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 data sets. To address this problem, we propose a new set of algorithms based of Federated learning (FL). Our approach differs substantially from the standard FL algorithms, as it takes into account the predictable connectivity patterns that are immanent to the LEO constellations. Extensive numerical evaluations highlight the fast convergence speed and excellent asymptotic test accuracy of the proposed method. In particular, the achieved test accuracy is within 96 % to 99.6 % of the centralized solution and the proposed algorithm has less hyperparameters to tune than state-of-the-art asynchronous FL methods.**

***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 by private companies with the goal of providing ubiquitous connectivity and low latency Internet service [1], [2]. Their integration into terrestrial mobile networks is an active research area, covering various use cases such as Earth observation missions [3]–[6]. Presumably, machine learning (ML) will become an essential tool to manage these constellations and utilize their sensor measurements [7], [8].

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 modern deep neural networks [9], this involves high transmission costs and delays. Moreover, it might simply be prohibited to share the raw data with a central entity 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) [10], [11].

A core assumption of the FL setting is intermittent and unpredictable participation of the clients. In order to cope with that, asynchronous algorithms have been proposed recently [12]. 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 can be incorporated into FL algorithms when the training process is orchestrated by a GS. We conclusively show that this approach leads to superior training performance when compared to generic FL algorithms.

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 (HLRN).

II. System Model and Problem Formulation

Consider a LEO constellation of K satellites in L orbital planes. Each orbital plane l, $l \in \mathcal{L} = \{1, \ldots, L\}$, contains the satellites $\mathcal{O}_l$. Clearly, $\mathcal{O}_i \cap \mathcal{O}_j = \emptyset$ for all $i, j \in \mathcal{L} : i \neq j$, and $\bigcup_{l \in \mathcal{L}} \mathcal{O}_l = \mathcal{K} = \{1, \ldots, K\}$. Satellite k, $k \in \mathcal{K}$, has position $\boldsymbol{r}_k(t)$ at time t with coordinates in an Earth-centered inertial (ECI) coordinate system, i.e., a Cartesian coordinate system with origin at Earth's center of mass and fixed with respect to the stars. Satellites in the same orbital plane follow the same trajectory, i.e., $\boldsymbol{r}_{k_2}(t) = \boldsymbol{r}_{k_1}(t + \Delta_{k_1 k_2})$ for all $k_1, k_2 \in \mathcal{O}_l$, $l \in \mathcal{L}$, and some delay $\Delta_{k_1 k_2}$. In addition, there is a fixed position GS. Due to Earth's rotation in the ECI frame, the GS follows a time dependent trajectory $\boldsymbol{r}_g(t)$.

A ground to satellite link (GSL) is feasible if a satellite $k \in \mathcal{K}$ is visible from the GS at a minimum elevation angle α_e, expressed as $\frac{\pi}{2} - \angle(\boldsymbol{r}_g(t), \boldsymbol{r}_k(t) - \boldsymbol{r}_g(t)) \geq \alpha_e$ or:

$$\arccos\left(\frac{\langle \boldsymbol{r}_g(t), \boldsymbol{r}_k(t) - \boldsymbol{r}_g(t)\rangle}{\|\boldsymbol{r}_g(t)\| \, \|\boldsymbol{r}_k(t) - \boldsymbol{r}_g(t)\|}\right) \leq \frac{\pi}{2} - \alpha_e. \tag{1}$$

At time instant t, the set of feasible GSL is $\mathcal{C}_{\mathrm{GSL}}(t) = \{k \in \mathcal{K} : (1)\}$. In general, only a subset of satellites is connected to the GS at any point in time and the time between contacts is much longer than the actual online time.

Satellite $k \in \mathcal{K}$ collects data from its on-board instruments and stores it in a data set $\mathcal{D}_k$. Due to different orbits and orbital positions, the data sets of two distinct satellites $k_1, k_2 \in \mathcal{K}$ are disjunct, i.e., $\mathcal{D}_{k_1} \cap \mathcal{D}_{k_2} = \emptyset$, and possibly non-IID. After the data collection phase, the satellites collaboratively solve an optimization problem of the form

$$\min_{\boldsymbol{\theta} \in \mathbb{R}^d} \frac{1}{n} \sum_{\boldsymbol{x} \in \mathcal{D}} f(\boldsymbol{x}; \boldsymbol{\theta}) = \min_{\boldsymbol{\theta} \in \mathbb{R}^d} \sum_{k \in \mathcal{K}} \frac{n_k}{n} \sum_{\boldsymbol{x} \in \mathcal{D}_k} \frac{1}{n_k} f(\boldsymbol{x}; \boldsymbol{\theta}) \tag{2}$$

with the goal of training a machine learning model, where $\mathcal{D} = \bigcup_{k\in\mathcal{K}} \mathcal{D}_k$, $n_k = |\mathcal{D}_k|$, and $n = |\mathcal{D}| = \sum_{k\in\mathcal{K}} n_k$. 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 (labeled) data point $\boldsymbol{x}$ and model parameters $\boldsymbol{\theta}$. This process is orchestrated by the GS and performed iteratively without sharing data sets between satellites. In particular, at iteration i, the GS receives parameter updates $\{\boldsymbol{\theta}^i_{k_1}, \boldsymbol{\theta}^i_{k_2}, \dots\}$ computed by a subset of currently visible satellites $k_1, k_2, \dots$ based on their local version of the global model and their local data set. The GS incorporates these updates into the global model parameters $\boldsymbol{\theta}^i$ where the ultimate goal is that $\lim_{i\to\infty} \boldsymbol{\theta}^i = \boldsymbol{\theta}^\star$ with $\boldsymbol{\theta}^\star$ being a optimal solution to (2).

We assume that the satellites have very limited computational resources available to compute model updates. This could be due to, e.g., a small form factor that limits heat dissipation, other computational tasks running in parallel, and energy constraints. Hence, we consider the case where the satellites work on (2) between visits to the GS and use the contact time to do a parameter exchange. The communication protocol during one pass is the following. Upon contact, satellite k makes its presence known to the GS and, optionally, sends its updated model parameters. Then, the GS updates the global model $(\boldsymbol{\theta}^i, \boldsymbol{\theta}^i_k) \mapsto \boldsymbol{\theta}^{i+1}$ and decides whether satellite k should continue working on (2) until its next GS contact. If yes, it transmits the updated parameter vector to satellite k. This is formalized in the next two subsections.

A. Satellite Operation for Federated Learning

Satellite k receives the current global model parameters $\boldsymbol{\theta}^\tau$ and the corresponding global epoch τ from the GS. Then, it performs one or more iterations of minibatch stochastic gradient descent (SGD) over its complete local data set. In particular, in each iteration ("epoch") the local data set 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 [9], [11]. This is repeated until a predefined number of epochs is performed or until the GS is visited again. Since the time point of the next GS contact is deterministic, the satellite can terminate the SGD procedure in time for the next contact.

To avoid large deviations from the global model, L^2-regularization is used [12] and the satellite computes its gradient updates based on the surrogate objective with a regularization parameter λ:

$$g_{\boldsymbol{\theta}'}(\mathcal{B};\boldsymbol{\theta}) = \frac{1}{|\mathcal{B}|}\sum_{\boldsymbol{x}\in\mathcal{B}} f(\boldsymbol{x};\boldsymbol{\theta}) + \frac{\lambda}{2}\left\|\boldsymbol{\theta} - \boldsymbol{\theta}'\right\|_2^2. \tag{3}$$

Upon connecting to the GS, the satellite sends the locally updated model $\boldsymbol{\theta}^\tau_k$ and the time stamp τ to the GS. Then, the satellite waits for a new version of the global model that might be sent during the current GS connection or at a later contact. Please refer to Algorithm 1 for pseudo-code of this algorithm.

B. Ground Station Operation

The GS implements an asynchronous FL procedure, i.e., satellites can work, in principle, on different version of the global model and updates are incorporated into the global model as they arrive at the GS. In contrast, in a synchronous FL procedures, e.g., FedAvg [11], all workers operate on the same version of the global model which is only updated once all scheduled workers have sent their updates.

Algorithm 1 Client Update Procedure at Satellite k

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

Algorithm 2 Ground Station Operation

1: **Initialize** epoch $i = 0$, model $\boldsymbol{\theta}^0$, wall time t
2: **while** stopping criterion not met **do**
3: $\quad$Wait for any satellite. Upon connection to satellite k:
4: $\quad$**if** received model update $(\boldsymbol{\theta}^\tau_k, \tau)$ **then**
5: $\quad\quad i \leftarrow i+1$
6: $\quad\quad\boldsymbol{\theta}^i \leftarrow$ ServerUpdate$(i, \tau, \boldsymbol{\theta}^{i-1}, \boldsymbol{\theta}^\tau_k)$
7: $\quad$**end if**
8: $\quad$**if** Schedule(k, t) **then**
9: $\quad\quad$Transmit $(\boldsymbol{\theta}^i, i)$ to satellite k
10: $\quad$**end if**
11: **end while**

Algorithm 2 outlines the operation of the GS. The implementation of the ServerUpdate and Schedule procedure depends on the FL scheme and will be discussed in Section III. The stopping criterion in line 2 can be any of the usual termination criteria in ML, e.g., number of epochs, elapsed wall time, or early stopping [9, §7.8]. We assume that the communication in Algorithm 2 is non-blocking, i.e., communication delay is hidden from Algorithm 2. Thus, the only part of Algorithm 2 that might lead to delays for other satellites are lines 3–6. Due to the computational simplicity of these operations, we assume this delay is negligible.

An obvious extension of this procedure is to allow several parameter exchanges during a single satellite pass. This complicates scheduling but likely results in faster convergence. Due to space limitations, we postpone this extension to future work.

III. Federated Learning

This FL scenario is characterized by long delays, ranging from the time of one orbital period to half a day, occurring between the assignment of a computation task to a satellite and receiving its result. The typical operation of FL algorithms is to assign a task to a subset of workers and then wait for all of them to return their result before continuing. This is known as synchronous FL and prone to the straggler problem [13]. Applied to the task at hand, it is apparent that a straightforward adaption of a synchronous algorithm will lead to slow convergence. Nevertheless, we sketch the operation of

Algorithm 3 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 $\boldsymbol{\theta}_k^i$ from satellite k
13: $\boldsymbol{\theta}^{i+1} \leftarrow \boldsymbol{\theta}^{i+1} + \frac{n_k}{n}\boldsymbol{\theta}_k^i$
14: $\mathcal{R}_i \leftarrow \mathcal{R}_i \setminus \{k\}$
15: **end if**
16: **end while**
17: **end while**

the well-known FedAvg algorithm [11] in Section III-A and use it as baseline.

One way to address the straggler problem is to incorporate client updates whenever they arrive in an asynchronous fashion. Such an algorithm was first published in [12] under the name FedAsync. It is shown in [12] to often outperform FedAvg and will be adapted to the satellite setting in Section III-B. However, this algorithm was designed under the premise that device availability is driven by a random, unpredictable process. While the satellite scenario best described as asynchronous it is far from being random as the client updates come in a highly predictable manner. We exploit this in Section III-C to unroll the FedAvg algorithm to work asynchronously.

A. Federated Averaging Algorithm

The FedAvg server selects, in iteration 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 the rule

$$\boldsymbol{\theta}^{i+1} = \sum_{k \in \mathcal{S}_i} \frac{n_k}{\sum_{k \in \mathcal{S}_i} n_k} \boldsymbol{\theta}_k^i. \tag{4}$$

A naïve adaption of this algorithm to the satellite scenario is given in Algorithm 3. The difference to the original FedAvg algorithm is that the communication is asynchronous to allow scheduling of satellites that are not simultaneously visible to the GS, while the update is still computed synchronously.

Observe that the server update loop in lines 6–16 is blocking, i.e., it waits for all scheduled satellites to connect twice to the GS before scheduling a new set of satellites. The implication is that very careful scheduling is necessary to avoid long delays. An alternative is to use a completely asynchronous algorithm.

B. Asynchronous Federated Learning Algorithm

The FedAsync algorithm from [12] incorporates client updates asynchronously as they are received. At iteration i, it obtains the updated global model from the convex combination of the received client update and the current global model parameters, i.e.,

$$\boldsymbol{\theta}^{i+1} = (1 - \alpha)\boldsymbol{\theta}^i + \alpha\boldsymbol{\theta}_k^i \tag{5}$$

Algorithm 4 FedAsync Update Procedure [12]

Require: Mixing factor $\alpha' \in (0, 1)$, staleness function $s(t) \in [0, 1]$
procedure SERVERUPDATE($i, \tau, \boldsymbol{\theta}, \boldsymbol{\theta}_k$)
$\alpha \leftarrow \alpha' s(t_i - t_\tau)$ ▷ Account for staleness
return $(1 - \alpha)\boldsymbol{\theta} + \alpha\boldsymbol{\theta}_k$
end procedure

where $\alpha \in (0, 1)$ defines how much weight is given to incoming client updates. Updates that are based on older versions of the global model are likely to introduce an error into the solution and, hence, should receive less weight. This staleness is incorporated by augmenting the base weight α' by a so-called staleness function $s(t)$ prior to updating the global model, i.e., $\alpha = \alpha' \cdot s(t_i - t_\tau)$, where i is the current model epoch, τ is the epoch the client update is based on (cf. Algorithm 2), and t_j is the time epoch j was processed at the GS. Pseudo-code for this implementation of the SERVERUPDATE procedure is given in Algorithm 4.

As no client update can be fresher than one orbital period, a hinged staleness function as proposed in [12, §5.2] seems appropriate. In particular, we choose

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

for some small $\varepsilon \geq 0$ and a positive constant a, where $T_{\text{o,max}}$ is the maximum orbital period in the constellation $\{\mathcal{O}_l\}_{l \in \mathcal{L}}$. 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.

The update rule in (5) is considerably different to the FedAvg rule in (4), which is motivated by (2) to obtain a solution that is unbiased towards any particular local data set. Given the fact that the timing of all future model parameter updates can be predicted, it would be desirable to choose α for each update individually such that (4) is approximated. However, this is impossible using the update rule (5) as can be verified easily.

C. Unrolled Federated Averaging Algorithm

Consider a near-polar Walker Delta Pattern Constellation [14] 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, without loss of generality, 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 $1 \to 2 \to 3 \to \cdots \to K$, then this sequence is repeated in every following orbital period.

In this case, the FedAvg update rule in (4) with full client participation can be implemented as a moving average 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 $\boldsymbol{\theta}_k^{i_2}$ received at t_{i_2} is based on $\boldsymbol{\theta}^{i_1+1}$ and can be incorporated in the global model as

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

where the weight $\alpha_k = \frac{n_k}{n}$ accounts for different data set sizes. There are exactly K updates in each orbital period and, due to

Algorithm 5 Unrolled FedAvg Update Procedure (FedSat)

procedure SERVERUPDATE($\cdot, \cdot, \boldsymbol{\theta}, \boldsymbol{\theta}_k$)
  Let $\boldsymbol{\theta}_k^{\text{old}}$ be the previous model update by satellite k
  return $\boldsymbol{\theta} - \alpha_k \left(\boldsymbol{\theta}_k^{\text{old}} - \boldsymbol{\theta}_k\right)$ ▷ α_k as in (8)
end procedure

the periodicity of the satellite-GS contact order, the resulting model after a multiple of K epochs is expected to be close the result from Algorithm 3. This update procedure is summarized in Algorithm 5. Observe that the difference $\boldsymbol{\theta}_k^{\text{old}} - \boldsymbol{\theta}_k$ can be replaced by the accumulated gradient update of satellite k. Hence, it is not necessary to actually store the last model update of each satellite as indicated by Algorithm 5.

Now, consider adding a second orbital shell $\mathcal{C}_2$ to the constellation with twice the orbital period of the first shell $\mathcal{C}_1$. Satellites in $\mathcal{C}_2$ visit the GS only half as often as satellites in $\mathcal{C}_1$. According to the discussion in [15], directly applying (7) might lead to a bias towards the data stored in $\mathcal{C}_1$. Inspired by the approach in [15], this might be alleviated by weighting model updates of satellites in $\mathcal{C}_2$ with $2\frac{n_k}{\tilde{n}}$ instead of $\frac{n_k}{\tilde{n}}$, where $\tilde{n}$ is chosen such that the sum of all weights is one. Generalizing this approach to arbitrary constellations, we can choose the weight of satellite k's update in (7) as

$$\alpha_k = \frac{\tilde{\alpha}_k}{\sum_k \tilde{\alpha}_k} \qquad \text{with} \quad \tilde{\alpha}_k = \frac{n_k}{\nu_k}, \tag{8}$$

where n_k is the size of the local data set and ν_k is the number of GS contacts of satellite k over a fixed period T_ν. The choice of this interval T_ν strongly depends on the particular ML scenario and the satellite constellation. In general, a longer period will help to average out short term fluctuations. On the other hand, in case of strong asynchronicity and non-IID data, this might lead to biases if some satellites have strong influence on the model over a short period of time. Especially in the beginning of the training process, this might drive the model towards a local solution in favor of some satellite's local data distribution. In this case, better results might be obtained by using a shorter averaging period. In some cases, it might even be beneficial to update the weights throughout the training.

IV. Empirical Results

We numerically evaluate the performance of the algorithms presented in Section III for a logistic regression model trained on the MNIST dataset [16], [17]. This is a classification task with 784 inputs and 10 outputs, amounting to 7850 trainable parameters. The test/train split is 89 %/11 % and the performance is evaluated in terms of the test accuracy over the training time. The expected test accuracy of this model when trained centrally is 0.87. All results are averaged over 10 different random initializations of model parameters. The training data set is distributed randomly over all workers with equal local data set sizes. Each satellite operates according to Algorithm 1 where a single pass over the local data set is done in batches of size 10 between GS contacts. The learning rate $\eta = 0.1$ and $\lambda = 0.001$ in all experiments. The mixing parameter α for Algorithm 4 was fine-tuned for each experiment individually. We rely on the FedML framework [18] for our FL implementation. In the results, we refer to Algorithm 3 as "FedAvg", to Algorithm 4 as "FedAsync" and to Algorithm 5 as "FedSat". The variant with weight augmentation as in (8) is denoted as "FedSat+". The scheduler includes all satellites in each round. FedAsync is used with the hinged staleness function in (6) with parameters $\varepsilon = 0.01$ and $a = 5(1+\epsilon)T_{o,\max}$, where, by Kepler's third law, $T_{o,\max} \approx 127\,\text{min}$.

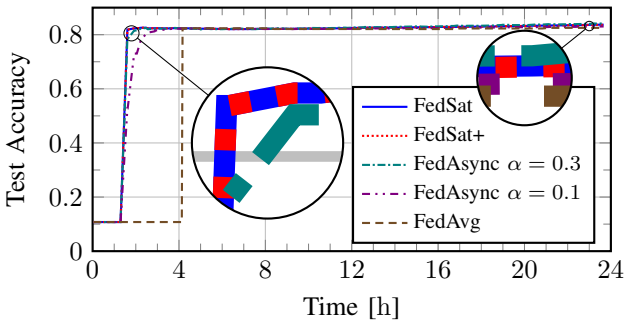

Fig. 1. Test accuracy for a GS at the North Pole with IID data distribution.

A satellite constellation with two orbital shells at altitude 500 km and 2000 km, respectively, containing five satellites each is considered. Both are Walker Delta constellations [14] 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, the data set was split such that labels 0–4 and 5–9 are distributed to satellites in the 500 km and 2000 km orbital shells, respectively.

First, consider the case where the GS is located at the North Pole. As discussed in Section III-C, this is a very symmetrical scenario and provides good insight into the general performance of the discussed algorithms. Figure 1 shows how the accuracy of the model evolves over time for an IID data distribution, i.e., all satellites have data that was uniformly sampled from the MNIST dataset. It can be observed that all algorithms converge towards a very similar accuracy after $2T_{o,\max}$. In the case of FedSat, the achieved test accuracy is 96.2 % of the centralized accuracy. The convergence of FedAvg is almost instantaneous after $2T_{o,\max}$ which is due to its synchronous operation. However, to achieve exactly the same accuracy as FedAvg a longer training time would be necessary. Among the asynchronous algorithms, neither FedSat nor FedAsync has a clear edge, except that FedAsync requires an additional hyperparameter α to be fine-tuned.

In Fig. 2, the same experiment is repeated with a non-IID data distribution between orbital shells. As before, FedAvg exhibits almost instantaneous convergence after a delay of $2T_{o,\max}$. However, the test accuracy of FedAvg improves after additional $2T_{o,\max}$ which gives a good indication that it will be unsuitable to train more complicated models in this satellite setup. As for the other algorithms, FedSat+ is fastest to converge but asymptotically achieves a strictly worse accuracy than FedAvg and FedSat. In particular, FedSat's accuracy is lower bounded by FedAvg, as should be expected from its derivation, and comes within 99.6 % of the centralized performance. FedAsync shows a performance similar to FedSat and FedSat+ at the beginning but fails to converge to a stable solution and oscillates between FedSat and FedSat+. Interestingly, the optimal $\alpha = 0.1$ is different than in the IID case. Choosing $\alpha = 0.3$ instead

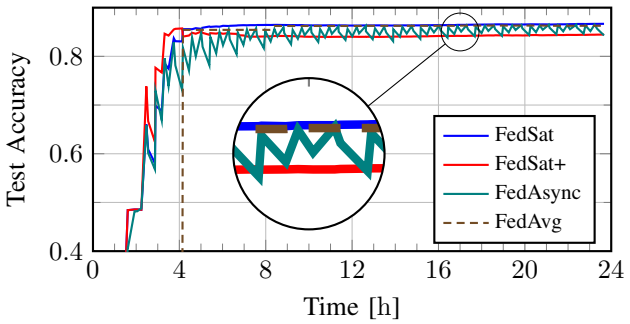

Fig. 2. Test accuracy for a GS at the North Pole with Non-IID data distribution.

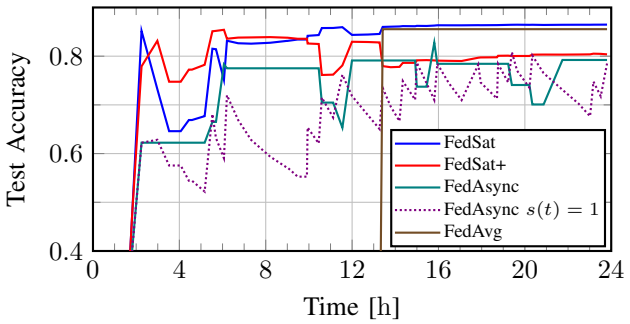

Fig. 3. Test accuracy for a GS in Bremen with Non-IID data distribution.

leads to increased oscillation (not shown). We conclude that FedAsync is quite sensitive to the choice of α and that this parameters needs to be tuned for every scenario separately. Further, the strictly inferior long-term solution of FedSat+ combined with the fast initial training success indicates that a multi-stage strategy for the weight augmentation is a promising training strategy.

Next, we repeat the non-IID experiment for a GS based in Bremen, Germany, to assess the performance in a less remote location. Here, the satellite contacts are far less symmetrical than at the North Pole which leads to a considerably more difficult training scenario. This can be observed from the evolution of the test accuracy in Fig. 3, especially in comparison to the previous experiments. In principle, the observations made in Fig. 2 carry over to this case, but in a more pronounced way. Again, FedSat+ exhibits a good initial performance but converges to an inferior solution. FedSat achieves 99.4 % of the centrally trained model's test accuracy. FedAsync (with $\alpha = 0.1$) converges to an even worse solution than FedSat+. We also evaluate the effect of proposed staleness function in (6). It can be seen from the plot labelled "FedAsync $s(t) = 1$" that without this function the convergence behavior worsens. FedSat achieves the best asymptotic accuracy and converges at a similar speed than the other algorithms. Finally, FedAvg requires a very long time for its initial convergence and lower bounds the performance of FedSat. It is to be expected that it converges to the same accuracy as FedSat given more time.

In conclusion, these experiments verify the theoretical observations from Section III. We have observed that a naïve implementation of FedAvg [11] leads to tremendous delays and that directly applying a generic asynchronous algorithm leads to unstable convergence behavior. Instead, the proposed unrolling of FedAvg in Algorithm 5, denoted as FedSat, exhibits very good training performance that might be improved by carefully augmenting the learning rate based on (8).

V. Conclusions

We have considered FL in LEO constellations where satellites collaboratively train a ML model without sharing their local data sets. 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. This reduces the training time of FedAvg by several hours and leads to an algorithm that outperforms FedAsync, a generic asynchronous algorithms, 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] M. Mitry, "Routers in space: Kepler communications' CubeSats will create an internet for other satellites," *IEEE Spectr.*, vol. 57, no. 2, pp. 38–43, Feb. 2020.

[2] 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.

[3] I. Leyva-Mayorga *et al.*, "LEO small-satellite constellations for 5G and beyond-5G communications," *IEEE Access*, vol. 8, Oct. 2020.

[4] Y. Qian, "Integrated terrestrial-satellite communication networks and services," *IEEE Wireless Commun.*, vol. 27, no. 6, Dec. 2020.

[5] 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, 2021.

[6] 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.

[7] M. A. Vazquez *et al.*, "Machine learning for satellite communications operations," *IEEE Wireless Commun.*, vol. 59, no. 2, Feb. 2021.

[8] D. J. Lary *et al.*, "Machine learning applications for earth observation," in *Earth Observation Open Science and Innovation*. New York; Berlin, Germany; Vienna, Austria: Springer-Verlag, 2018, pp. 165–218.

[9] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. Cambridge, MA, USA: MIT Press, 2016.

[10] 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.

[11] 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.

[12] C. Xie, O. Koyejo, and I. Gupta, "Asynchronous federated optimization," in *Proc. Annu. Workshop Optim. Mach. Learn. (OPT2020)*, Dec. 2020.

[13] P. Kairouz *et al.*, *Advances and Open Problems in Federated Learning*, ser. FnT Mach. Learn. Now, 2021, vol. 14, no. 1–2.

[14] J. G. Walker, "Satellite constellations," *J. Brit. Interplanetary Soc.*, vol. 37, pp. 559–571, Dec. 1984.

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

[16] Y. LeCun, C. Cortes, and C. J. C. Burges. The MNIST database of handwritten digits. [Online]. Available: http://yann.lecun.com/exdb/mnist/

[17] T. Li *et al.*, "Federated optimization in heterogeneous networks," in *Proc. Mach. Learn. Syst. (MLSys 2020)*, vol. 2, 2020, pp. 429–450.

[18] C. He *et al.*, "FedML: A research library and benchmark for federated machine learning," 2020, arXiv:2007.13518.