On-Board Federated Learning for Satellite Clusters With Inter-Satellite Links

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

***Abstract*— The emergence of mega-constellations of interconnected satellites has a major impact on the integration of cellular wireless and non-terrestrial networks, while simultaneously offering previously inconceivable data gathering capabilities. This paper studies the problem of running a federated learning (FL) algorithm within low Earth orbit satellite constellations connected with intra-orbit inter-satellite links (ISL), aiming to efficiently process collected data in situ. Satellites apply on-board machine learning and transmit local parameters to the parameter server (PS). The main contribution is a novel approach to enhance FL in satellite constellations using intra-orbit ISLs. The key idea is to rely on predictability of satellite visits to create a system design in which ISLs mitigate the impact of intermittent connectivity and transmit aggregated parameters to the PS. We first devise a synchronous FL, which is extended towards an asynchronous FL for the case of sparse satellite visits to the PS. An efficient use of the satellite resources is attained by sparsification-based compression the aggregated parameters within each orbit. Performance is evaluated in terms of accuracy and required data transmission size. We observe a sevenfold increase in convergence speed over the state-of-the-art using ISLs, and $10\times$ reduction in communication load through the proposed in-network aggregation strategy.**

***Index Terms*— Low Earth orbit, mega-constellations, intra-orbit inter-satellite links, federated learning, sparsification.**

I. Introduction

SATELLITE constellations have been an essential component of modern communication and remote sensing systems for decades. Recent advances in satellite technology and the emergence of interconnected mega constellations in low earth orbit (LEO), are revolutionizing the way we collect and process data from space [2], [3], [4]. Unlike the previous cellular generations that were exclusively focused on terrestrial networks, mega-constellations and Non-Terrestrial Networks (NTN) are seen as the integral part of 5G and the upcoming 6G wireless systems [5]. Consisting of thousands of satellites, these constellations have the potential to process vast amounts of data, e.g., high-resolution hyperspectral images. Conventional central processing of this collected data involves significant challenges, including communication delays, limited bandwidth and storage, as well as data ownership concerns [6], [7]. The on-board intelligence of satellites increases steadily [4], [8], [9], e.g., PhiSat-1 of European Space Agency (ESA) mission, pushing towards in-orbit data preservation and learning to conserve bandwidth and energy, avoid overloading of ground-satellite links (GSLs), and enable native artificial intelligence in space.

Satellite federated learning (SFL) has emerged as a promising solution to address these challenges [7], [10], as an instance of distributed machine learning (ML) that enables satellites to collaboratively learn a model without exchanging raw data. With federated learning (FL), each satellite trains a local model with its own data and sends only the updated model parameters to be aggregated at a central *parameter server (PS)*. FL has the potential to reduce both communication cost and training delay. Nonetheless, the intermittent connectivity between satellites and the PS introduce extended delays when implementing conventional FL in satellite constellations. The first step towards SFL was made in [10], where each satellite acts as an individual collaborator towards the PS, located within a terrestrial ground station (GS). Each LEO satellite has only a very short communication window per orbital period towards the terrestrial PS. This, as well as the fact the link between a GS and a satellite vanishes behind the horizon for several hours after a few orbital periods, leads to a connectivity bottleneck that severely inhibits convergence speed of a plain FL. Thus, instead of using conventional synchronous FL, [10] advantageously uses the sporadic, but predictable, satellite connectivity to roll out an asynchronous aggregation.

Newer satellites, especially within the context of mega constellations [11], [12], [13], rely increasingly on inter-satellite links (ISLs) and multi-hop routing. In this paper, we consider a SFL setup with ISLs to facilitate the efficient implementation of both, synchronous and asynchronous SFL. The focus is on scenarios with connectivity between adjacent satellites within the same circular orbital plane. In this case, these connected satellites have stable relative positions, resulting in an approximately fixed distance from each other and, thus, in stationary link budgets. This is in stark contrast to links

Manuscript received 10 July 2023; revised 16 November 2023 and 21 December 2023; accepted 8 January 2024. Date of publication 19 January 2024; date of current version 18 June 2024. 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). An earlier version of this paper was presented in part at the IEEE International Conference on Communications (ICC 2022), Seoul, South Korea, May 2022 [DOI: 10.1109/ICC45855.2022.9838619]. The associate editor coordinating the review of this article and approving it for publication was M. Erol-Kantarci. *(Corresponding author: Nasrin Razmi.)*

Nasrin Razmi, Bho Matthiesen, and Armin Dekorsy are with the Gauss-Olbers Center and the Department of Communications Engineering, University of Bremen, 28359 Bremen, Germany (e-mail: razmi@ant.uni-bremen.de; matthiesen@ant.uni-bremen.de; dekorsy@ant.uni-bremen.de).

Petar Popovski is with the Department of Electronic Systems, Aalborg University, 9100 Aalborg, Denmark, and also with the Department of Communications Engineering, University of Bremen, 28359 Bremen, Germany (e-mail: petarp@es.aau.dk).

Color versions of one or more figures in this article are available at https://doi.org/10.1109/TCOMM.2024.3356429.

Digital Object Identifier 10.1109/TCOMM.2024.3356429

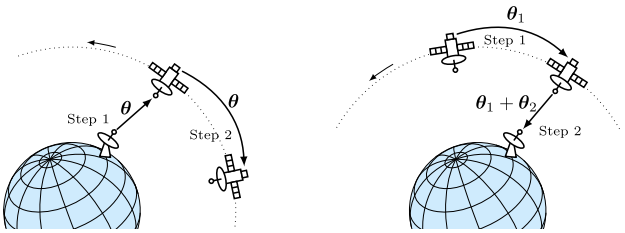


Fig. 1. Global model distribution (left) and collection of local updates (right) using intra-orbit inter-satellite communication.

across different circular orbital planes. These inter-orbit ISLs are, in the best case, constantly changing in distance and, in the worst case, have a very short lifespan [3], [14], [15], [16]. However, we explicitly note that the focus on intra-orbit ISLs does not exclude scenarios where additional inter-orbit ISLs are available. Indeed, the current work is directly applicable to those scenarios and, in some cases, it might be even preferable to employing all available links, as this will likely result in a considerably higher orchestration complexity.

A direct implementation of multi-hop routing leads to network traffic growing quadratically in the number of satellites and a high load on links towards the PS, as each client update will be treated as a common unicast message. Leveraging the structure of FL traffic along with in-network aggregation [17], communication can be limited to a single outgoing message per satellite and a global iteration during the aggregation phase. Moreover, most of these transmissions use intra-orbit ISLs instead of the more challenging PS link, resulting energy saving and reduced communication load at the PS.

Fig. 1 illustrates the key idea through an example of connectivity bottleneck in SFL using two satellites at 550 km altitude, spaced 45° apart within the same orbital plane. The PS is located in a GS and both satellites are participating in a FL procedure with their local data sets. A single pass over the GS, i.e., the time the satellite and the GS can communicate, is less than 10 minutes, while a single orbital period is 95 minutes. Assume that the computation of this update takes 15 minutes. Let one of the satellites be selected by the PS to compute an update to the global model. The satellite will collect the current global model on the first pass and return the result on the next pass. This incurs a delay of roughly one orbital period, which is more than one hour in excess of the computation time. If both satellites are supposed to compute an update then, upon the first pass, each satellite will collect the current model version and start computing. The first satellite will pass the PS for the second time and deliver the update. However, in a synchronous FL procedure, the PS will not update the global model until all updates are received. Hence, the first satellite will not have a new model version available to iterate upon during the offline period following the second pass. Differently from this, when ISLs are available, the first satellite can transmit this global model to the second satellite directly after receiving it. Then, both satellites compute the update in parallel and the second satellite can collect and deliver both updates to the GS during its first pass. This is illustrated in Fig. 1. The time for a single global iteration is reduced from over one orbital period to less than 20 minutes by this approach, while the traffic at the PS is reduced by 50 %. Implementing this idea requires careful system design and route planning, relying upon the inherent determinism of satellite movement.

The objective of this paper is to present a system design for SFL within a LEO satellite mega-constellation in which the satellites within the same orbital plane are connected via ISLs to the adjacent nodes, forming a ring network. Each satellite has the capability to communicate with an out-of-orbit entity, such as GS, that orchestrates the training process. The external orchestration is needed as the satellites are not necessarily able to communicate across orbital planes; at the same time it creates a connectivity bottleneck. The contributions of this paper are:

- Design of a distributed system for SFL, supporting client clustering, synchronous and asynchronous orchestration, and consistent decentralized routing decisions.
- Development of a communication scheme for SFL that takes advantage of intra-orbit ISLs. Due to the co-design of predictive routing and in-network aggregation, the convergence time is reduced markedly, while not increasing the communication load. We have also devised a failure handling procedure.
- Extension of the proposed communication scheme to accommodate gradient sparsification and in-network aggregation for bandwidth-efficiency. This includes the development of a novel estimator for the number of non-zero elements in the sum of sparse vectors.
- An effective method to prevent biased solutions for asynchronous aggregation in satellite constellations connected with ISLs. This is necessary after improving connectivity with ISLs, as simple opportunistic scheduling can result in a small subset of clusters dominating the training process for several hours.
- We have evaluated the performance of the proposed system in several setups. The numerical results highlight a major increase in convergence speed ($\sim$7 times) due to the use of ISLs and $\sim$10 times reduction of communication load based on our in-network aggregation approach.

We remark that the proposed system is agnostic to the actual federated optimization (FO) procedure, as long as it supports partial aggregation as discussed in Section IV-A. Hence, the system performance can be further improved using conventional fine-tuning of FO algorithms and ML model-specific hyperparameters [18]. Finally, we note that the primary performance metric in this paper is the convergence time measured by a (simulated) wall clock. Conventional metrics for FL algorithms are model test accuracy versus the number of global iterations or the number of gradient computations. This is sensible as the focus in these studies is on improved computational efficiency. Instead, the key challenge that sets SFL apart from conventional scenarios is the connectivity bottleneck implied by the laws of orbital mechanics, potentially leading to extensive delays between iterations. Success in overcoming these obstacles is best measured in the number of global iterations the distributed system can manage within a certain period. As with any communication system, another important metric is bandwidth-efficiency, especially in GSLs.

A. Related Work

A detailed technical model of SFL was introduced in [10], while [6] considered it in a more general context of satellite-based computing networks. A broad overview of the different scenarios encountered in SFL, together with a discussion of the technical challenges for each scenario, is presented in [7]. The technical details of several of the ideas sketched in [7] are provided in this paper. Aerial FL [19] is a setup that is complementary to SFL, in which the PS is operated within satellites or aerial stations to orchestrate a planet-side FL process. It also contains scenarios where the aerial stations act as clients participating in the training process. The focus of [19] is, however, on using non-terrestrial networks as access network for terrestrial FL nodes. Closely related to that are the satellite-assisted internet of remote things system architectures considered in [20] and [21], where satellites serve as PS and access network, respectively.

A coarse classification of current work on SFL can be made based on the presence and usage of ISLs. The model in [10] considers the case without ISLs, identifies the connectivity bottleneck, and proposes a satellite-specific asynchronous FL algorithm as solution. This work is extended in [22] with a scheduling algorithm that exploits the inherent determinism of satellite trajectories. This scheduler can be combined with the clustered approach to SFL presented here. Indeed, while the current work aims at improving worker-PS connectivity to facilitate faster convergence and, optimally, synchronous orchestration, the algorithm in [22] focuses on reducing staleness by leveraging on the predictable connectivity. Another scheduling approach based on buffered asynchronous FL is proposed in [23], aiming to balance local model staleness and idle times. Lacking a thorough benchmark against the state-of-the-art, the gain of the complicated scheduling algorithm in [23] remains an open question. Staleness in asynchronous SFL is further investigated in [24], which introduces an asynchronous update rule based on the notion that staleness effects in SFL are similar in consecutive training epochs. In [25], the connection bottleneck is tackled by combining synchronous orchestration with a dynamic aggregation rule that ignores stragglers. However, the primary reason for the feasibility of synchronous terrestrial orchestration is the usage of multiple geographically distributed GSs that act as a distributed PS. While the authors observe correctly that latency between GSs is small compared to GSLs, mechanisms to either ensure consistency between GSs or a hierarchical FL approach would be required in a practical system implementation of [25].

An alternative means to improving connectivity is the usage of ISLs instead of a distributed PS, as proposed in the conference version of this paper [1]. The communication strategy from [1] is adopted in [26] and combined with a distributed PS implemented within interconnected high-altitude platforms (HAPs). This is extended in [27] to asynchronous aggregation with multiple HAPs. A modified version of [1] is proposed in [28], consisting of a decentralized implementation of predictive routing, which might lead to inconsistent routing decisions, and the absence of incremental aggregation (see Section IV-A), which leads to a quadratic traffic growth within each orbital plane. FL in a fully connected, ultra-dense satellite constellation, employing both intra- and inter-orbit ISLs, is considered in [29]. There, only satellites within close vicinity of the GS are participating in each epoch of the training process. A decentralized learning system, without PS, leveraging inter- and intra-orbit ISLs is proposed in [30]. Decentralized learning in LEO satellite constellations under very realistic satellite system assumptions is considered in [31] for a semi-supervised classification task. Finally, [32] treats FL-aware routing and resource allocation for SFL.

B. Organization

The rest of this paper is organized as follows. In Section II, we present the system model, which includes models for the constellation, communications, and computation. Section III describes the different orchestration approaches at the PS and rigorously defines its operation. In Section IV, the client process is defined and an efficient communication scheme for SFL is developed. Section V discusses incorporating gradient compression in the communication scheme for increased bandwidth efficiency, using gradient sparsification as an example. Finally, we evaluate the performance of our framework in Section VI and conclude the paper in Section VII.

C. Notation

Scalars are represented in a normal font x, while vectors in bold $\boldsymbol{x}$. The Euclidean norm of a vector $\boldsymbol{x}$ is $||\boldsymbol{x}||$. The angle between two vectors $\boldsymbol{x_1}$ and $\boldsymbol{x_2}$ is $\angle(\boldsymbol{x_1}, \boldsymbol{x_2})$. Sets are denoted by $\mathcal{X}$, and the cardinality of $\mathcal{X}$ is $|\mathcal{X}|$. Removing an element x_i from the set $\mathcal{X}$ is denoted by $\mathcal{X} \setminus \{x_i\}$. In a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ with vertices $\mathcal{V}$ and edges $\mathcal{E}$, the neighborhood of any vertex $v \in \mathcal{V}$ is denoted as $\mathcal{N}(v)$. If $\mathcal{G}$ is directed, $\mathcal{N}^-(v)$ and $\mathcal{N}^+(v)$ denote the incoming and outgoing neighborhood of v, respectively. The operators $\Pr(\cdot)$, $\mathbb{E}$, and $I(\cdot)$ are the probability, expected value, and indicator function, respectively. Integer rounding is denoted by $\lfloor\cdot\rfloor$ and $\lceil\cdot\rceil$.

II. System Model

The constellation has P orbital planes, where orbit p, $p \in \{1, \dots, P\}$, contains K_p satellites $\mathcal{K}_p$ such that $\mathcal{K}_p \cap \mathcal{K}_q = \emptyset$, for all $q \neq p$. The set of all satellites within the constellation is denoted as $\mathcal{K} = \bigcup_{p=1}^{P} \mathcal{K}_p = \{k_{1,1}, \dots, k_{P,K_P}\}$, with the total number of satellites $K = \sum_{p=1}^{P} K_p$. Each satellite k follows a trajectory $\boldsymbol{s}_k(t)$ around Earth with orbital period $T_p = 2\pi\sqrt{\frac{a_k^3}{\mu}}$, i.e., $\boldsymbol{s}_k(t) \approx \boldsymbol{s}_k(t + nT_p)$ for all integer n, where a_k is the semi-major axis of satellite k and $\mu = 3.98 \times 10^{14}\,\mathrm{m}^3/\mathrm{s}^2$ is the geocentric gravitational constant. For circular orbits, the semi-major axis is $a_k = r_E + h_k$ with h_k being the satellite's altitude above the Earth's surface and $r_E = 6371\,\mathrm{km}$ the Earth radius. The satellites within an orbital plane p follow the same trajectory and are assigned unique IDs $\mathcal{K}_p = \{k_{p,1}, \dots, k_{p,K_p}\}$ such that satellite $k_{p,i+1}$ is behind $k_{p,i}$. If the satellites are distributed equidistantly within the orbital plane, we have

$\boldsymbol{s}_{k_{p,1}}(t) \approx \boldsymbol{s}_{k_{p,2}}(t - T_p/K_p) \approx \boldsymbol{s}_{k_{p,3}}(t - 2T_p/K_p) \approx \ldots$.[1] Coordinates are in an Earth-centric reference frame.

A. Communication Model

The number of communication devices per satellite depends on the specific mission requirements. In this paper, we assume each satellite has three communication devices, two of which are for intra-orbit communications. The third one is for communication outside of its orbital plane, either a GSL if the PS is located in a GS or an ISL towards a satellite in another orbit if the PS is located in a satellite. It is worth noting that if the proposed schemes are applied for satellites equipped with more than three communication devices, only the three required ones are used. Communication with an Earth-based GS is feasible if the satellite is visible from the GS at an elevation angle $\frac{\pi}{2} - \angle(\boldsymbol{s}_{GS}, \boldsymbol{s}_k(t) - \boldsymbol{s}_{GS}) \geq \alpha_e$, where α_e is the minimum elevation angle [3], [33] and $\boldsymbol{s}_{GS}$ is the position of the GS. While this condition is satisfied, we assume communication is possible at a fixed rate.[2] For ISLs, communication is feasible if the line of sight is not obstructed by the Earth. With lower atmospheric layers degrading the link quality, a sensible assumption is to consider an ISL to be feasible if it does not enter the atmosphere below the thermosphere [14], starting at approximately 80 km above sea level. This translates to a maximum slant range $d_{\mathrm{Th}}(t; k_1, k_2) = \sqrt{\|\boldsymbol{s}_{k_1}(t)\|^2 - r_T^2} + \sqrt{\|\boldsymbol{s}_{k_2}(t)\|^2 - r_T^2}$ for any two satellites k_1, k_2, where $r_T = r_E + 80\,\mathrm{km}$. For circular orbits, this threshold is the constant $d_{\mathrm{Th}}(k_1, k_2) = \sqrt{(h_{k_1} + r_E)^2 - r_T^2} + \sqrt{(h_{k_2} + r_E)^2 - r_T^2}$. We assume communication at a fixed rate is possible between satellites k_1 and k_2 if their distance at time t is $d(t; k_1, k_2) \leq d_{\mathrm{Th}}(t; k_1, k_2)$.

The most stable link usage is to connect each satellite to its two closest orbital neighbors, effectively forming a ring network [16]. The two neighbors of satellite $k_{p,i}$ are $\mathcal{N}(k_{p_i}) = \{k_{p,i-1}, k_{p,i+1}\}$. Here, the satellite indices $i-1$ and $i+1$ are modulo K_p, which is a convention we will adopt throughout this paper until further notice. Following the previous discussion, the data rate between any two satellites $k_1, k_2 \in \mathcal{K}$ within the constellation is fixed to $r(t; k_1, k_2) = \rho_{k_1,k_2}$ if $k_2 \in \mathcal{N}(k_1)$ and communication is feasible, and zero otherwise. For the out-of-orbit communication link, we assume there is a single communication partner of interest, denoted as the PS. Then, the rate function of satellite k's, $k \in \mathcal{K}$, link towards the PS is similarly defined as $r_{\mathrm{PS}}(t; k) = \rho_{k,\mathrm{PS}}$ if communication is feasible and zero otherwise.

B. Computation Model

The satellites within the constellation collaboratively train a ML model from data $\mathcal{D}$ collected at the satellites. The ML model is known at all satellites and fully defined by its model parameter vector $\boldsymbol{w} \in \mathbb{R}^{n_d}$. The goal is to find a solution to the optimization problem

$$\min_{\boldsymbol{w}} \quad F(\boldsymbol{w}), \tag{1}$$

where the global loss function $F(\boldsymbol{w}) = \frac{1}{D}\sum_{\boldsymbol{x}\in\mathcal{D}} f(\boldsymbol{x}; \boldsymbol{w})$, with $D = |\mathcal{D}|$, measures the performance of the ML model with respect to the data set $\mathcal{D}$ for a certain set of model parameters $\boldsymbol{w}$ and a potentially nonconvex per-sample loss function $f(\boldsymbol{x}; \boldsymbol{w})$.

Regarding (1), we make two fundamental assumptions: *(i)* the computational resources at the satellites are limited such that a distributed solution of (1) is necessary; *(ii)* the data set $\mathcal{D}$ is distributed across the constellation and communicating this data is not feasible; i.e., each satellite has a local data set $\mathcal{D}_k$ such that $\mathcal{D} = \bigcup_{k\in\mathcal{K}} \mathcal{D}_k$, which is not shared with any other participants in the training process. Due to the local data sets assumption and the limited connectivity, this distributed ML scenario is an instance of a FL [34]. However, in contrast to a conventional FL setup, here client participation is under central control, the connectivity is mostly deterministic and predictable, and the number of devices is orders of magnitude lower.

In FL, (1) is solved iteratively using a modified distributed stochastic gradient descent (DSGD) procedure.[3] This algorithm is motivated by the linearity of the gradient and the observation that the objective function is separable as $F(\boldsymbol{w}) = \frac{1}{D}\sum_{k\in\mathcal{K}} D_k F_k(\boldsymbol{w})$ with $F_k(\boldsymbol{w}) = \frac{1}{D_k}\sum_{\boldsymbol{x}\in\mathcal{D}_k} f(\boldsymbol{x}; \boldsymbol{w})$ and $D_k = |\mathcal{D}_k|$. The optimization process is orchestrated by a central PS, which maintains the current iteration of the global model parameters $\boldsymbol{w}$, distributes these to the clients for further refinement, and collects the results. In contrast to conventional FL, we assume that every client participates in every iteration of the solution process due to the relatively small number of clients. Since satellites within the constellation are only connected to their orbital neighbors, communication across orbital planes is only feasible via the out-of-constellation link. Hence, it is only natural to assume the PS to be either located in a GS or in a satellite outside the constellation. Details of the PS operation will be discussed in Section III.

1) Client Operation: Each satellite $k \in \mathcal{K}$ runs a process to handle all application-layer communications related to the FL training. This procedure will be designed in Section IV. Upon receiving an updated global parameter vector $\boldsymbol{w}^n$, it launches the learning procedure outlined in Algorithm 1 in a separate thread for concurrent execution. This learning procedure then computes an update to $\boldsymbol{w}^n$ based on the local data set $\mathcal{D}_k$. After initialization in line 2, the loss function $F_k(\boldsymbol{w})$ is minimized in I local epochs using permutation-based mini-batch stochastic gradient descent (SGD) in lines 3–10. More specifically, in each local epoch, satellite k shuffles its data set $\mathcal{D}_k$ randomly and then divides it into mini-batches of size $|\mathcal{B}|$. Subsequently, it performs a gradient step based on

[1] Under the assumption of perfect Keplerian orbits, we can replace '$\approx$' with '$=$' in all statements on trajectories. With real-world orbits being subject to orbital perturbations and station keeping maneuvers, these relations do not hold exactly and we only state them here to introduce notation and some general concepts on an abstraction level suitable for this paper.

[2] The fixed rate assumption in the out-of-constellation link is made for the sake of simplicity. It has no direct impact on the developed SFL framework and can be relaxed easily to a variable rate if the system supports adaptive coding and modulation.

[3] While this paper is focusing on DSGD-based optimization, the extension to many other iterative distributed optimization algorithms for training ML models is straightforward.

Algorithm 1 Satellite Learning Procedure

1: **procedure** ClientOpt($\boldsymbol{w}$)
2: **initialize** $\boldsymbol{w}_k^{n,0} = \boldsymbol{w}^n, \quad i = 0$
3: **for** I epochs **do** ▷ I epochs of mini-batch SGD
4: $\tilde{\mathcal{D}}_k \leftarrow$ Randomly shuffle $\mathcal{D}_k$
5: $\mathscr{B} \leftarrow$ Partition $\tilde{\mathcal{D}}_k$ into mini-batches of size B
6: **for** each batch $\mathcal{B} \in \mathscr{B}$ **do**
7: $\boldsymbol{w}_k^{n,i+1} \leftarrow \boldsymbol{w}_k^{n,i} - \frac{\eta}{|\mathcal{B}|} \nabla_{\boldsymbol{w}} \left(\sum_{\boldsymbol{x} \in \mathcal{B}} f(\boldsymbol{x}; \boldsymbol{w}) \right)$
8: $i \leftarrow i + 1$
9: **end for**
10: **end for**
11: $\boldsymbol{w}_k^n \leftarrow \boldsymbol{w}_k^{n,i}$
12: $\boldsymbol{g}_k(\boldsymbol{w}_k^n) \leftarrow \boldsymbol{w}_k^n - \boldsymbol{w}_k^{n,0}$ ▷ Compute effective gradient
13: $\bar{\boldsymbol{g}}_k(\boldsymbol{w}_k^n) \leftarrow$ CompressGradient($\boldsymbol{g}_k(\boldsymbol{w}_k^n)$) ▷ Apply gradient compression (e.g., sparsification)
15: **return** $D_k \bar{\boldsymbol{g}}_k(\boldsymbol{w}_k^n)$
16: **end procedure**

the empirical average per-sample loss for each mini-batch, i.e.,

$$\boldsymbol{w}_k^{n,i+1} \leftarrow \boldsymbol{w}_k^{n,i} - \frac{\eta}{|\mathcal{B}|} \nabla_{\boldsymbol{w}} \left(\sum_{\boldsymbol{x} \in \mathcal{B}} f(\boldsymbol{x}; \boldsymbol{w}) \right), \tag{2}$$

where η is the learning rate. Instead of directly transmitting the updated model parameters $\boldsymbol{w}_k^n$, as represented in line 11, the effective gradient $\boldsymbol{g}_k(\boldsymbol{w}_k^n)$ is computed in line 12. While both representations $\boldsymbol{w}_k^n$ and $\boldsymbol{g}_k(\boldsymbol{w}_k^n)$ are theoretically equivalent, the latter is often easier to compress. This is optionally done in line 13 by calling the procedure CompressGradient, which will be defined in Section V. Without compression, CompressGradient is simply the identity function, i.e., $\bar{\boldsymbol{g}}_k(\boldsymbol{w}_k^n) = \boldsymbol{g}_k(\boldsymbol{w}_k^n)$. In a slight modification of the usual approach, Algorithm 1 returns the compressed effective gradient scaled by D_k in line 15.

Note that the subsequent results do not rely on the explicit implementation of the SGD procedure in lines 3–10. The only requirement is that the updated model parameters can be incorporated into the global model based on effective gradients using the update rules presented in Section III. However, we will use the implementation in Algorithm 1 throughout this paper.

For the routing procedure developed in Section IV, we will require an accurate estimate of the time to run Algorithm 1. Apart from scheduling delays due to multi-task computing, the runtime directly depends on the number of processor cycles for each operation in Algorithm 1. These are hardware-dependent, deterministic, and can be determined offline before deployment [35]. First, consider a single epoch. The local data set is shuffled and divided into $\lceil \frac{D_k}{B} \rceil$ mini-batches. This process takes c_{epoch} CPU cycles per sample. Computation of the stochastic gradients requires, in total, $n_d D_k c_{\text{s}}$ CPU cycles, where c_{s} is the number of cycles to process one sample for a single dimension of $\boldsymbol{w}$. Executing the gradient step takes, per mini-batch, $n_d c_{\text{step}}$ clock cycles. Thus, one epoch requires a total of $D_k c_{\text{epoch}} + n_d D_k c_{\text{s}} + \lceil \frac{D_k}{B} \rceil n_d c_{\text{step}}$ CPU cycles. After I epochs, a final gradient step taking $n_d c_{\text{step}}$ cycles is performed to compute the effective gradient. The gradient compression takes an additional c_{compress} cycles (see Section V) and is assumed zero if no compression is used. Passing the result to the communication stack takes, together with other overhead occurring due to, e.g., process setup and termination, a total of c_{os} cycles. The runtime of Algorithm 1 is

$$t_l(k) = \frac{I D_k (c_{\text{epoch}} + n_d c_{\text{s}}) + c_{\text{step}} n_d \left(I \lceil \frac{D_k}{B} \rceil + 1 \right) + c_{\text{compress}} + c_{\text{os}}}{\nu_k}, \tag{3}$$

where ν_k is the CPU frequency at satellite k.

III. Orchestration of Satellite Federated Learning

FL uses a conventional client-server architecture to orchestrate the training process. While the clients compute the stochastic gradient steps for (1), the parameter server (PS) is responsible for aggregating these gradient steps, updating the global model parameters, and distributing the updated parameter vector to the clients. In a conventional FL setup, the PS is also responsible for client scheduling, modeled as an uniform sampling of a subset of clients in each global iteration. Given that FL operates on a massive number of clients, this is equivalent to a two-stage SGD step, the first step being a client selection and the second computation. Instead, SFL operates with significantly fewer (orders of magnitude) clients and each client has a much larger share of the total data, exhibiting a non-negligible contribution to the unbiased model convergence. Hence, it is reasonable that all clients participate in every global iteration.

Connectivity towards the workers is necessary for synchronization of the training process. In SFL, the communication window from a single LEO satellite towards a ground-based PS is usually in the order of a few minutes, followed by an offline period due to Earth blockage, ranging from one orbital period up to several hours. As shown in [10], the conventional FedAvg operation of collecting all local updates before creating a new global model iteration leads to severe delays in the training process. This bottleneck can be partially mitigated by modifying FedAvg for *asynchronous* operation [10]. Compared to synchronous FL, asynchronous FL has a slower convergence speed in terms of gradient steps. For ground-orchestrated SFL without ISLs, this decrease is greatly outweighed by the reduction in the delay between gradient steps, resulting in much faster overall convergence speed, measured in wall time. Leveraging ISLs, the optimal PS operation very much depends on the PS location and resulting connectivity patterns [7]. We will introduce both orchestration approaches in a unified manner in Section IV and V. The actual aggregation rule at the PS is easily exchangeable as long as additivity of individual client updates holds. This broadens the contribution of Section IV, as it allows to improve the PS operation while preserving the dense connectivity patterns enabled by ISLs.

A. Synchronous Orchestration

A synchronous FL PS, exemplified by FedAvg [34], repeats the following steps until a termination criterion for the global model is met: 1) Transmit the global model parameters $\boldsymbol{w}$ to the scheduled clients; 2) Wait for the clients to run Algorithm 1 and return their results; 3) Aggregate the received gradients and update the global model parameters. The difference between FL algorithms is in the computation of gradients in Algorithm 1 and the update rule in Step 3). While we focus on FedAvg here, the extension to many other algorithms is trivial.

Consider global iteration n and assume all clients are scheduled to participate in this iteration. Plain FedAvg implements the update rule $\boldsymbol{w}^{n+1} = \frac{1}{D}\sum_{k=1}^{K} D_k \boldsymbol{w}_k^n$. An equivalent update rule based on effective gradients $\boldsymbol{g}_k(\boldsymbol{w}_k^n)$, as represented in Algorithm 1, is

$$\begin{aligned}\boldsymbol{w}^{n+1} &= \frac{1}{D}\sum_{k=1}^{K} D_k \left(\boldsymbol{w}_k^n - \boldsymbol{w}_k^{n,0}\right) + \frac{1}{D}\sum_{k=1}^{K} D_k \boldsymbol{w}_k^{n,0} \\ &= \boldsymbol{w}^n + \frac{1}{D}\sum_{k=1}^{K} D_k \boldsymbol{g}_k(\boldsymbol{w}_k^n). \end{aligned} \tag{4}$$

A common generalization is to add a global server learning rate η_s to this update rule, i.e., $\boldsymbol{w}^{n+1} = \boldsymbol{w}^n - \eta_s \boldsymbol{\gamma}^n$, where $\boldsymbol{\gamma}^n = -\frac{1}{D}\sum_{k=1}^{K} D_k \boldsymbol{g}_k(\boldsymbol{w}_k^n)$ [18].

The complete algorithm for PS operation in the SFL scenario, in relation to the client update procedure in Algorithm 1, is given in Algorithm 2. This is a modified version of the delay-tolerant FedAvg implementation in [10]. It is initialized in line 2, where a set of client clusters $\mathscr{C}$ is defined. This is required for the efficient use of ISLs in Section IV. The idea is to treat each cluster as if it were an individual user, receiving the current parameter vector only once and also delivering a single effective gradient per global iteration. For now, as well as for scenarios without ISLs, it can be assumed that satellite/client k is mapped to cluster $\mathcal{C}_k$, with P clusters in total. The PS maintains the training process until the sequence $\{\boldsymbol{w}^1, \boldsymbol{w}^2, \dots\}$ satisfies the termination criterion in line 3. Each iteration of this outer loop corresponds to a global iteration, counted as n. The sets $\mathcal{T}^n$ and $\mathcal{R}^n$ track the transmission of $\boldsymbol{w}^n$ and reception of the gradient update to $\boldsymbol{w}^n$ per client group, respectively. Hence, the inner loop in line 5–21 runs until updates have been received from all clusters in $\mathscr{C}$. This loop blocks until a satellite connects to the PS in line 6. Note that this could also mean continuing a connection that was not terminated the previous iteration. A message from the satellite is received that either requests the transmission of the current parameter vector or contains the update from the satellite's cluster.

If the satellite requests transmission of $\boldsymbol{w}^n$ and it was not yet delivered to its cluster, it will be transmitted in line 10. Successful reception must be acknowledged by the satellite. A possible implementation is the Bundle protocol's custody transfer [36], [37]. Then, this cluster is marked as having received the transmission in line 11. All subsequent transmission requests for $\boldsymbol{w}^n$ to satellites of this cluster will be rejected by terminating the connection in line 19. If the satellite transmits an update $\boldsymbol{\gamma}_p$ to $\boldsymbol{w}^n$ and the cluster has not yet transmitted an update in the iteration, the compression applied in line 13 of Algorithm 1 is decoded in line 13. Without compression, UNCOMPRESSGRADIENT is the identity function. Then, the update rule (4) is applied in line 14. incrementally (see also line 4) and relies on the received gradient already being scaled by D_k. Unless the current iteration is finished, the connection is terminated in Algorithm 2.

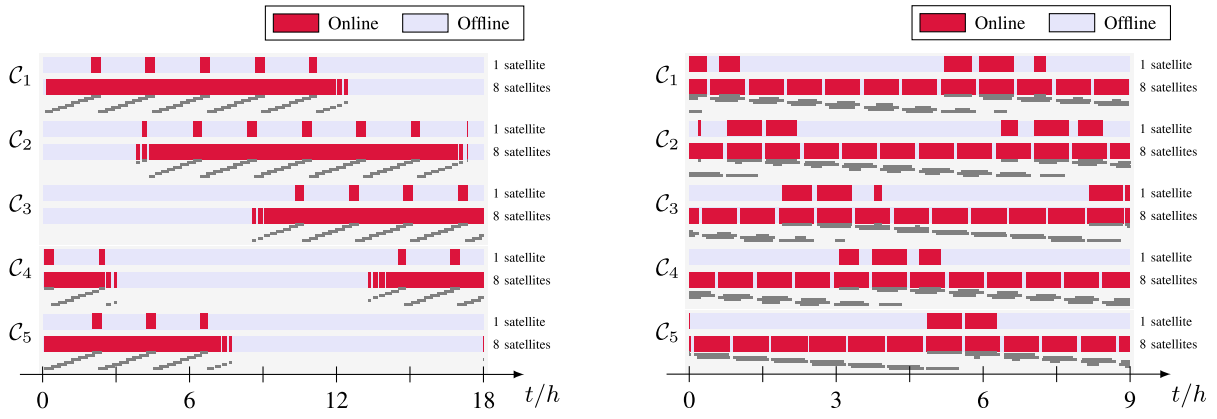
Algorithm 2 Synchronous PS Operation

1: **initialize** global iteration $n = 0$, model $\boldsymbol{w}^0$,
2: and client clusters $\mathscr{C} = \{\mathcal{C}_1, \mathcal{C}_2, \dots\}$
3: **while** termination criterion not met **do**
4: Set $n \leftarrow n+1$, $\mathcal{T}^n = \mathcal{R}^n = \emptyset$, $\boldsymbol{w}^n \leftarrow \boldsymbol{w}^{n-1}$
5: **while** $|\mathcal{R}^n| < |\mathscr{C}|$ **do**
6: Wait until connection from satellite k:
7: Receive message m
8: Find p such that $k \in \mathcal{C}_p$
9: **if** $p \notin \mathcal{T}^n$ **and** m is request for data **then**
10: Transmit $\boldsymbol{w}^{n-1}$ to satellite k
11: Upon successful transfer, add p to $\mathcal{T}^n$
12: **else if** $p \notin \mathcal{R}^n$ **and** m contains gradient update $\boldsymbol{\gamma}_p$ **then**
13: $\tilde{\boldsymbol{\gamma}}_p \leftarrow$ UNCOMPRESSGRADIENT$(\boldsymbol{\gamma}_p)$
14: $\boldsymbol{w}^n \leftarrow \boldsymbol{w}^n + \frac{1}{D}\tilde{\boldsymbol{\gamma}}_p$
15: Add p to $\mathcal{R}^n$
16: Acknowledge reception to k
17: **end if**
18: **if** $|\mathcal{R}^n| < |\mathscr{C}|$ **then**
19: Terminate connection
20: **end if**
21: **end while**
22: **end while**

B. Asynchronous Orchestration

As opposed to synchronous operation, in asynchronous PS operation, the PS does not delay the global model update until all clients have delivered their local updates. Instead, it opportunistically incorporates received gradient updates into a new iteration of the global model, which is subsequently distributed to the clients. Consequently, clients simultaneously operate on different version of the model parameters and gradient updates are typically based on an outdated version of the model parameters, reducing the convergence speed and, potentially, numerical problems. For SFL, this *stalenenss*, i.e., the age of the local model with respect to the current global model, is bounded due to the quasiperiodicity of this scenario. Hence, we are operating in the partially asynchronous domain, which leads to, generally speaking, much better convergence properties than in totally asynchronous scenarios with unbounded delays [38].

Inspired by [10], an asynchronous version of Algorithm 2 is proposed in Algorithm 3. It consists of the same functional blocks as Algorithm 2: Wait for incoming connections in lines 4–6, incorporate and acknowledge gradient updates in lines 8–11, run until global convergence (lines 13–21), and transmit the current version of the model parameters in



(a) PS located in a GS in Bremen, Germany.

(b) PS located in a satellite in equatorial circular orbit with altitude 500 km.

Fig. 2. Connectivity towards the PS from within a 60°: 40/5/1 Walker delta constellation. That is, a constellation of 40 satellites having 60° inclined circular orbits and altitude 2000 km. The satellites are distributed evenly among five orbital planes, which are spaced equidistantly around Earth. Clusters $\mathcal{C}_p$ are defined as either a single satellite per orbital plane or all satellites within an orbital plane. In the second case, a cluster is considered having a connection to the PS if at least one satellite of the cluster can communicate with the PS. Per-satellite connectivity towards the PS is displayed in gray below the cluster connectivity.

Algorithm 3 Asynchronous PS Operation

1: **initialize** global iteration $n = 0$, model $\boldsymbol{w}^0$, $\mathcal{A} = \mathcal{B} = \emptyset$,
2: and client clusters $\mathscr{C} = \{\mathcal{C}_1, \mathcal{C}_2, \dots\}$
3: **loop**
4: Wait until connection from satellite k:
5: Receive message m
6: Find p such that $k \in \mathcal{C}_p$
7: **if** $p \in \mathcal{A}$ and m contains gradient update $\boldsymbol{\gamma}_p$ **then**
8: $\tilde{\boldsymbol{\gamma}}_p \leftarrow \textsc{UncompressGradient}(\boldsymbol{\gamma}_p)$
9: $\boldsymbol{w}^n \leftarrow \boldsymbol{w}^n + \frac{1}{D}\tilde{\boldsymbol{\gamma}}_p$
10: $\mathcal{A} \leftarrow \mathcal{A} \setminus \{p\}$
11: Acknowledge reception to k
12: Wait for new message m
13: **if** termination criterion is met **then**
14: $\mathcal{B} \leftarrow \{1, 2, \dots, P\} \setminus \mathcal{A}$
15: **if** $\mathcal{A} = \emptyset$ **then**
16: Terminate loop (and connection)
17: **end if**
18: **else**
19: $\mathcal{B} \leftarrow \emptyset$
20: **end if**
21: **end if**
22: **if** $p \notin \mathcal{A} \cup \mathcal{B}$ and m is request for data **then**
23: Transmit $\boldsymbol{w}^{n-1}$ to satellite k
24: Upon successful transfer, add p to $\mathcal{A}$
25: **end if**
26: Terminate connection
27: **end loop**

lines 23–24. The main difference is that, in Algorithm 2, the nested loop ensures that every cluster adds their update to the global model before this new version is transmitted to any client. Instead, Algorithm 3 directly incorporates every received update and immediately starts using this new version to answer requests for data. To avoid race conditions within the clusters, Algorithm 3 uses the set $\mathcal{A}$ to track active clusters, i.e., clusters that received the model parameters and did not yet return an update. Another difference is that the global termination criterion, after being first met, might not remain valid after receiving the outstanding updates from active clusters. To handle this, whenever a new global model satisfies the termination criterion, all inactive clusters in $\mathcal{B}$, including the currently connected, are blocked from further computations. This is repeated until no active cluster remains. If the termination criterion is violated for any update, all blocks are removed in line 19 and the algorithm resumes normal operation.

IV. Intra-Orbit Aggregation for Satellite FL

Convergence in SFL is mainly impaired by the connectivity bottleneck between satellites and the PS. We have discussed algorithmic approaches to this obstacle in the previous section. With the availability of ISLs, these can be complemented by multi-hop routing to close gaps in connectivity. Figure 2 illustrates the potential gain for a constellation having five orbital planes, each with eight satellites, displaying the connection of a single satellite from each orbital plane towards a PS. Using multi-hop routes within the orbital plane leads to significantly prolonged online periods as compared to the sporadic point-to-point connectivity. For simplicity, we focus on synchronous orchestration first and review the specifics for asynchronous orchestration in Section IV-F.

Communication for FL involves two primary tasks, parameter vector distribution and collection of gradient updates. A direct approach to implementing these, leveraging ISLs for multi-hop routing, are conventional delay-tolerant networking techniques for satellite networks, e.g., contact graph routing in combination with the Bundle protocol [39], [40], [41]. Then, parameter vector distribution is a multicast message from the PS towards all satellites in $\mathcal{K}$, while gradient updates are communicated as unicast messages from individual satellites towards the PS. However, the number of these unicast transmissions scales quadratically in the number of satellites

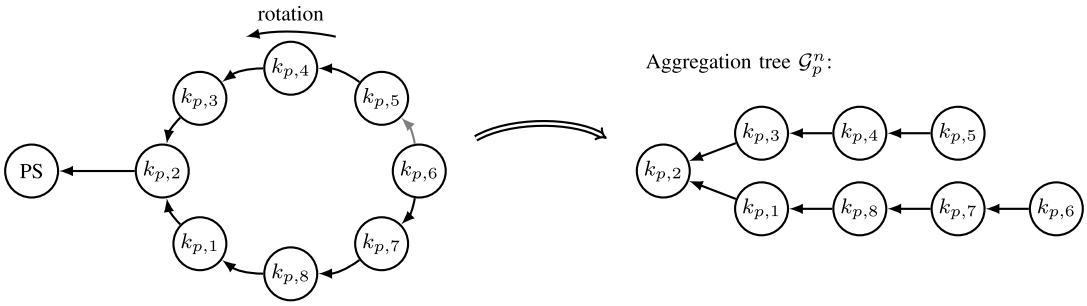


Fig. 3. Routing tree for incremental aggregation in orbital plane p. Satellite $k_{p,2}$ acts as sink node. Satellite $k_{p,6}$ has two shortest-path routes to sink. This is resolved unambiguously by the routing algorithm.

per orbital plane. This can be reduced to a linear increase with in-network aggregation.

A. Incremental Aggregation

Consider the PS update rule from (4), within the global iteration n. The PS is primarily interested in the sum of effective gradients instead of individual gradient updates. Hence, the updates from all satellites within an orbital plane $\mathcal{K}_p$ can be collected at a single satellite $s_p^n \in \mathcal{K}_p$ for transmission to the PS. Instead of transmitting the individual gradient updates $\{D_k \bar{\boldsymbol{g}}_k(\boldsymbol{w}_k^n)\}_{k\in\mathcal{K}_p}$, this satellite sends the linear combination $\sum_{k\in\mathcal{K}_p} D_k \bar{\boldsymbol{g}}_k(\boldsymbol{w}_k^n)$ to the PS. Following Section III, the PS then treats the satellites in $\mathcal{K}_p$ as the client cluster $\mathcal{C}_p$ and incorporates their joint update.

This approach can be extended to intra-cluster communication. Suppose the sink satellite s_p^n is known to all satellites in $\mathcal{K}_p$. Then, each satellite can easily determine a shortest-path in-tree, rooted at and oriented towards s_p^n, for $\mathcal{K}_p$ [42, §9.6]. For K_p odd, this tree is unique. For K_p even, two equal length paths exist for the satellite farthest away from s_p^n. Denote this satellite $k_{p,i}$ and resolve this ambiguity by choosing the tree where $k_{p,i+1}$ is the parent node of $k_{p,i}$. We denote this directed graph as the *aggregation tree* $\mathcal{G}_p^n = (\mathcal{K}_p, \mathcal{E}_p^n)$, with vertex set $\mathcal{K}_p$ and edge set $\mathcal{E}_p^n$. Its computation is illustrated in Figure 3.

Let $\mathcal{N}^-_{\mathcal{G}_p^n}(k) = \{y \in \mathcal{K}_p : (y,k) \in \mathcal{E}_p^n\}$ be the incoming neighborhood of k on $\mathcal{G}_p^n$. Based on $\mathcal{G}_p^n$, each satellite $k \in \mathcal{K}_p$ knows exactly which gradient updates it needs to forward. After local computation completes, node k waits until all updates from the satellites in $\mathcal{N}^-_{\mathcal{G}_p^n}(k)$ are received via ISLs. This might have already happened during the local computation phase. Then, the outgoing partial aggregate of satellite k is computed as

$$\boldsymbol{\gamma}_k^n = D_k \bar{\boldsymbol{g}}_k(\boldsymbol{w}_k^n) + \sum_{y\in\mathcal{N}^-_{\mathcal{G}_p^n}(k)} \boldsymbol{\gamma}_y^n \tag{5}$$

and transmitted via ISL to the next hop $p \in \mathcal{N}^+_{\mathcal{G}_p^n}(k) = \{y \in \mathcal{K}_p : (k,y) \in \mathcal{E}_p^n\}$. If k is the current[4] sink node, i.e., $|\mathcal{N}^+_{\mathcal{G}_p^n}(k)| = 0$, $\boldsymbol{\gamma}_k^n$ is the joint aggregate of client cluster $\mathcal{C}_g = \mathcal{K}_p$ and transmitted to the PS once the link becomes available.

[4]Please refer to Section IV-D for a failure handling procedure that might change the sink node after initial assignment.

B. Parameter Vector Distribution

Algorithms 2 and 3 treat client clusters as if they were a single node, i.e., the PS expects to receive a single (joint) update per client cluster and transmits the model parameter vector only once per global iteration to each cluster. The previous subsection provides ample reason for the first design choice. Transmitting the parameter vector only once is also sensible for several reasons. First of all, it is what ideally happens if the PS transmits a multicast message towards the nodes in $\mathcal{K}_p$ using the Bundle protocol. As noted before, using the Bundle protocol's custody transfer (or similar) to safeguard point-to-point transmissions involving the PS is a fundamental assumption in this paper. Second, using multicast messages and multi-hop routing considerably reduces delay and the communication effort for the PS. Finally, this provides a natural leader election mechanism for coordinating the sink node selection.

The PS sends the model parameters $\boldsymbol{w}^n$ in iteration n to a single satellite k per cluster $\mathcal{C}_p$. Satellite k is then responsible for determining an appropriate sink node s_p^n using the procedure described in Section IV-C. Subsequently, satellite k propagates its routing decision together with $\boldsymbol{w}^n$ through both ISLs to its neighbors $\mathcal{N}(k)$ and starts the local training ClientOpt($\boldsymbol{w}^n$). All other satellites, upon reception of new model parameters $\boldsymbol{w}^n$ via one of the ISLs, forward the model parameters to their next neighbor that did not receive $\boldsymbol{w}^n$ and start the local training ClientOpt($\boldsymbol{w}^n$). Subsequent receptions of $\boldsymbol{w}^n$ are silently dropped upon reception. This ensures that the propagation of $\boldsymbol{w}^n$ through the orbital plane stops once every satellite received it.

C. Predictive Routing

The final missing piece to this routing approach is a procedure to determine the sink node s_p^n. This decision must be made before the local training procedure completes on any satellite in the client cluster. From a timing perspective, the ideal sink node completes computing (5) immediately before its link towards the PS becomes available. Predicting the future state of a satellite constellation is possible with high accuracy due to the determinism of satellite movement [43], [44], [45]. Here, we assume the availability of an orbital propagator with low computational complexity and (relatively) low precision like SGP4 [44], [46]. We also assume availability of fairly recent information on the constellation state, e.g., in the form

of two-line element sets (TLEs) [46], [47], distributed by the satellite operator. The sink node can be determined from the orbital positions of the PS and satellites $\mathcal{K}_p$ at the time when intra-orbit aggregation completes. This requires a prediction of the time T_p^n to 1) distribute the model parameters; 2) compute the local updates; and 3) deliver these updates to candidate sink nodes. The computation time can be estimated using $t_l(k)$ in (3). The time to transmit a vector $\boldsymbol{v}$ via intra-orbit ISL between two satellites k_1 and $k_2 \in \mathcal{N}(k_1)$ is upper bounded as $t_c(\boldsymbol{v}, t; k_1, k_2) \leq \frac{S(\boldsymbol{v})}{\rho_{k_1,k_2}} + \frac{\max_t d(t;k_1,k_2)}{c_0}$, where c_0 is the vacuum speed of light, $S(\boldsymbol{v})$ is the storage size of $\boldsymbol{v}$ and $d(t; k_1, k_2)$ is constant in t for circular orbits. For uncompressed gradients, $S(\boldsymbol{w}^n) = n_d\omega$, where ω is the storage size of a single element of model parameter, usually 16 –bit or 32 –bit floating point number.

A precise solution for T_p^n should take into account that satellites start their learning procedure at different times depending on the number of hops the global model parameters need to travel. Even when distribution takes place over shortest-distance paths, the completion time is different for each potential sink node in $\mathcal{K}_p$. Hence, predictions on the constellation state must be made for up to K_p time instants and the computational complexity for determining the sink node scales at least linearly in K_p. A considerably simpler approach is to assume distribution takes place over shortest-distance paths and make a worst-case estimate for the aggregation phase. Combined with assuming a symmetric orbital distribution of satellites and equal ISL capabilities, i.e., $d_p = \max_t d(t; k_1, k_2)$ and $\rho_p = \rho_{k_1,k_2}$ for all $k_1 \in \mathcal{K}_p, k_2 \in \mathcal{N}(k_1)$, we obtain

$$T_p^n \leq \left\lceil \frac{K_p}{2} \right\rceil \left(\frac{S(\boldsymbol{w}^n)}{\rho_p} + \frac{d_p}{c_0} \right) + \max_{k \in \mathcal{K}_p} t_l(k) + \left\lceil \frac{K_p}{2} \right\rceil \left(\frac{\max_{k \in \mathcal{K}_p} S(\bar{\boldsymbol{g}}_k(\boldsymbol{w}_k^n))}{\rho_p} + \frac{d_p}{c_0} \right). \quad (6)$$

Under constant-length gradient compression and equal in-cluster computation times, becomes:

$$T_p^n \leq \hat{T}_p^n = t_l(k_1) + \left\lceil \frac{K_p}{2} \right\rceil \left(\frac{S(\boldsymbol{w}^n) + S(\bar{\boldsymbol{g}}_{k_1}(\boldsymbol{w}_{k_1}^n))}{\rho_p} + \frac{2d_p}{c_0} \right) \quad (7)$$

for any $k_1 \in \mathcal{K}_p$. This bound overestimates the time for communications. However, this part is likely small compared to the computation time. Taking into account the inaccuracies of orbital prediction,[5] local clock deviations, and computational delays due to, e.g., multitasking, interrupts, or priority scheduling, slightly overestimating the time to completion seems reasonable. The major advantage of using the simple estimate in (7) is that the constellation state needs only be predicted for a single time instant. Thus, the computational effort remains constant in K_p.

Based on the estimate $\hat{T}_p^n$, the satellite that took custody of $\boldsymbol{w}^n$ computes the positions of the PS and all satellites in its orbital plane at time $t_p^n + \hat{T}_p^n$, where t_p^n is the expected local time after determining s_p^n. Among the satellites in communication range of the PS, the satellite with the longest remaining window for communication is selected as sink node s_p^n. Should this window be of insufficient length to transmit the parameter vector or if no satellite is in communications range of the PS, the next satellite of that orbital plane to contact the PS is selected as sink s_p^n. Then, the custodian satellite starts the model distribution process as described in Section IV-B.

[5]TLE data has an initial accuracy of roughly 1 km and then decays quickly [46].

D. Failure Handling

If the sink node completes the incremental aggregation after its communication window to the PS has closed, a critical routing error may occur. This can happen due to random factors in learning and transmission, such as multiprocessing loads and queuing delays. As a result, the sink may fail to deliver the aggregated parameters to the PS within the designated, initially planned time.

To deal with these situations, we introduce failure handling. A straightforward scheme, termed pass-to-neighbor, could work as follows. If the sink satellite s_p^n is unable to deliver the aggregated parameters to the PS, it transfers those parameters to a neighboring satellite. If the PS is visible, this neighbor sends them to the PS; otherwise, it relays them further to the next neighbor. This process of passing to the next neighbor, within a fixed direction, continues until a satellite can forward the parameters to the PS. Pass-to-neighbor may lengthen the whole FL process if it takes an excessive time to find a satellite that can forward the aggregated parameters to the PS.

A more involved, yet practical, scheme is *determine-new-sink*. Here, after finalizing the parameter aggregation, the sink satellite s_p^n can designate a new sink $\hat{s}_p^n$, if the delay dictates that. In this regard, the sink s_p^n uses visibility of satellites and the time required for ISL transmissions to calculate the time it takes for the aggregated parameters to reach a potential new sink, chosen from the set of satellites in the orbit p, i.e. $k \in \mathcal{K}_p = \{1, 2, \cdots, K_p\}$, as

$$t(k) = t_0 + h(k, s_p^n) \left(\frac{S(\gamma_{s_p^n}^n)}{\rho_p} + \frac{d_p}{c_0} \right) + t_g, \quad (8)$$

where t_0 is the current time of s_p^n and $h(k, s_p^n)$ is the number of hops between satellite k and the sink s_p^n in the orbit. The parameter t_g is a guard time, introduced to mitigate other delays in the orbit that may occur due to the parameter transmission from s_p^n to the newly designated sink $\hat{s}_p^n$. The satellite k is chosen as the new sink $\hat{s}_p^n$ if it satisfies both the conditions of having the lowest $t(k)$, the time at which it can first visit the PS. Subsequently, the aggregated parameters along with the ID of new sink $\hat{s}_p^n$ are sent from s_p^n to $\hat{s}_p^n$. This is done via a multi-hop connection over the ISLs of satellites between s_p^n and $\hat{s}_p^n$. The new sink satellite $\hat{s}_p^n$ transmits the parameters to the PS when the PS becomes visible.

E. Algorithm

The satellite operation described above is summarized in Algorithm 4. This procedure runs until a stop message is received from the PS, either directly or via ISL. The parameter vector distribution is handled in lines 3–16. The training

Algorithm 4 Satellite Operation

1: **Initialize** global iteration $n = 0$, satellite ID $k = k_{p,i}$
2: and orbital plane ID p
3: **Wait for** incoming ISL **or** start of PS connectivity window
4: **if** received $(s_p^{n+1}, \boldsymbol{w}^{n+1})$ from satellite l **then**
5: $n \leftarrow n + 1$
6: Forward $(s_p^n, \boldsymbol{w}^n)$ to $\mathcal{N}(k) \setminus \{l\}$
7: **else if** connected to PS **then**
8: Request parameters $\boldsymbol{w}^{n+1}$ and wait for reply
9: **if** received $\boldsymbol{w}^{n+1}$ **then**
10: Acknowledge reception to PS and set $n \leftarrow n + 1$
11: Compute $t_p^n + \hat{T}_p^n$ and determine s_p^n ▷ cf. Section IV-C
12: Transmit $(s_p^n, \boldsymbol{w}_p^n)$ to $\mathcal{N}(k)$
13: **else**
14: Goto line 3
15: **end if**
16: **end if**
17: **Execute concurrently** $D_k \bar{\boldsymbol{g}}_k(\boldsymbol{w}_k^n) \leftarrow$ ClientOpt$(\boldsymbol{w}^n)$
18: Compute aggregation tree $\mathcal{G}_p^n$ ▷ cf. Section IV-A
19: **if** k is sink satellite **then**
20: Initialize failure handling ▷ cf. Section IV-D
21: **Wait for [** ClientOpt **and** results from $\mathcal{N}_{\mathcal{G}_p^n}^-(k)$ **]**
until *failure*
22: **if** failure **then**
23: Compute $\boldsymbol{\gamma}_k^n$ as in (5)
24: Calculate $\hat{s}_p^n$ based on (8)
25: Transmit $(\hat{s}_p^n, \gamma_{s_p^n}^n)$ to the next satellite
26: **else**
27: Compute $\boldsymbol{\gamma}_k^n$ as in (5)
28: **Wait for** connection to PS
29: Transmit $\boldsymbol{\gamma}_k^n$ and wait for acknowledgement
30: **end if**
31: **else**
32: **Wait for** [ClientOpt **and** results from $\mathcal{N}_{\mathcal{G}_p^n}^-(k)$] **or** $(\hat{s}_p^n, \gamma_{s_p^n}^n)$
33: **if** $(\hat{s}_p^n, \gamma_{s_p^n}^n)$ **then**
34: **if** k is $\hat{s}_p^n$ **then**
35: Goto line 28
36: **else**
37: Transmit $(\hat{s}_p^n, \gamma_{s_p^n}^n)$ to the next satellite
38: **end if**
39: **else**
40: Compute $\boldsymbol{\gamma}_k^n$ as in (5)
41: Transmit $\boldsymbol{\gamma}_k^n$ to next hop $p \in \mathcal{N}_{\mathcal{G}_p^n}^+(k)$
42: **end if**
43: **end if**
44: Goto line 3

procedure is launched concurrently, e.g., in a separate thread, in line 17. Algorithm 4 continues immediately with line 18, which starts the incremental aggregation phase by computing $\mathcal{G}_p^n$. Lines 23–25 are for failure handling.

If the satellite is the current sink node, it waits for incoming results and its own computation to finish. Should the designated PS communication window pass in the meantime, the failure handling is applied according to the procedure described in Section IV-D. It triggers the failure handling procedure in lines 23–25, which calculates the new sink. Otherwise, once all results are ready, the sink node computes the final cluster aggregate in line 27, waits for the PS to become available, and transmits the results.

All other satellites execute lines 31–43. If the local training is complete and the expected incoming partial aggregate is received, the satellite computes (5) and forwards the result to the next hop in line 41. It then returns to line 3 to wait for a new global iteration or take over as sink node. If the satellite is tasked with taking over as new sink node, it performs as line 35.

F. Asynchronous Orchestration

The primary reason for asynchronous orchestration is long connectivity gaps due to orbital mechanics. These outages are considerably reduced by the techniques developed in this section. As there is no apparent benefit of per-client asynchronous updates over using (synchronous) client clusters, the system performance under asynchronous aggregation is expected to benefit significantly from incremental intra-orbit aggregation. In fact, the proposed FL system can be implemented such that the clients are agnostic to the PS operation.

However, in some scenarios the improved connectivity can result in a large number of updates from a small group of client clusters. This can lead to heavily biased solutions and other convergence issues. One such problematic scenario could be Fig. 2a combined with a short learning time $t_l(k)$. A simple solution to this issue is to set a maximum update frequency per cluster, i.e., fix a minimum time T_u between updates. In a completely trusted system (as is likely the case in SFL), this can be implemented by 1) modifying line 11 in Algorithm 4 to use $\max\{\hat{T}_p^n,\ T_u\}$ instead of $\hat{T}_p^n$; 2) distributing the time stamp $t_p^n + T_u$ together with the model parameters and designated sink node; and 3) not sending any updates to the PS before $t_p^n + T_u$, unless this would result in missing the planned communication window (modify line 28 in Algorithm 4). Alternatively, the PS could simply refuse sending the current global model parameters to a cluster if the update frequency is too high.

V. Sparsification-Based Gradient Compression

Synchronizing large-scale ML models requires transmitting massive amounts of data. A widely employed method to reduce the communication cost is to truncate the effective gradients prior to transmission to only contain the elements with largest magnitude, while setting the other elements to zero. This is known as *sparsification* [48], [49], as the resulting vectors are communicated in sparse vector encoding. With practical sparsification ratios q in the range of 1 % to 10 %, the size of a single effective gradient vector can be reduced by approximately 80 % to 98 %.

However, while a single sparsified vector has a deterministic length of $\lfloor n_d q \rfloor$ nonzero elements, the sum of multiple sparse length-$\lfloor n_d q \rfloor$ vectors has nondeterministic length. This leads to variable transmission lengths and, hence, uncertainties in the predictive routing procedure. In this section, we develop an estimator for sparse vectors subject to incremental aggregation based on probabilistic modeling of the weight vectors. Before, we briefly review FL gradient sparsification.

A. Gradient Sparsification for FL

Each client transmits only the $\lfloor n_d q \rfloor$ largest magnitude elements from its effective gradient vector $\boldsymbol{g}_k(\boldsymbol{w}_k^n)$. This truncation operation is denoted as Top_q. A common method to improve the training performance under gradient sparsification is to track the accumulated sparsification error in a residual vector $\boldsymbol{\Delta}_k^n$. The effect is that small magnitude elements, which would be ignored in every iteration, are aggregated over several iterations and will survive sparsification at some point.

The complete sparsification procedure, for an effective gradient $\boldsymbol{g}_k(\boldsymbol{w}_k^n)$, is to accumulate the previous sparsification error into the effective gradient as $\boldsymbol{g}_k^{\mathrm{acc}}(\boldsymbol{w}_k^n) \leftarrow \boldsymbol{g}_k(\boldsymbol{w}_k^n) + \boldsymbol{\Delta}_k^{n-1}$, then compute the sparse gradient for transmission as $\bar{\boldsymbol{g}}_k(\boldsymbol{w}_k^n) \leftarrow \mathsf{Top}_q(\boldsymbol{g}_k^{\mathrm{acc}}(\boldsymbol{w}_k^n))$, and update the sparsification error as $\boldsymbol{\Delta}_k^n \leftarrow \boldsymbol{g}_k^{\mathrm{acc}}(\boldsymbol{w}_k^n) - \bar{\boldsymbol{g}}_k(\boldsymbol{w}_k^n)$. Convergence of SGD with this sparsification procedure is established in [49]. The CompressGradient procedure in Algorithm 1 is implemented as exactly these three steps, with $\bar{\boldsymbol{g}}_k(\boldsymbol{w}_k^n)$ being the return value. Then, UncompressGradient simply converts the sparse vector back to its full-length representation, and the summation in (5) is implemented as a conventional sparse vector addition [50, §2].

B. Predictive Routing for Sparse Incremental Aggregation

The predictive routing procedure in Section IV-C relies on knowledge of the storage size $S(\bar{\boldsymbol{g}}_k(\boldsymbol{w}_k^n))$. With sparsification applied, the estimates in (7) and (8) remain no longer valid, as the relevant length $S(\boldsymbol{\gamma}_k^n)$ is no longer constant over the aggregation path. To this end, first consider the following lemma.

Lemma 1: Consider L independent and identically distributed (i.i.d.) random vectors $\boldsymbol{X}_1, \ldots, \boldsymbol{X}_L$ of dimension n_d. Let $\tilde{\boldsymbol{X}}_l = \mathsf{Top}_q(\boldsymbol{X}_l)$ for all $l = 1, \ldots, L$. Then, the expected number of nonzero elements in $\boldsymbol{X}_\Sigma = \sum_{l=1}^L \tilde{\boldsymbol{X}}_l$ is $n_d - n_d \left(1 - \frac{n_a}{n_d}\right)^L$, where $n_a = \lfloor n_d q \rfloor$ is the number of nonzero elements in each summand.

Proof: Consider the vector $\boldsymbol{X}_l = (X_{l,1}, \ldots, X_{l,n_d})$ and let $A_{l,i}$ be the event that $\boldsymbol{X}_{l,i}$ is zero after the Top_q operation. The probability that $A_{l,i}$ occurs is $1 - \frac{n_a}{n_d}$ [51, Lemma 13.1]. Further, let A_i be the event that element i is zero after Top_q in all L vectors. Then, due to independence, $\Pr(A_i) = \Pr\left(\bigcap_{l=1}^L A_{l,i}\right) = \prod_l \Pr(A_{l,i}) = \left(1 - \frac{n_a}{n_d}\right)^L$. Observe that A_i is the event that the ith element of $\boldsymbol{X}_\Sigma$ is zero. Thus, the expected number of zero elements in $\boldsymbol{X}_\Sigma$ is $\mathbb{E}\left[\sum_{i=1}^{n_d} I(A_i)\right] = \sum_i \mathbb{E}\left[I(A_i)\right] = \sum_i \Pr(A_i) = n_d \left(1 - \frac{n_a}{n_d}\right)^L$, where $I(\cdot)$ is the indicator function that takes value 1 if A_i occurs and 0 otherwise. ∎

Based on this lemma, we can make a reasonable estimate on the number of transmitted bits during incremental aggregation.

Proposition 1: The expected total number of transmitted bits over H hops of incremental aggregation is upper bounded as $n_d(\omega + \lceil \log_2 n_d \rceil) \left[H + 1 - \frac{n_d}{n_a}\left[1 - \left(1 - \frac{n_a}{n_d}\right)^{H+1}\right]\right]$, where $n_a = \lfloor n_d q \rfloor$.

Proof: Let $P_h = (k_1, k_2, \ldots, k_{h+1})$, for $h = 1, 2, \ldots, H$, be an increasing sequence of nested paths. Consider incremental aggregation over path P_H starting at k_1. Denote by $S(P_h)$ the total number of bits transmitted over path P_h. Then, $S(P_1) = S(\boldsymbol{\gamma}_{k_1}^n)$ and $S(P_h) = S(\boldsymbol{\gamma}_h^n) + S(P_{h-1})$ for $h > 1$, where $\boldsymbol{\gamma}_k^n$ is the outgoing aggregate at node k as defined in (5). At k_1, the outgoing aggregate has storage size $S(\boldsymbol{\gamma}_{k_1}^n) = n_a(\omega + \lceil \log_2 n_d \rceil)$, where $\lceil \log_2 n_d \rceil$ accounts for the storage space of element indices in the sparse representation [50, §2].

At subsequent nodes k_h, $h > 1$, the outgoing aggregate has size $S(\boldsymbol{\gamma}_{k_h}^n) = S(D_{k_h} \bar{\boldsymbol{g}}_{k_h}(\boldsymbol{w}_{k_h}^n) + \boldsymbol{\gamma}_{k_{h-1}}^n)$. Modelling $\boldsymbol{w}_k^n$ as an i.i.d. random vector and assuming the gradients of the nodes along P_H are independent, we obtain $\mathbb{E}[S(\boldsymbol{\gamma}_{k_h}^n)] = (\omega + \lceil \log_2 n_d \rceil)\left(n_d - n_d\left(1 - \frac{n_a}{n_d}\right)^h\right)$ from Lemma 1. Further, observe that $S(\boldsymbol{\gamma}_{k_1}^n) = n_d - n_d\left(1 - \frac{n_a}{n_d}\right)$. Then, by recursion, $\mathbb{E}[S(P_H)] = \sum_{h=1}^H \mathbb{E}[S(\boldsymbol{\gamma}_h^n)] = n_d(\omega + \lceil \log_2 n_d \rceil)\left[H - \sum_{h=1}^H \left(1 - \frac{n_a}{n_d}\right)^h\right]$, and due to the geometric sum identity, $\mathbb{E}[S(P_H)] = n_d(\omega + \lceil \log_2 n_d \rceil)\left[H + 1 - \frac{n_d}{n_a}\left(1 - \left(1 - \frac{n_a}{n_d}\right)^{H+1}\right)\right]$.

Finally, observe that for dependent gradients the positions of the nonzeros after Top_q will be correlated. Hence, with the notation from the proof of Lemma 1, $\Pr(A_{l,i}) \leq \Pr(A_{l,i} | A_{l,i-1} \cdots A_{l,1})$. Thus, $\Pr(A_i) \geq \left(1 - \frac{n_a}{n_d}\right)^L$ and $\mathbb{E}\left[\sum_i I(A_i)\right] \geq n_d \left(1 - \frac{n_a}{n_d}\right)^L$. ∎

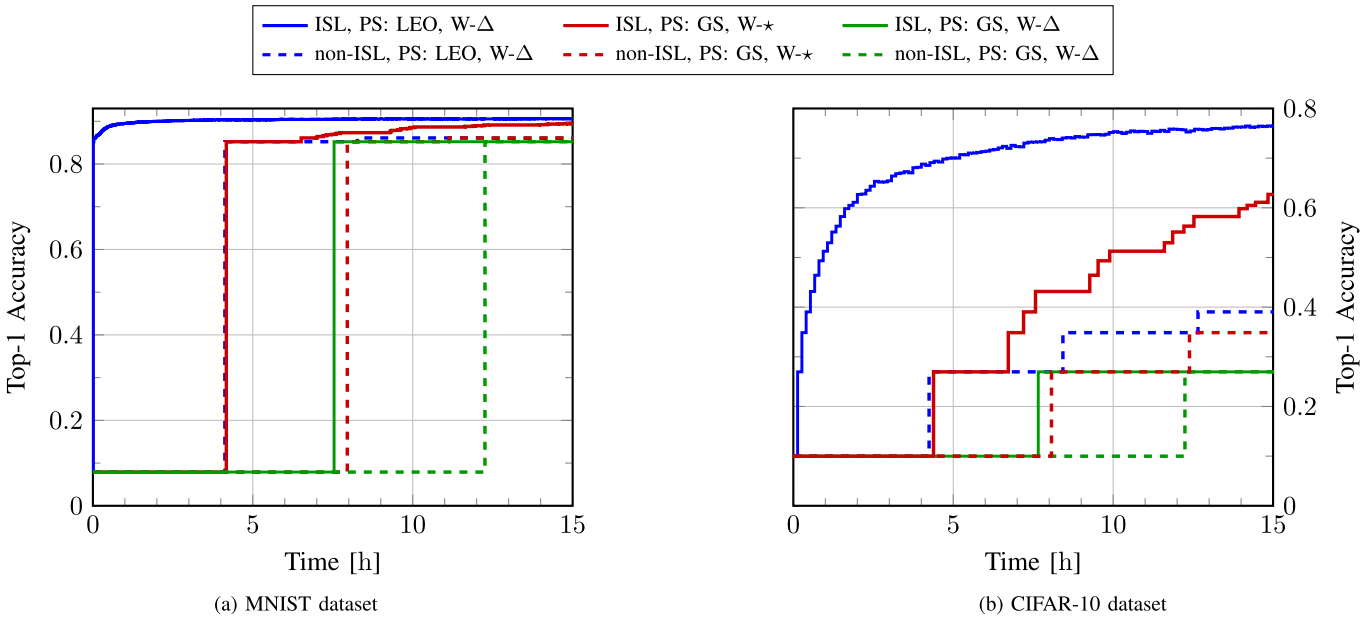
Leveraging Proposition 1, we replace $\left\lceil \frac{K_p}{2} \right\rceil S(\bar{\boldsymbol{g}}_{k_1}(\boldsymbol{w}_{k_1}^n)$ in (7) and (8) with

$$n_d(\omega + \lceil \log_2 n_d \rceil)\left[\left\lceil \frac{K_p}{2} \right\rceil + 1 - \frac{n_d}{n_a}\left[1 - \left(1 - \frac{n_a}{n_d}\right)^{\left\lceil \frac{K_p}{2} \right\rceil + 1}\right]\right], \tag{9}$$

to account for sparsification in the predicted routing procedure. This will overestimate the communication effort as it does not account for the dependence between gradients. As discussed in Section IV-C, a certain amount of "slack" is inconsequential and might even improve overall performance due to reducing the probability of timing failures. However, should tighter bounds become necessary, training a simple ML estimator for the aggregated vector length, e.g., using reinforcement learning, appears sensible. Instead, tighter analytical bounds would require assumptions on a random distribution for the elements of $\boldsymbol{w}$ and experimental calibration of correlation between gradients.

VI. Performance Evaluation

We evaluate the performance of the proposed system design for four representative scenarios. The worker satellites are organized either in a 60°: 40/5/1 Walker delta or a 85°: 40/5/1 Walker star constellation, both at an altitude of 2000 km. The notation $i : t/p/f$ indicates a constellation with p evenly spaced circular orbital planes at inclination i, each having



(a) MNIST dataset

(b) CIFAR-10 dataset

Fig. 4. Test accuracy with respect to wall-clock time. Synchronous orchestration for the MNIST and CIFAR-10 datasets with non-i.i.d. distributions, considering both terrestrial and non-terrestrial PS. i.e., the GS in Bremen and an LEO satellite. Note that ISL and non-ISL stand for the FedAvg with and without ISL algorithms respectively.

t/p equidistant satellites. The phasing parameter f defines the relative shift in right ascension of the ascending node between adjacent orbital planes and amounts to 9° in this particular setup [52]. Subsequently, we identify these constellations as W-Δ and W-$\star$, respectively. These constellations are combined with a PS located either in a terrestrial GS located in Bremen, Germany, or in a LEO satellite orbiting at an altitude of 500 km in the equatorial plane.

Communication links are modelled as complex Gaussian channels with free-space path loss (FSPL). Then, the maximum achievable rate between nods k and i using a bandwidth B is $\rho(k,i) = B\log_2(1+\mathrm{SNR}(k,i))$. The $\mathrm{SNR}(k,i) = \frac{P_t G_k(i) G_i(k)}{N_0 L(k,i)}$ is defined by the transmit power P_t, the noise spectral density $N_0 = k_B T B$ at receiver temperature T, and average antenna gains $G_j(l)$ at node j towards node l, with k_B being the Boltzman constant. The FSPL is $L(k,i) = (4\pi f_c d(k,i)/c_0)^2$, where f_c is the carrier frequency and $d(k,i)$ the distance between nodes k and i [16], [53]. We assume fixed rate links operating at the minimum rate supported by the link. This is equivalent to selecting $d(k,i)$ as the maximum communication distance $d_{\mathrm{Th}}(k,i)$ between these nodes. Following [52], we set $f_c = 20\,\mathrm{GHz}$, $B = 500\,\mathrm{MHz}$, $P_t = 40\,\mathrm{dBm}$, $T = 354\,\mathrm{K}$, and the antenna gains to 32.13 dBi.

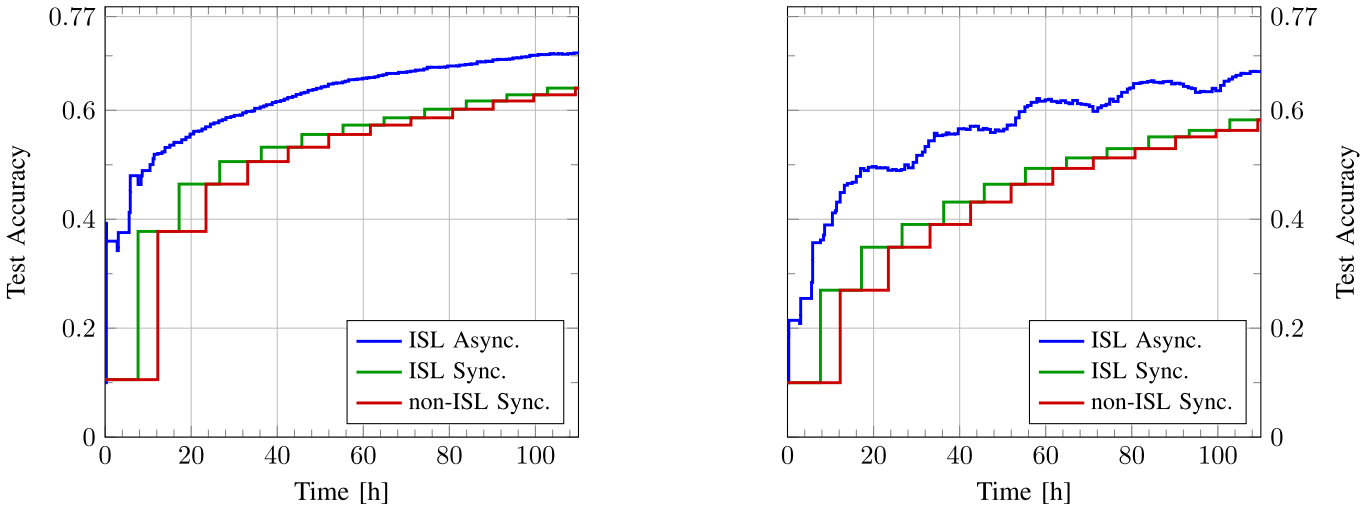
Numerical ML performance is evaluated based on the two most widely used benchmarks. The first is a conventional 7850-parameter logistic regression model trained on the MNIST dataset, 28×28 pixel greyscale images of handwritten digits ranging from 0 to 9 [54]. The other is a deep CNN with 122 570 parameters from [55], trained on the CIFAR-10 dataset [56], which includes 10 classes with 32×32 RGB images. The data samples are either evenly distributed at random among the satellites, which is the i.i.d. setting, or in a non-i.i.d. fashion using a Dirichlet distribution with parameter 0.5 [57], [58]. We use a batch size of ten, run five local epochs, and take the learning rate as 0.1. A computation time t_l of 60 s and 480 s is assumed for MNIST and CIFAR-10, respectively. The simulation is build upon the FedML framework [59].

A. Synchronous and Asynchronous Orchestration

We start by evaluating the benefits of ISLs for synchronous orchestration as detailed in Algorithm 2. Without employing ISLs, this is the vanilla FL approach as first proposed in [34]. Directly applying it to SFL, i.e., each satellite contacts the PS directly, leads to very slow convergence speed, especially in scenarios with terrestrial orchestration [10]. The question is whether the usage of ISLs resolves the connectivity bottleneck sufficiently to make synchronous orchestration feasible.

To this end, we measure the test accuracy with respect to the wall clock time for synchronous terrestrial orchestration in the W-$\star$ and W-Δ constellations, with and without ISLs. We complement this by an experiment with a PS in LEO. Results for non-i.i.d. MNIST and CIFAR-10 are displayed in Fig. 4. All scenarios show the distinct step function behavior first observed in [10] for the non-ISL scenarios. This is caused by the PS being forced to wait for all results before updating the global model. As expected, the usage of ISLs, as proposed in Section IV, shortens the convergence time significantly in all scenarios. With MNIST training usually showing very fast convergence, it can be easily observed that ISLs reduce the time to convergence in all scenarios by 4 hours. This leads to almost instantaneous convergence in the LEO scenario, at least in relation to the training duration in conventional SFL, and reduces the convergence time by up to 50 % for terrestrial orchestration.

The CIFAR-10 experiment was chosen to evaluate the convergence behavior of a more involved ML model. As such, convergence requires a large number of global iterations. From



(a) i.i.d. distribution

(b) non-i.i.d. distribution

Fig. 5. Comparison of synchronous and asynchronous orchestration. The figure displays test accuracy with respect to wall-clock time for a W-Δ constellation with CIFAR-10 dataset, distributed i.i.d. and non-i.i.d., and PS located in Bremen.

Fig. 4b, we observe a more nuanced convergence behavior as compared to the MNIST experiment. In particular, LEO-based orchestration benefits most from ISLs, while W-Δ profits only in the initial learning phase. Surprisingly, it basically shows the same convergence speed as without ISLs, simply shifted in time by a few hours. The reason behind this is best understood from Fig. 2, which displays exactly the connectivity patterns of these two scenarios. Connectivity in the LEO PS scenario changes from a sporadic towards a near-persistent connectivity pattern. Instead, for terrestrial orchestration, the connectivity windows grow significantly in duration, but the overall connectivity pattern is still sporadic. That is, the offline periods of individual orbital groups still lead to blocking in synchronous aggregation algorithms. However, this behavior is not universal to terrestrial orchestration. It depends on the constellation design and GS location, as can be seen by the improved convergence speed in the W-$\star$ scenario. Moreover, combining synchronous orchestration and ISLs with the scheduling approach from [22] is expected to alleviate this problem in training scenarios with limited computational complexity.

Following the discussion in Section III and [7], a viable alternative to synchronous orchestration in sporadic connectivity scenarios is asynchronous PS operation. This is evaluated in Fig. 5 for W-Δ with terrestrial orchestration, with maximum time between updates T_u set to 147 minutes. Convergence with asynchronous aggregation is improved over synchronous operation, but still involves at least a ten-fold increase in convergence time. Especially initial convergence speed, i.e., within the first few hours of training, is greatly increased. This leads to the conclusion that asynchronous aggregation is a potential solution if the system design suffers from sporadic connectivity. However, if changing the PS location is an option, it might be preferable to asynchronous orchestration.

B. Failure Handling

To evaluate the performance of failure handling schemes, we consider single orbit with an inclination of 85° and altitude of 2000 km, where the PS is located in a GS in Bremen. We model the computation time for learning at any satellite k as $T_l(k) = t_l(k) + X(k)$, where $t_l(k)$ is the deterministic time for learning, and $X(k)$ accounts for random additional computation time due to e.g. multiprocessing loads.The distribution of $X(k)$ is modelled using a Gamma distribution, a common approach for modeling service times, with shape and scale parameters denoted as $\alpha(k)$ and $\theta(k)$ respectively [60]. Hence, the cumulative distribution function (CDF) of $X(k)$ is $F_X(x) = \Gamma(\alpha(k))^{-1}\ \gamma(\alpha(k), \frac{x}{\theta(k)})$ if x is positive, and $F_X(x) = 0$ otherwise, where $\alpha(k) > 0$, $\theta(k) > 0$, $\Gamma(z)$ is the Gamma function, and $\gamma(s, x)$ is the lower incomplete Gamma function. We set the values of $t_l(k)$, $\alpha(k)$, and $\theta(k)$ for any satellite k to 480 seconds, 25, and 25 seconds respectively. The random communication delays in any satellite k, denoted as $Y(k)$, are taken into account in addition to the deterministic time $t_c(k, j)$ for transmitting the aggregated parameters from satellites k to j. In this regard, total communication time for each satellite k is modelled as $T_c(k, j) = t_c(k, j) + Y(k)$, where $Y(k)$ follows an exponential distribution with parameter $\lambda(k) > 0$, with CDF $F_Y(y) = 1 - \exp(-y\lambda(k))$ if y is positive and $F_Y(y) = 0$ otherwise. Here, we set $t_c(k, j)$ and $\lambda(k)$ for any satellite k to 50 seconds and 0.025 respectively. It is worth mentioning that, when the delays in the transmissions and learning procedures of satellites are very small, the system may not need any failure handling scheme. In this regard, we have set the parameters values such that the effect of failure can be seen. We also set t_g to 0.

Figure 6 shows the required time for handling the failure with respect to the number of satellites in the orbit, for the pass-to-neighbor and determine-new-sink schemes. For any number of satellite in Fig. 6, we calculate the average required time based on overall 20 000 random values for $X(k)$ and $Y(k)$ for the 20 different randomly chosen global iterations, starting at different times. As we can see in Fig. 6, with the increase in the number of satellites, the determine-new-sink scheme reduces the required time by a factor of approximately 4.5, as compared to the pass-to-neighbor in any orbit. This is

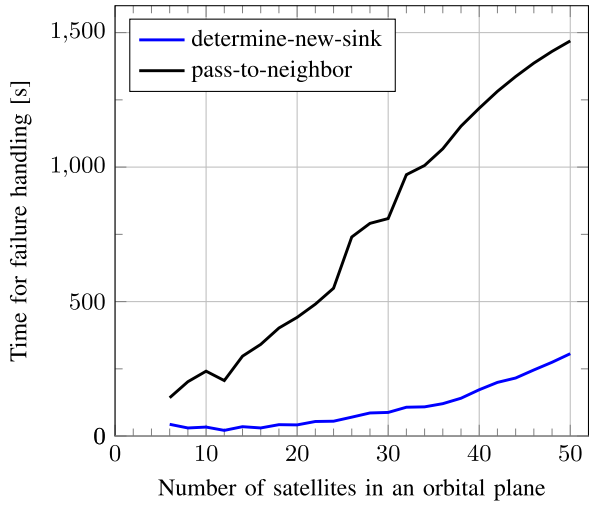


Fig. 6. The time required for failure handling (FH) with respect to the number of satellites in an orbital plane.

particularly effective when considering the delays incurred in constellations with multiple orbits, as the determine-new-sink failure handling scheme significantly reduces the required time within each orbit, thereby substantially reducing the overall required time.

C. Gradient Sparsification and Transmission Load

Figure 7 shows the transmission load required for collecting the gradients in one orbit and sending them towards the PS, within a single global iteration. The results are based on running FedAvg on the MNIST dataset with non-i.i.d. distribution. The figure depicts the total transmitted data with and without incremental aggregation (IA) as a function of the number of satellites within an orbital plane. We consider three different cases, without sparsification as well as sparsification with ratios $q = 0.1$ and $q = 0.01$. When IA is used, each satellite aggregates its gradient updates with the received ones, according to the method described in Section IV-A. Without IA, each satellite only forwards its gradients and the received ones to the other satellite or the PS, without performing any aggregation.

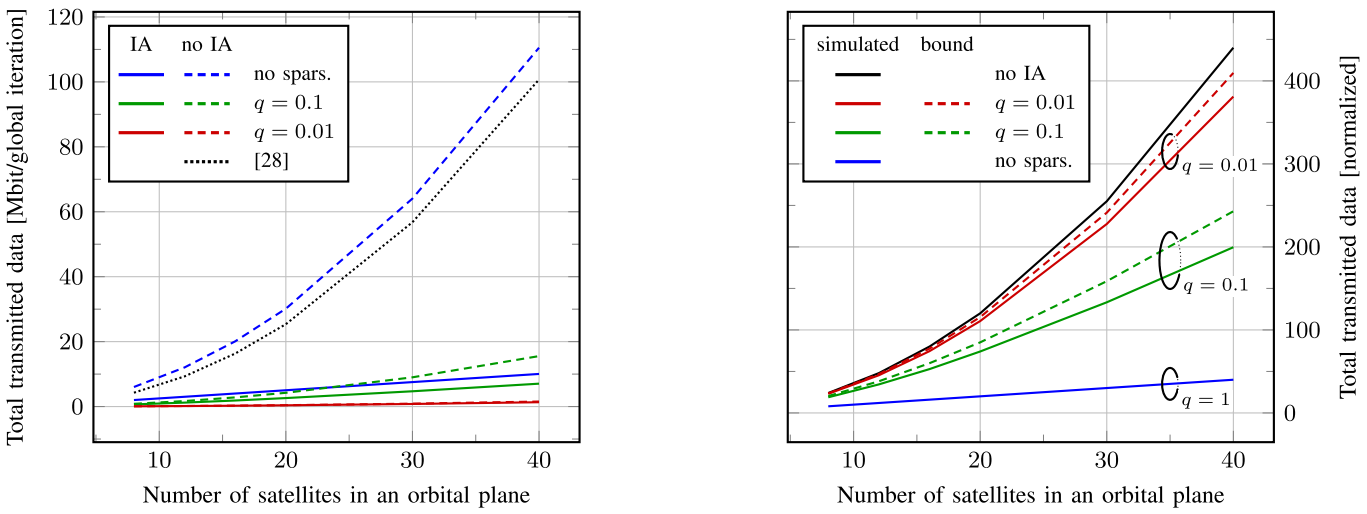
Figure 7a shows the communication effort in terms of the total transmitted data. We set the storage size ω of model parameters to 32 bit. The sparsified vectors need an additional index field per entry, which is chosen to be 13 bits. Hence, each nonzero element in a sparsified vector requires a total of 45 bits. The effect of sparsification on transmission cost is well known and not further discussed here. A notable observation is that IA has comparable (and even stronger) effect than removing 90 % of the data from each vector (with sparsification). This effect continues to hold in the combination of sparsification and IA, leading to an even further reduction of communication cost and, more importantly, a linear cost increase in the number of satellites.

In addition, we compare the communication efficiency of our approach to the state-of-the-art in [28]. There, each satellite within an orbital plane transmits its updated local parameters to a designated sink satellite through multihop connections over neighbouring satellites. In contrast to our approach, each parameter vector is transmitted through unicast transmissions as in the baseline approach. However, contrary to the baseline, the sink satellite performs partial aggregation before transmitting the updated parameter vector to the PS. The result is a quadratic growth in total communication effort, with slightly better performance than the baseline. As the number of satellites per orbit increases, we observe a tenfold reduction in communication effort due to IA over [28] in Fig. 7a.

The impact of IA appears to decrease in Fig. 7a with stricter sparsification, i.e., the effect of IA seems small for $q = 0.1$ and negligible for $q = 0.01$. This, however, is only partially true and near impossible to assess from Fig. 7a. For this reason, Fig. 7b shows the total transmitted data normalized to the size of a single parameter vector ready for transmission. The purpose of this is to evaluate the increase in communication efficiency due to IA independently from the compression achieved by sparsification. The normalized baseline, i.e., multiple unicast transmission without IA, is identical with and without sparsification and, hence, only shown once. Based on this normalization, the impact of IA on the communication efficiency is proportional to the size of the gap between the simulated result (solid color) and the black baseline. The initial impression from Fig. 7a continues to hold, i.e., the efficiency of IA decreases with increasing sparsification ratio, i.e., lower q. However, this does not imply IA is expendable in any way. For 40 satellites per orbital plane, we observe a reduction in communication cost of 55 % for $q = 0.1$ purely due to IA. Without sparsification, this reduction amounts to 91 %, while it is still a notable 13 % for $q = 0.01$. An important aspect to consider is that a sparsification ratio of $q = 0.01$ leads to slower convergence in training, as opposed to $q = 0.1$, which might deteriorate the savings in communication cost. Instead, IA comes at no cost to the training process and, thus, can mitigate some of the adverse effects that the lossy compression (sparsification) has on a training process within a fixed communications budget.

A possible cause for the reduced efficiency of IA at small q is indicated by the dashed lines in Fig. 7b. These display the estimated communication effort based on the bound in Proposition 1 and (9). We observe that this bound becomes less tight as the number of satellites increases and as the vectors become sparser.[6] This discrepancy between bound and simulation is most likely caused by the fact that the computed local updates are correlated instead of being stochastically independent as assumed in Proposition 1. In other words, a large gap between bound and simulation indicates a stronger correlation between the position of the nonzero elements in the gradients, while a small gap points to their distribution being closer to independence. Now, with higher sparsification ratios (small q), the probability of two vectors having overlapping nonzero entries becomes increasingly small and, hence, each IA step increases the number of nonzero entries in the resulting vector significantly. To see how this reduces the effect of IA, recall that a sparse vector is represented by the nonzero values

[6]Recalling the discussion in Section V, this has no direct impact on the communication effort and should only lead to small timing errors in the predictive routing procedure.



(a) Absolute

(b) Relative

Fig. 7. Total transmitted data with respect to the number of satellites in an orbit per global iteration for three different sparsification ratios; a) $q = 0.01$ and b) $q = 0.1$, and c) without sparsification (q=1). IA and no IA stand for the proposed algorithm with and without incremental aggregation respectively.

together with the indices of these nonzero positions. When the indices of non-zero elements differ among satellites, the transmitted vector after IA must contain all nonzero elements of the received vector together with the nonzero elements of the satellite's own sparsified result. With decreasing overlap between the positions of nonzero elements, this becomes closer to separate transmission of both vectors.

Figure 7 deals with the amount of total transmitted data necessary to deliver all gradient updates from a client cluster towards the PS. We have observed that conventional approaches scale quadratically in the number of satellites per cluster. Instead, the proposed IA method scales linearly and, thus, offers a massive increase in communication efficiency. However, through the narrow lens of communication efficiency in terms of total transmitted data, the same efficiency is achieved by not using clusters at all and, instead, have each satellite communicate directly with the PS. To take a slightly different angle, recall that the majority of transmissions for clustered SFL is done over stable ISLs, while the direct satellite-to-PS approach requires more costly GSL or a dynamic long-range ISL. Consequently, the proposed framework reduces the communication load at the PS, in comparison to direct connectivity, directly proportional to the cluster size.

VII. Discussion and Conclusion

We have designed a clustered FL system tailored to distributed ML in modern satellite megaconstellations. It efficiently uses intra-orbit ISLs to avoid connectivity bottlenecks that impair convergence speed. Owing to a FL-specific in-network aggregation strategy, this is achieved without increasing the total transmitted amount of data. Indeed, it can be expected that this method actually decreases the communication cost due to relying on cheaper-to-operate ISLs instead of long distance GSLs. Moreover, the communication load is evenly spread among the network instead of focusing all communications on direct worker-to-PS links. This strategy requires a careful predictive route planning in order to operate successfully in a dynamic network topology imposed by orbital mechanics. We have developed the necessary routing algorithms for this in conjunction with a rigorous distributed system design that inflicts a low overhead for synchronization and consensus finding. The proposed system is compatible with a wide variety of FL algorithms, supports gradient sparsification, and includes facilities for asynchronous clustered aggregation. While not explicitly mentioned, the system's extension to incorporate worker scheduling is rather straightforward. We have evaluated the performance of the proposed system for a few carefully chosen examples. The results highlight the major increase in convergence speed to ISLs and the bandwidth effectiveness of our in-network aggregation approach.

Regarding future work, it is interesting to revise the assumption adopted in this paper that all participants in the FL process are trustworthy, under complete control, and the software operates nominally. For instance, if the satellites belong to different operators, the use of ISLs may incur a certain cost and/or a group for satellites may decide not to participate in the FL process, which would require an extension of the proposed algorithms. Moreover, incorporating inter-orbit ISLs in the communication scheme has the potential to replace out-of-constellation orchestration and further increase the resource-efficiency of SFL. This, however, comes at the cost of higher management complexity, will likely require investigating several special cases, e.g., with and without cross-seam SFLs, and opens the possibility for SFL-specific network topology design. It is also noteworthy that our scenario with one GS can be extended to the case with multiple GSs to improve the satellite-PS connectivity. Two principal approaches to employ multiple GSs as PS can be considered: (1) the GSs forward the communication towards a centralized cloud PS, or (2) they act together as a distributed PS. The first case is incremental with respect to the proposed algorithms as it involves multi-hop routing through a terrestrial network. The second case implies a stronger separation of ground- and

space-segment and requires accurate synchronization to maintain model parameters integrity.

REFERENCES

[1] N. Razmi, B. Matthiesen, A. Dekorsy, and P. Popovski, "On-board federated learning for dense LEO constellations," in *Proc. IEEE Int. Conf. Commun. (ICC)*, Seoul, South Korea, May 2022, pp. 4715–4720.

[2] M. N. Sweeting, "Modern small satellites-changing the economics of space," *Proc. IEEE*, vol. 106, no. 3, pp. 343–361, Mar. 2018.

[3] I. Leyva-Mayorga et al., "LEO small-satellite constellations for 5G and beyond-5G communications," *IEEE Access*, vol. 8, pp. 184955–184964, 2020.

[4] M. Y. Abdelsadek et al., "Future space networks: Toward the next giant leap for humankind," *IEEE Trans. Commun.*, vol. 71, no. 2, pp. 949–1007, Feb. 2023.

[5] M. Mozaffari, X. Lin, and S. Hayes, "Toward 6G with connected sky: UAVs and beyond," *IEEE Commun. Mag.*, vol. 59, no. 12, pp. 74–80, Dec. 2021.

[6] H. Chen, M. Xiao, and Z. Pang, "Satellite-based computing networks with federated learning," *IEEE Wireless Commun.*, vol. 29, no. 1, pp. 78–84, Feb. 2022.

[7] B. Matthiesen, N. Razmi, I. Leyva-Mayorga, A. Dekorsy, and P. Popovski, "Federated learning in satellite constellations," *IEEE Netw.*, early access, 2023.

[8] D. Izzo et al., "Selected trends in artificial intelligence for space applications," in *Artifical Intelligence for Space: AI4SPACE*. CRC Press, 2022, pp. 21–52.

[9] G. Fontanesi et al., "Artificial intelligence for satellite communication and non-terrestrial networks: A survey," Apr. 2023, *arXiv:2304.13008*.

[10] N. Razmi, B. Matthiesen, A. Dekorsy, and P. Popovski, "Ground-assisted federated learning in LEO satellite constellations," *IEEE Wireless Commun. Lett.*, vol. 11, no. 4, pp. 717–721, Apr. 2022.

[11] R. Radhakrishnan et al., "Survey of inter-satellite communication for small satellite systems: Physical layer to network layer view," *IEEE Commun. Surveys Tuts.*, vol. 18, no. 4, pp. 2442–2473, 4th Quart., 2016.

[12] Y. Lee and J. P. Choi, "Connectivity analysis of mega-constellation satellite networks with optical intersatellite links," *IEEE Trans. Aerosp. Electron. Syst.*, vol. 57, no. 6, pp. 4213–4226, Dec. 2021.

[13] R. Li, B. Lin, Y. Liu, M. Dong, and S. Zhao, "A survey on laser space network: Terminals, links, and architectures," *IEEE Access*, vol. 10, pp. 34815–34834, 2022.

[14] D. Bhattacherjee and A. Singla, "Network topology design at 27,000 kmhour," in *Proc. Int. Conf. Emerg. Netw. Express Technol.*, Orlando, FL, USA, Dec. 2019, pp. 341–354.

[15] A. U. Chaudhry and H. Yanikomeroglu, "Laser intersatellite links in a starlink constellation: A classification and analysis," *IEEE Veh. Technol. Mag.*, vol. 16, no. 2, pp. 48–56, Jun. 2021.

[16] I. Leyva-Mayorga, B. Soret, and P. Popovski, "Inter-plane inter-satellite connectivity in dense LEO constellations," *IEEE Trans. Wireless Commun.*, vol. 20, no. 6, pp. 3430–3443, Jun. 2021.

[17] E. Fasolo, M. Rossi, J. Widmer, and M. Zorzi, "In-network aggregation techniques for wireless sensor networks: A survey," *IEEE Wireless Commun.*, vol. 14, no. 2, pp. 70–87, Apr. 2007.

[18] J. Wang et al., "A field guide to federated optimization," 2021, *arXiv:2107.06917*.

[19] Q.-V. Pham, M. Zeng, T. Huynh-The, Z. Han, and W.-J. Hwang, "Aerial access networks for federated learning: Applications and challenges," *IEEE Netw.*, vol. 36, no. 3, pp. 159–166, May 2022.

[20] Q. Fang, Z. Zhai, S. Yu, Q. Wu, X. Gong, and X. Chen, "Olive branch learning: A topology-aware federated learning framework for space-air-ground integrated network," *IEEE Trans. Wireless Commun.*, vol. 22, no. 7, pp. 4534–4551, 2022.

[21] R. Uddin and S. Kumar, "SDN-based federated learning approach for satellite-IoT framework to enhance data security and privacy in space communication," in *Proc. IEEE Int. Conf. Wireless Space Extreme Environments (WiSEE)*, Winnipeg, MB, Canada, Oct. 2022, pp. 71–76.

[22] N. Razmi, B. Matthiesen, A. Dekorsy, and P. Popovski, "Scheduling for ground-assisted federated learning in LEO satellite constellations," in *Proc. 30th Eur. Signal Process. Conf. (EUSIPCO)*, Belgrade, Serbia, Aug. 2022, pp. 1102–1106.

[23] J. So, K. Hsieh, B. Arzani, S. Noghabi, S. Avestimehr, and R. Chandra, "FedSpace: An efficient federated learning framework at satellites and ground stations," Feb. 2022, *arXiv:2202.01267*.

[24] L. Wu and J. Zhang, "FedGSM: Efficient federated learning for LEO constellations with gradient staleness mitigation," 2023, *arXiv:2304.08537*.

[25] J. Lin, J. Xu, Y. Li, and Z. Xu, "Federated learning with dynamic aggregation based on connection density at satellites and ground stations," in *Proc. IEEE Int. Conf. Satell. Comput. (Satellite)*, Shenzhen, China, Nov. 2022, pp. 31–36.

[26] M. Elmahallawy and T. Luo, "FedHAP: Fast federated learning for LEO constellations using collaborative HAPs," 2022, *arXiv:2205.07216*.

[27] M. Elmahallawy and T. Luo, "AsyncFLEO: Asynchronous federated learning for LEO satellite constellations with high-altitude platforms," in *Proc. IEEE Int. Conf. Big Data (Big Data)*, Osaka, Japan, Dec. 2022, pp. 5478–5487.

[28] M. Elmahallawy and T. Luo, "Optimizing federated learning in LEO satellite constellations via intra-plane model propagation and sink satellite scheduling," 2023, *arXiv:2302.13447*.

[29] C.-Y. Chen, L.-H. Shen, K.-T. Feng, L.-L. Yang, and J.-M. Wu, "Edge selection and clustering for federated learning in optical inter-LEO satellite constellation," 2023, *arXiv:2303.16071*.

[30] C. Wu, Y. Zhu, and F. Wang, "DSFL: Decentralized satellite federated learning for energy-aware LEO constellation computing," in *Proc. IEEE Int. Conf. Satell. Comput. (Satellite)*, Shenzhen, China, Nov. 2022, pp. 25–30.

[31] J. Östman, P. Gomez, V. Magal Shreenath, and G. Meoni, "Decentralised semi-supervised onboard learning for scene classification in low-Earth orbit," 2023, *arXiv:2305.04059*.

[32] P. Wang, H. Li, and B. Chen, "FL-task-aware routing and resource reservation over satellite networks," in *Proc. IEEE Global Commun. Conf. (GLOBECOM)*, Rio de Janeiro, Brazil, Dec. 2022, pp. 2382–2387.

[33] *Study on New Radio (NR) to Support Non-Terrestrial Networks*, document 3GPP TR 38.811 V15.4.0, Sep. 2020.

[34] B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. Arcas, "Communication-efficient learning of deep networks from decentralized data," in *Proc. Artif. Intell. Statist. (AISTATS)*, Orlando, FL, USA, Apr. 2017.

[35] A. P. Miettinen and J. K. Nurminen, "Energy efficiency of mobile clients in cloud computing," in *Proc. USENIX Workshop Hot Topics Cloud Comput.*, Boston, MA, USA, Jun. 2010, pp. 1–7.

[36] K. Scott and S. C. Burleigh, *Bundle Protocol Specification*, document RFC 5050, Nov. 2007.

[37] S. Burleigh, K. Fall, and E. J. Birrane, *Bundle Protocol Version 7*, document RFC 9171, Jan. 2022.

[38] D. P. Bertsekas and J. N. Tsitsiklis, *Parallel Distrib. Computation: Numerical Methods*. Athena Scientific, 2015.

[39] L. Torgerson et al., *Delay-Tolerant Networking Architecture*, document RFC 4838, Apr. 2007.

[40] G. Araniti et al., "Contact graph routing in DTN space networks: Overview, enhancements and performance," *IEEE Commun. Mag.*, vol. 53, no. 3, pp. 38–46, Mar. 2015.

[41] J. A. Fraire, O. De Jonckère, and S. C. Burleigh, "Routing in the space Internet: A contact graph routing tutorial," *J. Netw. Comput. Appl.*, vol. 174, Jan. 2021, Art. no. 102884.

[42] N. Deo, *Graph Theory With Applications to Engineering and Computer Science*. Upper Saddle River, NJ, USA: Prentice-Hall, 1974.

[43] J. R. Vetter, "Fifty years of orbit determination: Development of modern astrodynamics methods," *Johns Hopkins APL Tech. Dig.*, vol. 27, no. 3, pp. 239–252, 2007.

[44] D. Vallado and P. Crawford, "SGP4 orbit determination," in *Proc. Astrodyn. Spec. Conf. Exhibit*, Aug. 2008, p. 6770.

[45] D. A. Vallado and W. D. McClain, *Fundamentals Astrodynamicsand Appl.*, 4th ed. Microcosm Press, 2013.

[46] D. A. Vallado, P. Crawford, R. Hujsak, and T. S. Kelso, "Revisiting spacetrack report #3: Rev 3," in *Proc. AIAA/AAS Astrodyn. Specialist Conf.*, Aug. 2006, pp. 1–88.

[47] F. R. Hoots and R. L. Roehrich, "Models for propagation of NORAD element sets," Project Space Track, Aerosp. Defense Command, Colorado Springs, CO, USA, Tech. Rep. Spacetrack Rep. no. 3, Dec. 1980.

[48] A. F. Aji and K. Heafield, "Sparse communication for distributed gradient descent," in *Proc. Conf. Empir. Methods Nat. Lang. Process.*, Copenhagen, Denmark, Sep. 2017, pp. 440–445.

[49] D. Alistarh, T. Hoefler, M. Johansson, N. Konstantinov, S. Khirirat, and C. Renggli, "The convergence of sparsified gradient methods," in *Proc. Conf. Adv. Neural Inf. Process. Syst.*, Dec. 2018, pp. 5976–5986.

[50] I. S. Duff, A. M. Erisman, and J. K. Reid, *Direct Methods for Sparse Matrices*, 2nd ed. Oxford, U.K.: Oxford Univ. Press, Jan. 2017.

[51] A. W. van der Vaart, *Asymptotic Statistics*. Cambridge, U.K.: Cambridge Univ. Press, 1998.

[52] I. Leyva-Mayorga et al., "NGSO constellation design for global connectivity," in *Non-Geostationary Satellite Communications Systems*, E. Lagunas, S. Chatzinotas, K. An, and B. F. Beidas, Eds. Hertfordshire, U.K.: IET, Dec. 2022, ch. 9, pp. 189–236.

[53] L. J. Ippolito Jr., *Satellite Communications Systems Engineering*. Hoboken, NJ, USA: Wiley, 2017.

[54] Y. LeCun, C. Cortes, and C. J. C. Burges. *The MNIST Database of Handwritten Digits*. Accessed: Dec. 15, 2021. [Online]. Available: http://yann.lecun.com/exdb/mnist/

[55] Y. Fraboni, R. Vidal, L. Kameni, and M. Lorenzi, "A general theory for federated optimization with asynchronous and heterogeneous clients updates," *J. Mach. Learn. Res.*, vol. 24, no. 110, pp. 1–43, 2023.

[56] A. Krizhevsky and G. Hinton, "Learning multiple layers of features from tiny images," M.S. thesis, Dept. Comput. Sci., Univ. Toronto, Toronto, ON, Canada, Apr. 2009. [Online]. Available: https://www.cs.toronto.edu/~kriz/cifar.html

[57] M. Yurochkin, M. Agarwal, S. Ghosh, K. Greenewald, N. Hoang, and Y. Khazaeni, "Bayesian nonparametric federated learning of neural networks," in *Proc. Int. Conf. Mach. Learn.*, San Francisco, CA, USA, Jun. 2019, pp. 7252–7261.

[58] H. Wang, M. Yurochkin, Y. Sun, D. Papailiopoulos, and Y. Khazaeni, "Federated learning with matched averaging," in *Proc. Int. Conf. Learn. Represent. (ICLR)*, Apr. 2020, pp. 1–16.

[59] C. He et al., "FedML: A research library and benchmark for federated machine learning," in *Proc. Conf. Neural Inf. Process. Syst. (NeurIPS)*, Dec. 2020, pp. 1–13.

[60] J. R. Lorch and A. J. Smith, "Improving dynamic voltage scaling algorithms with PACE," *ACM SIGMETRICS Perform. Eval. Rev.*, vol. 29, no. 1, pp. 50–61, Jun. 2001.

Nasrin Razmi (Graduate Student Member, IEEE) received the B.Sc. degree in electrical engineering from Urmia University, in 2011, and the M.Sc. degree in electrical engineering from the K. N. Toosi University of Technology (KNTU), Tehran, Iran, in 2013. She is currently pursuing the Ph.D. degree with University of Bremen, Bremen, Germany. Her research interests include wireless communication and machine learning. She received the award from the IEEE Communications Society (ComSoc) for being selected as one of the top 15 finalists to present the Ph.D. research project in four minutes at IEEE GLOBECOM 2023.

Bho Matthiesen (Member, IEEE) received the Diplom-Ingenieur (M.Sc.) and Ph.D. (Hons.) degrees in electrical engineering from Technische Universität Dresden, Dresden, Germany, in 2012 and 2019, respectively. From 2008 to 2010, he was a Student Research Assistant with Technische Universtät Dresden. From 2010 to 2011, he was a Research Assistant with the Fraunhofer Institute for Telecommunications, Berlin, Germany. From 2012 to 2019, he was a Research Associate with the Chair of Communications Theory, Technische Universität Dresden. He is currently a Research Group Leader and a Lecturer with the U Bremen Excellence Chair of Petar Popovski, Department of Communications Engineering, University of Bremen, Germany. His research interests are in communication theory, wireless communications, and optimization theory. He is a member of IEEE ComSoc, IEEE SPS, EURASIP, and VDE/ITG. He received the Best Paper Award at the 17th IEEE International Conference on Industrial and Information Systems (ICIIS) in 2023. He is an exemplary reviewer of the IEEE WIRELESS COMMUNICATIONS LETTERS, from 2020 to 2021. He was an invited speaker at the second 6G Wireless Summit 2020; and a tutorial presenter at IEEE VTC2020-Fall, IEEE ICC 2021, IEEE ICASSP 2021, and IEEE ICC 2023. He served as the Publication Co-Chair for the International Symposium on Wireless Communication Systems (ISWCS) 2021, the International ITG 26th Workshop on Smart Antennas, and the 13th Conference on Systems, Communications, and Coding (WSA & SCC 2023). He is an Associate Editor of the *EURASIP Journal on Wireless Communications and Networking* and *Wireless Personal Communications* (Springer) and an Editorial Board Member of *Scientific Reports* (Nature Portfolio).

Armin Dekorsy (Senior Member, IEEE) received the B.Sc. degree from the University of Applied Sciences Konstanz, the M.Sc. degree from the University of Paderborn, and the Ph.D. degree from the University of Bremen, all in communications engineering. He is a Professor with the University of Bremen, where he ia slao the Director of the Gauss-Olbers Space-Technology Transfer-Center and leads the Department of Communications Engineering. With over 11 years of industry experience, he has held distinguished research positions, including a DMTS with Bell Labs and a Research Coordinator Europe with Qualcomm. He has actively participated in over 60 cooperative international research projects. He has coauthored the textbook *Nachrichtenübertragung* (Release 6, Springer Vieweg), a bestseller in the German-speaking world on communication technologies. His research focuses on signal processing and communications for 5G/6G, industrial radio, and 3D networks. He is recognized as a Senior Member of the IEEE Communications and Signal Processing Society and a member of the VDE/ITG Expert Committee Information and System Theory.

Petar Popovski (Fellow, IEEE) received the Dipl.-Ing and M.Sc. degrees in communication engineering from the University of Sts. Cyril and Methodius in Skopje and the Ph.D. degree from Aalborg University in 2005. He is a Professor with Aalborg University, where he heads the Section on Connectivity; and the Visiting Excellence Chair with the University of Bremen. He has authored the book *Wireless Connectivity: An Intuitive and Fundamental Guide*. His research interests are in the area of wireless communication and communication theory. He received an ERC Consolidator Grant in 2015, the Danish Elite Researcher Award in 2016, the IEEE Fred W. Ellersick Prize in 2016, the IEEE Stephen O. Rice Prize in 2018, the Technical Achievement Award from the IEEE Technical Committee on Smart Grid Communications in 2019, the Danish Telecommunication Prize in 2020, and the Villum Investigator Grant in 2021. He is the Chair of the IEEE Communication Theory Technical Committee. He was the General Chair for IEEE SmartGridComm 2018 and IEEE Communication Theory Workshop 2019. He was a Member-at-Large at the Board of Governors in IEEE Communication Society, from 2019 to 2021. He is the Editor-in-Chief of IEEE JOURNAL ON SELECTED AREAS IN COMMUNICATIONS.