Ground-Assisted Federated Learning in LEO Satellite Constellations

Nasrin Razmi, *Student Member, IEEE*, Bho Matthiesen, *Member, IEEE*, Armin Dekorsy, *Senior Member, IEEE*, and Petar Popovski, *Fellow, IEEE*

***Abstract*—In Low Earth Orbit (LEO) mega constellations, there are relevant use cases, such as inference based on satellite imaging, in which a large number of satellites collaboratively train a machine learning model without sharing their local datasets. To address this problem, we propose a new set of algorithms based on Federated learning (FL), including a novel asynchronous FL procedure based on FedAvg that exhibits better robustness against heterogeneous scenarios than the state-of-the-art. Extensive numerical evaluations based on MNIST and CIFAR-10 datasets highlight the fast convergence speed and excellent asymptotic test accuracy of the proposed method.**

***Index Terms*—Satellite communication, Low Earth Orbit (LEO), Federated Optimization**

I. Introduction

Constellations of small satellites flying in Low Earth Orbit (LEO) are a cost-efficient and versatile alternative to traditional big satellites in medium Earth and geostationary orbits. Several of these constellations are currently deployed with the goal of providing ubiquitous connectivity and low latency Internet service [1]. Their integration into terrestrial mobile networks is an active research area, covering various use cases such as Earth observation missions [2]–[6]. Presumably, machine learning (ML) will become an essential tool to manage these constellations and utilize their sensor measurements [7]–[9].

The traditional approach to ML is to aggregate all data in a central location and then solve the learning problem. Considering the vast amounts of data necessary to train deep neural networks [10], this involves high transmission costs and delays. Moreover, considering the emergence of variety of private owners of small satellites, it might be prohibited to share the data due to privacy or data ownership concerns. The obvious solution to this dilemma is to train locally and aggregate the derived model parameters only. This is achieved by solving the ML problem collaboratively and only sharing updated model parameters. The distributed ML paradigm taking data heterogeneity and limited connectivity into account is known as federated learning (FL) [11], [12]. Applying distributed ML to satellite constellations is only natural when considering the general trend towards edge computing [13]. For example, in ESA's PhiSat-1 mission, raw Earth observation data is pre-processed using deep ML models on the satellites and only relevant information is transmitted to the ground [8], [9]. Since the raw data remains on the satellites, improving the employed ML models based on new observations requires on-board re-training.

A core assumption of the general FL setting is intermittent and unpredictable participation of the clients, i.e., the satellites in the considered scenario. In order to cope with that, asynchronous algorithms have been proposed recently [14]. However, the distinctive feature of the LEO learning scenario is the predictable availability of clients combined with very long periods between visits to the same ground station (GS). In this paper, we investigate how this predictive availability impacts the FL scenario when the training process is orchestrated by a GS and propose a novel asynchronous algorithm. We conclusively show that our approach leads to superior training performance when compared to state-of-the-art FL algorithms. In particular, our key contributions are that we: 1) define the LEO FL scenario and identify core challenges compared to conventional FL; 2) propose an algorithmic framework and communication protocol for satellite FL; 3) adapt FedAvg [12] and FedAsync [14] to this scenario and propose a novel asynchronous variant of FedAvg that is particularly well suited for ground-assisted FL in satellite constellations, and 4) numerically evaluate the discussed algorithms to verify our theoretical considerations. These results show that the proposed asynchronous FL algorithm has higher robustness against heterogeneous scenarios than FedAsync.

II. System Model and Background on FL

Consider a LEO constellation of K satellites in L orbital planes. In an Earth-centered inertial coordinate system, satellite k, $k \in \mathcal{K} = \{1, \dots, K\}$, has trajectory $\boldsymbol{r}_k(t)$ and the GS, although fixed in a constant location on Earth, has trajectory $\boldsymbol{r}_g(t)$. A ground to satellite link is feasible if satellite k is visible from the GS at a minimum elevation angle α_e, i.e., $\frac{\pi}{2} - \angle(\boldsymbol{r}_g(t), \boldsymbol{r}_k(t) - \boldsymbol{r}_g(t)) \geq \alpha_e$. In general, only a subset of satellites is connected to the GS at once and the time between contacts is much longer than the actual online time.

Satellite k collects data from its on-board instruments and stores it in a dataset $\mathcal{D}_k$. Due to different orbits and orbital positions, the datasets of two distinct satellites are disjunct and possibly non-IID. After the data collection phase, the satellites collaboratively solve an optimization problem of the form

$$\min_{\boldsymbol{\theta}\in\mathbb{R}^d} \frac{1}{n} \sum_{\boldsymbol{x}\in\mathcal{D}} f(\boldsymbol{x};\boldsymbol{\theta}) = \min_{\boldsymbol{\theta}\in\mathbb{R}^d} \sum_{k\in\mathcal{K}} \frac{n_k}{n} \sum_{\boldsymbol{x}\in\mathcal{D}_k} \frac{1}{n_k} f(\boldsymbol{x};\boldsymbol{\theta}) \qquad (1)$$

N. Razmi, B. Matthiesen, and A. Dekorsy are with the Gauss-Olbers Center, c/o University of Bremen, and the Department of Communications Engineering, University of Bremen, 28359 Bremen, Germany (e-mail: {razmi,matthiesen,dekorsy}@ant.uni-bremen.de). P. Popovski is with the Department of Electronic Systems, Aalborg University, 9100 Aalborg, Denmark (e-mail: petarp@es.aau.dk). P. Popovski is also holder of the U Bremen Excellence Chair in the Department of Communications Engineering, University of Bremen, 28359 Bremen, Germany.

This work is supported in part by the German Research Foundation (DFG) under Germany's Excellence Strategy (EXC 2077 at University of Bremen, University Allowance) and by the North-German Supercomputing Alliance.

Algorithm 1 Worker SGD Procedure

```
1: Receive $(\boldsymbol{\theta}^i, i)$ from the PS
2: $\boldsymbol{\theta}_k^{i,0} \leftarrow \boldsymbol{\theta}^i, \quad j \leftarrow 0$
3: while stopping criterion not met do
4:     $\tilde{\mathcal{D}}_k \leftarrow$ Randomly shuffle $\mathcal{D}_k$
5:     $\mathscr{B} \leftarrow$ Partition $\tilde{\mathcal{D}}_k$ into minibatches of size $B$
6:     for each batch $\mathcal{B} \in \mathscr{B}$ do
7:         $\boldsymbol{\theta}_k^{\tau,j+1} \leftarrow \boldsymbol{\theta}_k^{i,j} - \eta \nabla_{\boldsymbol{\theta}}\, g_{\boldsymbol{\theta}^i}(\mathcal{B}; \boldsymbol{\theta}_k^{i,j})$        ▷ cf. (2)
8:         $j \leftarrow j + 1$
9:     end for
10: end while
11: Push $(\boldsymbol{\theta}_k^{i,j}, i)$ to the PS
```

with the goal of training a machine learning model, where $\mathcal{D} = \bigcup_{k\in\mathcal{K}} \mathcal{D}_k \subset \mathbb{R}^m$, $n_k = |\mathcal{D}_k|$, and $n = \sum_{k\in\mathcal{K}} n_k$. The objective $\frac{1}{n}\sum_{\boldsymbol{x}\in\mathcal{D}} f(\boldsymbol{x};\boldsymbol{\theta})$ is an empirical loss function defined by the training task, where $f(\boldsymbol{x};\boldsymbol{\theta})$ is the training loss for a data point $\boldsymbol{x} \in \mathcal{D}$ and model parameters $\boldsymbol{\theta}$ with dimension d. This process is orchestrated by the GS and performed iteratively without sharing datasets between satellites. We assume that the satellites have very limited computational resources available for the solution of (1). Hence, we consider the case where the satellites work on (1) between visits to the GS and use the contact time to do an exchange of model parameters $\boldsymbol{\theta}$.

A. Federated Learning Background

Solving the ML training problem (1) distributedly under the assumptions of intermittent connectivity, heterogeneous datasets, and without sharing local raw data is known as FL. The most widely employed approach to this problem is the FedAvg algorithm [12]. A parameter server (PS) manages the learning process and keeps a global version of the current model parameters $\boldsymbol{\theta}^i$ to be learned. In each global iteration i, the "epoch", the PS selects a subset $\mathcal{S}_i$ of available workers to participate in the next round. It transmits the current version of the global model to the selected workers and then waits for all of them to return their results.

The workers perform one or multiple iterations of minibatch stochastic gradient descent (SGD) over their local dataset. In particular, in each local epoch the local dataset is partitioned in $\lceil \frac{n_k}{B} \rceil$ random batches $\mathcal{B} \in \mathscr{B}$ of size $B = |\mathcal{B}|$ and, for each minibatch, a SGD step is performed with learning rate η [10], [12]. The loss function is based on (1) and defined as

$$g_{\boldsymbol{\theta}'}(\mathcal{B};\boldsymbol{\theta}) = |\mathcal{B}|^{-1} \sum\nolimits_{\boldsymbol{x}\in\mathcal{B}} f(\boldsymbol{x};\boldsymbol{\theta}) + \tilde{g}_{\boldsymbol{\theta}'}(\mathcal{B};\boldsymbol{\theta}) \tag{2}$$

where $\tilde{g}$ is an optional regularization term [15]. Upon termination, the updated local model parameters are transmitted to the PS. The whole procedure is given in Algorithm 1.

After receiving results from all scheduled workers, the PS aggregates the results into a new version of the global model

$$\boldsymbol{\theta}^{i+1} = \sum_{k\in\mathcal{S}_i} \frac{n_k}{\sum_{k\in\mathcal{S}_i} n_k} \boldsymbol{\theta}_k^i, \tag{3}$$

where $\boldsymbol{\theta}_k^i$ are the local model parameters of worker k, and $\boldsymbol{\theta}^{i+1}$ is the new set of global model parameters. After this aggregation step, the PS starts the next epoch.

This is known as synchronous FL and can lead to slow convergence speed if the PS has to wait for stragglers. One way to address this problem is to incorporate client updates whenever they arrive in an asynchronous fashion. Such an algorithm was first published in [14] under the name FedAsync and is shown to outperform FedAvg in some cases. While the client operation in FedAsync is as in Algorithm 1, the PS operates differently and periodically assigns computing tasks to some workers by transmitting the current version of the global model parameters along with the epoch. Client updates are incorporated asynchronously as they arrive. In particular, the update from client k in epoch i is incorporated as

$$\boldsymbol{\theta}^{i+1} = (1-\alpha)\boldsymbol{\theta}^i + \alpha\boldsymbol{\theta}_k^i \tag{4}$$

where the mixing factor $\alpha \in (0,1)$ determines how much weight is given to incoming client updates. This factor is determined as $\alpha = \alpha' \cdot s(i - \tau_k)$, where α' is a fixed base weight, i is the current epoch, τ is the epoch the worker received the global model, and $s(i) \in (0,1]$ is a problem-specific staleness function that may be used to reduce the weight given to updates based on older version of the global model. The rationale behind this is that such updates are likely to introduce an error into the solution as the global model parameters have already advanced further towards the solution.

III. Federated Learning on Satellites

The FL algorithms discussed in Section II-A were designed under the premise that device availability is driven by a random process and that parallel communication is possible without significant delay. However, the satellite scenario is fundamentally different in several aspects: the number of workers is a few magnitudes smaller than in terrestrial applications, devices are always available for computation tasks, but communication is only possible during a small and highly predictable time window. In addition, at each time instant only a very small fraction of workers is within range of communication.

While this scenario is best addressed by an asynchronous FL algorithm, we also consider the synchronous FedAvg algorithm as baseline. We first outline the communication protocol, define the satellite operation, and discuss the application of FedAvg and FedAsync to the satellite scenario. In the next section, we design a novel asynchronous algorithm that leverages the predictable connectivity of satellite communications to implement FedAvg without unnecessary delays.

A. Communication Protocol and Satellite Operation

Communication is implemented in a client server protocol, where all connections are initiated by the satellite. Whenever the satellite is not working on a communication task, it tries to contact the GS. Hence, communication is either initiated when the GS comes within communication range or directly upon completion of a communication task. Upon connection, satellite k transmits a local model parameter update $\boldsymbol{\theta}_k^i$ if one is available and was not previously sent, where i denotes the current global epoch. Then, the GS updates the global model parameters $(\boldsymbol{\theta}^i, \boldsymbol{\theta}_k^i) \mapsto \boldsymbol{\theta}^{i+1}$ and decides whether satellite k should continue computation. If true, the GS transmits the updated global parameter vector to satellite k and terminates the connection. Otherwise, the connection is terminated and the satellite does not reestablish connection during this pass.

The computation task on the satellite is described in Algorithm 1. To avoid large deviations from the global model due to asynchronous operation and long delays between GS contacts,

Algorithm 2 Synchronous Ground Station Operation (FedAvg)

```
 1: Initialize epoch $i = 0$, model $\boldsymbol{\theta}^1$, wall time $t$
 2: while stopping criterion not met do
 3:     $i \leftarrow i + 1$
 4:     $\mathcal{S}_i = \text{SCHEDULE}(t)$        ▷ Predictive scheduling of workers
 5:     Initialize $\mathcal{R}_i = \mathcal{S}_i$, $\boldsymbol{\theta}^{i+1} = \mathbf{0}$
 6:     while $\mathcal{S}_i \cup \mathcal{R}_i \neq \emptyset$ do
 7:         Wait for any satellite. Upon connection to satellite $k$:
 8:         if $k \in \mathcal{S}_i$ then
 9:             Transmit $\boldsymbol{\theta}^i$ to satellite $k$
10:             $\mathcal{S}_i \leftarrow \mathcal{S}_i \setminus \{k\}$
11:         else if $k \in \mathcal{R}_i$ then
12:             Receive model update $\boldsymbol{\theta}^i_k$ from satellite $k$
13:             $\boldsymbol{\theta}^{i+1} \leftarrow \boldsymbol{\theta}^{i+1} + \frac{n_k}{n} \boldsymbol{\theta}^i_k$
14:             $\mathcal{R}_i \leftarrow \mathcal{R}_i \setminus \{k\}$
15:         end if
16:         if $\mathcal{S}_i \cup \mathcal{R}_i \neq \emptyset$ then
17:             Terminate connection to satellite $k$
18:         end if
19:     end while
20: end while
```

L^2-regularization on the model parameters is employed [15], i.e., the regularization term in (2) is chosen as

$$\tilde{g}_{\boldsymbol{\theta}'}(\mathcal{B}; \boldsymbol{\theta}) = \frac{\lambda}{2} \left\| \boldsymbol{\theta} - \boldsymbol{\theta}' \right\|_2^2 \tag{5}$$

with parameter λ. The stopping criterion in line 3 is a fixed number of iterations that should be chosen such that the computation is finished before the GS is visited again.

B. Synchronous Ground Station Operation

We start the discussion of GS operation by adapting FedAvg to the satellite scenario. Recall that the FedAvg server selects, in epoch i, a subset $\mathcal{S}_i$ of workers to perform updates on the current model $\boldsymbol{\theta}^i$ and then waits for the arrival of *all* scheduled results before updating the model according to (3). A naïve adaption of this algorithm to the satellite scenario is given in Algorithm 2. The main loop runs until convergence is determined in line 2 by any of the usual criteria, e.g., number of epochs, elapsed wall time, or early stopping [10, §7.8]. Workers for the new epoch are selected by the function SCHEDULE and stored in $\mathcal{S}_i$ and $\mathcal{R}_i$. The next version of the global model is initialized in line 5. The role $\mathcal{S}_i$ and $\mathcal{R}_i$ becomes apparent in the following lines: $\mathcal{S}_i$ contains the scheduled workers that have yet to receive the current global model parameters, while $\mathcal{R}_i$ holds the workers that have not yet returned their model update. Accordingly, the inner loop in lines 6–19 runs until both sets are empty. In line 7, the GS waits for any satellite to connect. If it is in $\mathcal{S}_i$, the current global model is sent, the satellite is removed from $\mathcal{S}_i$ and the connection is terminated. If not in $\mathcal{S}_i$ but in $\mathcal{R}_i$, the GS expects that the satellite transmitted a local model update that is incorporated in the new version of the global model parameters in line 13. Then, the satellite is removed from $\mathcal{R}_i$ and the algorithm returns to line 7.

The key difference to vanilla FedAvg is that the communication is asynchronous to allow scheduling of satellites not simultaneously visible to the GS, while the update is still computed synchronously. An important observation is that the work loop in lines 6–19 is blocking, i.e., it waits for all scheduled satellites to connect twice to the GS before starting a new epoch. Assuming the satellite does not finish computation within a single pass, this implies that one epoch takes at least one orbital period on average. However, with multiple satellites scheduled, this time increases.

Algorithm 3 Asynchronous Ground Station Operation

```
 1: Initialize epoch $i = 0$, model $\boldsymbol{\theta}^0$, wall time $t$
 2: loop
 3:     Wait for any satellite. Upon connection to satellite $k$:
 4:     if received model update $(\boldsymbol{\theta}^\tau_k, \tau)$ then
 5:         $i \leftarrow i + 1$
 6:         $\boldsymbol{\theta}^i \leftarrow \text{SERVERUPDATE}(i, \tau, \boldsymbol{\theta}^{i-1}, \boldsymbol{\theta}^\tau_k)$
 7:         if stopping criterion is met then
 8:             Exit loop: Go to line 15
 9:         end if
10:     end if
11:     if SCHEDULE$(k, t)$ then
12:         Transmit $(\boldsymbol{\theta}^i, i)$ to satellite $k$
13:     end if
14:     Terminate connection to satellite $k$
15: end loop
```

From an optimization theoretic perspective, Algorithm 2 is equivalent to FedProx [15], a FedAvg variant that uses the regularization in (5). Convergence follows from [15, Thm. 4].

C. Asynchronous Ground Station Operation

In contrast to Algorithm 2, asynchronous FL operation allows the satellites to work on different versions of the global model, leading to reduced delay. Algorithm 3 outlines the operation of the GS. In line 3, it waits for a satellite to connect. Communication is assumed as non-blocking, i.e., communication delay is hidden from Algorithm 3. If a model update was received, the epoch is advanced in line 5 and the global model is updated based on its current version and the received update. If the satellite is scheduled for further computation, the new global model is transmitted in line 12. The connection is terminated in line 14.

1) FedAsync: Implementation of the SERVERUPDATE and SCHEDULE procedures in Algorithm 3 depends on the FL scheme and the communication scenario. In case of FedAsync (cf. Section II-A), SERVERUPDATE first computes the mixing factor α based on a staleness function $s(i - \tau_k)$ and then returns (4). As no client update can be fresher than one orbital period, we propose to use a hinged staleness function following the definition in [14, §5.2]. In particular, let $s(i - \tau_k) = \tilde{s}(t_i - t_{\tau_k})$ where t_j is the time epoch j was processed at the GS and

$$\tilde{s}(t) = \begin{cases} 1 & \text{if } t \leq (1+\varepsilon) T_{\mathrm{o,max}} \\ (1 + a(t - (1+\varepsilon) T_{\mathrm{o,max}}))^{-1} & \text{otherwise} \end{cases} \tag{6}$$

for some small $\varepsilon \geq 0$, a positive constant a, and $T_{\mathrm{o,max}}$ being the maximum orbital period within the constellation. The scheduler can easily calculate the value of $s(\cdot)$ at the next pass of a given satellite. Hence, it is possible to conserve energy and computational resources by setting SCHEDULE(k, t) to false if the weight α will be below a certain threshold.

IV. UNROLLED FEDERATED AVERAGING ALGORITHM

Synchronous FL procedures applied to ground-assisted satellite in-constellation learning suffer from high latencies. This can be alleviated by asynchronous FL procedures like FedAsync. However, under full client participation and adequately chosen hyperparameters, FedAvg has stronger convergence properties than FedAsync. Hence, it is desirable to implement FedAvg in an asynchronous way. Leveraging on the predictable connectivity of satellites, this is indeed possible.

First, consider a near-polar Walker Delta Pattern Constellation [16] with single orbital shell and a GS located at the North Pole. This is a symmetrical scenario where every satellite visits the GS exactly once per orbital period. Moreover, the sequence of connecting satellites to the GS is constant, i.e., if the satellites are ordered such that, within some interval $[t, t+T_o]$ with T_o being the orbital period, the sequence of satellite contacts is $1 \rightarrow 2 \rightarrow 3 \rightarrow \cdots \rightarrow K$, then this sequence is repeated in every following orbital period.

In this case, the FedAvg update rule in (3) with full client participation can be implemented incrementally without requiring synchronicity in the update phase. In particular, suppose satellite k visits the GS at time t_{i_1} (with epoch i_1) and again at $t_{i_2} = t_{i_1} + T_o$. Then, the client update $\boldsymbol{\theta}_k^{i_2}$ is based on $\boldsymbol{\theta}^{i_1+1}$ and can be incorporated in the global model as

$$\boldsymbol{\theta}^{i_2+1} = \boldsymbol{\theta}^{i_2} - \alpha_k(\boldsymbol{\theta}_k^{i_1} - \boldsymbol{\theta}_k^{i_2}), \qquad (7)$$

where the weight $\alpha_k = \frac{n_k}{n}$ accounts for different dataset sizes. There are exactly K updates in each orbital period and, due to the periodicity of the satellite-GS contact order, the resulting model after a multiple of K epochs should be close to that from Algorithm 2. More generally, consider a constellation where the revisit period of each satellite with respect to an arbitrarily located (but fixed) GS is approaching the same value. Then, the proposed procedure in Algorithm 3 with SERVERUPDATE function as in (7) converges as established in [17, §2].

Finally, consider a constellation with multiple orbital shells. The assumption about equal revisit rates usually does not hold in this case. Following the discussion in [17] this does not prevent convergence but might result in a biased solution. However, this is also the case for many other state-of-the-art FL algorithms, including the methods presented here. Indeed, the numerical results in the next section will show that this effect is much less pronounced in this algorithm than in the asynchronous baseline FedAsync.

V. Empirical Results

We numerically evaluate the performance of the proposed algorithms in terms of the test accuracy on the MNIST [18] and CIFAR-10 [19] datasets. For MNIST, we train a logistic regression model with 7850 trainable parameters [15]. The expected accuracy of centralized training is around 89 %. For CIFAR, we train a ResNet-18 that can achieve an accuracy of slightly above 90 % when trained centrally [20]. The training dataset is distributed randomly over all workers with equal local dataset sizes. Each satellite operates according to Algorithm 1 with $\eta = 0.1$ and $\lambda = 0$, where a single pass over the local dataset is done in batches of size 10 between GS contacts. We rely on the FedML framework [21] for our FL implementation. In the results, we refer to Algorithm 2 as "FedAvg," to the asynchronous baseline in Section III-C1 as "FedAsync," and to the algorithm in Section IV as "FedSat." The staleness function for FedAsync, if used, has parameters $\varepsilon = 0.01$ and $a = 5(1+\epsilon)T_{o,\max}$, where, by Kepler's third law, $T_{o,\max} \approx 127\,\mathrm{min}$. The mixing parameter α for FedAsync was fine-tuned for each experiment individually.

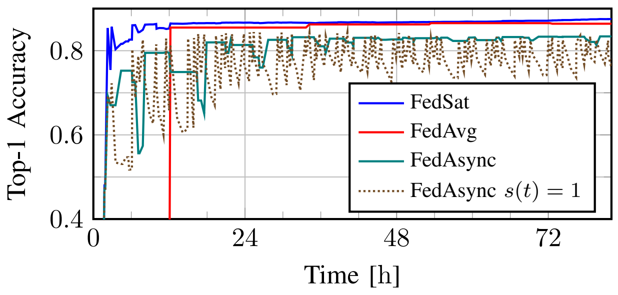

Fig. 1. Top-1 accuracy for a GS in Bremen with Non-IID MNIST data.

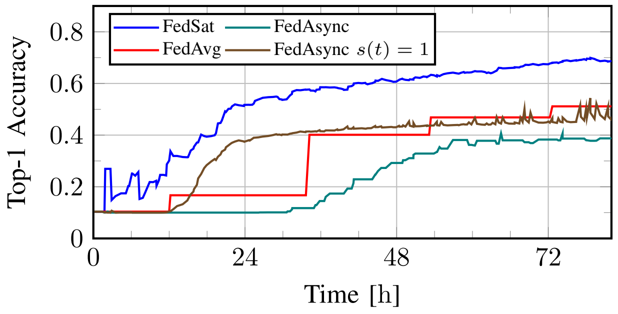

Fig. 2. Top-1 accuracy for a GS in Bremen with Non-IID CIFAR data.

A satellite constellation with two orbital shells at altitudes 500 km and 2000 km, respectively, containing five satellites each is considered. Both are Walker Delta constellations [16] with inclination angle of 80° and five orbital planes. They are shifted such that the minimum difference in right ascension of the ascending node (RAAN) between shells is 36°. The minimum elevation angle α_e is 10°. For the non-IID cases, half of the available classes are distributed to the 500 km orbital shell and the other half to that at 2000 km.

First, consider the case where the GS is located in Bremen, Germany, and the data has non-IID distribution. This scenario poses considerable challenges to the algorithms due to non-uniform device participation and heterogeneous datasets. Figures 1 and 2 display the test accuracy for MNIST and CIFAR, respectively. FedAvg exhibits almost instantaneous convergence for MNIST after a delay of $2T_{o,\max}$, which is due to the simple model. Instead, in the CIFAR experiment, the accuracy resembles a step function with very slow convergence. This clearly shows the inadequacy of FedAvg, and synchronous algorithms in general, in ground-assisted satellite learning. In both experiments, FedAsync exhibits faster convergence than FedAvg but shows inferior training performance. Interestingly, the staleness function proposed in (6) is necessary for stable convergence for MNIST but has negative impact on CIFAR training. The mixing factor α' is set to 0.5 and 0.1 for MNIST and CIFAR, respectively, and the learning rate η for FedAsync is 0.01. The proposed FedSat algorithm shows superior training performance, both in convergence speed and final test accuracy.

Next, we consider a homogeneous scenario with IID data distribution and GS at the North Pole. The accuracy results are shown in Fig. 3. The FedAvg behavior is as before, but with shorter time periods between updates. This is because the constellation revolves around the GS. Among the asynchronous algorithms, FedAsync has a minor edge over FedSat. However, this requires fine-tuning of the additional hyperparameter α,

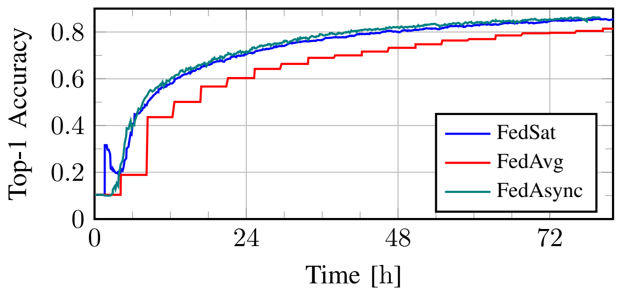

Fig. 3. Top-1 accuracy for a GS at the North Pole with IID CIFAR data. FedAsync with $\alpha = 0.3$ and without staleness function.

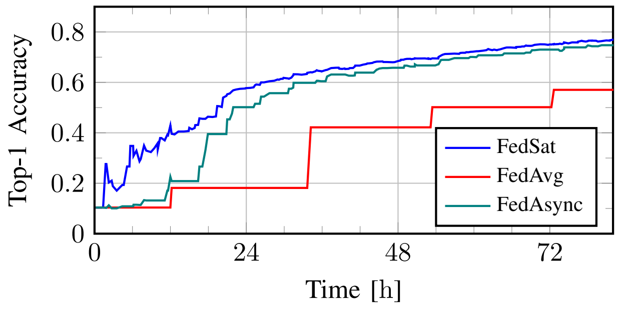

Fig. 4. Top-1 accuracy for a GS in Bremen with IID CIFAR data. FedAsync with $\alpha = 0.3$ and without staleness function.

which has optimal value 0.3 in this case. Finally, Fig. 4 displays results for the same data distribution as before, but for a GS in Bremen. This introduces non-uniform device participation into the previous experiment and could be considered the middle ground between both experiments. The core observation to be made is that FedAsync now performs strictly worse than FedSat. We conclude from this that the proposed method exhibits considerably higher robustness against heterogeneity, which is an important property for the scenario at hand.

In conclusion, these experiments verify our theoretical considerations. We have observed that a naïve implementation of FedAvg [12] leads to tremendous delays and that the state-of-the-art in asynchronous FL algorithms, i.e., FedAsync, struggles to deal with the inherent heterogeneity of the satellite learning scenario. We conjecture that this is not only the case for satellite constellations but also for general FL scenarios with heterogeneity. This is supported by an additional simulation in the Fig. 5. Instead, the proposed algorithm shows excellent performance in all experiments.[1]

VI. Conclusions

We have considered FL in LEO constellations where satellites collaboratively train a ML model without sharing their local datasets. Unique challenges compared to terrestrial networks were identified and addressed by adapting FedAvg and FedAsync to this setting. We have demonstrated how to unroll FedAvg by exploiting the deterministic worker availability and, effectively, convert it from a synchronous to an asynchronous learning algorithm without sacrificing training performance.

[1]The oscillatory behavior of FedAsync in Fig. 5 for non-IID data could not be avoided by tuning α and η. Careful selection of a staleness function and a decaying learning rate might help to dampen this behavior. However, the general trend is apparent and supports our conclusions.

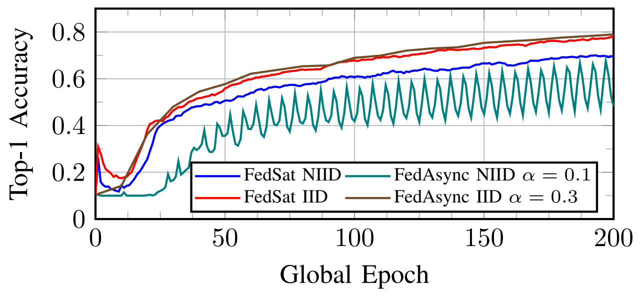

Fig. 5. Top-1 accuracy for uniform client sampling with IID and Non-IID CIFAR data. FedAsync without staleness function.

This reduces the training time of FedAvg by several hours and leads to an algorithm that outperforms FedAsync both in convergence time and test accuracy. The proposed algorithm also has less hyperparameters to tune than FedAsync.

In this initial work, several topics were left open for future work, including proper scheduling of workers, multiple data exchanges during a single GS pass, and employing multiple GS. These approaches could lead to considerably faster training.

References

[1] I. del Portillo, B. Cameron, and E. Crawley, "A technical comparison of three low earth orbit satellite constellation systems to provide global broadband," *Acta Astronaut.*, vol. 159, pp. 123–135, Mar. 2019.
[2] I. Leyva-Mayorga *et al.*, "LEO small-satellite constellations for 5G and beyond-5G communications," *IEEE Access*, vol. 8, Oct. 2020.
[3] Y. Qian, "Integrated terrestrial-satellite communication networks and services," *IEEE Wireless Commun.*, vol. 27, no. 6, Dec. 2020.
[4] O. Kodheli *et al.*, "Satellite communications in the new space era: A survey and future challenges," *IEEE Commun. Surveys Tuts.*, vol. 23, no. 1, pp. 70–109, Firstquarter 2021.
[5] B. Di, L. Song, Y. Li, and H. V. Poor, "Ultra-dense LEO: Integration of satellite access networks into 5G and beyond," *IEEE Wireless Commun.*, vol. 26, no. 2, pp. 62–69, Apr. 2019.
[6] Z. Lin *et al.*, "Supporting IoT with rate-splitting multiple access in satellite and aerial-integrated networks," *IEEE Internet Things J.*, vol. 8, no. 14, pp. 11 123–11 134, Jul. 2021.
[7] M. A. Vazquez *et al.*, "Machine learning for satellite communications operations," *IEEE Wireless Commun.*, vol. 59, no. 2, Feb. 2021.
[8] G. Giuffrida *et al.*, "CloudScout: A deep neural network for on-board cloud detection on hyperspectral images," *Remote Sens.*, vol. 12, 2020.
[9] G. Mateo-Garcia *et al.*, "Towards global flood mapping onboard low cost satellites with machine learning," *Sci. Rep.*, vol. 11, Mar. 2021.
[10] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*, 2016.
[11] J. Konečný, H. B. McMahan, and D. Ramage, "Federated optimization: Distributed optimization beyond the datacenter," in *Proc. 8th NIPS Workshop Optim. Mach. Learn. (OPT2015)*, Dec. 2015.
[12] H. B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. Agüera y Arcas, "Communication-efficient learning of deep networks from decentralized data," in *Proc. 20th Int. Conf. Artificial Intell. Statist. (AISTATS)*, ser. Proc. Mach. Learn. Res. (PMLR), vol. 54, Apr. 2017.
[13] X. Wang *et al.*, "Convergence of edge computing and deep learning: A comprehensive survey," *IEEE Commun. Surveys Tuts.*, vol. 22, 2020.
[14] C. Xie, O. Koyejo, and I. Gupta, "Asynchronous federated optimization," in *Proc. Annu. Workshop Optim. Mach. Learn. (OPT2020)*, Dec. 2020.
[15] T. Li *et al.*, "Federated optimization in heterogeneous networks," in *Proc. Mach. Learn. Syst. (MLSys 2020)*, Austin, TX, Mar. 2020, pp. 429–450.
[16] J. G. Walker, "Satellite constellations," *J. Brit. Interplanet. Soc.*, 1984.
[17] A. Nedić, D. Bertsekas, and V. Borkar, "Distributed asynchronous incremental subgradient methods," in *Studies in Computational Mathematics*. Elsevier, 2001, vol. 8, pp. 381–407.
[18] Y. LeCun, C. Cortes, and C. J. C. Burges. The MNIST database of handwritten digits. [Online]. Available: http://yann.lecun.com/exdb/mnist/
[19] A. Krizhevsky, "Learning multiple layers of features from tiny images," Univ. Toronto, Tech. Rep., 2009.
[20] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," Microsoft Research, Tech. Rep., 2015.
[21] C. He *et al.*, "FedML: A research library and benchmark for federated machine learning," 2020, arXiv:2007.13518.