device or network backbones. We combine all above factors into one notation, matrix $\mathbf{J}_t \in \mathbb{Z}^{N\times G}$, which indicates the feasible sensor-gateway pairs at wall-clock time t:

$$\mathbf{J}_{t,ij} = \begin{cases} 1 & \text{if sensor } i \text{ and gateway } j \text{ are connectable at time } t \\ 0 & \text{otherwise.} \end{cases} \quad (2)$$

During training, a sensor is associated with only one gateway at one time. The gateway triggers local training on the device, and the device needs to upload the returned model to the same gateway. But sensors can switch to another gateway between aggregations. We use another matrix notation $\mathbf{I}_t \in \mathbb{Z}^{N\times G}$ with the same shape as $\mathbf{J}_t$ as decision variables for real-time sensor-gateway connections:

$$\mathbf{I}_{t,ij} = \begin{cases} 1 & \text{if sensor } i \text{ is connected to gateway } j \text{ at time } t \\ 0 & \text{otherwise.} \end{cases} \quad (3)$$

For example, in Fig. 2, sensor device 2 can reach both gateway 1 and 2, but communicates with gateway 1 in the current round. In this case, $\mathbf{J}_{t,21} = \mathbf{J}_{t,22} = 1, \mathbf{I}_{t,21} = 1, \mathbf{I}_{t,22} = 0$.

Computational and Communication Models. Computational and communication delays play the major role as system heterogeneities. In this paper, we adopt general models while more specific parameterized CPU or network models (such as [14, 40]) can be applied when more information is given. We are more interested in the communication delays happened on the last-hop links, where bandwidth is more limited and the transmitters (sensor devices) enjoy lower transmission power. During runtime, we measure *round latency* τ_{ij} as the time to complete one gateway round between gateway j and sensor i, which has three segments: (i) τ_{ji}^D, the downlink delay to transmit a model from gateway j to device i, (ii) τ_i^C, the computational delay of device i to perform local training and (iii) τ_{ij}^U, the uplink delay to transmit the updated model in a reverse direction. In Fig. 2, the downlink and uplink transmissions are represented with the red and blue arrows, while local training uses green arrow. The round latency is simply the sum of red, green and blue arrows: $\tau_{ij} = \tau_{ji}^D + \tau_i^C + \tau_{ij}^U$. The computational delay at gateways and cloud for aggregation are neglected since gateways and cloud generally have more computational resource.

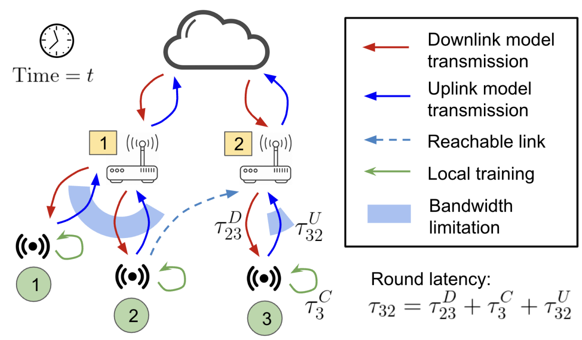

Figure 2: An example of hierarchical FL deployment.

Bandwidth Limitation. Bandwidth limitation places an upper bound on the throughput or data rate. Different from the synchronous design, we cannot accurately model the throughput at t with asynchrony. Therefore, given round latency τ_{ij} and a FL model with size M, we estimate the average data rate on link i, j as $R_{ij} = M/\tau_{ij}$.

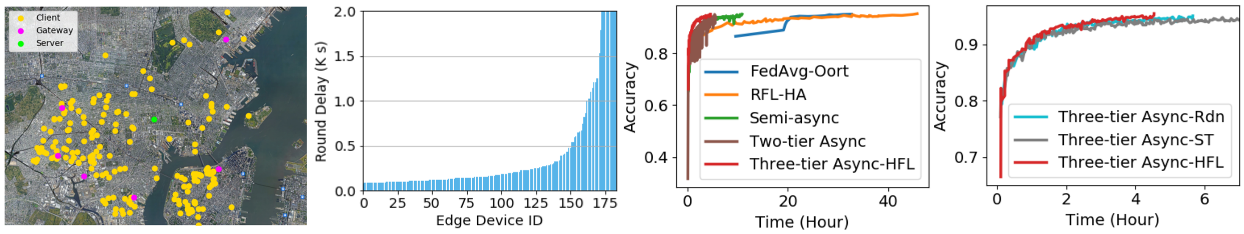

Figure 3: Left: NYCMesh topology. Second left: The round delay distribution of all edge devices. Right: Convergence performance under wall-clock time in the NYCMesh motivating study.

The total data rate of all selected devices at gateway j can be computed as $R_j = \sum_{i=1}^{N} \boldsymbol{I}_{ij} R_{ij}$. B_j is an upper bound of *average* data rate on last-hop links at gateway j. In Fig. 2, devices 1 and 2 are subject to B_1, which is depicted as a blue arc.

3.3 Two-Tier Asynchronous FL

In asynchronous FL, each device downloads the latest global model from the cloud, runs local training, and uploads the model to the cloud where asynchronous aggregation is performed immediately. The latest asynchronous FL algorithms [16, 53] employ two common techniques as follows.

Firstly, in addition to the original loss term $L^i(\boldsymbol{\omega}^i)$, a **regularized loss term** penalizing the difference between current model weights $\boldsymbol{\omega}^i$ and the downloaded global model $\boldsymbol{\omega}_\tau$ is appended on device i:

$$g^i(\boldsymbol{\omega}^i; \boldsymbol{\omega}_\tau) = L^i(\boldsymbol{\omega}^i) + \frac{\rho}{2} \|\boldsymbol{\omega}^i - \boldsymbol{\omega}_\tau\|^2. \tag{4}$$

Here ρ is the regularization weight.

Secondly, the algorithm performs **staleness-aware weight aggregation** at the cloud:

$$\alpha_h \leftarrow \alpha \cdot s(h - \tau) \tag{5a}$$

$$\boldsymbol{\omega}_h \leftarrow (1 - \alpha_h)\boldsymbol{\omega}_{h-1} + \alpha_h \boldsymbol{\omega}_{new}, \tag{5b}$$

where ω_{new} is the newly received model weights, h is the current cloud epoch and α_h is the staleness-aware weight calculated by multiplying a constant α with the staleness function $s(h - \tau)$. Staleness refers to the difference in *the number of epochs* since its last global update. For example, h is the current global aggregating epoch while τ is the global epoch when the model is downloaded. Intuitively, larger staleness means the model is more outdated and thus should be given less importance. Staleness-aware aggregation simulates an averaging process without synchrony. The staleness function $s(h - \tau)$ determines the exponential decay factor during model aggregation. We adopt the polynomial staleness function $s_q(h - \tau) = (h - \tau + 1)^{-q}$ parameterized by $q > 0$ as in [53].

4 A MOTIVATING STUDY

In this section, we conduct a motivating study of existing FL frameworks under hierarchical and unreliable networks, justifying the design of *Async-HFL* on both algorithmic and management aspects. While recent works have noticed the importance of accounting the latency factor during client selection [11, 32, 36], they only considered the delay distribution in a *data-center* setting which is significantly different from the ones in *real-world wireless networks*. Real-world measurements have shown that wireless networks follow the long-tail delay distribution and are highly unpredictable [45]. We implement the FL frameworks based on ns3-fl [19] and extract the three-tier topology from the installed node locations in NYCMesh [2] as depicted in Fig. 3 (left). We assume that edge devices are connected to the gateways via Wi-Fi, and the gateways are connected to the server via Ethernet. For each node, we retrieve its latitude, longitude, and height as input to the `HybridBuildingsPropagationLossModel` in ns-3 to obtain the average point-to-point latency. To include network uncertainties, we add a log-normal delay on top of the mean latency at each local training round. The delay distribution of all edge devices (assuming all devices are selected) in one training round is shown in Fig. 3 (second left). The simulated network delays mimic the measurement results in [45]. We use the human activity recognition dataset [7], assigning the data collected from one individual to one device thus presenting naturally non-iid data. The upper bound on the bandwidth of the gateways is set to 20KB/s for all experiments.

Table 3: Total communicated data size ratio (to Async-HFL) before reaching 95% test accuracy in the motivating study.

Sync-Oort	RFL-HA	Semi-async	Two-tier Async
0.79x	1.42x	1.66x	1.30x

In such setting, we experiment the performance of (i) **FedAvg under Oort**, the state-of-the-art latency-aware client selection algorithm [32], (ii) **RFL-HA** [49] with asynchronous cloud aggregation and synchronous gateway aggregation, (iii) **Semi-async**, with synchronous cloud aggregation and semi-asynchronous gateway aggregation as in [43], (iv) **two-tier Async FL** which naïvely extends the two-tier asynchronous algorithm [16, 53] to three-tier by letting the gateway just forward data, (v) **three-tier Async-HFL** with intermediate gateway aggregation proposed in our paper.

We report the wall-clock time convergence in Fig. 3 (right) and the total size of the data communicated in ratio to Async-HFL in Table 3. The convergence of FedAvg and RFL-HA are significantly slowed down even with the state-of-the-art client selection. For Semi-async, we test the waiting period of 50, 100, 150 seconds and select the best results. Semi-async accelerates convergence but still takes 0.39x longer to reach the same 95% accuracy than the fully asynchronous methods. Noticeably, compared to two-tier async, our three-tier Async-HFL achieves more stable and slightly faster convergence, while saving 0.3x communication load (equivalent to 426MB or 1498 multilayer perceptron models). This gain comes from the intermediate gateway aggregation. Introducing an additional

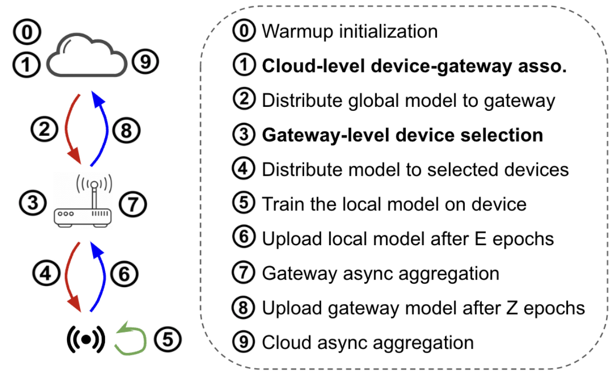

Figure 4: The step-by-step procedure of *Async-HFL* in one branch of the hierarchical network.

"averaging" step does not only smooth out the curve but avoids unnecessary back-and-forth transmission.

Aside from algorithmic design, framework management is also critical in hierarchical FL. We experiment with random (**Async-Random**), short-latency-first (**Async-ST**) gateway-level device selection and **Async-HFL** with full management. The convergence results are shown in Fig. 3 (right). With carefully designed modules, Async-HFL converges 1.24x faster than the random gateway-level device selection. Yet, a poor device selection like Async-ST that ignores data heterogeneity can lead to a 2.13x slower speed to reach 95% accuracy or even an unconverged model in the worst case.

5 *ASYNC-HFL* DESIGN

5.1 Overview

To address all challenges (C1)-(C4) systematically, we propose an end-to-end framework *Async-HFL* for efficient and robust FL especially in hierarchical and unreliable IoT networks. The *major differences* between *Async-HFL* and previous frameworks are the following: **(i)** *Async-HFL* quantifies non-iid data distribution by *learning utility*, which is a metric based on gradient diversity, **(ii)** *Async-HFL* incorporates strategic management components, the **cloud-level device-gateway association** (① in Fig. 4) and the **gateway-level device selection** (③ in Fig. 4), which are critical in jointly speeding up practical convergence under heterogeneous data and system characteristics.

Fig. 4 depicts the step-by-step procedure of *Async-HFL* in one round of cloud aggregation. For simplicity, we only show one branch of the hierarchical network. After the warmup initialization, we start from the cloud determining the low-level network topology, namely device-gateway association (①), and then distributing the latest global model to all gateways (②). Next, the gateway distributes the model to devices selected by the gateway (③-④). On the device, local training is performed for E epochs and an updated model is returned to the gateway (⑤-⑥). The gateway then integrates the newly received model immediately using asynchronous aggregation (⑦). After Z gateway updates, the current gateway model is uploaded to the cloud for global asynchronous aggregation (⑧-⑨).

We provide more details on the design of *Async-HFL* in this section. Section 5.2 presents the detailed asynchronous hierarchical algorithm with convergence proof. Section 5.3 presents the definition of the *learning utility* metric to quantify gradient diversity. Finally, Section 5.4 reveals the concrete design of device selection and device-gateway association modules.

5.2 Asynchronous Hierarchical FL Algorithm

In *Async-HFL*, besides the asynchronous cloud aggregation, we utilize an intermediate gateway layer and apply staleness-aware asynchronous aggregation for Z epochs at the gateway. Compared to having the gateways directly forwarding asynchronous model updates to the cloud, adding intermediate gateway aggregations reduces communication burden while making sure the convergence guarantees still apply after adding minimal assumptions (as detailed later). Steps ④-⑦ correspond to the two-tier asynchronous FL in Fig. 4, while our hierarchical algorithm includes steps ②, ④-⑨, spanning all three tiers in the IoT network.

The concrete algorithm implementation on cloud, gateways and devices is shown in Algorithm 1. The cloud and each gateway holds a `Cloud` or `Gateway` process, which completes the initialization and asynchronously triggers the `Updater` threads for aggregation. The `Updater` thread performs aggregation until the predetermined epoch number is reached. Each sensor device runs a `Sensor` process that locally solves a regularized optimization problem using stochastic gradient descent (SGD). While previous works have studied adaptively adjusting local epochs according to computational resources [37], namely trading model quality for faster return, such gain in *Async-HFL* might be trivial due to asynchronous aggregation and longer as well as unexpected network delays. Hence *Async-HFL* uses a fixed number of E and Z epochs on device and gateway-level.

Convergence Analysis. We now establish the theoretical convergence of *Async-HFL*'s algorithm. We set the staleness function $s(\cdot) = 1$ throughout this section. We require certain regularity conditions on the loss function, namely L-smoothness and μ-weak convexity and bounded gradients. Note that μ-weak convexity allows us to handle non-convex loss functions ($\mu > 0$), convex functions ($\mu = 0$) and strongly convex functions ($\mu < 0$). We list the additional assumptions and the convergence result below[1].

Assumption 1 (Bounded gradients). *The loss function at the cloud, L_N, and the regularized loss function at each device g_i, $\forall i \in \mathcal{N}$, have bounded gradients bounded by*

$$\|\nabla L_N(\boldsymbol{\omega})\|^2 \leq V_1, \quad \forall \boldsymbol{\omega} \in \mathbb{R}^d$$
$$\|\nabla g^i(\boldsymbol{\omega}; \boldsymbol{\omega}')\|^2 \leq V_2, \quad \forall \boldsymbol{\omega}, \boldsymbol{\omega}' \in \mathbb{R}^d, \forall i \in \mathcal{N}.$$

Assumption 2 (Bounded Delay). *The delays $h - \tau$ at the cloud model, and $z - \zeta$ at the gateway model are bounded*

$$h - \tau \leq K_c, \qquad z - \zeta \leq K_g \tag{6}$$

Assumption 3 (Regularization ρ is large). *ρ is large enough such that for some fixed constant $c > 0, \forall \tau, \zeta > 0, h \geq 1, i \in [N]$,*

$$-(1 + 2\rho + c)V_1 + \left(\rho^2 - \frac{\rho}{2}\right)\mathbb{E}[\|\boldsymbol{\omega}^i_{\tau,\zeta,h-1} - \boldsymbol{\omega}^i_{\tau,\zeta}\|^2] \geq 0 \tag{7}$$

[1]The complete proof is included in the supplementary material or can be found at https://arxiv.org/abs/2301.06646

Algorithm 1: Asynchronous Hierarchical FL

Input: $C, \mathcal{G}, \mathcal{N}, H, Z, E, \alpha, \beta, s(\cdot), g^i(\cdot), \omega_0$

Process `Cloud()`
- Send $(\omega_0, 0)$ to all gateways $j \in \mathcal{G}$
- Run `CloudUpdater()` asynchronously in parallel

Thread `CloudUpdater()`
- **for** *cloud epoch* $h \in [H]$ **do**
  - **if** *receive* (ω^j_{new}, τ) *from gateway j* **then**
    - $\omega_h \leftarrow (1-\alpha_h)\omega_{h-1} + \alpha \times s(h-\tau)\omega^j_{new}$
    - Send (ω_h, h) to all gateways $j \in \mathcal{G}$

Process `Gateway(`j`)`
- **if** *triggered by* `Cloud()` **then**
  - Receive global model and timestamp (ω_h, h)
  - Update $\tau \leftarrow h, \omega^j_{\tau,0} \leftarrow \omega_h$
  - Send $(\omega^j_{\tau,z}, z)$ to selected sensors i connected to j
  - Run `GatewayUpdater(`j`)` asynchronously in parallel

Thread `GatewayUpdater(`j`)`
- **for** *gateway epoch* $z \in [Z]$ **do**
  - Receive (ω^i_{new}, ζ) from sensor node i
  - $\omega^j_{\tau,z} \leftarrow (1-\beta^j_z)\omega^j_{\tau,z-1} + \beta^j_z \omega^i_{new}$
  - Send $(\omega^j_{\tau,z}, z)$ to all sensor i connected to j
- Upload $(\omega^j_{\tau,Z}, \tau)$ to cloud

Process `Sensor(`i`)`
- **if** *triggered by* `Gateway(`j`)` **then**
  - Receive $(\omega^j_{\tau,z}, z)$ from gateway j
  - Update $\zeta \leftarrow z, \omega^i_{\tau,\zeta,0} \leftarrow \omega^j_{\tau,z}$
  - **for** *device epoch* $e \in [E]$ **do**
    - $\omega^i_{\tau,\zeta,e} \leftarrow \omega^i_{\tau,\zeta,e-1} - \gamma \nabla g^i(\omega^i_{\tau,\zeta,e-1}; \omega^j_{\tau,z})$
  - Upload $(\omega^i_{\tau,\zeta,E}, \zeta)$ to gateway j

Theorem 5.1. *For L-smooth and μ-weakly convex loss function ℓ, under Assumptions 1-3, with* $\gamma \leq L^{-1}, \alpha \leq K_c^{-3/2}$ *and* $\beta \leq K_g^{-3/2}$, *after running Algorithm 1 for H, Z and E cloud, gateway and device epochs, we obtain*

$$\min_{h=0}^{H-1} \mathbb{E}[\|\nabla L_N(\omega_h)\|^2] \leq \frac{\mathbb{E}[L_N(\omega_0) - L_N(\omega_H)]}{\alpha\beta\gamma c H Z E} + \Xi \tag{8}$$

Theorem 5.1 extends the proof of the two-tier asynchronous FL algorithm in [53] to three tiers, namely the device-gateway-cloud architecture. Adding the extra gateway level only requires a bounded delay assumption at the gateway and adds constant terms in Ξ due to the gateway, without sacrificing the convergence rate. We are able to ensure convergence in spite of mild assumptions, for instance, constant staleness $s(\cdot) = 1$ and weak convexity, and using stronger assumptions would enable even stronger results both theoretically and empirically.

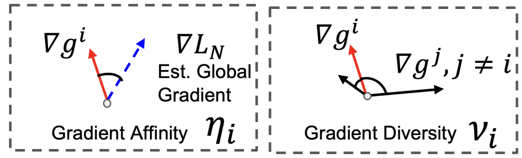

Figure 5: Visualization of the defined learning utility metric combining gradient affinity and gradient diversity.

5.3 Learning Utility

To quantify the learning-wise contribution of aggregating each device's local model to the global model, we need a metric that accounts for data heterogeneity. State-of-the-art asynchronous client selection algorithms [25] use local loss values to indicate the learning contribution of a device. Here in *Async-HFL*, we propose the *learning utility* metric which takes into account gradient diversity. Compared to loss values, gradient diversity contains finer-grained information about data heterogeneity.

We extract the latest gradient on device i: $\nabla g^i(\omega^i_{\tau,\zeta,E}, \omega^i_{\tau,z})$, and global gradient as a sum of all latest gradients: $\nabla L_N = \frac{1}{N}\sum_{i=1}^N \nabla g^i$. The *learning utility* metric u_i is defined for each device i based on the gradient affinity with the global gradient, η_i, and the gradient diversity with the other devices, ν_i:

$$u_i = \eta_i + \nu_i, \tag{9a}$$

$$\eta_i = \nabla g^{i\top} \nabla L_N, \tag{9b}$$

$$\nu_i = \frac{-1}{N-1}\sum_{j\neq i}^{N-1} \nabla g^{i\top} \nabla g^j. \tag{9c}$$

To assist understanding, a visualization example is shown in Fig. 5. The gradient affinity, η_i, evaluates the similarity between device i's gradients and the global gradients, taking the dot product of ∇g^i and ∇L_N (Equation (9b)). On the other hand, the term ν_i sums up the pairwise dissimilarity between i and all other devices to evaluate the diversity of gradients (Equation (9c)). By combining η_i and ν_i, the learning utility u_i is a device-specific metric that favors the devices with close-to-global or largely diverse data distribution. The idea of learning utility is inspired from online coreset selection [56], where the goal is to select a finite number of individual samples that preserve the maximal knowledge about data distribution to store in memory. We stress that our learning utility metric jointly considers the norm and the distribution of gradients, thus integrating more information than just the norm of gradients or loss value.

5.4 Device Selection and Device-Gateway Association

After identifying the learning utility metric to model data heterogeneity, in this section, we present the design of gateway-level device selection and cloud-level device-gateway association to enhance practical convergence of *Async-HFL*. Both modules are designed to account diverse data distribution, heterogeneous latencies and unexpected stragglers. Fig. 6 presents an overview of the design and the necessary information to be collected and exchanged. As shown, the gateway-level device selection module executes after

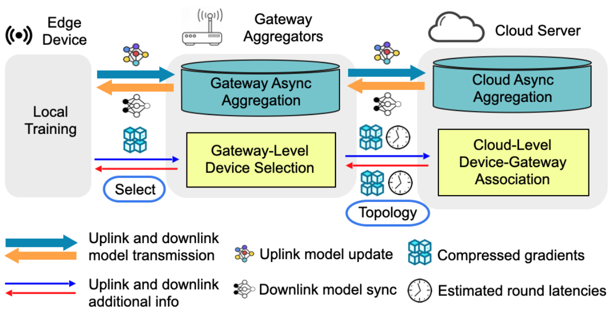

Figure 6: The overview of the distributed design of gateway-level device selection and cloud-level device-gateway association. The thick arrows represent necessary communications for FL while the thin arrows stand for communication overhead.

the previous gateway aggregation, and thus adjusts device participation in real time. The device-gateway association module is fired less frequently, once after a certain number of cloud epochs, and thus manages network topology for longer-term performance.

Gateway-Level Device Selection. Given the current set of devices connected to gateway j at time t ($\mathbf{I}_{t,ij} = 1$), we select a subset to trigger asynchronous training. Suppose $d_i = 1$ denotes that device i is selected and the latest model is transmitted from the gateway to the device, but the new updated model has not returned from the device. Once returned, the gateway-level device selection module records the compressed gradients information ∇g^i from the device, from which the *learning utility* u_i is updated. We also keep track of the moving average of device's round latency τ_{ij} at gateway j. In real-time selection, devices presenting large *learning utility* and short round latency are preferred, as these devices are able to contribute significantly to the convergence in a fast manner. We model the selection problem as an Integer Linear Program with variables of device selecting status d_i:

$$\text{(Device Selection at} j) \ \max \sum_{\mathbf{I}_{t,ij}=1} d_i u_i \left(1/\tau_{ij}\right)^{\kappa} \tag{10a}$$

$$\text{s.t.} \quad d_i R_{ij} \le B_j, \quad \forall i \in \left\{ i | \mathbf{I}_{t,ij} = 1 \right\} \tag{10b}$$

$$d_i \in \{0,1\} \quad \forall i \in \left\{ i | \mathbf{I}_{t,ij} = 1 \right\} \tag{10c}$$

Equation (10a) defines the objective combining *learning utility* and round latency, with κ as a hyperparameter that curves the contribution of round latency. Equation (10b) imposes the bandwidth upper bound at gateway j. The problem has at most $O(|N|)$ variables and linear constraints.

Cloud-Level Device-Gateway Association. Given the feasible links $\mathbf{J}_t$ at time t, we need to determine the device-gateway association $\mathbf{I}_t$ used in the following cloud epochs. We remind the readers that $\mathbf{J}_t$ denotes the real-time link availability, so unexpected device or link failures are reflected in $\mathbf{J}_t$ and our association solver is able to consider them timely. At the cloud, we retrieve the gradients and round latency information from the corresponding gateways, and send back the decided association $\mathbf{I}_t$. Previous works have shown empirically that a stronger similarity of the gateway data distribution to the global distribution leads to a faster method convergence [18]. To "shape" the gateway distribution while fully utilizing bandwidth, we formulate a multi-objective optimization problem:

$$\text{(Association at cloud)} \ \max u_{slack} - \phi R_{slack} \tag{11a}$$

$$\text{s.t.} \sum_{i=1}^{N} \mathbf{I}_{t,ij} \, u_i \ge u_{slack}, \quad \forall j \in \mathcal{G} \tag{11b}$$

$$\sum_{i=1}^{N} \mathbf{I}_{t,ij} \, R_{ij}/B_j \le R_{slack}, \quad \forall j \in \mathcal{G} \tag{11c}$$

$$\mathbf{I}_{t,ij} \le \mathbf{J}_{t,ij}, \quad \forall i \in \mathcal{N}, j \in \mathcal{G} \tag{11d}$$

$$\sum_{j=1}^{G} \mathbf{I}_{t,ij} \le 1, \quad \forall i \in \mathcal{N} \tag{11e}$$

$$\mathbf{I}_{t,ij} \in \{0,1\}, \quad \forall i \in \mathcal{N} \tag{11f}$$

The total objective in Equation (11a) balances the learning utility and the throughput of all associated devices at each gateway. Using slack variables, we are able to disassemble the max-min operation thus keep the problem an Integer Linear Program. The first objective u_{slack} is a slack variable defined as the minimal *learning utility* among all gateways (Equation (11b)). The second objective R_{slack} is a slack variable for the maximal associated throughput ratio (R_j/B_j) among all gateways (Equation (11c)). Our goal is to make a balanced allocation of devices (considering both learning utility and data rate) which are proportional to the gateways' bandwidth limitations. ϕ is used to tune the importance ratio between sub-objectives. Equation (11d) limits $\mathbf{I}_t$ to use feasible links defined by $\mathbf{J}_t$. Equation (11e) forces each device to connect to at most one gateway. The problem has $O(|G||N|)$ variables and linear constraints.

The two Integer Linear Programs are in the form of 0-1 Knapsack problem [21] which can be approached by a large number of algorithms ranging from optimal solver, greedy heuristics to meta-heuristics. In this paper, we implement both problems in the Gurobi solver [22] and show the computation overhead is negligible in a 200-node network compared to the savings in convergence time.

Minimizing Communication Overhead. The thin arrows in Fig. 6 show the communication overhead, including latest gradients ∇g^i, round latencies τ_{ij} and network topology $\boldsymbol{J}_t, \boldsymbol{I}_t$. Among these meta information for management, gradients act as the major source of overhead. To minimize the communication overhead of *Async-HFL*, we first collect all devices' gradients during warmup, then perform Principle Component Analysis (PCA) on these gradients. Afterwards, we distribute the PCA parameters to all local devices. During the real training session, only the principle components of gradients are exchanged with gateways and clouds. An overhead analysis of *Async-HFL* is presented in Section 6.7.

6 EVALUATION

6.1 Datasets and Models

To simulate heterogeneous data distribution, we retrieve non-iid datasets for four typical categories of IoT applications. The information of the datasets from each category, the partition settings and the models are summarized below.

Application #1: Image Classification. We select MNIST [17], FashionMNIST [52], CIFAR-10 [31] datasets for evaluation. We

Table 4: Statistics of federated datasets. Model size refers to the size of the packet that contains all weights in the model.

Dataset	Devices	Avg. Samples/Device	Data Partitions	Models	Size
MNIST	184	600	Synthetic (assign 2 classes to each device)	CNN	1.6MB
FashionMNIST	184	600	Synthetic (assign 2 classes to each device)	CNN	1.7MB
CIFAR-10	50	1000	Synthetic (assign 2 classes to each device)	ResNet-18	43MB
Shakespeare	143	2892.5	Natural (each device is a speaking role)	LSTM	208KB
HAR	30	308.5	Natural (each device is a human subject)	MLP	285KB
HPWREN	26	6377.4	Natural (each device is a station)	LSTM	292KB

apply CNNs with two convolutional layers for MNIST and Fashion-MNIST, and the canonical ResNet-18 [24] for CIFAR-10. All three image classification datasets are partitioned synthetically with 2 classes randomly assigned to each device, and the local samples are dynamically updated from the same distribution.

Application #2: Next-Character Prediction. We adopt the Shakespeare [10] dataset, where the goal is to correctly predict the next character given a sequence of 80 characters. Local data is partitioned by assigning the dialogue of one role to one device. We apply a two-layer LSTM classifier containing 100 hidden units with an 8D embedding layer, which is the base model for this application.

Application #3: Human Activity Recognition. We use the HAR dataset [7] collected from 30 volunteers. We assign the data from one individual to one device and apply a typical multilayer perceptron (MLP) model with two fully-connected layers.

Application #4: Time-Series Prediction. We build a time-series prediction task using the historical data collected by the High Performance Wireless & Education Network (HPWREN) [1]. HPWREN is a large-scale environmental monitoring sensor network spanning 20k sq. miles and collecting readings of temperature, humidity, wind speed, etc., every half an hour. Each reading has 11 features and we combine the readings in the past 24 hours (in total 48 readings) to be one sample. Each device holds the data collected at one station. The goal is to predict the next reading. We use the mean squared error (MSE) loss and a one-layer LSTM with 128 hidden units.

6.2 System Implementation

As each trial of FL on a large physical deployment can take up to days, we mainly use simulations to mimic practical system and network heterogeneities. We further implement and validate *Async-HFL* on a smaller physical deployment.

Large-Scale Simulation Setup[2]. We implement our discrete event-based simulator based on ns3-fl [19], the state-of-the-art FL simulator using PyTorch for FL experiments and ns-3 [3] for network simulations. Note, that in contrast to most existing frameworks that simulate communication rounds [29, 38, 48, 53], ns3-fl simulates the *wall-clock* computation and communication time based on models from realistic measurements. Approaches showing superb convergence with regard to rounds might perform poorly under wall-clock time if failing to consider system heterogeneities. The network topology is configured based on NYCMesh as described in Section 4 with 184 edge devices, 6 gateways, and 1 server.

[2]Implementation of the large-scale simulation is available at https://github.com/Orienfish/Async-HFL.

Table 5: Important parameters setup on various datasets.

Dataset	Target Acc./Err.	Gateway	γ	ρ
MNIST	95	1M	0.01	0.1
FashionMNIST	75	1M	0.01	0.1
CIFAR-10	50	20M	0.001	1.0
Shakespeare	35	20k	0.01	0.2
HAR	95	20K	0.003	0.1
HPWREN	1.5e-5 (pred. err.)	20K	0.001	0.1

Physical Deployment Setup[3]. We implement *Async-HFL* on Raspberry Pi (RPi) 4B and 400 based on the state-of-the-art framework FedML [13]. The physical deployment consists of 7 RPi 4Bs and 3 RPi 400s, distributing in 7 different houses and all connecting to the home Wi-Fi router. We ensure the variances of networking conditions by setting up some RPis in the farther end of the backyard, some in the bedroom in the vicinity of the router. We stress that the networking conditions may also be affected by the Wi-Fi traffic in real time. For example, the network delay could be longer if the residents are streaming a movie in the meantime. Such setup mimics the real-world scenarios where our application shares the bandwidth with other traffic and the network latency enjoys high diversity. In addition, during our experiment, we observe that RPis may fail to connect from the beginning, or (with rare probabilities) encounter unexpected suspension in the middle of one trial. Creating two virtual clients on each RPi, we are able to obtain a total of 20 clients on the RPi setup.

Apart from the RPis, we set up 20 more clients by requesting 1, 2 or 4 CPU cores from a CPU cluster and each accompanied by 4 GB RAM. Different from the RPi clients, where networking conditions vary largely, the CPU cluster has a stable internet connection but the computational delay varies depending on the requested resource. Since we do not have access to the home Wi-Fi router, we deploy the implementation of gateways and the cloud on an Ethernet-connected desktop, with an Intel Core i7-8700@3.2GHz, 16GB RAM and a NVIDIA GeForce GTX 1060 6GB GPU.

6.3 Experimental Setup

Baselines. Given that the major design of *Async-HFL* is around device selection and association, we adopt state-of-the-art client-selection methods from the synchronous, hybrid, semi-asynchronous, and asynchronous FL schemes to compare. We add the prefix **sync** to the baselines using synchronous aggregations at both gateway and cloud. Conversely, **async** indicates asynchronous

[3]Implementation of the physical deployment is available at https://github.com/Orienfish/FedML.

Table 6: Convergence speedup on large-scale simulations and various datasets. Bolded numbers reflect the best baseline result on each dataset.

Dataset	Convergence time speedup of *Async-HFL* with respect to baselines							
	Async-HL	Async-Random	Semi-async	RFL-HA	Sync-Oort	Sync-TiFL	Sync-DivFL	Sync-Random
MNIST	**1.11x**	1.27x	6.2x	32.5x	40.0x	27.13x	63.4x	67.3x
FashionMNIST	**1.08x**	1.49x	8.3x	36.7x	20.5x	32.8x	73.4x	96.8x
CIFAR-10	**1.09x**	1.40x	2.3x	12.3x	44.3x	59.0x	62.0x	61.7x
Shakespeare	1.19x	1.79x	0.59x	0.71x	**0.31x**	2.39x	5.87x	5.46x
HAR	1.31x	**1.22x**	2.7x	7.4x	10.3x	21.6x	22.5x	24.1x
HPWREN	**1.11x**	1.48x	2.4x	12.8x	19.5x	26.5x	27.7x	31.4x

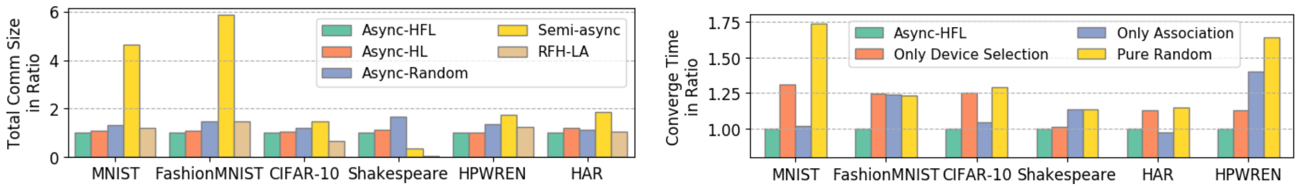

Figure 7: Left: Total communicated data size in ratio compared to *Async-HFL* on large-scale simulations and various datasets. Right: Convergence time in ratio compared to *Async-HFL* using various combinations of device selection and device-gateway association.

aggregations at both gateway and cloud. The only two baselines not following this naming rule are RFL-HA and Semi-async as follows.

- **Sync-Random/TiFL/Oort/DivFL** makes random device-gateway association while device selections are made via random selection, TiFL [11], Oort [32] and DivFL [8] respectively. TiFL groups devices with similar delays to one tier and greedily selects high-loss devices in one tier until reaching the throughput limit. Oort uses a multi-arm bandits based algorithm to balance loss and latency. DivFL utilizes a greedy method to maximize a submodular function which takes the diversity of gradients into account.
- **RFL-HA [49]** uses synchronous aggregation at gateways and asynchronous aggregation at cloud. While applying a random device selection at the gateway level, RFL-HA utilizes a re-clustering heuristic to adjust device-gateway associations.
- **Semi-async** performs semi-asynchronous aggregations at gateways as in [43] and synchronous aggregations at cloud. Random choices are applied for device selection and device-gateway association. We experiment with the semi-period of 50, 100, 150 seconds and pick the best results.
- **Async-Random/HL** uses random device-gateway association and random or high-loss first device selection under the asynchronous scheme. Prioritizing the nodes with high loss or large gradients' norm is the state-of-the-art approach for asynchronous FL [25]. We did not compare with [61] as their algorithm depends on completely different metrics.

Evaluation Metrics. For the simulation, we compare the convergence time, i.e., the wall-clock time to reach a predetermined accuracy or test loss (i.e., loss on the test dataset). Detailed parameters setup are listed in Table 5. The target accuracy or loss is close to the optimal value that is reached by FedAvg. We also compare the total communicated data size to account communication efficiency. For the physical deployment, we quantify convergence using the accuracy or test loss at the same wall-clock elapsed time. We also study the execution time, which is indicative of energy consumption on real platforms.

6.4 Results on Large-Scale Simulations

Convergence Results. We first report the simulated convergence time on all datasets using NYCMesh topology. We set device epochs $E = 5$ and gateway epochs $Z = 20$ for asynchronous, $Z = 5$ for semi-asynchronous and synchronous gateway aggregations. Each method is tested with three random trials and we report the average convergence time and the corresponding speedup in Table 6. On all three image classification datasets, *Async-HFL* achieves a minimum 1.11x, 1.08x, 1.09x speedup over the best baseline. On HAR and HPWREN datasets (with smaller number of devices, see Table 4) *Async-HFL* surpasses the best baseline by 0.22x and 0.11x respectively. Hence the design of *Async-HFL* to balance learning utility and system characteristics works under both synthetic and nature data partition. The only exception is Shakespeare, where Sync-Oort, RFL-HA and semi-async reach the target accuracy faster than *Async-HFL*. Nevertheless, we emphasize that *Async-HFL* still has 1.19x speedup compared to the state-of-the-art asynchronous FL. We speculate that the relative slower convergence of all asynchronous methods on Shakespeare roots in the essential converging difficulty of two-tier asynchronous FL algorithms. In the convergence curve of Shakespeare, we observe the first test accuracy increase with asynchronous methods after around 100 cloud epochs, while the synchronous baseline improves test accuracy at the first cloud epoch. We will refine the algorithmic design of *Async-HFL* for efficient convergence on Shakespeare in our future work. In general, synchronous methods take much longer to reach the target accuracy due to the long waiting time, even with delay-aware client selection methods such as Oort and TiFL. Both RFL-HA and Semi-async leverage hybrid aggregation scheme, thus converge slower than the fully asynchronous methods while faster than the synchronous approaches, except on Shakespeare. The convergence speedup of *Async-HFL* over state-of-the-art asynchronous methods is 1.08-1.31x on all datasets.

The total communicated data size of non-synchronous methods compared to *Async-HFL* is shown in Fig. 7 (left). We are only interested in comparing non-synchronous methods as synchronous aggregation trades long waiting time for less cloud epochs and communication savings. Therefore, synchronous methods take much

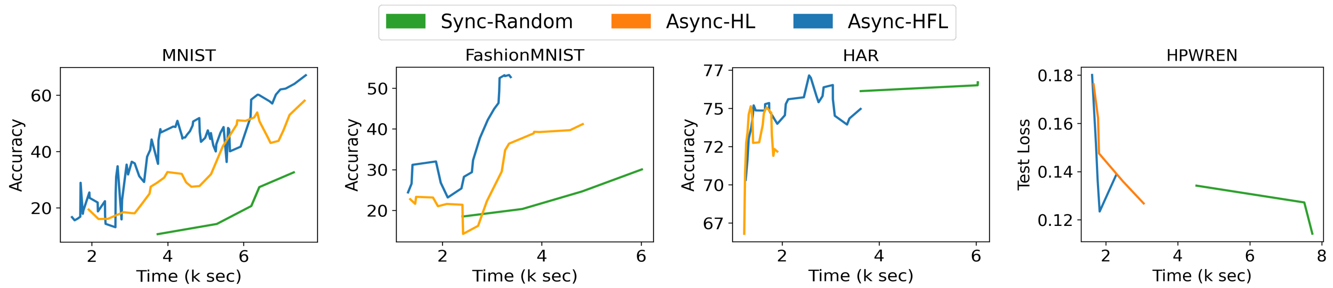

Figure 8: Convergence results under wall-clock time on the physical deployment.

less communication to converge but the slowdown is usually unacceptable as shown in Table 6. *Async-HFL* saves total communicated data size by 2.6%-21.6% and 14.5%-66.8% compared to Async-HL and Async-Random on all datasets. The total transmitted data size of semi-async and RFL-HA on Shakespeare is significantly lower than asynchronous approaches due to their fast convergence. Note, that on CIFAR-10, although RFL-HA consumes only 0.65x exchanged data compared to *Async-HFL*, it takes 11.3x longer to reach the same accuracy. The hybrid schemes of RFL-HA and Semi-async do not end up with faster convergence nor communication savings on MNIST, FashionMNIST, HPWREN and HAR.

Performance Breakdown. *Async-HFL* includes two modules to boost the FL performance: gateway-level device selection and cloud-level device-gateway association. To evaluate the contribution of each component separately, we compare the convergence time on all datasets using *(i)* pure random selections, *(ii)* only the device selection, *(iii)* only the device-gateway association, and *(iv)* the complete *Async-HFL* as shown in Fig. 7 (right). The target accuracy and bandwidth are set to the same as in Table 5. Each module contributes various extents on different datasets. On MNIST, CIFAR-10 and HAR, the device-gateway association improves convergence more significantly by balancing the network topology, achieving 1.71x, 1.23x and 1.17x speedup by itself. On FashionMNIST, using one module does not change much, but applying both modules leads to a 1.23x speedup. On Shakespeare, the speedup is mainly supported by our coreset device selection with a 1.12x speedup by itself. On HPWREN, applying a single device selection or device-gateway association module leads to a 1.45x or 1.17x speedup, while using both modules contributes to a 1.64x total speedup. Hence both device selection and association components are necessary in *Async-HFL* to deal with various data and system heterogeneities.

6.5 Results on Physical Deployment

Convergence Results. We validate *Async-HFL* on the physical deployment running MNIST, FashionMNIST, HAR and HPWREN datasets. The accuracy or test loss under wall-clock time are summarized in Fig. 8. We run *Async-HFL* and Async-HL for 30 cloud rounds, Sync-Random for 3 cloud rounds, unless the system is suspended due to stragglers. Note, that Async-HL is the state-of-the-art asynchronous baseline and presents the second best result in simulations. *Async-HFL* ends up with 70%, 56% and 75% accuracies on MNIST, FashionMNIST and HAR, while Async-HL only reaches 62%, 36% and 73% at similar time (after 7.6K, 3.4K and 2K seconds).

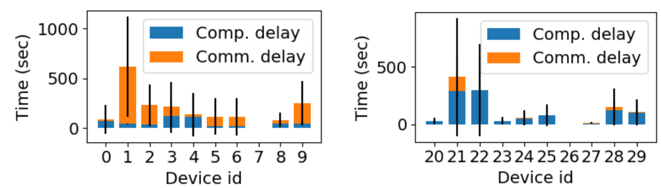

Figure 9: Round latency results on the physical deployment running MNIST. Left: Time breakups on RPis. Right: Time breakups on CPUs.

For the synchronous baseline, we are only able to obtain very limited traces due to straggler effects. After setting a timeout limit for synchronous aggregations, we acquire the curves in Fig. 8 with very slow convergence. The HPWREN dataset is very computational challenging for all methods and a lot of devices drop off due to no communication for a long time. *Async-HFL* strives for convergence within 2K seconds, while Async-HL and Sync-Random reach similar test loss after 3K and 7.7K seconds. While a small-scale physical deployment can be largely affected by uncertainties, we are able to observe consistently better convergence using *Async-HFL* over the baselines on all four datasets. The results demonstrate the robustness of *Async-HFL* under delay heterogeneities and stragglers. This is because the *Async-HFL* performance is dynamically guided by its two modules: (i) the gateway-level device selection module, which timely adjusts device participation, and (ii) the cloud-level device-gateway association, which considers device dropouts via taking $\mathbf{J}_t$ as input.

Round Latency. Fig. 9 displays the round latency measurements of our practical setup, which demonstrates how challenging our physical deployment is. To remind the reader, round latency is the time to complete one gateway round of downloading the model to device, training the model on device, then returning the updated model back to the gateway. The measurement supports our argument that IoT networks present heterogeneous system and network characteristics. In more details, Fig. 9 left and right show the round latency breakup on ten representatives of RPis and CPU clusters respectively. The missing columns indicate failed devices. Our physical deployment setup covers two typical scenarios with very different breakups. For RPis, the major heterogeneity comes from the network side, as we setup the RPis at various places with different distances to the router. For CPU clusters, the computational delay rather than communication delay presents more variations due to varying number of requested CPU cores. For FL, both heterogeneities cause the largely varied and unstable round latency distribution that *Async-HFL* targets to address.

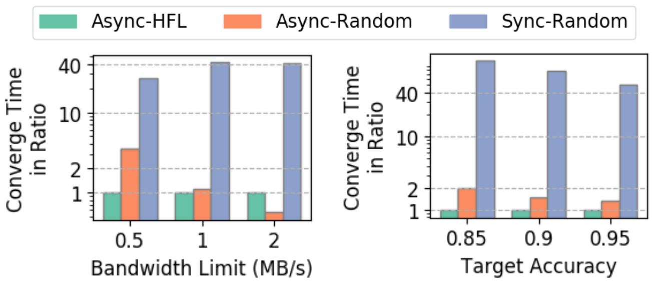

Figure 10: Convergence time in ratio compared to *Async-HFL* running MNIST under various bandwidth limits at gateways (left) and target accuracy (right) at cloud.

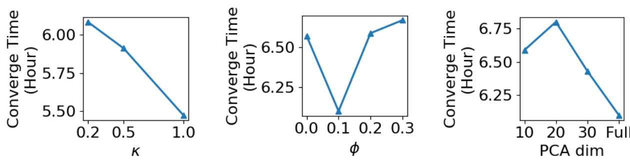

Figure 11: Sensitivity experiments of *Async-HFL* on HAR dataset.

6.6 Sensitivity Analysis.

Bandwidth Limitation. Fig. 10 (left) shows the convergence time to reach 95% on MNIST when altering bandwidth limits at all gateways. The speedup over Async-Random is more significant under more restricted bandwidth (3.47x under 0.5MB/s vs. 0.56x under 2MB/s), as the benefit of intelligently selecting subset of devices reveals more with limited resource. Compared to the Sync-Random baseline, the speedup gap closes under 0.5MB/s bandwidth. When only a limited number of devices can be selected, FedAvg (Sync-Random) gets a stable convergence via averaging the models from multiple devices.

Target Accuracy. Fig. 10 (right) shows the convergence time to reach various target accuracies on MNIST with the same set of other settings. The speedup over Async-Random is 1.99x, 1.50x and 1.34x for reaching 85%, 90% and 95%. Under the same settings, the speedup over Sync-Random is 106x, 78x and 51x. The results demonstrate *Async-HFL*'s fast convergence in the early stage, which can be attributed to prioritizing diverse and fast devices in *Async-HFL*'s management design.

Hyperparameters κ and ϕ. We experiment the impact of κ (Equation (10a)) and ϕ (Equation (11a)) on the final convergence time, as both parameters determine the balance between data heterogeneity (learning utility) and system heterogeneity (round latency). We use the HAR dataset with configurations in Table 5. Fig. 11 (left) shows the wall-clock time to reach the same accuracy using $\kappa = 0.2, 0.5, 1.0$. A larger κ increases the weight of delays during gateway-level device selection thus results in faster convergence. Fig. 11 (middle) depicts the wall-clock convergence time using $\phi = 0, 0.1, 0.2, 0.3$. $\phi = 0$ means only data heterogeneity is considered, while a larger ϕ increases the contribution of bandwidth limitation during cloud-level device-gateway association. A proper ϕ (in this case, $\phi = 0.1$) leads to the best convergence performance by jointly considering data and system aspects.

6.7 Overhead Analysis

As shown in Fig. 6, the major communication overhead of *Async-HFL* comes from exchanging the gradients. Using PCA to compress the gradients, the effect of various PCA dimensions on convergence time while processing the HAR dataset is shown in Fig. 11 (right). A PCA compression of 30 dimensions introduces a communication overhead of <0.5%, while the increase on convergence time (compared to using the full gradients) is less than 6%. Hence, the PCA compression strategy effectively reduces communication overhead while preserving convergence speed. On the computational side, the device selection algorithm consumes 1.6, 1.4, 4.3, 0.1 seconds per selection on the MNIST, FashionMNIST, CIFAR-10 and Shakespeare datasets. The time consumption of cloud-level association is 1.1, 0.9, 8.7 and 0.3 seconds per selection on the server for the above datasets. These additional computational times are negligible on the physical deployment with an average 120.26 seconds of round latency.

7 CONCLUSION

In this paper, we propose *Async-HFL*, the first end-to-end asynchronous hierarchical Federated Learning framework which jointly considers data, system heterogeneities, stragglers and scalability in IoT networks. *Async-HFL* performs asynchronous aggregations on both gateways and cloud, thus achieving faster convergence with heterogeneous delays and being robust to stragglers. With the *learning utility* metric to quantify gradient diversity, we design the device selection and device-gateway association modules to balance *learning utility*, round latencies and unexpected stragglers, collaboratively optimizing practical model convergence. We conduct comprehensive simulations based on ns-3 and NYCMesh to evaluate the *Async-HFL* under various network characteristics. Our results show a 1.08-1.31x speedup in terms of *wall-clock* convergence time and 2.6-21.6% communication savings compared to state-of-the-art asynchronous FL algorithms. Our physical deployment proves robust convergence under unexpected stragglers.

ACKNOWLEDGMENTS

This work was initiated during Xiaofan Yu's internship at Arm Research in summer of 2021. The research was supported in part by National Science Foundation under Grants #2112665 (TILOS AI Research Institute), #2003279, #1911095, #1826967, #2100237, #2112167.

REFERENCES

[1] 2022. High Performance Wireless Research & Education Network (HPWREN). http://hpwren.ucsd.edu/ [Online].
[2] 2022. New York City (NYC) Mesh. https://www.nycmesh.net/map/ [Online].
[3] 2022. ns-3: a discrete-event network simulator for internet systems. https://www.nsnam.org/ [Online].
[4] Mehdi Salehi Heydar Abad, Emre Ozfatura, Deniz Gunduz, and Ozgur Ercetin. 2020. Hierarchical federated learning across heterogeneous cellular networks. In *ICASSP*. IEEE, 8866–8870.
[5] Alaa Awad Abdellatif, Naram Mhaisen, Amr Mohamed, Aiman Erbad, Mohsen Guizani, Zaher Dawy, and Wassim Nasreddine. 2022. Communication-efficient hierarchical federated learning for IoT heterogeneous systems with imbalanced data. *Future Generation Computer Systems* 128 (2022), 406–419.
[6] Irfan Ahmad and Karunakar Pothuganti. 2020. Design & implementation of real time autonomous car by using image processing & IoT. In *ICSSIT*. IEEE, 107–113.
[7] Davide Anguita, Alessandro Ghio, Luca Oneto, Xavier Parra Perez, and Jorge Luis Reyes Ortiz. 2013. A public domain dataset for human activity recognition using smartphones. In *ESANN*. 437–442.

[8] Ravikumar Balakrishnan, Tian Li, Tianyi Zhou, Nageen Himayat, Virginia Smith, and Jeff Bilmes. 2021. Diverse Client Selection for Federated Learning via Submodular Maximization. In *ICLR*.
[9] Sándor Beniczky, Philippa Karoly, Ewan Nurse, Philippe Ryvlin, and Mark Cook. 2021. Machine learning and wearable devices of the future. *Epilepsia* 62 (2021), S116–S124.
[10] Sebastian Caldas, Sai Meher Karthik Duddu, Peter Wu, Tian Li, Jakub Konečnỳ, H Brendan McMahan, Virginia Smith, and Ameet Talwalkar. 2018. Leaf: A benchmark for federated settings. *arXiv preprint arXiv:1812.01097* (2018).
[11] Zheng Chai, Ahsan Ali, Syed Zawad, Stacey Truex, Ali Anwar, Nathalie Baracaldo, Yi Zhou, Heiko Ludwig, Feng Yan, and Yue Cheng. 2020. Tifl: A tier-based federated learning system. In *HPDC*. 125–136.
[12] Zheng Chai, Yujing Chen, Liang Zhao, Yue Cheng, and Huzefa Rangwala. 2020. Fedat: A communication-efficient federated learning method with asynchronous tiers under non-iid data. *arXiv preprin arxiv:2010.05958* (2020).
[13] et al. Chaoyang He. 2020. Fedml: A research library and benchmark for federated machine learning. *arXiv preprint arXiv:2007.13518* (2020).
[14] Mingzhe Chen, H Vincent Poor, Walid Saad, and Shuguang Cui. 2020. Convergence time minimization of federated learning over wireless networks. In *ICC*. IEEE, 1–6.
[15] Shuai Chen, Xiumin Wang, Pan Zhou, Weiwei Wu, Weiwei Lin, and Zhenyu Wang. 2022. Heterogeneous Semi-Asynchronous Federated Learning in Internet of Things: A Multi-Armed Bandit Approach. *IEEE Transactions on Emerging Topics in Computational Intelligence* 6, 5 (2022), 1113–1124.
[16] Zheyi Chen, Weixian Liao, Kun Hua, Chao Lu, and Wei Yu. 2021. Towards asynchronous federated learning for heterogeneous edge-powered internet of things. *Digital Communications and Networks* 7, 3 (2021), 317–326.
[17] Li Deng. 2012. The mnist database of handwritten digit images for machine learning research. *IEEE Signal Processing Magazine* (2012).
[18] Yongheng Deng, Feng Lyu, Ju Ren, Yongmin Zhang, Yuezhi Zhou, Yaoxue Zhang, and Yuanyuan Yang. 2021. SHARE: Shaping Data Distribution at Edge for Communication-Efficient Hierarchical Federated Learning. In *ICDCS*. IEEE, 24–34.
[19] Emily Ekaireb, Xiaofan Yu, Kazim Ergun, Quanling Zhao, Kai Lee, Muhammad Huzaifa, and Tajana Rosing. 2022. ns3-fl: Simulating Federated Learning with ns-3. In *WNS-3*. 97–104.
[20] Chenyuan Feng, Howard H Yang, Deshun Hu, Zhiwei Zhao, Tony QS Quek, and Geyong Min. 2022. Mobility-aware cluster federated learning in hierarchical wireless networks. *IEEE Transactions on Wireless Communications* 21, 10 (2022), 8441–8458.
[21] Arnaud Fréville. 2004. The multidimensional 0–1 knapsack problem: An overview. *European Journal of Operational Research* 155 (2004).
[22] Gurobi Optimization, LLC. 2022. Gurobi Optimizer Reference Manual. https://www.gurobi.com
[23] Jiangshan Hao, Yanchao Zhao, and Jiale Zhang. 2020. Time efficient federated learning with semi-asynchronous communication. In *ICPADS*. IEEE.
[24] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep residual learning for image recognition. In *CVPR*. 770–778.
[25] Chung-Hsuan Hu, Zheng Chen, and Erik G Larsson. 2021. Device scheduling and update aggregation policies for asynchronous federated learning. In *SPAWC*. IEEE, 281–285.
[26] Dzmitry Huba, John Nguyen, Kshitiz Malik, Ruiyu Zhu, Mike Rabbat, Ashkan Yousefpour, Carole-Jean Wu, Hongyuan Zhan, Pavel Ustinov, Harish Srinivas, et al. 2022. Papaya: Practical, private, and scalable federated learning. *Proceedings of Machine Learning and Systems* 4 (2022), 814–832.
[27] Ahmed Imteaj and M Hadi Amini. 2020. Fedar: Activity and resource-aware federated learning model for distributed mobile robots. In *ICMLA*. IEEE.
[28] Nabaa Ali Jasim, Haider TH, and Salim AL Rikabi. 2021. Design and implementation of smart city applications based on the internet of things. *iJIM* 15, 13 (2021).
[29] Sai Praneeth Karimireddy, Satyen Kale, Mehryar Mohri, Sashank Reddi, Sebastian Stich, and Ananda Theertha Suresh. 2020. Scaffold: Stochastic controlled averaging for federated learning. In *ICML*. PMLR, 5132–5143.
[30] Latif U Khan, Shashi Raj Pandey, Nguyen H Tran, Walid Saad, Zhu Han, Minh NH Nguyen, and Choong Seon Hong. 2020. Federated learning for edge networks: Resource optimization and incentive mechanism. *IEEE Communications Magazine* 58, 10 (2020), 88–93.
[31] Alex Krizhevsky, Geoffrey Hinton, et al. 2009. Learning multiple layers of features from tiny images. (2009).
[32] Fan Lai, Xiangfeng Zhu, Harsha V Madhyastha, and Mosharaf Chowdhury. 2021. Oort: Efficient federated learning via guided participant selection. In *OSDI*. 19–35.
[33] Hyun-Suk Lee and Jang-Won Lee. 2021. Adaptive transmission scheduling in wireless networks for asynchronous federated learning. *IEEE Journal on Selected Areas in Communications* 39, 12 (2021), 3673–3687.
[34] Ang Li, Jingwei Sun, Pengcheng Li, Yu Pu, Hai Li, and Yiran Chen. 2021. Hermes: an efficient federated learning framework for heterogeneous mobile clients. In *MobiCom*. 420–437.
[35] Ang Li, Jingwei Sun, Xiao Zeng, Mi Zhang, Hai Li, and Yiran Chen. 2021. Fedmask: Joint computation and communication-efficient personalized federated learning via heterogeneous masking. In *SenSys*. 42–55.
[36] Chenning Li, Xiao Zeng, Mi Zhang, and Zhichao Cao. 2022. PyramidFL: A fine-grained client selection framework for efficient federated learning. In *MobiCom*. 158–171.
[37] Tian Li, Anit Kumar Sahu, Manzil Zaheer, Maziar Sanjabi, Ameet Talwalkar, and Virginia Smith. 2020. Federated optimization in heterogeneous networks. *Proceedings of Machine learning and systems* 2 (2020), 429–450.
[38] Tian Li, Maziar Sanjabi, Ahmad Beirami, and Virginia Smith. 2019. Fair resource allocation in federated learning. *arXiv preprint arXiv:1905.10497* (2019).
[39] Lumin Liu, Jun Zhang, SH Song, and Khaled B Letaief. 2020. Client-edge-cloud hierarchical federated learning. In *ICC*. IEEE, 1–6.
[40] Siqi Luo, Xu Chen, Qiong Wu, Zhi Zhou, and Shuai Yu. 2020. Hfel: Joint edge association and resource allocation for cost-efficient hierarchical federated edge learning. *IEEE Transactions on Wireless Communications* 19, 10 (2020), 6535–6548.
[41] Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, and Blaise Aguera y Arcas. 2017. Communication-efficient learning of deep networks from decentralized data. In *AISTATS*. PMLR, 1273–1282.
[42] Aritra Mitra, Rayana Jaafar, George J Pappas, and Hamed Hassani. 2021. Linear convergence in federated learning: Tackling client heterogeneity and sparse gradients. *Advances in Neural Information Processing Systems* 34 (2021), 14606–14619.
[43] John Nguyen, Kshitiz Malik, Hongyuan Zhan, Ashkan Yousefpour, Mike Rabbat, Mani Malek, and Dzmitry Huba. 2022. Federated learning with buffered asynchronous aggregation. In *AISTATS*. PMLR, 3581–3607.
[44] Monica Ribero and Haris Vikalo. 2020. Communication-efficient federated learning via optimal client sampling. *arXiv preprint arXiv:2007.15197* (2020).
[45] Kaixin Sui, Mengyu Zhou, Dapeng Liu, Minghua Ma, Dan Pei, Youjian Zhao, Zimu Li, and Thomas Moscibroda. 2016. Characterizing and improving wifi latency in large-scale operational networks. In *MobiSys*. 347–360.
[46] Alysa Ziying Tan, Han Yu, Lizhen Cui, and Qiang Yang. 2022. Towards personalized federated learning. *IEEE Trans. Neural Netw. Learn. Syst.* (2022).
[47] Hao Wang, Zakhary Kaplan, Di Niu, and Baochun Li. 2020. Optimizing federated learning on non-iid data with reinforcement learning. In *INFOCOM*. IEEE, 1698–1707.
[48] Jianyu Wang, Qinghua Liu, Hao Liang, Gauri Joshi, and H Vincent Poor. 2020. Tackling the objective inconsistency problem in heterogeneous federated optimization. *Adv Neural Inf Process Syst* 33 (2020), 7611–7623.
[49] Zhiyuan Wang, Hongli Xu, Jianchun Liu, He Huang, Chunming Qiao, and Yangming Zhao. 2021. Resource-efficient federated learning with hierarchical aggregation in edge computing. In *INFOCOM*. IEEE, 1–10.
[50] Zhongyu Wang, Zhaoyang Zhang, Yuqing Tian, Qianqian Yang, Hangguan Shan, Wei Wang, and Tony QS Quek. 2022. Asynchronous federated learning over wireless communication networks. *IEEE Transactions on Wireless Communications* 21, 9 (2022), 6961–6978.
[51] Wentai Wu, Ligang He, Weiwei Lin, Rui Mao, Carsten Maple, and Stephen Jarvis. 2020. Safa: a semi-asynchronous protocol for fast federated learning with low overhead. *IEEE Trans. Comput.* 70, 5 (2020), 655–668.
[52] Han Xiao, Kashif Rasul, and Roland Vollgraf. 2017. Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms.
[53] Cong Xie, Sanmi Koyejo, and Indranil Gupta. 2019. Asynchronous federated optimization. *arXiv preprint arXiv:1903.03934* (2019).
[54] Bo Xu, Wenchao Xia, Jun Zhang, Tony QS Quek, and Hongbo Zhu. 2021. Online client scheduling for fast federated learning. *IEEE Wirel. Commun. Lett.* 10, 7 (2021), 1434–1438.
[55] Bo Xu, Wenchao Xia, Jun Zhang, Xinghua Sun, and Hongbo Zhu. 2021. Dynamic client association for energy-aware hierarchical federated learning. In *WCNC*. IEEE, 1–6.
[56] Jaehong Yoon, Divyam Madaan, Eunho Yang, and Sung Ju Hwang. 2021. Online coreset selection for rehearsal-based continual learning. *arXiv preprint arXiv:2106.01085* (2021).
[57] Linlin You, Sheng Liu, Yi Chang, and Chau Yuen. 2022. A triple-step asynchronous federated learning mechanism for client activation, interaction optimization, and aggregation enhancement. *IEEE Internet of Things Journal* (2022).
[58] Yu Zhang, Morning Duan, Duo Liu, Li Li, Ao Ren, Xianzhang Chen, Yujuan Tan, and Chengliang Wang. 2021. CSAFL: A clustered semi-asynchronous federated learning framework. In *IJCNN*. IEEE, 1–10.
[59] Zhengyi Zhong, Weidong Bao, Ji Wang, Xiaomin Zhu, and Xiongtao Zhang. 2022. FLEE: A hierarchical federated learning framework for distributed deep neural network over cloud, edge and end device. *ACM TIST* (2022).
[60] Chendi Zhou, Hao Tian, Hong Zhang, Jin Zhang, Mianxiong Dong, and Juncheng Jia. 2021. TEA-fed: time-efficient asynchronous federated learning for edge computing. In *ACM CF*. 30–37.
[61] Hongbin Zhu, Yong Zhou, Hua Qian, Yuanming Shi, Xu Chen, and Yang Yang. 2022. Online client selection for asynchronous federated learning with fairness consideration. *IEEE Transactions on Wireless Communications* (2022).

A PROOF OF THEOREM 5.1

In the main paper, we establish the convergence of Async-HFL under the three assumptions. Our analysis extends the proof in [53] from a single level of asynchrony to two levels. The proof involves bounding the deviation of cloud loss function L_N between steps in terms of its gradient norm ∇L_N. We proceed from the lowest level, the device, establishing progress across device steps e. Then, using the progress after E device steps, we can then bound progress at the next level, which is the gateway. Note that moving to the next level, the cloud is much simpler as the update steps and assumptions for cloud and gateway level are similar. We now describe the complete proof starting from the progress on any device in the next subsection.

A.1 Device steps

On each device, we only perform SGD steps, therefore, we can utilize Assumptions 1 and 3 to bound the progress after each gradient update.

Lemma A.1 (Single-step Device). *Under Assumptions 1 and 3, the progress in terms of L_N after one device step is given by*

$$\mathbb{E}[L_N(\boldsymbol{\omega}^i_{\tau,\zeta,e})] \leq \mathbb{E}[L_N(\boldsymbol{\omega}^i_{\tau,\zeta,e-1})] - \gamma c\mathbb{E}[\|\nabla L_N(\boldsymbol{\omega}^i_{\tau,\zeta,e-1})\|^2] + \gamma^2 O(\rho E^2 V_2) \quad \forall \tau, \zeta \geq 0, e \geq 1, i \in \mathcal{N} \tag{12}$$

Proof. Since the loss is L-smooth, $L_N(\boldsymbol{\omega})$ and $g_{N,\boldsymbol{\omega}'}(\boldsymbol{\omega})$ are L-smooth and $(2L+2\rho)$-smooth. To simplify the proof, we drop the subscript τ and superscript i from the following steps. Since $g_{N,\boldsymbol{\omega}'}(\boldsymbol{\omega}) \geq L_N(\boldsymbol{\omega}) \forall \boldsymbol{\omega}, \boldsymbol{\omega'}$, we have

$$\begin{aligned}
\mathbb{E}[L_N(\boldsymbol{\omega}_{\zeta,e})] &\leq \mathbb{E}[g_{N,\boldsymbol{\omega}_\zeta}(\boldsymbol{\omega}_{\zeta,e})] \\
&\leq \mathbb{E}[g_{N,\boldsymbol{\omega}_\zeta}(\boldsymbol{\omega}_{\zeta,e-1})] + \gamma^2(L+\rho)\mathbb{E}[\|\nabla g_{\boldsymbol{\omega}_\zeta}(\boldsymbol{\omega}_{\zeta,e-1})\|^2] \\
&\quad - \gamma\mathbb{E}[\left\langle \nabla g_{N,\boldsymbol{\omega}_\zeta}(\boldsymbol{\omega}_{\zeta,e-1}), \nabla g_{\boldsymbol{\omega}_\zeta}(\boldsymbol{\omega}_{\zeta,e-1})\right\rangle] \\
&\leq \mathbb{E}[L_N(\boldsymbol{\omega}_{\zeta,e-1})] + \frac{\rho}{2}\mathbb{E}[\|\boldsymbol{\omega}_\zeta - \boldsymbol{\omega}_{\zeta,e-1}\|^2] \\
&\quad + \gamma^2(L+\rho)\mathbb{E}[\|\nabla g_{\boldsymbol{\omega}_\zeta}(\boldsymbol{\omega}_{\zeta,e-1})\|^2] \\
&\quad - \gamma\mathbb{E}[\left\langle \nabla g_{N,\boldsymbol{\omega}_\zeta}(\boldsymbol{\omega}_{\zeta,e-1}), \nabla g_{\boldsymbol{\omega}_\zeta}(\boldsymbol{\omega}_{\zeta,e-1})\right\rangle] \\
&\leq \mathbb{E}[L_N(\boldsymbol{\omega}_{\zeta,e-1})] + \frac{\rho\gamma^2E^2V_2}{2} + \gamma^2(L+\rho)V_2 \\
&\quad - \gamma\mathbb{E}[\left\langle \nabla g_{N,\boldsymbol{\omega}_\zeta}(\boldsymbol{\omega}_{\zeta,e-1}), \nabla g_{\boldsymbol{\omega}_\zeta}(\boldsymbol{\omega}_{\zeta,e-1})\right\rangle]
\end{aligned}$$

For the second step, we use smoothness of $g_{N,\boldsymbol{\omega}_\zeta}$. For the third step, we expand $g_{N,\boldsymbol{\omega}_\zeta}$ and for the fourth step, we use Assumption 1 to bound $\mathbb{E}[\|\nabla g_{\boldsymbol{\omega}_\zeta}(\boldsymbol{\omega}_{\zeta,e-1})\|^2]$ and $\mathbb{E}[\|\boldsymbol{\omega}_\zeta - \boldsymbol{\omega}_{\zeta,e-1}\|^2]$.

Note that, $\mathbb{E}[\|\boldsymbol{\omega}_\zeta - \boldsymbol{\omega}_{\zeta,e-1}\|^2] = \mathbb{E}[\|\sum_{r=1}^{e-1}\gamma\nabla g_{\boldsymbol{\omega}_\zeta}(\boldsymbol{\omega}_{\zeta,r})\|^2] \leq \gamma^2(e-1)\sum_{r=1}^{e-1}\mathbb{E}[\|\nabla g_{\boldsymbol{\omega}_\zeta}(\boldsymbol{\omega}_{\zeta,r})\|^2] \leq \gamma^2(e-1)^2V_2 \leq \gamma^2E^2V_2$

We now utilize Assumption 3 to bound the remaining terms and introduce ∇L_N. Consider the following inequality for some $c > 0$,

$$\begin{aligned}
&\mathbb{E}[\left\langle \nabla g_{N,\boldsymbol{\omega}_\zeta}(\boldsymbol{\omega}_{\zeta,e-1}), \nabla g_{\boldsymbol{\omega}_\zeta}(\boldsymbol{\omega}_{\zeta,e-1})\right\rangle] - c\mathbb{E}[\|\nabla L_N(\boldsymbol{\omega}_{\zeta,e-1})\|^2] \\
&= \mathbb{E}[\langle \nabla L_N(\boldsymbol{\omega}_{\zeta,e-1}) + \rho(\boldsymbol{\omega}_{\zeta,e-1} - \boldsymbol{\omega}_\zeta), \\
&\quad \nabla L(\boldsymbol{\omega}_{\zeta,e-1}) + \rho(\boldsymbol{\omega}_{\zeta,e-1} - \boldsymbol{\omega}_\zeta)\rangle] - c\mathbb{E}[\|\nabla L_N(\boldsymbol{\omega}_{\zeta,e-1})\|^2] \\
&= \mathbb{E}[\left\langle \nabla L_N(\boldsymbol{\omega}_{\zeta,e-1}), \nabla L(\boldsymbol{\omega}_{\zeta,e-1})\right\rangle] + \rho^2\mathbb{E}[\|\boldsymbol{\omega}_{\zeta,e-1} - \boldsymbol{\omega}_\zeta\|^2] \\
&\quad + \rho\mathbb{E}[\left\langle \nabla L_N(\boldsymbol{\omega}_{\zeta,e-1}) + \nabla L(\boldsymbol{\omega}_{\zeta,e-1}), \boldsymbol{\omega}_{\zeta,e-1} - \boldsymbol{\omega}_\zeta\right\rangle] \\
&\quad - c\mathbb{E}[\|\nabla L_N(\boldsymbol{\omega}_{\zeta,e-1})\|^2] \\
&\geq -\frac{1}{2}\mathbb{E}[\|\nabla L_N(\boldsymbol{\omega}_{\zeta,e-1})\|^2] - \frac{1}{2}\mathbb{E}[\|\nabla L(\boldsymbol{\omega}_{\zeta,e-1})\|^2] \\
&\quad + \rho^2\mathbb{E}[\|\boldsymbol{\omega}_{\zeta,e-1} - \boldsymbol{\omega}_\zeta\|^2] - \frac{\rho}{2}\mathbb{E}[\|\boldsymbol{\omega}_{\zeta,e-1} - \boldsymbol{\omega}_\zeta\|^2] \\
&\quad - \frac{\rho}{2}\mathbb{E}[\|\nabla L_N(\boldsymbol{\omega}_{\zeta,e-1}) + \nabla L(\boldsymbol{\omega}_{\zeta,e-1})\|^2] - c\mathbb{E}[\|\nabla L_N(\boldsymbol{\omega}_{\zeta,e-1})\|^2] \\
&\geq -\frac{1}{2}\mathbb{E}[\|\nabla L_N(\boldsymbol{\omega}_{\zeta,e-1})\|^2] - \frac{1}{2}\mathbb{E}[\|\nabla L(\boldsymbol{\omega}_{\zeta,e-1})\|^2] \\
&\quad + \rho^2\mathbb{E}[\|\boldsymbol{\omega}_{\zeta,e-1} - \boldsymbol{\omega}_\zeta\|^2] - \frac{\rho}{2}\mathbb{E}[\|\boldsymbol{\omega}_{\zeta,e-1} - \boldsymbol{\omega}_\zeta\|^2] \\
&\quad - \rho\mathbb{E}[\|\nabla L_N(\boldsymbol{\omega}_{\zeta,e-1})\|^2] - \rho\mathbb{E}[\|\nabla L(\boldsymbol{\omega}_{\zeta,e-1})\|^2] \\
&\quad - c\mathbb{E}[\|\nabla L_N(\boldsymbol{\omega}_{\zeta,e-1})\|^2] \\
&\geq -V_1(\rho + \frac{1}{2} + c) - V_2(\rho + \frac{1}{2}) + (\rho^2 - \frac{\rho}{2})\mathbb{E}[\|\boldsymbol{\omega}_{\zeta,e-1} - \boldsymbol{\omega}_\zeta\|^2]
\end{aligned}$$

In the first two steps, we expand the definitions of $g_{N,\boldsymbol{\omega}}$ and $g_{\boldsymbol{\omega}}$. In the second step and third step, we use $\langle a,b\rangle \geq -\frac{\|a\|^2}{2} - \frac{\|b\|^2}{2}$. In the fourth step, we use Assumption 1 to obtain the bounds. Finally, by Assumption 3, we have a $c > 0$ such that the last inequality is always ≥ 0. Therefore,

$$\mathbb{E}[\left\langle \nabla g_{N,\boldsymbol{\omega}_\zeta}(\boldsymbol{\omega}_{\zeta,e-1}), \nabla g_{\boldsymbol{\omega}_\zeta}(\boldsymbol{\omega}_{\zeta,e-1})\right\rangle] \leq c\mathbb{E}[\|\nabla L_N(\boldsymbol{\omega}_{\zeta,e-1})\|^2]$$

Using this inequality, we have proved the lemma.

$$\mathbb{E}[L_N(\boldsymbol{\omega}^i_{\tau,\zeta,e})] \leq \mathbb{E}[L_N(\boldsymbol{\omega}^i_{\tau,\zeta,e-1})] - \gamma c\mathbb{E}[\|\nabla L_N(\boldsymbol{\omega}^i_{\tau,\zeta,e-1})\|^2] + \gamma^2\rho O(E^2V_2)$$

□

Using the progress made in a single step, we can bound the update for all local steps at a single device, by summing the above lemma over E steps.

Lemma A.2 (All-steps Device). *Under Assumptions 2 and 3, after E steps at the device level, the progress made in the objective function is given by*

$$\mathbb{E}[L_N(\boldsymbol{\omega}^i_{\tau,\zeta,E})] \leq \mathbb{E}[L_N(\boldsymbol{\omega}^i_{\tau,\zeta})] - \gamma c\sum_{e=1}^{E}\mathbb{E}[\|\nabla L_N(\boldsymbol{\omega}^i_{\tau,\zeta,e-1})\|^2] + \gamma^2 O(\rho E^3 V_2) \quad \forall \tau, \zeta \geq 0, i \in \mathcal{N} \tag{13}$$

A.2 Gateway steps

When we move a level up the hierarchy, the update is simply a convex combination of the current model at the gateway and the trained model received from a device. Note that this is the first place in the proof where we need to handle asynchrony. Using Assumption 2, we first explicitly derive the delay recursion in the following intermediate Lemma.

Lemma A.3 (Gateway delay recursion). *Let $\omega^j_{\tau,\zeta,E}$ be the update received at gateway j at step z, then for $\mathcal{R}^j_{\tau,z} \triangleq \mathbb{E}[\|\omega^j_{\tau,\zeta} - \omega_{\tau,z-1}\|^2]$, under Assumptions 1 and 2, we can bound the norm of update at gateway at gateway step z as,*

$$\mathcal{R}^j_{\tau,z} \leq K_g\beta^2(\gamma^2 O(K_g E^2 V_2) + 4\sum_{r=1}^{K_g-1}\mathcal{R}^j_{\tau,z-r}),\ \forall\tau\geq 0, z\geq 1, j\in\mathcal{G} \tag{14}$$

Proof. We drop the superscript j and subscript τ and only consider the update at the given machine. From Assumption 2, we can see that $z-1-\zeta \leq K_g$, therefore, there have been at most K_g gateway updates between ζ and $z-1$. Utilizing this fact, we can rewrite $\mathcal{R}_z$ as

$$\begin{aligned}\mathcal{R}_z =&\mathbb{E}[\|\sum_{r=0}^{z-\zeta-2}(\omega_{\zeta+r+1}-\omega_{\zeta+r})\|^2]\\ \leq&(z-\zeta-2)\sum_{r=0}^{z-\zeta-2}\mathbb{E}[\|\omega_{\zeta+r+1}-\omega_{\zeta+r}\|^2]\end{aligned}$$

We utilize triangle inequality to split the terms inside the norm. Let $\omega_{\zeta_{z-r},E}$ be the update received at step $z-r$ at the gateway then, we have,

$$\mathcal{R}_z \leq (z-\zeta-1)\beta^2\sum_{r=1}^{z-\zeta-1}\mathbb{E}[\|\omega_{\zeta_{z-r+1},E}-\omega_{z-r}\|^2] \tag{15}$$

Now, consider a single term in the summation given in Eq. (15).

$$\begin{aligned}\mathbb{E}[\|\omega_{\zeta_{z-r+1},E}-\omega_{z-r}\|^2] &\leq 2\mathbb{E}[\|\omega_{\zeta_{z-r+1},E}-\omega_{\zeta_{z-r+1}}\|^2]\\ &\quad+2\mathbb{E}[\|\omega_{z-r}-\omega_{\zeta_{z-r+1}}\|^2]\\ &\leq \gamma^2 O(E^2V_2)+2\mathcal{R}_{z-r+1}\end{aligned}$$

We use the fact that E steps on a device can be represented as sum of E gradient updates where norm of each gradient is bounded by $O(V_2)$ and use a triangle inequality from Assumption 1. Plugging this into Eq. (15), we obtain,

$$\begin{aligned}\mathcal{R}_z \leq&(z-\zeta-2)\beta^2\sum_{r=1}^{z-\zeta-1}(\gamma^2O(E^2V_2)+2\mathcal{R}_{z-r})\\ \leq& K_g\beta^2(\gamma^2O(K_gE^2V_2)+4\sum_{r=1}^{K_g-1}\mathcal{R}_{z-r})\end{aligned}$$

We finally use the fact that delay is bounded by K_g from Assumption 2.

□

Handling the delay recursion, by means of the above Lemma, we utilize the update step on a single gateway and the progress made by a training model at a given device, from Lemma A.2, to obtain the single-step progress at the gateway level.

Lemma A.4 (Single-step Gateway). *Under Assumptions 1,2 and 3, for a single step at the gateway, we have*

$$\mathbb{E}[L_N(\omega^j_{\tau,z})-L_N(\omega^j_{\tau,z-1})] \leq -\gamma\beta c\sum_{e=0}^{E-1}\mathbb{E}[\|\nabla L_N(\omega^{i_{j,z}}_{\tau,\zeta,e})\|^2]+\Delta_z \quad \forall\tau\geq 0, z\geq 1, j\in\mathcal{G} \tag{16}$$

where the device $i_{j,z}$ was used for the z^{th} gateway update and

$$\Delta_z = \beta\gamma^2 O(\rho E^3 V_2)+\beta O(\sqrt{V_1})\sqrt{\mathcal{R}^j_{\tau,z}}+\beta O(L+\rho)\mathcal{R}^j_{\tau,z}$$

Proof. We will again omit τ and j to simplify the proof. Consider the function value after a single update at gateway at step z upon receiving the update $\omega_{\zeta,E}$ from the device.

$$\mathbb{E}[L_N(\omega_z)-L_N(\omega_{z-1})] = \mathbb{E}[g_N(\omega_z;\omega_{z-1})-L_N(\omega_{z-1})] \tag{17}$$

$$\leq \mathbb{E}[(1-\beta)g_N(\omega_{z-1};\omega_{z-1})+\beta g_N(\omega_{\zeta,E};\omega_{z-1})-L_N(\omega_{z-1})] \tag{18}$$

$$\leq \beta\mathbb{E}[L_N(\omega_{\zeta,E})-L_N(\omega_{z-1})]+\frac{\beta\rho}{2}\mathbb{E}[\|\omega_{\zeta,E}-\omega_{z-1}\|^2] \tag{19}$$

$$\begin{aligned}&\leq \beta\mathbb{E}[L_N(\omega_{\zeta,E})-L_N(\omega_\zeta)]+\beta\mathbb{E}[L_N(\omega_\zeta)-L_N(\omega_{z-1})]\\ &\quad+\frac{\beta\rho}{2}\mathbb{E}[\|\omega_{\zeta,E}-\omega_{z-1}\|^2]\end{aligned} \tag{20}$$

We first use the fact that $g_N(\omega_1;\omega_2)\geq L_N(\omega_1), \forall\omega_1,\omega_2\in\mathbb{R}^d$. Then, we utilize convexity of g_N followed by expanding g_N into its components. For the first term in Eq. (20), we can use Lemma A.2, as it is the difference of function values after E steps at the device.

For the second term in Eq. (20), we can use the triangle inequality and bound the gradient norm from Assumption 1, similar to the proof of Lemma A.3, to obtain,

$$\begin{aligned}\mathbb{E}[\|\omega_{\zeta,E}-\omega_{z-1}\|^2] &\leq 2\mathbb{E}[\|\omega_{\zeta,E}-\omega_\zeta\|^2]+2\mathcal{R}_z\\ &\leq 2\gamma^2O(E^2V_2)+2\mathcal{R}_z\end{aligned}$$

We handle the first term in (20) separately using L-smoothness of L_N.

$$\begin{aligned}\mathbb{E}[L_N(\omega_\zeta)-L_N(\omega_{z-1})] \leq&\mathbb{E}[\langle\nabla L_N(\omega_{z-1}),\omega_\zeta-\omega_\zeta\rangle]\\ &+\frac{L}{2}\mathbb{E}[\|\omega_\zeta-\omega_\zeta\|^2]\\ \leq&\mathbb{E}[\|\nabla L_N(\omega_{z-1})\|\|\omega_\zeta-\omega_\zeta\|]+\frac{L}{2}\mathcal{R}_z\\ \leq&O(\sqrt{V_1})\mathbb{E}[\|\omega_\zeta-\omega_\zeta\|]+\frac{L}{2}\mathcal{R}_z\\ \leq&O(\sqrt{V_1})\sqrt{\mathcal{R}_z}+\frac{L}{2}\mathcal{R}_z\end{aligned}$$

We use Cauchy-Schwartz inequality to bound $\langle a,b\rangle \leq \|a\|\|b\|, \forall a,b\in\mathbb{R}^d$. Further, we utilize Assumption 1 to bound $\|\nabla g_N(\cdot)\|$ and then use Jensen's inequality to obtain $\mathbb{E}[\|a\|] \leq \sqrt{\mathbb{R}[\|a\|^2]}$.

Substituting these values in Eq. (20), we complete the proof. □

Using the above Lemmas A.4 and A.3, we can bound the progress of gateway after Z steps. For this purpose, we need to bound the sum of the recursion terms in Lemma A.3. The following Lemma provides this bound.

Lemma A.5 (Sum of gateway recursion). *Under Assumptions 1 and 2, and when $\beta \leq O(K_g^{-3/2})$, we have,*

$$\sum_{z=1}^{Z} \mathcal{R}_z \leq \gamma^2\beta^2 O(K_g^2 E^2 Z V_2) \tag{21}$$

$$\sum_{z=1}^{Z} \sqrt{\mathcal{R}_z} \leq \gamma\beta O(K_g E Z \sqrt{V_2}) \tag{22}$$

Proof. We first start with Lemma A.3 and sum over $z = 1$ to Z

$$\sum_{z=1}^{Z} \mathcal{R}_z \leq \gamma^2\beta^2 O(K_g^2 E^2 Z V_2) + \beta^2 K_g^2 \sum_{z=1}^{Z} \mathcal{R}_z \tag{23}$$

$$\sum_{z=1}^{Z} \mathcal{R}_z \leq (1-\beta^2 K_g^2)^{-1}\gamma^2\beta^2 O(K_g^2 E^2 Z V_2) \tag{24}$$

Here, summing over the recursion, we need to control the coefficient of the sum on RHS. If this coefficient is < 1, then we are done, which is exactly the case when $\beta = O(K_g^{-3/2})$ as $\beta^2 K_g^2 = O(K_g^{-1})$ which can be made to be a constant less than 1.

Now, for the sum over square roots, we first use Jensen's inequality to obtain, $\sqrt{\sum_{i=1}^n a_i} \leq \sum_{i=1}^n \sqrt{a_i}, \forall a_i \geq 0$ on Lemma A.3.

$$\sqrt{\mathcal{R}_z} \leq \gamma\beta O(K_g E \sqrt{V_2}) + \beta\sqrt{K_g} \sum_{r=1}^{K_g-1} \sqrt{\mathcal{R}_{z-r}} \tag{25}$$

$$\sum_{z=1}^{Z} \sqrt{\mathcal{R}_z} \leq \gamma\beta O(K_g E \sqrt{V_2}) + \beta K_g^{3/2} \sum_{z=1}^{Z} \sqrt{\mathcal{R}_z} \tag{26}$$

$$\leq (1-\beta K_g^{3/2})^{-1}\gamma\beta O(K_g E \sqrt{V_2}) \tag{27}$$

Now, we again control the coefficient of the sum on RHS and for $\beta = O(K_g^{-3/2})$, this coefficient is a constant.

□

Lemma A.6 (All-steps Gateway). *From Lemmas A.4 and A.5, with $\beta = O(K_g^{-3/2})$, we have,*

$$\mathbb{E}[L_N(\omega^j_{\tau,Z}) - L_N(\omega^j_\tau)] \leq -\gamma\beta c \sum_{z=1}^{Z}\sum_{e=0}^{E-1} \mathbb{E}[\|\nabla L_N(\omega^{i_{j,z}}_{\tau,\zeta,e})\|^2] + \Delta_\tau$$
$$\forall \tau \geq 0, j \in \mathcal{G}$$

where

$$\Delta_\tau = \beta\gamma^2 O(\rho E^3 Z V_2) + \gamma\beta^2 O(K_g E Z \sqrt{V_1 V_2}) + \gamma^2\beta^3 O((L+\rho)K_g^2 E^2 Z V_2) \tag{28}$$

Proof. We ignore τ and j to simplify the proof. We can simply sum over the terms in Lemma A.4, to obtain,

$$\mathbb{E}[L_N(\omega^j_{\tau,Z}) - L_N(\omega^j_\tau)] \leq -\gamma\beta c \sum_{z=1}^{Z}\sum_{e=0}^{E-1} \mathbb{E}[\|\nabla L_N(\omega^{i_{j,z}}_{\tau,\zeta,e})\|^2] + \sum_{z=1}^{Z}\Delta_z \tag{29}$$

Now, to compute the sum of Δ_z terms,

$$\begin{aligned}\sum_{z=1}^{Z}\Delta_z &= \sum_{z=1}^{Z}(\beta\gamma^2 O(\rho E^3 V_2) + \beta O(\sqrt{V_1})\sqrt{\mathcal{R}_z} + \beta O(L+\rho)\mathcal{R}_z)\\ &= \beta\gamma^2 O(\rho E^3 Z V_2) + \beta O(\sqrt{V_1})\sum_{z=1}^{Z}\sqrt{\mathcal{R}_z} + \beta O(L+\rho)\sum_{z=1}^{Z}\mathcal{R}_z\\ &= \beta\gamma^2 O(\rho E^3 Z V_2) + \gamma\beta^2 O(K_g E Z\sqrt{V_1 V_2})\\ &\quad + \gamma^2\beta^3 O((L+\rho)K_g^2 E^2 Z V_2)\end{aligned}$$

□

A.3 Cloud steps

In this section, we try to bound the progress made in a single cloud step. First, we bound certain quantities which will appear in the proof.

Lemma A.7 (Gateway iterate difference). *Under Assumptions 1 and 2, we have,*

$$\mathbb{E}[\|\omega_{\tau,Z} - \omega_\tau\|^2] \leq \gamma^2\beta^2(O(Z^2E^2V_2) + \beta^2 O(K_g^2 E^2 Z^2 V_2)) \tag{30}$$

Proof. Note that this term corresponds to the difference in iterates after Z steps on gateway on model received at cloud step τ. We can split this difference into a telescopic sum of single steps on the gateway and then use triangle inequality.

$$\mathbb{E}[\|\omega_{\tau,Z} - \omega_\tau\|^2] = \mathbb{E}[\|\sum_{r=1}^{Z}\omega_{\tau,r} - \omega_{\tau,r-1}\|^2] \tag{31}$$

$$\leq Z\sum_{r=1}^{Z}\mathbb{E}[\|\omega_{\tau,r} - \omega_{\tau,r-1}\|^2] \tag{32}$$

$$\leq Z\beta^2\sum_{r=1}^{Z}\mathbb{E}[\|\omega_{\tau,\zeta_r,E} - \omega_{\tau,r-1}\|^2] \tag{33}$$

Here, we assume that $\omega_{\tau,\zeta_r,E}$ is the iterate received at r^{th} step on the gateway. Now, we can use arguments similar to Lemma A.6 to establish a recursion for each term in the sum in Eq. (33).

$$\begin{aligned}\mathbb{E}[\|\omega_{\tau,\zeta_r,E} - \omega_{\tau,r-1}\|^2] &\leq \mathbb{E}[\|\omega_{\tau,\zeta_r,E} - \omega_{\tau,\zeta_r} + \omega_{\tau,\zeta_r} - \omega_{\tau,r-1}\|^2]\\ &\leq 2\mathbb{E}[\|\omega_{\tau,\zeta_r,E} - \omega_{\tau,\zeta_r}\|^2]\\ &\quad + 2\mathbb{E}[\|\omega_{\tau,\zeta_r} - \omega_{\tau,r-1}\|^2]\\ &\leq 2\gamma^2 O(E^2V_2) + 2\mathcal{R}_r\end{aligned}$$

Plugging the above expression into Eq. (33), and using Lemma A.5 to bound $\sum_{z=1}^{Z}\mathcal{R}_z$, we obtain the result. □

Lemma A.8 (Cloud delay recursion). *Under Assumptions 1,3 and 2, where we define $Q_h = \mathbb{E}[\|\omega_\tau - \omega_{h-1}\|^2]$, where $\omega_{\tau,Z}$ is the iterate received at cloud step h, then*

$$Q_h \leq \gamma^2\alpha^2\beta^2(O(K_c^2Z^2E^2V_2) + \beta^2 O(K_c^2K_g^2E^2Z^2V_2)) + 2\alpha^2 K_c\sum_{r=1}^{K_c-1}Q_{h-r} \tag{34}$$

Proof. By Assumption 2, $h-1-\tau \le K_c$, therefore, there have been atmost K_c cloud steps between ω_τ and ω_{h-1} and we can express this difference as a telescopic sum of consecutive steps on the cloud followed by triangle inequality.

$$Q_h = \mathbb{E}[\|\sum_{r=0}^{h-\tau}(\omega_{h-r} - \omega_{h-r-1})\|^2] \tag{35}$$

$$\le (h-\tau)\sum_{r=0}^{h-\tau}\mathbb{E}[\|\omega_{h-r} - \omega_{h-r-1}\|^2] \tag{36}$$

Now, we can analyze each term in the sum in Eq. (36) separately. Let $\omega_{\tau_{h-r},Z}$ be the model received at the cloud at step $h-r$, then

$$\begin{aligned}\mathbb{E}[\|\omega_{h-r} - \omega_{h-r-1}\|^2] &\le \alpha^2\mathbb{E}[\|\omega_{\tau_{h-r},Z} - \omega_{h-r-1}\|^2] \\ &\le \alpha^2\mathbb{E}[\|\omega_{\tau_{h-r},Z} - \omega_{\tau_{h-r}} + \omega_{\tau_{h-r}} - \omega_{h-r-1}\|^2] \\ &\le 2\alpha^2\Big(\mathbb{E}[\|\omega_{\tau_{h-r},Z} - \omega_{\tau_{h-r}}\|^2 \\ &\quad + \mathbb{E}[\|\omega_{\tau_{h-r}} - \omega_{h-r-1}\|^2]\Big) \\ &\le 2\alpha^2\Big(\gamma^2\beta^2(O(Z^2E^2V_2) + \beta^2 O(K_g^2E^2Z^2V_2)) \\ &\quad + Q_{h-r}\Big)\end{aligned}$$

We utilize the update equation at the cloud with convexity of the $\|\cdot\|$ along with the triangle inequality and Lemma A.7 in the above steps.

Plugging this into Eq. (36) and using Assumption 2, we get our result.

$$\begin{aligned}Q_h &\le 2(h-\tau)\alpha^2\Big((h-\tau)\gamma^2\beta^2(O(Z^2E^2V_2) + \beta^2 O(K_g^2E^2Z^2V_2)) \\ &\quad + \sum_{r=1}^{h-\tau} Q_{h-r}\Big) \qquad (37) \\ &\le \gamma^2\alpha^2\beta^2(O(K_c^2Z^2E^2V_2) + \beta^2 O(K_c^2K_g^2E^2Z^2V_2)) \\ &\quad + 2\alpha^2K_c\sum_{r=1}^{K_c-1} Q_{h-r} \qquad (38)\end{aligned}$$

□

Using these lemmas, we can analyze the progress made in a single step at the cloud.

Lemma A.9 (Single-step Cloud). *Under Assumptions 1,3 and 2, after a single step on the cloud, we have,*

$$\begin{aligned}\mathbb{E}[L_N(\omega_h) - L_N(\omega_{h-1})] &\le -\gamma\alpha\beta c\sum_{z=0}^{Z-1}\sum_{e=0}^{E-1}\mathbb{E}[\|\nabla L_N(\omega_{\tau,\zeta,e})\|^2] \\ &\quad + \Xi_h, \quad \forall h > 0 \qquad (39)\end{aligned}$$

where

$$\begin{aligned}\Xi_h =& \alpha\beta\gamma^2 O(\rho E^3ZV_2) + \alpha\gamma\beta^2 O(K_gEZ\sqrt{V_1V_2}) \\ &+ \alpha\gamma^2\beta^3 O((L+\rho)K_g^2E^2ZV_2) \\ &+ \alpha\rho\gamma^2\beta^2(O(Z^2E^2V_2) + \beta^2 O(K_g^2E^2Z^2V_2)) \\ &+ O(\alpha\sqrt{V_1})\sqrt{Q_h} + \alpha O(\rho+L)Q_h\end{aligned}$$

Proof. The proof is similar to that of Lemma A.4 until 20, so we state the corresponding equation for a single cloud step.

$$\begin{aligned}&\mathbb{E}[L_N(\omega_h) - L_N(\omega_{h-1})] \qquad (40) \\ &\le \alpha\mathbb{E}[L_N(\omega_{\tau,Z}) - L_N(\omega_\tau)] + \alpha\mathbb{E}[L_N(\omega_\tau) - L_N(\omega_{h-1})] \\ &\quad + \frac{\alpha\rho}{2}\mathbb{E}[\|\omega_{\tau,Z} - \omega_{h-1}\|^2] \qquad (41)\end{aligned}$$

Now, we analyze the terms in above equation similar to Lemma A.4 For the second term in Eq. (41), we can use Lemma A.6, as it is the difference of function values after Z steps at the gateway.

For the third term in Eq. (41), we can use the triangle inequality and use Lemmas A.7 and A.8,

$$\begin{aligned}\mathbb{E}[\|\omega_{\tau,Z} - \omega_{h-1}\|^2] &\le 2\mathbb{E}[\|\omega_{\tau,Z} - \omega_\tau\|^2] + 2Q_h \\ &\le \gamma^2\beta^2(O(Z^2E^2V_2) + \beta^2 O(K_g^2E^2Z^2V_2)) + 2Q_h\end{aligned}$$

We handle the first term in Eq. (41) separately using L-smoothness of L_N.

$$\begin{aligned}\mathbb{E}[L_N(\omega_\tau) - L_N(\omega_{h-1})] \le& \mathbb{E}[\langle\nabla L_N(\omega_{h-1}), \omega_\tau - \omega_{h-1}\rangle \\ &+ \frac{L}{2}\mathbb{E}[\|\omega_\tau - \omega_{h-1}\|^2] \\ \le& \mathbb{E}[\|\nabla L_N(\omega_{h-1})\|\|\omega_\tau - \omega_{h-1}\|] + \frac{L}{2}Q_h \\ \le& O(\sqrt{V_1})\mathbb{E}[\|\omega_\tau - \omega_{h-1}\|] + \frac{L}{2}Q_h \\ \le& O(\sqrt{V_1})\sqrt{Q_h} + \frac{L}{2}Q_h\end{aligned}$$

We use Cauchy-Schwartz inequality to bound $\langle a, b\rangle \le \|a\|\|b\|, \forall a, b \in \mathbb{R}^d$. Further, we utilize Assumption 1 to bound $\|\nabla g_N(\cdot)\|$ and then use Jensen's inequality to obtain $\mathbb{E}[\|a\|] \le \sqrt{\mathbb{R}[\|a\|^2]}$.

Substituting these values in Eq. (41), we complete the proof.

□

We now bound the sum of the recursion terms Q_h and $\sqrt{Q_h}$.

Lemma A.10 (Sum of Cloud recursion). *Under Assumptions 1,2 and 3, and when $\alpha \le O(K_c^{-3/2})$, we have,*

$$\sum_{h=1}^{H} Q_h \le \gamma^2\alpha^2\beta^2(O(HK_c^2Z^2E^2V_2) + \beta^2 O(HK_c^2K_g^2E^2Z^2V_2)) \tag{42}$$

$$\sum_{h=1}^{H}\sqrt{Q_h} \le \gamma\alpha\beta(O(HK_cZE\sqrt{V_2}) + \beta O(HK_cK_gEZ\sqrt{V_2})) \tag{43}$$

Proof. We first start with Lemma A.8 and sum over $h=1$ to H

$$\begin{aligned}\sum_{h=1}^{H} Q_h \le& \gamma^2\alpha^2\beta^2(O(HK_c^2Z^2E^2V_2) + \beta^2 O(HK_c^2K_g^2E^2Z^2V_2)) \\ &+ 2\alpha^2K_c^2\sum_{h=1}^{H} Q_h \\ \sum_{h=1}^{H} Q_h \le& (1-2\alpha^2K_c^2)^{-1}\gamma^2\alpha^2\beta^2(O(HK_c^2Z^2E^2V_2) \\ &+ \beta^2 O(HK_c^2K_g^2E^2Z^2V_2))\end{aligned}$$

Here, summing over the recursion, we need to control the coefficient of the sum on RHS. If this coefficient is < 1, then we are done, which is exactly the case when $\alpha = O(K_c^{-3/2})$ as $\alpha^2 K_c^2 = O(K_c^{-1})$ which can be made to be a constant less than 1.

Now, for the sum over square roots, we first use Jensen's inequality to obtain, $\sqrt{\sum_{i=1}^n a_i} \leq \sum_{i=1}^n \sqrt{a_i}, \forall a_i \geq 0$ on Lemma A.8.

$$\begin{aligned}
\sqrt{Q_h} &\leq \gamma\alpha\beta(O(K_c ZE\sqrt{V_2}) + \beta O(K_c K_g EZ\sqrt{V_2})) \\
&\quad + \alpha\sqrt{K_c}\sum_{r=1}^{K_c-1}\sqrt{Q_{h-r}} \\
\sum_{h=1}^{H}\sqrt{Q_h} &\leq \gamma\alpha\beta(O(HK_c ZE\sqrt{V_2}) + \beta O(HK_c K_g EZ\sqrt{V_2})) \\
&\quad + \alpha K_c^{3/2}\sum_{h=1}^{H}\sqrt{Q_h} \\
&\leq (1-\alpha K_c^{3/2})^{-1}\gamma\alpha\beta(O(HK_c ZE\sqrt{V_2}) \\
&\quad + \beta O(HK_c K_g EZ\sqrt{V_2})) \\
&\leq \gamma\alpha\beta(O(HK_c ZE\sqrt{V_2}) + \beta O(HK_c K_g EZ\sqrt{V_2}))
\end{aligned}$$

Now, we again control the coefficient of the sum on RHS and for $\alpha = O(K_c^{-3/2})$, this coefficient is a constant. □

A.4 Completing the proof

The all-step progress for the cloud, after H steps, is our Theorem 5.1. We now provide its proof.

We first sum Lemma A.9 over $h = 1$ to H, to obtain,

$$\begin{aligned}
\mathbb{E}[L_N(\boldsymbol{\omega}_H) - L_N(\boldsymbol{\omega}_0)] \leq -\gamma\alpha\beta c\sum_{h=1}^{H}\sum_{z=0}^{Z-1}\sum_{e=0}^{E-1}\mathbb{E}[\|\nabla L_N(\boldsymbol{\omega}_{\tau,\zeta,e})\|^2] \\
+ \sum_{h=1}^{H}\Xi_h
\end{aligned} \tag{44}$$

We first compute $\sum_{h=1}^{H}\Xi_h$ using Lemma A.10 to bound the sum of recursion terms in Q_h and $\sqrt{Q_h}$.

$$\begin{aligned}
\sum_{h=1}^{H}\Xi_h &= \alpha\beta\gamma^2 O(\rho HE^3 ZV_2) + \alpha\gamma\beta^2 O(K_g HEZ\sqrt{V_1V_2}) \\
&\quad + \alpha\gamma^2\beta^3 O((L+\rho)K_g^2 HE^2 ZV_2) \\
&\quad + \alpha\rho\gamma^2\beta^2(O(Z^2HE^2V_2) + \beta^2 O(K_g^2 HE^2 Z^2 V_2)) + O(\alpha\sqrt{V_1})\sum_{h=1}^{H}\sqrt{Q_h} \\
&\quad + \alpha O(\rho + L)\sum_{h=1}^{H}Q_h \\
&\leq \alpha\beta\gamma^2 O(\rho HE^3 ZV_2) + \alpha\gamma\beta^2 O(K_g HEZ\sqrt{V_1V_2}) \\
&\quad + \alpha\gamma^2\beta^3 O((L+\rho)K_g^2 HE^2 ZV_2) \\
&\quad + \alpha\rho\gamma^2\beta^2(O(Z^2HE^2V_2) + \beta^2 O(K_g^2 HE^2 Z^2 V_2)) \\
&\quad + \gamma\alpha^2\beta(O(HK_c ZE\sqrt{V_1V_2}) + \beta O(HK_c K_g EZ\sqrt{V_1V_2})) \\
&\quad + O(\rho + L)\gamma^2\alpha^3\beta^2(O(HK_c^2 Z^2 E^2 V_2) \\
&\quad + \beta^2 O(HK_c^2 K_g^2 E^2 Z^2 V_2))
\end{aligned}$$

Now, consider the following term,

$$\begin{aligned}
\min_{h=0}^{H-1}\mathbb{E}[\|\nabla L_N(\boldsymbol{\omega}_h)\|^2] &\leq \frac{1}{H}\sum_{h=0}^{H-1}\mathbb{E}[\|\nabla L_N(\boldsymbol{\omega}_h)\|^2] \\
&\leq \frac{1}{HZ}\sum_{h=0}^{H-1}\sum_{z=0}^{Z-1}\mathbb{E}[\|\nabla L_N(\boldsymbol{\omega}_{\tau,z})\|^2] \\
&\leq \frac{1}{HZE}\sum_{h=0}^{H-1}\sum_{z=0}^{Z-1}\sum_{e=0}^{E-1}\mathbb{E}[\|\nabla L_N(\boldsymbol{\omega}_{\tau,\zeta,e})\|^2]
\end{aligned}$$

We obtained the above inequality using the fact that $\min_{i\in[n]} a_i \leq \frac{1}{n}\sum_{i=1}^{n} a_i$ and using convexity of $\|\cdot\|$. We can now use the results from Eq. (44) to obtain

$$\min_{h=0}^{H-1}\mathbb{E}[\|\nabla L_N(\boldsymbol{\omega}_h)\|^2] \leq \frac{\mathbb{E}[L_N(\boldsymbol{\omega}_0) - L_N(\boldsymbol{\omega}_H)]}{\alpha\beta\gamma cHZE} + \frac{\sum_{h=1}^{H}\Xi_h}{\alpha\beta\gamma cHZE} \tag{45}$$

We set the constant term to be Ξ.