MOHAWK: Mobility and Heterogeneity-Aware Dynamic Community Selection for Hierarchical Federated Learning

Allen-Jasmin Farcas
allen.farcas@utexas.edu
The University of Texas at Austin
Austin, TX, USA

Myungjin Lee
myungjinle@cisco.com
Cisco Systems
San Francisco, CA, USA

Ramana Rao Kompella
rkompell@cisco.com
Cisco Systems
San Francisco, CA, USA

Hugo Latapie
hlatapie@cisco.com
Cisco Systems
San Francisco, CA, USA

Gustavo de Veciana
deveciana@utexas.edu
The University of Texas at Austin
Austin, TX, USA

Radu Marculescu
radum@utexas.edu
The University of Texas at Austin
Austin, TX, USA

ABSTRACT

The recent developments in Federated Learning (FL) focus on optimizing the learning process for data, hardware, and model heterogeneity. However, most approaches assume all devices are stationary, charging, and always connected to the Wi-Fi when training on local data. We argue that when real devices move around, the FL process is negatively impacted and the device energy spent for communication is increased. To mitigate such effects, we propose a dynamic community selection algorithm which improves the communication energy efficiency and two new aggregation strategies that boost the learning performance in Hierarchical FL (HFL). For real mobility traces, we show that compared to state-of-the-art HFL solutions, our approach is scalable, achieves better accuracy on multiple datasets, converges up to 3.88× faster, and is significantly more energy efficient for both IID and non-IID scenarios.[1]

CCS CONCEPTS

• **Computing methodologies → Machine learning**; **Supervised learning**; • **Security and privacy**;

KEYWORDS

Federated Learning, Edge Devices, Data Heterogeneity, Data Privacy, Communication Cost, Energy Efficiency, Mobile Devices, Internet-of-Things

ACM Reference Format:
Allen-Jasmin Farcas, Myungjin Lee, Ramana Rao Kompella, Hugo Latapie, Gustavo de Veciana, and Radu Marculescu. 2023. MOHAWK: Mobility and Heterogeneity-Aware Dynamic Community Selection for Hierarchical Federated Learning. In *International Conference on Internet-of-Things Design and Implementation (IoTDI '23), May 09–12, 2023, San Antonio, TX, USA.* ACM, New York, NY, USA, 13 pages. https://doi.org/10.1145/3576842.3582378

[1]Code is available at: https://github.com/SLDGroup/MOHAWK

IoTDI '23, May 09–12, 2023, San Antonio, TX, USA
© 2023 Copyright held by the owner/author(s).
ACM ISBN 979-8-4007-0037-8/23/05.
https://doi.org/10.1145/3576842.3582378

1 INTRODUCTION

Federated learning (FL) trains machine learning (ML) models using the local data available on edge devices and then aggregates the updated local models in the cloud to obtain a global model. For instance, Federated Average (FedAvg) [23] simply averages the updated local model parameters to obtain a new global model. Many improvements to the original FedAvg algorithm [23] have been proposed to boost the learning in FL systems, *e.g.*, FedProx [17], FedMax [5], FedNova [28] to mention a few. Challenges like hardware, model, and data heterogeneity can negatively impact the learning and communication in FL [16] since in real-world scenarios all types of heterogeneity appear naturally. In particular, the impact on communication is the most important one [16, 20], especially since the edge devices may struggle to communicate the trained model back and forth with the cloud. Additionally, hardware heterogeneity impacts the devices communication with different connectivity technologies (*e.g.*, Wi-Fi, 4G or 5G, etc.). Finally, real data distributions can make the learning process harder to converge, hence requiring more communication rounds until a certain accuracy threshold is reached. This induces an even bigger impact on communication as each participating device needs to download and upload the local model multiple times.

Another line of work brings the computation even closer to the edge by using hierarchical FL to reduce the communication overhead of FL. Hierarchical FL (HFL), first pioneered in [18], uses edge Access Points (APs) as intermediary aggregation points before transmitting the edge models to the cloud for global aggregation. This enables edge devices to use higher communication speeds with local APs (instead of communicating directly with the cloud).

Previous HFL solutions [1, 18] make every device communicate with the same pre-assigned AP during all communication rounds. This limiting assumption does not consider the physical distance between devices and its assigned AP, hence it directly impacts the capacity and energy consumption spent in communication. Other HFL solutions consider the distance between the device and the AP to select the AP [22], but ignore the real mobility patterns of the various devices (*i.e.*, devices are assumed to be stationary and uniformly distributed in a given area). Other HFL solutions consider devices that can randomly change their current AP with one of their neighbors [7]. Such state-of-the-art approaches consider that any AP has *only* two neighboring APs to which devices can randomly connect. This assumption is not realistic since the APs can have

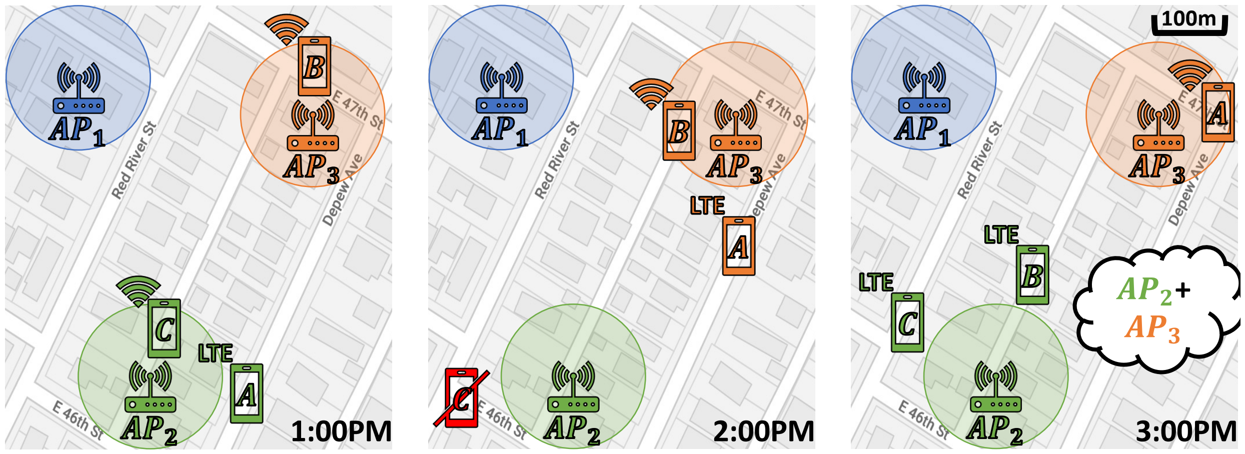

Figure 1: Our proposed MOHAWK framework selects at every time step dynamic communities of devices and their closest Access Point (AP). Within 100m range of an AP the devices have Wi-Fi connection, otherwise LTE. Dynamic edge aggregation allows devices like device C (at 2:00PM) that may disappear due to battery depletion to be reconsidered for learning the next time they appear, as long as the global model from the cloud was not updated in the meantime. Selective global aggregation only aggregates in the cloud the APs that aggregated at least one device since the last global aggregation (*e.g.*, AP_2 and AP_3).

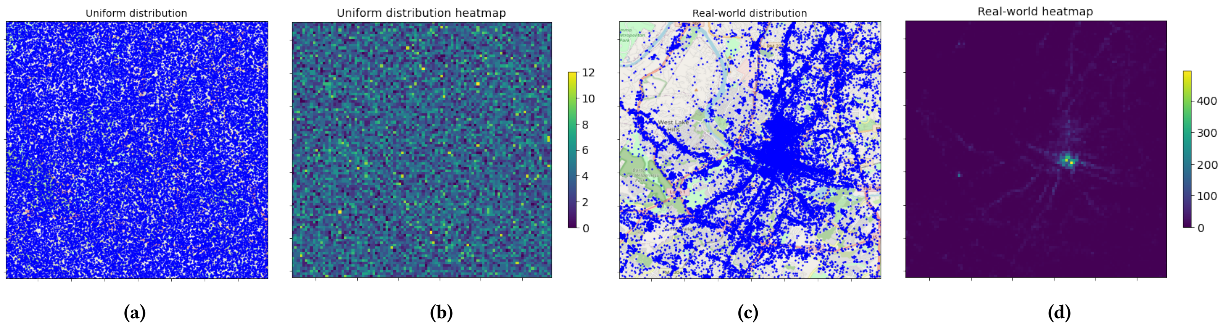

Figure 2: Uniform vs. real-world distribution of 37,994 Points of Interest (PoIs) where devices can be at in an urban area. (a) shows the uniform distribution of the 37,994 PoIs, while (b) is the heatmap of the distribution in (a) showing how uniformly spread the PoIs are. In contrast, (c) shows a real-world distribution of the PoIs and (d) is the heatmap of (c), showing how much non-uniformity there is, with hundreds of PoIs in the Downtown area and barely any PoIs in the outskirts of the city.

any number of neighbors and the device mobility is not dictated by fixed probabilities, but by human behavior.

Works such as [1] consider an area that is too small to be realistic (*i.e.*, 750m×750m) for the deployment of edge devices and APs. Even though the authors of [25] consider a larger area of 2km×2km, their focus is on simulated mobility and fully cooperative learning. Even with such solutions, there are still important problems associated with real device mobility that remain unaddressed. As mentioned in [20], the assumption that all data owners are willing to participate in the FL process *anytime and anywhere* is not realistic. We illustrate in Fig. 1 how some device C starts training at 1PM, has battery depleted at 2PM, and then becomes available again at 3PM. Due to the aforementioned limitations, current methods would *not* consider device C for aggregation; however, in order for devices to be able to learn *anytime* and *anywhere*, this is one of the first research questions that needs to be addressed.

Recent works consider a uniform distribution of locations the devices can be at, *i.e.*, Points of Interest (PoIs), and continuous availability of all participating devices in the FL process. However, this is not realistic since, as it can be seen in Fig. 2, the real-world distribution of the PoIs looks nothing like a uniform distribution. The presence of hubs can be seen especially in cities where Downtown and specific areas are more frequented by people. Having a realistic experimental validation is thus crucial to get us closer to learning anytime and anywhere.

To address these limiting factors, we propose a Mobility and Heterogeneity-Aware Dynamic Community Selection (MOHAWK) framework for Hierarchical Federated Learning that combines a dynamic community selection algorithm with *real device mobility* under heterogeneous environments using two new aggregation techniques which are essential for energy efficiency and scalability. As shown in Fig. 1, we provide a solution to adapt the learning process to missing and reappearing devices (such as device C at 2PM) which we shall refer to as *dynamic edge aggregation*. We also propose a *selective global aggregation* technique which only aggregates the model weights from the APs that aggregated any devices since the last global aggregation. Our dynamic edge aggregation enables more devices to participate with their local updates in the learning process, while the selective global aggregation results in a faster global convergence and scales to any number of APs. Our solution enables devices to learn continuously during the day, whenever they are available, spending less energy for communication, converging faster, and achieving better accuracy than other state-of-the-art HFL solutions.

The contributions of the paper are as follows:

- **Mobility and Heterogeneity-Aware Dynamic Community Selection (MOHAWK)**: A framework that combines a dynamic community selection algorithm for energy-efficient communication in mobile FL systems with two new aggregation strategies (*i.e.*, dynamic edge aggregation and selective global aggregation) that boost the learning performance under heterogeneous scenarios.
- **Federated Learning using Real Devices Mobility**: MOHAWK aims to include as many devices in the learning process as possible, thus providing a more inclusive (*i.e.*, fair) environment which guarantees that less energy will be wasted on training models that are ultimately not aggregated. To the best of our knowledge, this paper is the first to account for real devices mobility for FL.
- **Empirical Validation**: We show that MOHAWK converges up to 3.88× faster and is more scalable than state-of-the-art HFL solutions on MNIST, EMNIST, CIFAR10 and CIFAR100, while being more energy efficient.
- **A Hardware Prototype**: We provide real energy measurements on a hardware prototype with 36 Raspberry Pi devices that show up to 2.24× less average energy wasted per device for training local models compared to state-of-the-art HFL solutions.

To summarize, we provide a new energy-efficient solution for HFL which considers real mobility and availability of edge devices. The remainder of the paper is organized as follows: Section 2 discusses relevant prior work. In Section 3, we present our proposed approach. Section 4 shows our experimental results (both simulation and hardware prototype), while Section 5 summarizes our main contributions and outlines directions for future work.

2 RELATED WORK

2.1 Hierarchical Federated Learning

HierFAVG [18] considers a scenario with 50 devices and 5 APs, optimistically assuming that at every communication round every AP will handle exactly 5 devices. Looking at Fig. 2, this is a strong limitation. Abad et al. [1] consider 28 users uniformly distributed across a circular area with radius 750m; they fix 7 APs, each having 4 devices during each communication round. No device mobility is considered, and the results are provided only for CIFAR10 under the IID scenario with FedAvg used for aggregation. Hier-local-QSGD [19] is proposed for HFL with quantization using 20 clients and 4 edge servers, each server having 5 devices during each communication round. Hierarchical Federated Edge Learning (HFEL) [22] addresses resource allocation optimization and edge association problem solving. The authors consider up to 60 devices distributed randomly over a 500m x 500m area with up to 25 edge servers. However, HFEL does not consider any device mobility and thus their solution is purely a resource allocation optimization. We call the methods above *stationary* since all devices have a pre-assigned AP to communicate with. As mentioned in [2], the methods proposed in [1, 22] are appealing for computation offloading, but have limited applicability in the context of real mobile devices where there is almost no possibility to organize them.

Mobility-Aware Cluster FL (MACFL) [7] tries to relax the stationary scenario by allowing devices to move to a neighboring AP based on a fixed probability, and assuming each AP has *only* two neighbors. MACFL also uses 50 devices and 5 APs for their experiments on MNIST, just like HierFAVG [18]. We call MACFL *pseudo-mobile* because it allows some devices to change their APs, but has the same assumption as HierFAVG, *i.e.*, all devices are available at every communication round. In Federated Attentive Message Passing (FedAMP) [12], the authors propose a heuristic version of FedAMP called HeurFedAMP which considers a self-attention hyperparameter to control the weight of each message sent from a client to the cloud. This self-attention hyperparameter uses the cosine similarity between the local model and the global model and adjusts the weight of the local model in the aggregation based on how different the local model is from the global model. Inspired by HeurFedAMP, MACFL [7] uses the same attention scheme for every device and edge aggregation to help the learning process with clients randomly changing their APs.

To address the limitations of stationary and pseudo-mobile state-of-the-art solutions, we are the first to consider *real mobility* data for HFL together with realistic setups of devices and APs. Unlike [1, 7, 18, 19, 22], we do *not* assume that all devices are available at every communication round, as they get disconnected due to issues like battery depletion or temporary loss of signal.

2.2 Mobility Models and Scalability

Ochiai et al. [25] propose a fully distributed cooperative FL system organized by nodes that are physically nearby, *i.e.*, nodes communicate with other nodes if they are within the radio range, thus producing opportunistic contacts for learning. The authors of [25] simulate mobility using Random Waypoint (RWP) mobility [3] and Community Structured Environment (CSE) mobility [24]. In RWP the nodes are devices that walk around a given area, spending some time at different locations, while in CSE the nodes are devices assumed to be part of a few communities and they move from one community to another. For RWP, the authors in [25] use an area as large as 2km×2km and assume 100m as the radio range for communication. Ochiai et al. also show in [25] that the large area for RWP requires a longer time for more contacts among the nodes to obtain enough accuracy. Another realistic approach in [27] takes into account the real-world vehicle trace dataset to create

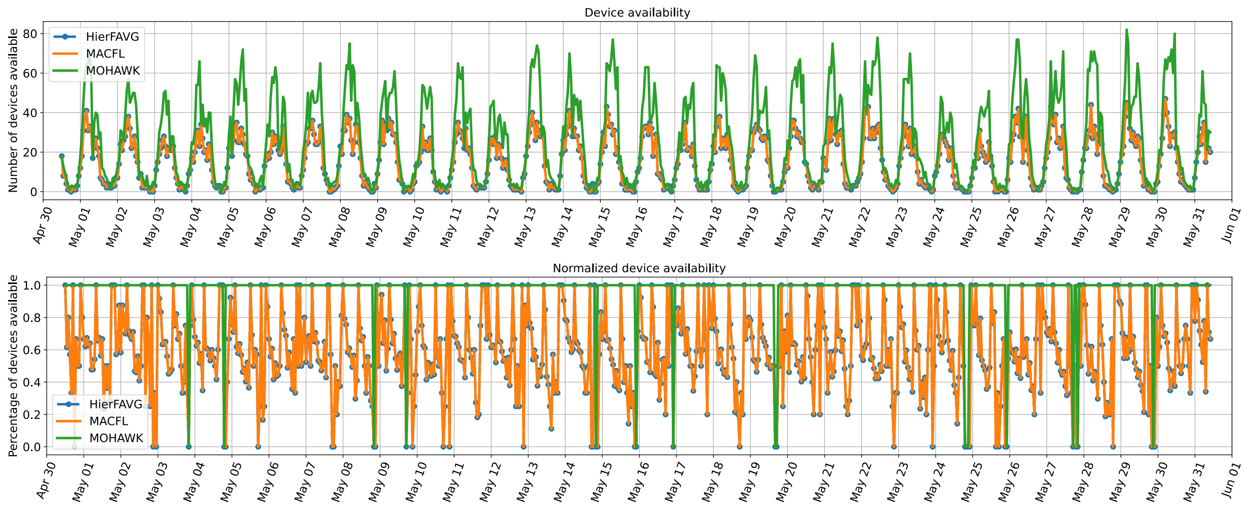

Figure 3: Device availability and normalized device availability during the month of May 2020. We observe a clear increase in the number of devices available for aggregation since our solution (*i.e.*, the green line) is always on top, while other (state-of-the-art) methods consider fewer devices for aggregation. Also, we note cases with no devices available (*e.g.*, May 3, 4, 5, etc.).

a FL system that considers the delay as a learning parameter, due to high mobility of vehicles. The authors use the Mobile Century Dataset [9] which contains 77 vehicles and traces approximately 20 miles of Interstate 880, but only 10 agents are selected during every communication round.

In contrast to [25], instead of using simulated mobility, we use the Foursquare *real-mobility dataset* [8]. We consider the top 1,000 devices that appear most times during the month of May 2020 within the metropolitan area of interest based on Foursquare data. The devices are smartphones associated with people moving around, *e.g.*, walking or driving. From these 1,000 devices for any given time step, there are about 76 devices present on the map, almost the same as the *total* number of devices used by [27]; this shows the scalability of our work.

2.3 Hardware Validation

Real hardware experiments are rarely reported in the FL literature. For instance, Luo et al. [21] use 20 Raspberry Pi 4 and 10 NVIDIA Jetson Nano devices and measure the average computation and communication time, without power or energy measurements. ClusterFL [26] uses a prototype built with 7 NVIDIA Jetson TX2 and 3 NVIDIA Jetson AGX to evaluate the impact of dynamic network conditions concluding that the real-world 4G LTE has substantially lower and more unstable bandwidth compared to Wi-Fi and Ethernet. Our proposed hardware prototype is equipped with 36 Raspberry Pi 3B+ devices, each of them having its own dedicated Smart Power 2 device to measure energy consumption.

3 METHODOLOGY

3.1 Device Availability and Distribution

Previous works [1, 7, 18, 19, 22] consider a uniform (random) distribution of devices that are always available for all their experiments. However, as shown in Fig. 2, it is unrealistic to assume a uniform distribution of PoIs over the entire area of interest. Besides considering unrealistically small areas of deployment (*e.g.*, 500m×500m [22]), the uniform distribution of devices also limits the usefulness of hierarchical approaches. We observe in all cities, a higher concentration of clients and their devices in some areas (*e.g.*, Downtown), and a sparser distribution of devices on the outskirts of a city. In Fig. 2 we show 37,994 PoIs where users may go, *e.g.*, McDonald's, Starbucks, parks, shopping malls to name a few. We consider all PoIs to be APs since all of them have at least one private Wi-Fi network available. So, it is natural when a client goes to such a PoI to not rely on its own cellular data and connect to the Wi-Fi network available at that location. Thus, for HFL we use dynamic connection between every device and its closest AP every time the device is available.

Device availability is also a big issue for FL systems, yet it is even less discussed in the literature. In Fig. 3, we show the availability of the top 1,000 devices that appear most times during the month of May 2020 from a total of 12,866 devices. As it can be seen, we have the lowest number of devices active at night and peak numbers of active devices during the afternoon. For both PoIs and device availability, we use data collected from Foursquare [8] over an area of 17.5km×17.5km. Previous HFL solutions consider that, at any given time, all APs have a fixed number of devices connected, *i.e.*, all devices are available at any time. As seen in Fig. 3, from 1,000 devices, we may end up having at most 76 devices present at the same time: sometimes, we may not have any devices available at all. To enable HFL anytime and anywhere, solutions should be robust to such dramatic variations in availability.

Since current FL solutions are oblivious to device availability issues, they typically aggregate at time $t + 1$, all devices that have been trained at t, assuming they are all still available. However, not

Table 1: Data transfer power model parameters [11].

Connection type	α_u [mW/Mbps]	α_d [mW/Mbps]	β [mW]
LTE	438.39	51.97	1288.04
Wi-Fi	283.17	137.01	132.86

Table 2: 4G LTE per user throughput ranges for different carriers [4]. We consider a mixed range from all carriers.

Carrier	Verizon	T-Mobile	AT&T	Sprint	t_{min}	t_{max}
t_u [Mbps]	15	16-17	11-12	7-8	**7**	**17**
t_d [Mbps]	36	23-24	25-26	12-30	**12**	**36**

only that some particular devices may not be present at $t+1$, but they may come back at a later time (*e.g.*, $t+2$), case in which the device will *not* be considered for aggregation. This implies such a device wastes energy training a model that is ultimately not used for aggregation. We show in Section 4.4 how much energy is wasted on such devices that start training at time t and do not aggregate since they are not present at $t+1$, but reappear at some other future times. We alleviate this issue by considering for edge aggregation all devices that started training since the last global aggregation.

In Fig. 3, we show how our solution considers up to 39.78% more devices than state-of-the-art solutions. For the normalized device availability, we consider 1.0 as the maximum number of devices available for aggregation at every time. We note that quite a few times all the plot lines go to zero, denoting times when no device out of a total of 1,000 devices considered is actually available.

3.2 Communication Model

Recent findings in [29] show that even in 2022, traditional technologies like 4G LTE and Wi-Fi 4/5 are still used by the majority of mobile users due to their more mature deployment and stable performance. This is why we use for modeling the power characteristics of LTE and Wi-Fi taken from [11]. The data transfer power model (best fit) parameters from [11] are summarized in Table 1. Assuming the upload throughput is t_u [Mbps] and the download throughput is t_d [Mbps], we have the power level [mW] for upload as $P_u = \alpha_u t_u + \beta$ and for download $P_d = \alpha_d t_d + \beta$ (as shown in [11]), where α_u and α_d are the power model parameters and β is the base power when throughput is 0 (see Table 1). Considering μ the model size [Mb] and Δ^{comm} [Mb] the extra bits required for communication using the FL framework, we compute the total communication energy [mJ] for a device as in Eq. 1 for a complete *communication round*, *i.e.*, when the device receives the model and uploads the updated model back to the AP:

$$E_{total} = P_u * (\mu + \Delta^{comm})/t_u + P_d * (\mu + \Delta^{comm})/t_d \tag{1}$$

Due to availability variations, a device may only download the model at time t and upload it at a later time $t+\lambda$ when it becomes available again. The total energy spent for communication at time t is $E_{total} = P_d * (\mu + \Delta^{comm})/t_d$ when the device only downloads the global model, while it becomes $E_{total} = P_u * (\mu + \Delta^{comm})/t_u$ at time $t+\lambda$ when the device is only uploading the local model.

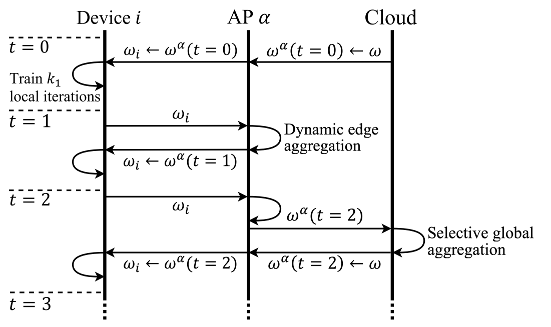
Device i
AP α
Cloud
$t=0$
$\omega_i \leftarrow \omega^\alpha(t=0)$
$\omega^\alpha(t=0) \leftarrow \omega$
Train k_1 local iterations
$t=1$
ω_i
Dynamic edge aggregation
$\omega_i \leftarrow \omega^\alpha(t=1)$
$t=2$
ω_i
$\omega^\alpha(t=2)$
Selective global aggregation
$\omega_i \leftarrow \omega^\alpha(t=2)$
$\omega^\alpha(t=2) \leftarrow \omega$
$t=3$

Figure 4: Sequence diagram showing the device-AP-cloud communication, with k_1 local epochs, $k_2 = 2$ dynamic edge aggregations, and one selective global aggregation at different time steps.

For realistic LTE connection throughputs, we consider the popular mobile carriers for their upload and download speeds, as summarized in Table 2. To set the minimum and maximum t_u and t_d, we consider the distance $\delta_{i,j}$ between an AP i and a device j. Given a random selection of 100 APs, we compute for the entire month of May 2020, the mean δ_{mean} and standard deviation δ_{std} distance between all devices and their selected APs. If a device is further than 100m from its AP, we select t_d and t_u based on Eq.2, where t_{min} and t_{max} are taken from Table 2:

$$t_d, t_u = \begin{cases} t_{max} & \text{if } \delta_{i,j} < \delta_{mean} - \delta_{std} \\ t_{min} & \text{if } \delta_{i,j} > \delta_{mean} + \delta_{std} \\ map(\delta_{i,j}; t_{max}, t_{min}) & \text{otherwise} \end{cases} \tag{2}$$

where map: $[\delta_{mean} - \delta_{std}, \delta_{mean} + \delta_{std}] \longrightarrow [t_{max}, t_{min}]$ is a function that maps linearly the distances within the range of $\delta_{mean} \pm \delta_{std}$ to the throughput speeds $[t_{max}, t_{min}]$. If a device is within 100m of its selected AP, we use a Wi-Fi speed of $t_u = t_d = 1000$ [Mbps]. We use the communication model during simulation to compute the energy consumption for communication (see Section 4.2).

3.3 MOHAWK Framework

In FL, we want to solve an optimization problem of the form:

$$\min_{\omega} f(\omega) = \frac{1}{|\mathbb{D}|} \sum_{i=1}^{|\mathbb{D}|} f_i(\omega, D_i) \tag{3}$$

where f_i is the loss function of device i evaluated on the local dataset D_i, $\mathbb{D}$ is the set that contains all devices and $|\cdot|$ denotes the number of elements of a set.

We solve the optimization problem in Eq. 3 over time. We show in Fig. 4 how the main steps of MOHAWK are performed at different time steps. We assume that device i selects AP α at time $t=0$. The first step is to download the global model weights ω from the cloud on AP α. Then, the AP model weights (at time $t=0$) denoted by $\omega^\alpha(t=0)$ are downloaded by the device i connected to AP α. The device i performs k_1 local epochs of stochastic gradient descent (SGD) on its local dataset. We assume the time it takes edge devices to train a model takes much longer than the communication of the model between the devices, APs, and the cloud. At $t=1$ we

Algorithm 1 Mobility and Hardware-Aware Dynamic Community Selection (MOHAWK)

```
1: Initialize global weights $\omega$ with random weights and download them on all APs $\omega^{\alpha} \leftarrow \omega, \forall \alpha \in \mathbb{A}$
2: Initialize time $t = 0$, set of APs for aggregation $\mathcal{A} = \emptyset$ and set of devices that trained $Tr = \emptyset$
3: for communication round $c = 1, 2, ..., C$ do
4:     for each device $i \in \mathbb{D}(t)$ in parallel do          ▷ For every device $i$ available at time $t$
5:         if $c == 1$ then
6:             Select $\alpha_i$ using Eq. 4          ▷ Dynamic community selection
7:         end if
8:         Download model $\omega^{\alpha_i}(t)$ from AP $\alpha_i$
9:         $\omega_i \leftarrow \mathtt{Train}(\omega^{\alpha_i}, k_1)$          ▷ Train for $k_1$ local epochs
10:    end for
11:    $Tr = Tr \cup \mathbb{D}(t)$          ▷ Save devices available at time $t$ as trained, but not aggregated
12:    $S^{\alpha} = \emptyset, \forall \alpha \in \mathbb{A}$
13:    $t = t + 1$          ▷ Proceed to the new time step $t + 1$
14:    for each device $i \in \mathbb{D}(t)$ in parallel do          ▷ For every device $i$ available at the new time $t$
15:        Select $\alpha_i$ using Eq. 4          ▷ Dynamic community selection
16:        $S^{\alpha_i} = S^{\alpha_i} \cup i$          ▷ Save device $i$ to aggregate at its selected AP $\alpha_i$
17:        $\mathcal{A} = \mathcal{A} \cup \alpha_i$          ▷ Save all APs that have devices to aggregate
18:    end for
19:    for each $\alpha \in \mathbb{A}$ in parallel do
20:        if $\alpha \in \mathcal{A}$ then
21:            $\omega^{\alpha}(t) = \sum_{\substack{i \in Tr \\ i \in S^{\alpha}}} p_i^{\alpha} \omega_i$          ▷ Dynamic edge aggregation
22:            $Tr = Tr \setminus S^{\alpha}$          ▷ Remove all aggregated devices $i$ from $Tr$
23:        else
24:            $\omega^{\alpha}(t) \leftarrow \omega^{\alpha}(t-1)$          ▷ Save the AP model weights
25:        end if
26:    end for
27:    if $c \bmod k_2 = 0$ then
28:        $\omega = \sum_{\alpha \in \mathcal{A}} q^{\alpha} \omega^{\alpha}(t)$          ▷ Selective global aggregation
29:        $\mathcal{A} = \emptyset, Tr = \emptyset$
30:        $\omega^{\alpha}(t) \leftarrow \omega, \forall \alpha \in \mathbb{A}$          ▷ Download global model on all APs
31:    end if
32: end for
```

assume device i selects the same AP α. Then, device i sends to AP α the updated local model weights ω_i at the next time step $t = 1$. The AP then performs one dynamic edge aggregation to update its own weights and then sends the updated weights $\omega^{\alpha}(t = 1)$ to device i. The device i performs again k_1 local epochs of SGD. At the next time step $t = 2$, the AP receives the updated local weights ω_i and performs another dynamic edge aggregation, *i.e.*, $k_2 = 2$. Since $k_2 = 2$, the AP α sends the updated model to the cloud for a selective global aggregation. The cloud sends the updated global model weights ω back to AP α, which sends the updated AP model weights $\omega^{\alpha}(t = 2)$ to device i and then the process repeats itself.

Dynamic Community Selection At every time t, we have only a subset of devices $\mathbb{D}(t) \subset \mathbb{D}$ available; due to real device mobility and availability, we have $|\mathbb{D}(t)| \ll |\mathbb{D}|$. Thus, we need to adapt the AP selection process to work dynamically for all APs $\alpha \in \mathbb{A}$, where $\mathbb{A}$ is the set containing all APs. As seen in Line 6 in Alg. 1, the first communication round begins with the *dynamic community selection* for each available device $i \in \mathbb{D}(t)$ by solving the following optimization problem:

$$\alpha_i = \underset{\alpha \in \mathbb{A}}{\text{argmin}}\ \mathrm{d}(\alpha, i) \tag{4}$$

where $d(\alpha, i) = \sqrt{(\alpha_x - i_x)^2 + (\alpha_y - i_y)^2}$ is the Euclidean distance between AP α and device i, and α_i is the selected AP for device i. We denote with S^{α} the set of all devices connected at AP α. Except the first communication round, at each new time step (Line 13 in Alg. 1), we perform for the available devices the dynamic community selection based on Eq. 4 (see Line 15 in Alg. 1). In other words, at every time step, each device selects its closest AP. This selection is performed locally on the device with the location of nearby APs known beforehand, since APs are assumed to be at fixed locations.

Dynamic Edge Aggregation Another implication of having $\mathbb{D}(t)$ devices available at time step t is that we may have $\mathbb{D}(t-1) \cap \mathbb{D}(t) = \emptyset$, thus the problem of how to perform edge aggregations in this context arises. We propose *dynamic edge aggregation*, which allows devices that started training their model at time t to aggregate their updates when they become available again, at $t + \lambda$, if only edge aggregations were performed during the time λ that has passed. Simply put, when a device becomes available, it selects an AP and that AP receives the last communication round when the device was available, *i.e.*, the last communication round during which the device received a model for local training. If this communication round occurred before a global aggregation, then the AP will consider the local model for dynamic edge aggregation. We implement the dynamic edge aggregation using a set Tr which contains *all devices* that have started training since the last global aggregation (Line 11 in Alg. 1). Thus, we perform dynamic edge aggregation as follows:

$$\omega^{\alpha}(t) = \sum_{\substack{i \in Tr \\ i \in S^{\alpha}}} p_i^{\alpha} \omega_i, \text{ where } p_i^{\alpha} = \frac{e^{-\sigma \cos(\omega^{\alpha}(t-1),\, \omega_i)}}{\sum_{i \in S^{\alpha}} e^{-\sigma \cos(\omega^{\alpha}(t-1),\, \omega_i)}} \tag{5}$$

where σ is a hyperparameter and $\cos(x, y) = \frac{<x,y>}{||x||^2\ ||y||^2}$ is the cosine similarity function. Inspired by FedAMP [12], we use a weighting based on the cosine similarity function to better address the mobile nature of the devices which may lead them to change the AP they connect to at every communication round they are available, see [7]. After a device i gets aggregated, it is removed from Tr (Line 22 in Alg. 1) and Tr gets reset every global aggregation (Line 29 in Alg. 1).

Selective Global Aggregation Finally, the device availability issue propagates to the AP level, since some APs, at certain time steps may not have any devices to aggregate; hence, we ask how we can perform the global aggregation in such a scenario. Since some APs will not have any update for the cloud, it makes sense to *not* aggregate them, preventing any communication with the cloud and thus saving energy and capacity for communication. We name this aggregation strategy *selective global aggregation* and we implement it using the subset of APs $\mathcal{A} \subseteq \mathbb{A}$ that performed at least one edge aggregation since the last global aggregation. We perform a selective global aggregation after k_2 dynamic edge aggregations. In Line 17 from Alg. 1, we save the APs that will be aggregated in the cloud, while in Line 28 we perform the selective global aggregation as follows:

$$\omega = \sum_{\alpha \in \mathcal{A}} q^{\alpha} \omega^{\alpha}(t), \text{ where } q^{\alpha} = \frac{e^{-\sigma \cos(\omega,\, \omega^{\alpha}(t))}}{\sum_{\alpha \in \mathcal{A}} e^{-\sigma \cos(\omega,\, \omega^{\alpha}(t))}} \tag{6}$$

where ω are the global model weights, σ is the same hyperparameter from Eq. 5, and *cos* is the cosine similarity function. Some APs (*e.g.*, from a dense area such as Downtown) may have aggregated many devices, while others may have aggregated very few devices. In order to not diverge too far from the global model, we weight the contributions of each AP based on the cosine similarity with the current global model weights. The cloud requests from all APs their updates, but, in the end, only the APs that performed at least one dynamic edge aggregation will actually send their updated models to the cloud for aggregation.

To summarize, we start with dynamic community selection and local training for k_1 local epochs. Then, we go to the next time step $t+1$, and since all devices changed their position and/or availability, we perform dynamic community selection for the available devices. On their newly selected APS, we run dynamic edge aggregation. We repeat this process k_2 times. Finally, we perform selective global aggregation with the APs that did at least one dynamic edge aggregation since the last selective global aggregation and send the updated global model to all APs α.

4 PERFORMANCE EVALUATION

4.1 Experimental Setup

We perform experiments using the Foursquare dataset [8] for the entire month of May 2020. Time t starts at May 1st 2020, 12:00AM UTC and ends at May 31st 2020 11:00PM UTC; we consider every hour from the month of May, summing up to 744 total time steps, thus having 744 communication rounds in total (*i.e.*, 31 days, each with 24 hours). We consider the difference between two consecutive time steps t and $t+1$ to be 1 hour. From 12,866 available devices, we select the top 1,000 that appear most times and from 37,994 APs we randomly select only 100 APs. We run each experiment three times with different seeds and report average values. All experiments are run using two GPU servers with 4×A6000 GPUs, 64 core AMD Threadripper PRO 3995WX CPU and 512GB RAM each. On each device, we use a simple convolutional model composed of one convolutional layer with 32 filters and MaxPooling, two convolutional blocks, each with two convolutional layers with 64 filters followed by MaxPooling and, finally, a fully connected layer with 512 neurons.

We use both independent and identically distributed (IID) and non-IID settings, similar to [10, 15], by controlling the α parameter from the Dirichlet distribution. We set $\alpha = 100$ for the IID scenario and $\alpha = 0.1$ for the non-IID scenario. We randomly sample from each class the number of images dictated by the Dirichlet distribution. Similar to [10, 15], we use 500 images per device for MNIST [14], CIFAR10 [13] and EMNIST [6] and we use 2500 images per device for CIFAR100 [13]. This enables in the IID case around 50 images per class for MNIST and CIFAR10, around 8 images per class for EMNIST and approximately 25 images per class for CIFAR100.

For HierFAVG, we consider (for all time steps) the same device-AP configuration, *i.e.*, we fix all devices with a pre-assigned AP. This forces every device to connect only to its pre-assigned AP irrespective of the distance between them (just as in [18]). For MACFL, at every time step, a device has a 50% probability of moving to one of the "neighboring" APs. We create a neighborhood of APs to match the setup from [7]. We connect AP_i to $AP_{i\pm1}$ such that every AP has two neighbors. Since APs are randomly selected for three different runs from approximatively 37,994 possibilities, AP_i could end up very far from $AP_{i\pm1}$. Since MACFL does not consider the real distance between the APs, the "virtual" assignment of APs is indeed matching the experimental setup provided in [7]. We consider the following hyperparameter values: $\sigma = 0.1$ (from Eq. 5 and Eq. 6) and learning rate 0.01. Following some ablation studies discussed in Section 4.3, we choose batch size 8 and $k_1 = 5$ for all experiments.

4.2 Empirical Results

Learning performance. The design choices of MOHAWK make our method better than other state-of-the-art HFL approaches. As shown in Table 3, for all four datasets running HFL (every hour for an entire month) we achieve very similar accuracy values for MNIST and better accuracy for all other datasets. We observe in some cases that the drop between IID and non-IID scenarios is

Table 3: Accuracies for IID and non-IID settings. MOHAWK obtains very similar or higher accuracy values in both settings.

Dataset		IID ($\alpha = 100$)			Non-IID ($\alpha = 0.1$)		
	k_2	2	5	10	2	5	10
MNIST	HierFAVG	99.12 ± 0.02	99.08 ± 0.09	99.06 ± 0.0	98.7 ± 0.04	98.69 ± 0.07	98.58 ± 0.03
	MACFL	99.12 ± 0.06	99.14 ± 0.01	99.16 ± 0.02	98.73 ± 0.08	98.72 ± 0.05	98.72 ± 0.03
	MOHAWK	**99.13 ± 0.06**	**99.17 ± 0.06**	**99.13 ± 0.0**	**99.15 ± 0.01**	**99.02 ± 0.06**	**98.88 ± 0.06**
EMNIST	HierFAVG	78.45 ± 0.16	78.26 ± 0.21	76.66 ± 0.48	77.07 ± 0.24	78.12 ± 0.23	76.6 ± 0.45
	MACFL	78.45 ± 0.033	78.5 ± 0.19	76.99 ± 0.47	77.43 ± 0.39	78.44 ± 0.24	76.98 ± 0.46
	MOHAWK	**78.76 ± 0.22**	**78.81 ± 0.22**	**78.02 ± 0.64**	**78.17 ± 0.24**	**78.64 ± 0.17**	**77.75 ± 0.4**
CIFAR10	HierFAVG	72.65 ± 0.61	69.69 ± 3.27	71.81 ± 0.44	67.09 ± 2.85	64.08 ± 0.12	63.28 ± 0.18
	MACFL	72.93 ± 0.58	72.91 ± 0.5 5	72.65 ± 0.53	65.36 ± 0.23	65.02 ± 0.33	64.65 ± 0.52
	MOHAWK	**78.69 ± 0.71**	**77.36 ± 0.62**	**75.8 ± 0.7**	**74.09 ± 0.66**	**70.82 ± 0.4**	**68.55 ± 0.76**
CIFAR100	HierFAVG	46.13 ± 0.25	45.74 ± 0.98	46.36 ± 0.73	45.67 ± 0.91	45.94 ± 0.37	45.88 ± 0.26
	MACFL	46.5 ± 0.07	46.61 ± 0.62	47.14 ± 0.5	46.19 ± 0.62	45.9 ± 0.23	46.67 ± 0.69
	MOHAWK	**46.81 ± 0.66**	**47.27 ± 1.00**	**47.55 ± 0.52**	**46.52 ± 0.32**	**47.16 ± 0.42**	**47.79 ± 0.56**

Table 4: Number of communication rounds required to achieve a threshold accuracy (Acc. thresh.) for both IID and non-IID settings. Overall, MOHAWK uses up to 3.88× less communication rounds.

Dataset		IID ($\alpha = 100$)			Non-IID ($\alpha = 0.1$)		
	k_2	2	5	10	2	5	10
MNIST	HierFAVG	46	50	70	162	185	210
Acc. thresh. 97%	MACFL	42	45	50	142	165	180
	MOHAWK	**12**	**20**	**30**	**40**	**75**	**110**
	Avg. improvement	**3.67×**	**2.38×**	**2×**	**3.8×**	**2.33×**	**1.77×**
EMNIST	HierFAVG	162	190	455	338	200	470
Acc. thresh. 75%	MACFL	150	165	335	330	170	410
	MOHAWK	**44**	**85**	**160**	**118**	**145**	**210**
	Avg. improvement	**3.55×**	**2.09×**	**2.47×**	**2.83×**	**1.28×**	**2.1×**
CIFAR10	HierFAVG	212	288	220	358	500	530
Acc. thresh. 60%	MACFL	192	210	210	428	455	510
	MOHAWK	**52**	**100**	**150**	**138**	**240**	**330**
	Avg. improvement	**3.88×**	**2.49×**	**1.43×**	**2.85×**	**1.99×**	**1.58×**
CIFAR100	HierFAVG	502	605	500	614	590	620
Acc. thresh. 45%	MACFL	422	455	390	524	595	520
	MOHAWK	**142**	**215**	**280**	**214**	**330**	**360**
	Avg. improvement	**3.25×**	**2.47×**	**1.59×**	**2.66×**	**1.8×**	**1.58×**

smaller for MOHAWK than for HierFAVG or MACFL. For CIFAR10 and $k_2 = 2$, the drop in accuracy between IID and non-IID for HierFAVG is 7.89%, for MACFL is 7.73%, while for MOHAWK is only 4.77%. This shows that MOHAWK has a better robustness against data heterogeneity. In our experiments, we observe that, for the same hyperparameters, when we increase k_2, the performance degrades for both baselines and MOHAWK. This confirms that even for real mobility and availability of devices, the findings from [7, 18] still hold: frequent edge aggregations (*e.g.*, $k_2 = 2$) are beneficial to the learning performance in HFL.

In Table 4, we show how much faster MOHAWK converges to a certain accuracy threshold when compared to the baselines. As highlighted in Table 4, for any given k_2 values MOHAWK manages to speedup convergence at least by 1.43× and up to 3.88×. The reason for such good convergence rates against state-of-the-art HFL solutions is the adaptation to real mobility and availability. By using dynamic edge aggregation and selective global aggregation, MOHAWK has the upper hand in every scenario. This also shows how real-world mobility and availability of devices impacts the current state-of-the-art HFL. Thus, we show the importance and need for mobility-aware HFL solutions like MOHAWK.

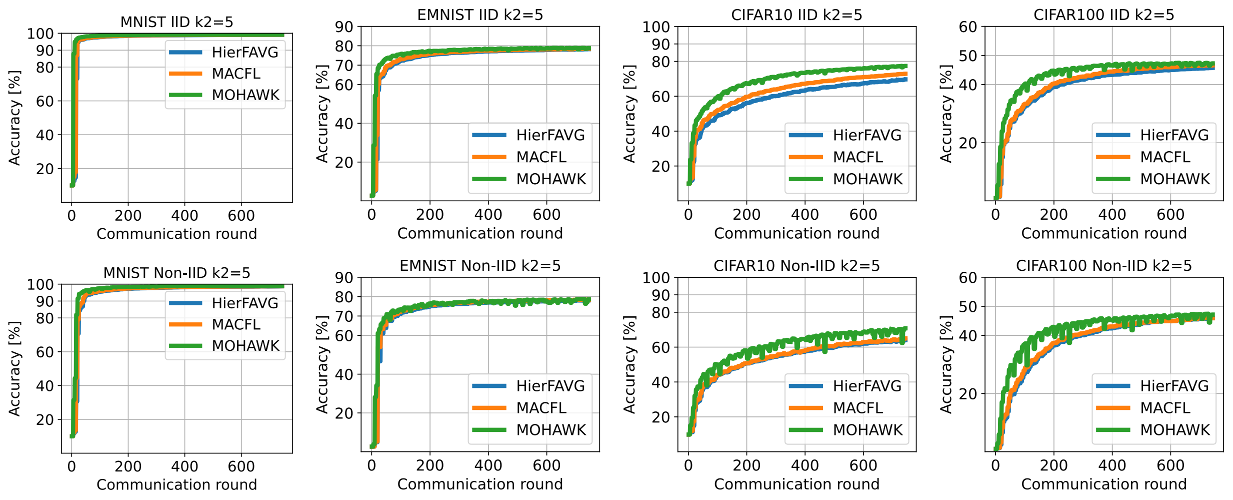

Figure 5: Accuracy results using IID and non-IID settings for $k_2 = 5$. MOHAWK converges faster and obtains a higher accuracy value in both IID and non-IID scenarios, for all datasets.

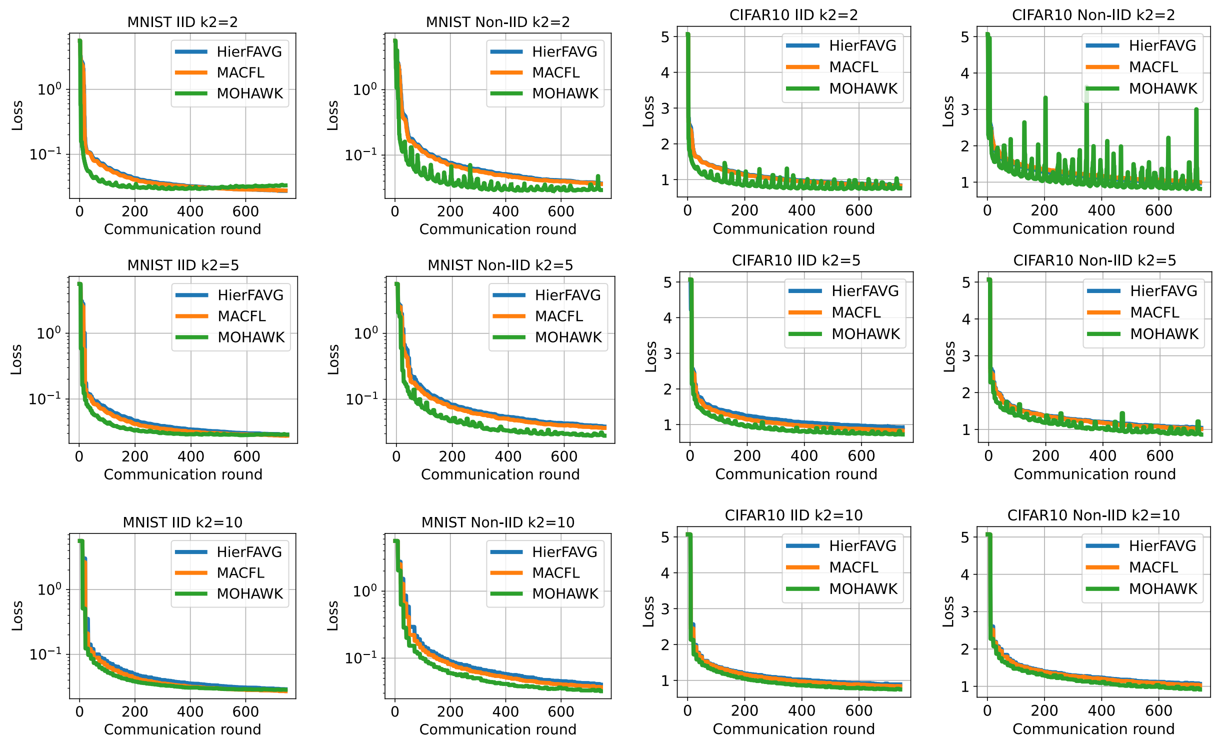

Figure 6: Global test loss for MOHAWK and various baselines. We observe that for very small k_2 values, *i.e.*, $k_2 = 2$, the global test loss may have a bigger variability since the number of aggregated APs is very low. The baselines aggregate all 100APs, even those that did not train at all, at the expense of a slower, but more stable convergence.

Table 5: Number of devices available at every time step and the average distance between devices and their selected APs. MOHAWK improves the device-AP distance by 7.74×, while including up to 39.78% more devices in the learning process.

Metric	k_2	HierFAVG	MACFL	MOHAWK	**Avg. Improv.**
Avg. number of devices available at every time step	2	15	15	18	**+12.84%**
	5	15	15	22	**+29.69%**
	10	15	15	26	**+39.78%**
Avg. distance [km]	-	7.08 ± 3.92	7.01 ± 3.83	0.91 ± 0.75	**7.74×**

Table 6: Average energy spent for communication per client to achieve a certain accuracy threshold on MNIST and CIFAR10 (in simulation) for both IID and non-IID settings. Overall, MOHAWK uses up to 3.87× less energy for communication to achieve a given accuracy threshold, *i.e.*, 97% for MNIST and 60% for CIFAR10.

Dataset	Data type	Model	k_2		
			2	5	10
MNIST Acc. thresh. 97%	IID (α = 100)	HierFAVG	602.97 ± 3.23 J	654.56 ± 3.72 J	892.31 ± 3.84 J
		MACFL	575.13 ± 1.8 J	588.46 ± 1.9 J	653.37 ± 2.74 J
		MOHAWK	**152.41 ± 2.91 J**	**263.47 ± 2.83 J**	**389.22 ± 4.11 J**
		Avg. improvement	**3.86×**	**2.36×**	**1.99×**
CIFAR10 Acc. thresh. 60%	IID (α = 100)	HierFAVG	2,659.61 ± 21.07 J	3,606.83 ± 25.54 J	2,764.25 ± 22.0 J
		MACFL	2,416.3 ± 8.79 J	2,629.54 ± 8.29 J	2,629.54 ± 8.29 J
		MOHAWK	**655.16 ± 6.13 J**	**1,250.28 ± 13.19 J**	**1,881.72 ± 17.53 J**
		Avg. improvement	**3.87×**	**2.49×**	**1.43×**
MNIST Acc. thresh. 97%	Non-IID (α = 0.1)	HierFAVG	2,051.28 ± 18.4 J	2,330.4 ± 19.64 J	2,633.58 ± 21.17 J
		MACFL	1,787.82 ± 4.53 J	2,087.03 ± 7.42 J	2,259.74 ± 8.49 J
		MOHAWK	**502.73 ± 5.44 J**	**943.85 ± 8.53 J**	**1,374.2 ± 12.45 J**
		Avg. improvement	**3.82×**	**2.34×**	**1.78×**
CIFAR10 Acc. thresh. 60%	Non-IID (α = 0.1)	HierFAVG	4,461.25 ± 34.07 J	6,242.22 ± 43.11 J	6,629.33 ± 44.98 J
		MACFL	5,325.98 ± 11.64 J	5,665.64 ± 12.3 J	6,356.33 ± 7.43 J
		MOHAWK	**1,704.53 ± 16.8 J**	**2,974.56 ± 29.87 J**	**4,143.05 ± 37.7 J**
		Avg. improvement	**2.87×**	**2×**	**1.57×**

In Fig. 5, we show the global test accuracy for both IID and non-IID scenarios (k_2 = 5 for all datasets). Overall, we can see higher accuracy levels and faster convergence over all datasets and data heterogeneity scenarios. In Fig. 6, for all k_2 values and data heterogeneity scenarios, we show the global test loss for MNIST and CIFAR10. We observe that for very small k_2 values there is more variability in the global test loss for MOHAWK due to the selective global aggregation. Since at night there are very few to no devices available, we aggregate just a few devices and then update the global model based on a small number of APs (since only a few of them do edge aggregations). This issue disappears with higher k_2 values since we allow more time for APs to perform edge aggregations (and hence, be considered for the selective global aggregation). The faster convergence due to the selective aggregation is also clearly visible for all k_2 values in Fig. 6.

Communication efficiency. For simulation, we estimate the energy consumption for communication using the models described in Section 3.2. In Table 5, we observe the increase in device availability compared to existing methods. When real-world mobility and device availability are present, our dynamic edge aggregation considers up to 39.78% more devices for aggregation (k_2 = 10). We can also see the dynamic community selection reduces the average distance between APs and the devices that connect to them by 7.74×. This forms communities of tightly grouped devices and their AP, thus allowing for faster communication between them. As it can be seen in Table 6, compared to state-of-the-art HFL solutions, MOHAWK improves the communication efficiency for HFL under mobile and heterogeneous environments by up to 3.87×. In Fig. 7, we show that to achieve a given accuracy threshold, the amount of energy spent for communication is drastically reduced (up to 3.87×) compared to current state-of-the-art approaches.

4.3 Ablation studies

Batch size variation. We perform an ablation study with k_2 = 5 to determine the best batch size to run MOHAWK with. For this, we fix k_1 = 1 to run the ablation experiments faster. We evaluate four different batch sizes on all datasets. As can be seen in Table 7, the best batch size by far, for all datasets, is the batch size of 8.

Variation of k_1 and k_2. Using k_1 = 1 results in lower accuracies for CIFAR10 and CIFAR100 so we followed up with another ablation

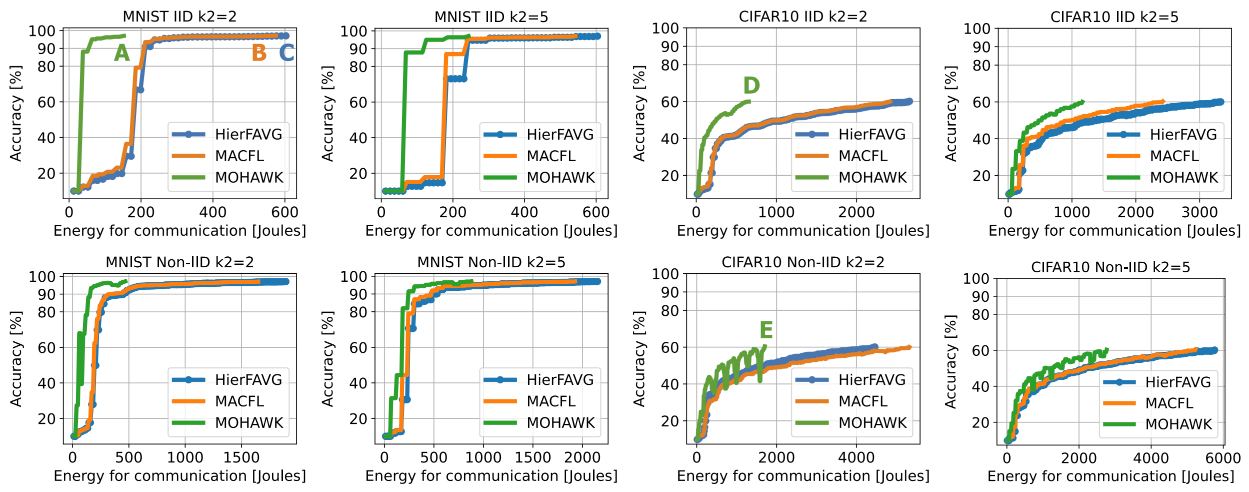

Figure 7: Average energy spent for communication [Joules] until reaching an accuracy threshold of 97% on MNIST and 60% on CIFAR10, with different k_2 values under data heterogeneity constraints. For MNIST, IID with $k_2 = 2$, MOHAWK achieves the required accuracy threshold A using up to 3.87× less energy for communication compared to the baselines B and C. Similar considerations for CIFAR10 IID (D) and non-IID (E).

Table 7: Ablation study for MOHAWK accuracy [%] using different batch sizes for $k_1 = 1$ and $k_2 = 5$. We observe that the best results over all datasets are for batch size 8.

	IID ($\alpha = 100$)				Non-IID ($\alpha = 0.1$)			
Batch size	64	32	16	8	64	32	16	8
MNIST	98.49	98.85	98.86	99.08	97.88	98.41	98.7	98.89
EMNIST	75.70	76.56	78.83	79.46	72.71	74.78	76.83	77.84
CIFAR10	50.67	64.93	68.88	60.7	44.55	47.95	52.13	55.88
CIFAR100	15.68	17.5	19.69	22.06	16.46	18.76	23.4	26.48

Table 8: Ablation study for MOHAWK accuracy [%] using different k_1 and k_2 values with batch size 64. We observe a big jump in accuracy from $k_1 = 1$ to $k_1 = 5$, hence we use $k_1 = 5$ in the main experiments for both MNIST and CIFAR10.

Dataset		IID ($\alpha = 100$)			Non-IID ($\alpha = 0.1$)		
	k_2	2	5	10	2	5	10
MNIST	$k_1 = 1$	98.74	98.49	98.32	98.4	97.9	97.45
	$k_1 = 5$	98.86	98.83	98.91	98.69	98.45	98.35
	$k_1 = 10$	98.91	98.79	98.69	98.75	98.44	98.39
CIFAR10	$k_1 = 1$	55.43	50.67	48.24	49.86	44.55	41.31
	$k_1 = 5$	73.76	68.36	64.97	64.22	57.62	53.85
	$k_1 = 10$	76.74	74.29	71.96	69.6	64.13	60.69

Table 9: Ablation study on MOHAWK scalability for larger, more realistic numbers of devices and APs on CIFAR10 with $k_1 = 5$ and $k_2 = 5$. Overall, we observe MOHAWK provides similar performance in all scenarios, while other approaches have large performance reductions.

Data	Devices	APs	HierFAVG	MACFL	MOHAWK
IID ($\alpha = 100$)	1,000	100	72.4	73.67	**78.22**
	1,000	500	58.24	58.61	**76.88**
	1,000	1,000	52.2	52.34	**76.82**
	10,000	1,000	56.53	57.21	**77.95**
Non-IID ($\alpha = 0.1$)	1,000	100	62.23	65.26	**71.09**
	1,000	500	50.85	51.0	**69.78**
	1,000	1,000	45.22	45.5	**69.29**
	10,000	1,000	48.47	49.62	**70.88**

study. We explore for MNIST and CIFAR10 different k_1 local epochs and k_2 dynamic edge aggregations to see which values work better. The jump in accuracy improvement from $k_1 = 1$ to $k_1 = 5$ proves to be much larger than the jump in accuracy between $k_1 = 5$ and $k_1 = 10$ (see Table 8). Since we consider real edge devices and the local updates happen every hour, we assume for simulations $k_1 = 5$ as a fair middle ground to use. Considering hardware heterogeneity, some low-budget devices may take up to one hour to train 5 local epochs, hence this also accounts for realistic training times.

Table 10: Hardware prototype experiment for $k_2 = 5$ in the IID setting using $k_1 = 1$ for MNIST and $k_1 = 5$ for CIFAR10. The wasted energy is the energy a device spends training and communicating with the cloud without the cloud actually considering the local model for aggregation. We observe MOHAWK not only uses far less energy overall, but it also wastes less energy.

Energy [Joules]	MNIST IID ($\alpha = 100$)			CIFAR10 IID ($\alpha = 100$)		
	HierFAVG	MOHAWK	Improv.	HierFAVG	MOHAWK	Improv.
Cumulative wasted /experiment	6,893,090 J	2,690,005 J	2.56×	37,021,657 J	14,969,761 J	2.47×
Cumulative consumed /experiment	10,563,326 J	7,707,298 J	1.37×	43,893,178 J	40,041,313 J	1.1×
Average wasted /device	94 ± 39 J	42 ± 23 J	2.24×	475 ± 112 J	243 ± 130 J	1.95×
Average consumed /device	137 ± 34 J	87 ± 12 J	1.57×	548 ± 100 J	470 ± 23 J	1.17×
Average wasted /comm. round	514 ± 464 J	121 ± 135 J	4.25×	2,615 ± 2,082 J	668 ± 610 J	3.91×
Average consumed /comm. round	727 ± 644 J	612 ± 520 J	1.19×	3,165 ± 2,325 J	3,262 ± 2,445 J	0.97×

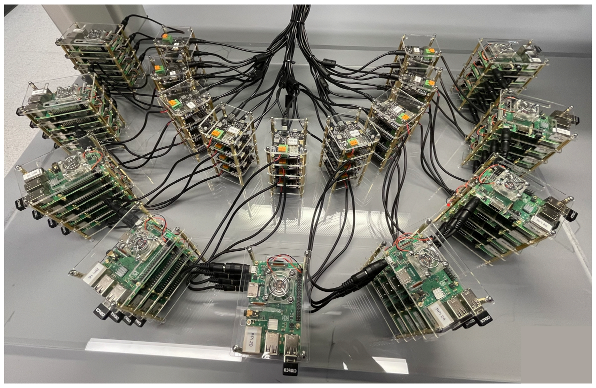

Figure 8: Hardware prototype with 36 Raspberry Pi 3B+ (outer semicircle of devices) and 36 Smart Power 2 devices (inner semicircle) used for real-time power and energy measurements.

Scalability of MOHAWK. We show in Table 9, using larger numbers of APs and devices, how scalable is MOHAWK. We observe similar performance in terms of accuracy on CIFAR10 for both IID and non-IID settings, while all other HFL solutions have a decrease in accuracy as we increase the number of APs. This is because MOHAWK uses selective global aggregation, which makes it robust to variabilities in the number of APs considered.

4.4 Hardware Prototyping and Validation

As can be seen in Fig. 8, we designed and built a custom testbed with 36 Raspberry Pi 3B+ devices. Each of the Raspberry Pi devices is connected to a Smart Power 2 device for real-time power and energy measurements. We run FL using a GPU server and communicate through wireless using a local router. We measure the total amount of energy spent in a 36 device experiment using 20APs. We use the GPU server to run the global server and the 20 APs.

We consider a device is *wasting energy* if it is training a model which is not ultimately aggregated. Thus, such devices are not improving the learning performance of the FL system, but are wasting their already limited resources (*e.g.*, battery, memory). To the best of our knowledge, no current HFL state-of-the-art methods account for this kind of wasted energy. This is why, on our hardware prototype, we run only HierFAVG as a baseline. We use $k_1 = 1$ for MNIST and $k_1 = 5$ for CIFAR10, running only the IID setting on both datasets. As seen in Table 10, MOHAWK achieves up to 4.25× less energy wasted (on average) per communication round and up to 2.24× less energy wasted (on average) per device. The total energy wasted over the entire experiment is reduced up to 2.56×; this shows how much more energy-efficient MOHAWK is in real scenarios. The energy measurements represent the energy spent on both training and communication while performing FL.

5 CONCLUSION

We have proposed a Mobility and Heterogeneity-Aware Dynamic Community Selection algorithm (MOHAWK) for mobile federated learning systems. Our approach takes into consideration the real devices mobility and selects the closest access point for every device to connect; this leads to significant reduction in the energy consumption for communication. To improve the learning performance, we have proposed two new aggregation strategies, namely, dynamic edge aggregation and selective global aggregation, that increase the number of devices aggregated at every time step by up to 39.78%; this also helps the global model learn up to 3.88× faster, while also achieving a higher accuracy, on average.

Limitations and future work. The current communication model can be improved in several ways, *e.g.*, by considering channel scheduling, and multi-hop networks of APs. For dynamic edge aggregation, we currently consider all available devices regardless of the quality and security vulnerability of their local model. A more robust and secure selection of devices could improve the overall performance. All these ideas are left for future work.

ACKNOWLEDGMENTS

This research was supported in part by NSF Grant CCF-2107085 and in part by Cisco Research, Inc. The work of G. de Veciana was supported by NSF Grant No. 2148224 which is supported in part by funds from OUSD R&E, NIST, and industry partners as specified in the Resilient & Intelligent NextG Systems (RINGS) program.

REFERENCES

[1] Mehdi Salehi Heydar Abad, Emre Ozfatura, Deniz Gunduz, and Ozgur Ercetin. 2020. Hierarchical federated learning across heterogeneous cellular networks. In *ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 8866–8870.

[2] Sawsan AbdulRahman, Hanine Tout, Hakima Ould-Slimane, Azzam Mourad, Chamseddine Talhi, and Mohsen Guizani. 2020. A survey on federated learning: The journey from centralized to distributed on-site learning and beyond. *IEEE Internet of Things Journal* 8, 7 (2020), 5476–5497.

[3] Tracy Camp, Jeff Boleng, and Vanessa Davies. 2002. A survey of mobility models for ad hoc network research. *Wireless communications and mobile computing* 2, 5 (2002), 483–502.

[4] Liane Cassavoy. 2021. *How Fast Is 4G LTE Wireless Service?* Accessed: 2023-02-26.

[5] Wei Chen, Kartikeya Bhardwaj, and Radu Marculescu. 2021. Fedmax: mitigating activation divergence for accurate and communication-efficient federated learning. In *Machine Learning and Knowledge Discovery in Databases: European Conference, ECML PKDD 2020, Ghent, Belgium, September 14–18, 2020, Proceedings, Part II*. Springer, 348–363.

[6] Gregory Cohen, Saeed Afshar, Jonathan Tapson, and Andre Van Schaik. 2017. EMNIST: Extending MNIST to handwritten letters. In *2017 international joint conference on neural networks (IJCNN)*. IEEE, 2921–2926.

[7] 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.

[8] Foursquare. 2023. *Independent Location Data & Location Technology Platform*. Accessed: 2023-02-26.

[9] Juan C Herrera, Daniel B Work, Ryan Herring, Xuegang Jeff Ban, Quinn Jacobson, and Alexandre M Bayen. 2010. Evaluation of traffic data obtained via GPS-enabled mobile phones: The Mobile Century field experiment. *Transportation Research Part C: Emerging Technologies* 18, 4 (2010), 568–583.

[10] Tzu-Ming Harry Hsu, Hang Qi, and Matthew Brown. 2019. Measuring the effects of non-identical data distribution for federated visual classification. *arXiv preprint arXiv:1909.06335* (2019).

[11] Junxian Huang, Feng Qian, Alexandre Gerber, Z Morley Mao, Subhabrata Sen, and Oliver Spatscheck. 2012. A close examination of performance and power characteristics of 4G LTE networks. In *Proceedings of the 10th international conference on Mobile systems, applications, and services*. 225–238.

[12] Yutao Huang, Lingyang Chu, Zirui Zhou, Lanjun Wang, Jiangchuan Liu, Jian Pei, and Yong Zhang. 2021. Personalized cross-silo federated learning on non-iid data. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 35. 7865–7873.

[13] Alex Krizhevsky, Vinod Nair, and Geoffrey Hinton. 2014. The CIFAR-10 dataset. *online: http://www. cs. toronto. edu/kriz/cifar. html* 55, 5 (2014).

[14] Yann LeCun. 1998. The MNIST database of handwritten digits. *http://yann. lecun. com/exdb/mnist/* (1998).

[15] Qinbin Li, Yiqun Diao, Quan Chen, and Bingsheng He. 2022. Federated learning on non-iid data silos: An experimental study. In *2022 IEEE 38th International Conference on Data Engineering (ICDE)*. IEEE, 965–978.

[16] Tian Li, Anit Kumar Sahu, Ameet Talwalkar, and Virginia Smith. 2020. Federated learning: Challenges, methods, and future directions. *IEEE signal processing magazine* 37, 3 (2020), 50–60.

[17] 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.

[18] Lumin Liu, Jun Zhang, SH Song, and Khaled B Letaief. 2020. Client-edge-cloud hierarchical federated learning. In *ICC 2020-2020 IEEE International Conference on Communications (ICC)*. IEEE, 1–6.

[19] Lumin Liu, Jun Zhang, Shenghui Song, and Khaled B Letaief. 2022. Hierarchical federated learning with quantization: Convergence analysis and system design. *IEEE Transactions on Wireless Communications* (2022).

[20] Yi Liu, Xingliang Yuan, Zehui Xiong, Jiawen Kang, Xiaofei Wang, and Dusit Niyato. 2020. Federated learning for 6G communications: Challenges, methods, and future directions. *China Communications* 17, 9 (2020), 105–118.

[21] Bing Luo, Xiang Li, Shiqiang Wang, Jianwei Huang, and Leandros Tassiulas. 2021. Cost-effective federated learning in mobile edge networks. *IEEE Journal on Selected Areas in Communications* 39, 12 (2021), 3606–3621.

[22] 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.

[23] Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, and Blaise Aguera y Arcas. 2017. Communication-efficient learning of deep networks from decentralized data. In *Artificial intelligence and statistics*. PMLR, 1273–1282.

[24] Hideya Ochiai and Hiroshi Esaki. 2008. Mobility entropy and message routing in community-structured delay tolerant networks. In *Proceedings of the 4th Asian Conference on Internet Engineering*. 93–102.

[25] Hideya Ochiai, Yuwei Sun, Qingzhe Jin, Nattanon Wongwiwatchai, and Hiroshi Esaki. 2022. Wireless ad hoc federated learning: A fully distributed cooperative machine learning. *arXiv preprint arXiv:2205.11779* (2022).

[26] Xiaomin Ouyang, Zhiyuan Xie, Jiayu Zhou, Jianwei Huang, and Guoliang Xing. 2021. Clusterfl: a similarity-aware federated learning system for human activity recognition. In *Proceedings of the 19th Annual International Conference on Mobile Systems, Applications, and Services*. 54–66.

[27] Md Ferdous Pervej, Jianlin Guo, Kyeong Jin Kim, Kieran Parsons, Philip Orlik, Stefano Di Cairano, Marcel Menner, Karl Berntorp, Yukimasa Nagai, and Huaiyu Dai. 2022. Mobility, Communication and Computation Aware Federated Learning for Internet of Vehicles. In *2022 IEEE Intelligent Vehicles Symposium (IV)*. IEEE, 750–757.

[28] Jianyu Wang, Qinghua Liu, Hao Liang, Gauri Joshi, and H Vincent Poor. 2020. Tackling the objective inconsistency problem in heterogeneous federated optimization. *Advances in neural information processing systems* 33 (2020), 7611–7623.

[29] Xinlei Yang, Hao Lin, Zhenhua Li, Feng Qian, Xingyao Li, Zhiming He, Xudong Wu, Xianlong Wang, Yunhao Liu, Zhi Liao, et al. 2022. Mobile access bandwidth in practice: Measurement, analysis, and implications. In *Proceedings of the ACM SIGCOMM 2022 Conference*. 114–128.