Author copy of paper published at *2024 IEEE/ACM Symposium on Edge Computing (SEC)*

HyperDrive: Scheduling Serverless Functions in the Edge-Cloud-Space 3D Continuum

Thomas Pusztai*
Distributed Systems Group, TU Wien
t.pusztai@dsg.tuwien.ac.at

Cynthia Marcelino*
Distributed Systems Group, TU Wien
c.marcelino@dsg.tuwien.ac.at

Stefan Nastic
Distributed Systems Group, TU Wien
snastic@dsg.tuwien.ac.at

***Abstract*—The number of Low Earth Orbit (LEO) satellites has grown enormously in the past years. Their abundance and low orbits allow for low latency communication with a satellite almost anywhere on Earth, and high-speed inter-satellite laser links (ISLs) enable a quick exchange of large amounts of data among satellites. As the computational capabilities of LEO satellites grow, they are becoming eligible as general-purpose compute nodes. In the 3D continuum, which combines Cloud and Edge nodes on Earth and satellites in space into a seamless computing fabric, workloads can be executed on any of the aforementioned compute nodes, depending on where it is most beneficial. However, scheduling on LEO satellites moving at approx. 27,000 km/h requires picking the satellite with the lowest latency to all data sources (ground and, possibly, earth observation satellites). Dissipating heat from onboard hardware is challenging when facing the sun and workloads must not drain the satellite's batteries. These factors make meeting SLOs more challenging than in the Edge-Cloud continuum, i.e., on Earth alone. We present HyperDrive, an SLO-aware scheduler for serverless functions specifically designed for the 3D continuum. It places functions on Cloud, Edge, or Space compute nodes, based on their availability and ability to meet the SLO requirements of the workflow. We evaluate HyperDrive using a wildfire disaster response use case with high Earth Observation data processing requirements and stringent SLOs, showing that it enables the design and execution of such next-generation 3D scenarios with 71% lower network latency than the best baseline scheduler.**

***Index Terms*—serverless computing, scheduling, 3D continuum, orbital edge computing, LEO satellites, SLOs**

I. Introduction

As of 2024, there are over 8,000 low Earth orbit (LEO) satellites orbiting the Earth [1]. Satellites have traditionally communicated with each other via ground stations. Lately, inter-satellite laser links (ISLs) aim to connect satellites and create a large orbital network topology [2]. Starlink is currently the largest LEO mega-constellation with about 7,000 satellites in orbit [3] and almost 12,000 total satellites approved by the FCC, which must be launched by 2028 [4]. By 2029 a second LEO mega-constellation is planned to be available with 3,236 satellites [5] and more competition is solicited by the FCC [3]. ISL capability allows LEO satellites to act as ground edge nodes, processing data directly in orbit and near the data source, such as Earth Observation (EO) satellite data. This opens up opportunities for new computing paradigms in space, such as Serverless Computing.

*The authors Cynthia Marcelino and Thomas Pusztai have contributed equally to this work.

Serverless Computing provides elastic scaling and infrastructure management. Serverless platforms automatically deploy functions, scaling up or down based on demand, thus avoiding idle resources [6]. To address the environmental heterogeneity of the Edge-Cloud-Space Continuum, Serverless platforms need scheduling mechanisms that identify environmental properties and their current conditions to deploy functions and meet their requirements [7].

Most common scheduling approaches focus on meeting requirements based on resources, network, application and energy [8, 9].

a) Resource-Aware: Schedulers [10] ensure that functions are executed on nodes capable of handling their computational requirements to prevent overloading any single node, which could lead to performance degradation or failures. In the Edge Cloud Space Continuum, resource-aware scheduling mechanisms [11, 12] dynamically allocate functions considering the infrastructure-specific resource characteristics such as CPU capacity and architecture, memory, and GPU. In Orbital Edge Computing (OEC), scheduling mechanisms [13, 14] address specific orbit characteristics such as satellite infrastructure resource and energy costs to transfer the data between satellites or to the ground stations. However, current approaches do not consider all aspects of Edge, Cloud, and Space as a unified continuum. They neglect the impact of resource temperature and heat generated by the task execution. Due to the substantial temperature variations on satellites between the daylight and eclipse periods of an orbit, tasks that require intense computation can produce too much heat, putting satellite components at risk of damage from overheating [15, 16].

b) Network-Aware: Nodes at the edge typically have different network characteristics than cloud nodes. These network characteristics include variations in end-to-end latency, bandwidth availability, and link reliability [8]. Network-aware schedulers [11, 17] consider these characteristics to optimize function placement, ensuring efficient and reliable communication. In OEC, schedulers [13] typically also address the intermittent ISL communication between satellites and high latency communication with ground stations. However, existing OEC schedulers are not built for serverless functions, so they cannot guarantee the complete execution of serverless workflows across the Edge Cloud Space Continuum. Existing schedulers do not consider the positions of satellites, which is essential to ensure the seamless execution of serverless

workflows from orbit to the Edge and Cloud. Therefore, existing schedulers fail to ensure that serverless functions can start, complete, and transfer all required data within the connectivity range of the satellite network.

c) Application and SLO-Aware: Applications have Service Level Objectives (SLOs) that define the expected performance and availability during their execution. To meet these requirements, SLO-aware schedulers [18, 19] need to consider not only infrastructure properties such as resource availability, but also workload characteristics. Although OEC schedulers ensure functions can execute in a specific node, they do not guarantee workload requirements, i.e., SLOs, such as maximum latency.

d) Energy-Aware: Schedulers consider the current power source and estimated task power consumption during the placement process. Energy-aware scheduling [20, 21] is crucial to prevent battery-powered devices from running out of power and to reduce overall power usage. By optimizing energy usage, schedulers ensure prolonged operational lifespans for edge devices and enhance sustainability, thus optimizing performance and longevity in the Edge Cloud Continuum. In OEC, energy-aware schedulers [13, 22] consider also the energy necessary to transmit the data either to other satellite nodes or to the ground station. However, existing schedulers overlook the satellite position during the energy consumption estimation. Despite tasks requiring a certain amount of power, the satellite can auto-recharge its batteries during the daylight periods.

Although current Serverless scheduling approaches address the heterogeneous devices on the Edge, they are not suitable for the specific environmental properties of the Edge Cloud Space 3D Continuum, such as satellite position and heat generation. Moreover, the current orbital scheduling approaches lack integration across the Edge Cloud and Space environment, essential for latency and function execution across the 3D Continuum.

In this paper, we introduce *HyperDrive*, a novel Serverless platform that seamlessly integrates Edge, Cloud, and Space Computing, creating a 3D Continuum. HyperDrive is part of Polaris[1], a SIG of the Linux Foundation Centaurus project[2], a novel open-source platform for building unified and highly scalable public or private distributed Cloud and Edge systems, which is now expanding into the 3D Continuum. HyperDrive leverages the specific capabilities of each layer of the 3D Continuum, such as Edge proximity to the data and satellite proximity to Earth observation data, to enable optimized serverless function deployment and execution.

The main contributions of this paper include:

1) The *architecture of the HyperDrive Serverless Platform*, which introduces novel components and mechanisms tailored to the unique characteristics of the 3D Continuum. HyperDrive enables functions to be seamlessly executed anywhere in the 3D Continuum, optimizing performance and reliability by ensuring that workflow SLOs are met.
2) The *HyperDrive scheduling model* is the foundation of our Serverless platform's scheduler, which is the main focus of our paper. The HyperDrive scheduling model considers constraints such as resource capacity, application SLO requirements, satellite temperature, and network load to minimize the end-to-end Serverless workflow latency.
3) Our *Heuristic Scheduling Algorithms for the 3D Continuum* enable the realization of the HyperDrive scheduling model using a flexible multi-criteria decision making (MCDM) approach. It first filters out nodes that are not capable of hosting a function and, then, scores the remaining nodes according to multiple criteria to find the best suited node for a function. Our prototype implementation is available as open-source[3]. HyperDrive achieves 71% lower end-to-end (E2E) network latency than the next best baseline approach.

This paper has eight sections. Section II presents the illustrative scenario and research challenges. Section III shows an overview of the HyperDrive Architecture for a Serverless Platform in the 3D Continuum. Section IV describes the Serverless Workflow Model, HyperDrive scheduling optimization model, and heuristic scheduling algorithms for the 3D Continuum. Section V details our implementation approach and describes the design of our experiments. Section VI discusses the results of the experiments, Section VII presents related work. Section VIII concludes the paper and outlines our future work.

II. Motivation

To further motivate our work we present an illustrative disaster response scenario and leverage it to derive research challenges.

A. Illustrative Scenario

Early detection of wildfires in remote areas is critical to mitigate their effects. Our scenario (Fig. 1) involves using a combination of drones, LEO satellites, and ground-based Edge nodes that compose a serverless workflow for real-time wildfire detection, inspired by [23, 24, 25, 26]. The drones operate in high-risk wildfire areas, such as California during the summer, monitoring specific zones and capturing video and sensor data to watch for signs of wildfires. They send the data to the nearest Edge node using streaming frameworks or, when out of range, transmit it to LEO satellites acting as in-orbit Edge nodes. Once a fire is detected, LEO satellites incorporate satellite Earth Observation (EO) data for processing. Our serverless workflow processes the data close to the source to improve latency and reduce network overhead. In some situations, functions are executed directly on LEO satellites due to the data's proximity to EO data and the high latency associated with downloading data to the ground.

Fig. 2 shows our Serverless workflow with four Serverless functions, partially executed on the Edge, partially executed in-orbit and partially executed in the Cloud. During the *Ingest*

[1]https://polaris-slo-cloud.github.io

[2]https://www.centaurusinfra.io

[3]https://github.com/polaris-slo-cloud/hyper-drive

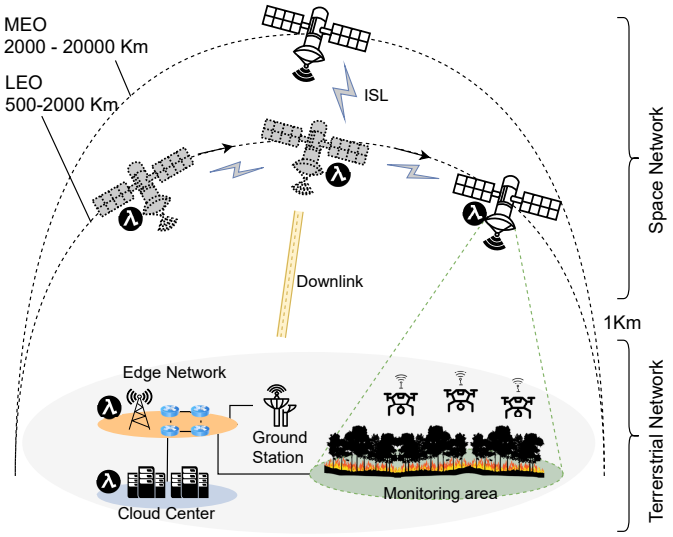

Fig. 1: Illustrative Scenario: Wildfire detection with on-ground and in-orbit Serverless Edge Computing

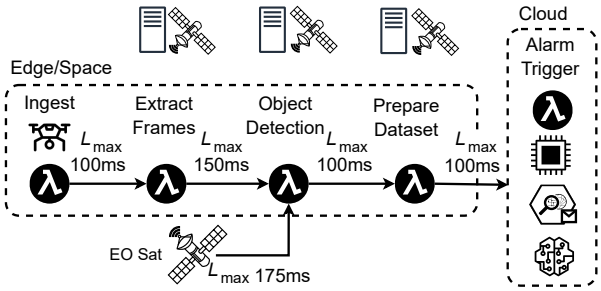

Fig. 2: Simplified Serverless Workflow for Wildfire Detection

stage, real-time videos are transmitted to Edge nodes on the ground or in-orbit. The *Extract Frames* function processes small video chunks received from Ingest stage and extracts image frames. *Object Detection* functions identify wildfire patterns in the extracted images, such as smoke, flames, or hotspots. The *Prepare Dataset* function prepares the data for resource-intensive tasks. The processed data is transmitted to the Cloud for storage and more resource-intensive tasks, such as machine learning model inference. In the Cloud, *Alarm trigger* functions evaluate the data and decide whether to trigger local emergency responses or deploy more drones to a specific area to confirm the wildfire before triggering an alarm.

Serverless computing allows dynamic scaling and processing close to the data source. By running Serverless functions directly on LEO satellites, we can combine data from the drones on the Earth and from EO satellites to process data as soon as they are produced. Atmospheric interference reduces link speeds to ground stations, typical speeds are around 300 Mbps [27]. Thus, downlinking data from EO satellites to Earth would take too long due to the large volume of data, e.g., each of the ESA Sentinel 2 satellites supplies high resolution images for a swath of 290 km in 13 spectral bands, producing about 1.5 TB of data per day [28, 29]. Since EO satellites only downlink to dedicated ground stations, the data may even be queued [30]. For Sentinel-2 "real-time" product availability is defined as "no later than 100 minutes after data sensing" [31], which violates the satellite data ingestion link SLO of the wildfire application. ISLs between EO satellites and LEO satellites are much better suited for large EO data volumes, since their speeds can be much higher – recently a 100 Gbps ISL from GEO to LEO has been demonstrated [32]. Hence, it is much faster to uplink a one GB ML model to the satellite than to downlink the EO data to a ground station. Drone videos are also moderate in size, e.g., a three minute 4K video from the FLAME2 dataset [33] amounts to 2.2 GB, which qualifies for uplinking to a LEO satellite in real-time.

Combining satellite EO data with drone data on LEO satellites allows reducing the time it takes to analyze and respond to wildfires. Additionally, it provides a reliable alternative when Edge nodes are out of range or experiencing connectivity issues. Scheduling the functions to execute in orbit ensures that wildfire detection and monitoring continue uninterrupted, even if ground-based infrastructure faces limitations. It allows immediate data processing and decision-making in orbit, reducing delays and ensuring continuous, real-time monitoring. As a result, we can decrease response times to wildfire threats. However, there are several challenges associated with scheduling Serverless functions in 3D continuum.

B. Research Challenges for Scheduling in the 3D Continuum

Based on the illustrative scenario, we identify several key requirements for scheduling serverless functions on LEO satellites in orbit as follows:

RC-1 Satellite Availability: Unlike Edge nodes, which have fixed positions, LEO satellites are constantly in motion as they orbit the Earth, which impacts their availability and communication windows [34]. A satellite must be within range of ① the drone, ② the EO satellite, and ③ the ground station to be considered available for scheduling. Specifically, the satellite needs to be within the drone's range to receive real-time video transmissions from Earth. At the same time, it must also be within the range of the EO satellite to receive and relay additional monitoring data. In addition, the satellite must be within range of ground stations, which have Cloud control planes for tasks such as scheduling. However, the term "in range" is more complex than direct line of sight. Since satellites can communicate via ISLs [35, 36], a satellite can be in range, if the bandwidth and latency via ISLs is acceptable for the purpose of the communication (e.g., data transfer). According to a recent study [37] Starlink's median roundtrip latency (client-LEO-Cloud) is 40-50 ms; the theoretical roundtrip latency between New York and London when routing exclusively through ISLs is 58-66 ms [35]. As satellites move in and out of range, the Serverless platform must continuously adapt, reallocating resources and re-establishing communication links. Therefore, satellite availability is more dynamic and complex compared to static Edge nodes.

RC-2 Power Supply: The scheduler must consider the satellite's power state, including its batteries' current charge

level and the overall health of its energy storage system. Given the increasing computing power in satellites and the strict size constraints for some of them [15], the scheduler must be aware of the energy requirements of specific serverless functions to ensure that the satellite has enough power reserves to execute these functions without depleting its energy resources. Finally, a CubeSat's solar panels produce only up to 7 W of power [38], while batteries can have a density of up to 190 Wh/kg [39]. This means that a satellite might not be designed to fully recharge their batteries in a daylight period of an orbit. Thus, the scheduler must evaluate whether the power expenditure of its workloads can be compensated with solar power before the battery depletes.

RC-3 Computing Capacity & Heat Generation: LEO satellites are deployed with fixed and limited resources that cannot be patched or upgraded throughout their lifetime. These satellites are built to consume minimal energy and are equipped with minimal components to reduce weight and, consequently, launch costs. As computing increases, the temperature also rises. Since there is no atmosphere in space, heat dissipation mainly occurs through thermal radiation and lack of exposure to the sun. LEO satellites typically face temperatures from $-120°$C in the shade to $+120°$C when in the sunlight [40]. This situation can lead to prolonged high temperatures, affecting the performance of critical components such as the CPU [15, 24, 41, 42]. Therefore, the scheduler must consider not only the existing processing capacity but also the current temperature of the components and how long they potentially need to dissipate the heat.

RC-4 Scalability: Due to the fixed number of satellites in orbit and the increased costs associated with launching new ones, horizontal scaling presents a significant challenge. Compared to the ground data centers, where additional servers can be easily deployed to meet increasing demand, the satellite network is limited by the number of satellites currently in orbit. This physical resource constraint and fixed number of nodes make it challenging to auto-scale effectively to meet varying workload demands [43, 44].

RC-5 SLO Awareness: Serverless workflows must meet specific Service Level Objectives (SLOs) to ensure performance and reliability. These SLOs typically include minimal latency and bandwidth, which are essential for maintaining optimal service performance. Maintaining SLOs on the ground can already be challenging [45, 46] and these challenges are exacerbated by the network specifics, orbital movements, battery, and heat conditions of satellites [47]. Therefore, to ensure performance and reliability, the scheduler must consider the state of multiple nodes when enforcing workload SLOs.

RC-6 Workflow Dependencies: In a mixed environment, Serverless workflows can be executed on ground-based or LEO Edge nodes. The scheduler needs to take into account the workflow composition to identify the dependencies and interactions between the functions. Additionally, the scheduler must consider the placement of these functions within the workflow to ensure that interdependent tasks are located closely together to minimize latency and maximize efficiency [11]

III. Architecture Overview of a Serverless Platform for the 3D Continuum

HyperDrive is a novel serverless platform specifically designed for the 3D Continuum, as shown in Fig. 3. To achieve that, our platform proposes six different layers: (a) an infrastructure layer that unifies the computing resources in the 3D Continuum, (b) a core platform layer for efficient and optimized function deployment and execution, (c) a function runtime layer for lightweight and low-latency execution, (d) a function model to allow developers change function behavior, (e) monitoring and tracing for real-time insights and (f) a stewardship layer layer composed of frameworks that enforce governance, security and compliance. Each platform layer introduces components to address the research challenges presented in Section II-B.

A. Infrastructure Layer

This layer includes common computing resources across the Edge-Cloud-Space 3D Continuum, such as computing, storage, and network. Each computing layer, i.e., Edge, Cloud, and Space within the 3D Continuum, has specific properties that require tailored resource management. In the Edge layer, the HyperDrive infrastructure layer manages battery power to prevent Edge devices from running out of power. For example, by providing battery level information to the scheduler so that only drones with enough battery capacity execute the `Ingest` function in the wildfire serverless workflow. In the Cloud, it handles heterogeneous provider-managed services such as AWS S3 storage and Azure storage for storing high-resolution satellite images or more intense computing tasks such as running inference on machine learning models. In the space layer, the platform manages thermal regulation and power to prevent satellite depletion. Furthermore, the infrastructure layer provides satellite positioning information, which is critical for HyperDrive scheduler to place functions within range to ensure efficient data exchange between the functions. These computing resources create a unified infrastructure layer that adapts to the heterogeneous and dynamic requirements of the 3D Continuum, enabling the HyperDrive Serverless Platform to adjust to resources based on demand, ensuring seamless execution across the Edge-Cloud-Space Continuum. This is a key prerequisite to achieving our vision of self-provisioning infrastructures [48].

B. Core Platform Layer

This layer incorporates components responsible for managing and orchestrating tasks across the 3D Continuum. It manages the configuration, deployment, computation balancing [49], and auto-scaling of serverless functions, handling their lifecycle and scaling resources up and down based on the workload demand, such as the wildfire serverless workflow. Moreover, the storage enables HyperDrive to store function deployment properties and specific function configurations, such as parameters, state management settings, and SLOs. To allocate functions effectively, HyperDrive scheduler considers the 3D Continuum requirements described in Section II-B. HyperDrive scheduler

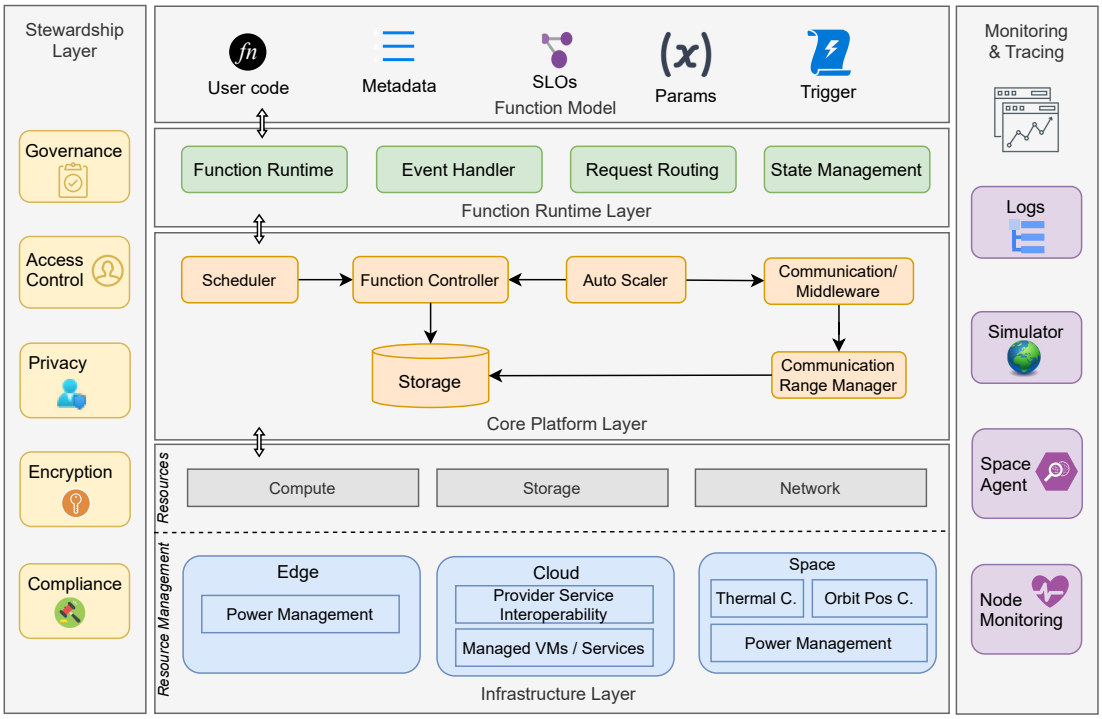

Fig. 3: Architecture Overview of a Serverless Platform for the Edge-Cloud-Space 3D Continuum

utilizes resource-based scheduling mechanisms, commonly used by various Edge and Cloud schedulers [11, 18, 19]. HyperDrive considers different requirements, including resource capacity, workload SLO, power supply, and satellite position, to make decisions using an MCDM approach. To ensure scalability in the large 3D Continuum, HyperDrive is a distributed scheduler that operates with multiple instances. Distributed scheduling requires keeping node state information in sync among the scheduler instances and handling scheduling conflicts. To address these two challenges, each HyperDrive scheduler instance obtains a function's candidate nodes and their states from the Monitoring Agent using sampling, similar to other distributed schedulers [19, 50], and handles conflicts using the MultiBind mechanism described in Section IV-C. By integrating Edge-Cloud-Space requirements, HyperDrive ensures the optimal placement and performance of Serverless functions within the 3D Continuum, thus meeting application demands and respecting boundaries between ground and space requirements such as latency and financial costs.

C. Function Runtime Layer

The Function Runtime layer consists of components such as Function Runtime, Event Handler, Request Routing, and State Management. The Function Runtime relies on lightweight frameworks such as WebAssembly to provide safety, isolation, and low-latency communication [51, 52]. In our illustrative scenario, the runtime utilizes function locality to reduce network overhead, ensuring satellites leverage local mechanisms such as inter-process communication (IPC) to exchange data between functions on the same host. Thus, the function runtime reduces latency and ensures that communication between co-located functions remains local, avoiding unnecessary ISL communication. Serverless stateless design pushes functions to leverage external services for state management [7, 53]. HyperDrive State Management leverages mechanisms such as short-term memory state [54, 55] to allow serverless workflows, like wildfire detection, to maintain their state between executions, thereby avoiding the overhead of external service communication. Due to the different properties, such as bandwidth, latency, and jitter, between Edge, Cloud, and Space, HyperDrive Request Routing optimizes load balancing by forwarding requests to functions in the vicinity, thus reducing latency by avoiding communication between functions cross-environment, such as Edge and space. The Event Handler manages events from different sources, such as image drones and EO data, to ensure proper function invocation. The components in this layer ensure a seamless execution of serverless functions to meet the workload requirements effectively. The function runtime layer offers lightweight mechanisms for executing functions on limited resource devices across the 3D Continuum.

D. Function Model

This layer introduces a function model that allows developers to define specific behaviors, such as SLOs and trigger types, in addition to the function code, parameters, and metadata. Developers can specify the type of event - such as streaming, asynchronous, or synchronous - that the function should process. In the 3D continuum, the function model enables users to react to specific satellite events, such as changes in orbit or satellite payload data received. Specifically, in the wildfire serverless

workflow, drones at the Edge trigger `ExtractFrames` function using video streams, while `ObjectDetection` are triggered by single image frames as data input. Moreover, developers may specify certain SLOs, such as a maximum latency of 100 ms between two functions, for instance, between `ExtractFrames` and `ObjectDetection`. Without coding effort, developers can indicate whether functions are stateless or stateful. The HyperDrive Function Model layer abstracts the underlying infrastructure, enabling developers to manage serverless workflows without the complexity of coding or infrastructure management. Finally, this layer offers specifically tailored programming modes, e.g., to facilitate dealing with large-scale, heterogeneous data sources [56].

E. Monitoring & Tracing

This layer is composed of components that enable real-time tracking and monitoring such as Space Agent, Node Monitoring, distributed logging systems, and a simulator that enables developers to simulate functions execution without deploying the function on the expensive and limited infrastructure, e.g., on the satellites. The Monitoring Agent is designed to track and analyze key performance metrics across the 3D Continuum, including Edge, Cloud, and space infrastructure. It watches computing capacity, memory usage, and resource utilization across all nodes to prevent overloading and ensure efficient function execution. Additionally, it monitors network quality of service (QoS) parameters, including bandwidth and latency, to maintain compliance with workload SLOs. By monitoring the common properties of different layers, the Monitoring Agent enables seamless integration and reliability across the 3D Continuum.

The Space Agent is specifically designed to address the requirements of in-orbit computing. It is responsible for tracking the unique properties of the space environment, including the availability of LEO satellites, taking into account their rapid movement in orbit and their limited communication windows. Additionally, the Space Agent manages ISLs and ground-satellite network graphs to ensure that the satellite can meet the user-defined latency SLOs. It also monitors the satellite power supply, identifying the current charge levels of batteries and their position in relation to solar energy generation, to ensure that serverless functions are assigned only to satellites with sufficient battery capacity. Furthermore, the Space Agent monitors satellite thermal levels to prevent overheating caused by high computational load or prolonged usage, which could result in execution failures and potentially lead to long-term hardware damage. By addressing these space-specific requirements, the Space Agent plays a crucial role in optimizing the scheduling, deployment and execution of serverless functions across the 3D Continuum.

F. Stewardship Layer

This layer ensures serverless functions' secure, compliant, and efficient operation across the 3D Continuum. Its components enforce compliance with environmental and data protection regulations relevant to the workflow, such as wildfire monitoring. Encryption leverages mechanisms to protect stored sensitive information, while privacy mechanisms ensure that personal or location-based data is handled confidentially by the system. Moreover, Access Control implements role-based access and fine-grained permissions to restrict unauthorized access and actions. At the same time, the Governance component oversees these processes, enforcing policies and standards to maintain system integrity, security, and performance across the platform under expected conditions but also under uncertainty [57].

IV. HyperDrive SLO-Aware Scheduler for 3D Continuum

The HyperDrive scheduler is designed to address the challenges that arise in the placement of serverless functions in the 3D Continuum first using an optimization problem and, then, using an MCDM approach. Without loss of generality, we assume that every serverless function is part of a serverless workflow, which we model as follows.

A. Serverless Workflow Model

A serverless workflow can be modeled as a directed acyclic graph (DAG) with every node representing an executable task, i.e., a serverless function or an operator, such as a condition, fork, or loop, and every link representing an invocation of the next node. The workflow DAG for our wildfire detection use case is is part of Fig. 2; all executable tasks are by nodes with a λ sign. For the purpose of scheduling we refer to a serverless function instance as a *task*.

The workflow graph can be annotated with metadata relevant to its tasks. Each task node is annotated with information such as container image, resource requirements, preferred location, and SLOs. Since many network connections in the 3D Continuum are not as reliable as within a Cloud data center, tasks need to be able to specify special needs regarding the network quality of service (QoS) for incoming and outgoing links. To this end each workflow link can be annotated with network SLOs, specifically with maximum allowed latency, minimum bandwidth, maximum jitter, and maximum packet drop percentage.

In many cases serverless functions do not only depend on data from the predecessor function(s), but also on an external data source. In the 3D Continuum such an external data source may be, e.g., an S3 storage in a Cloud data center or high resolution data from an EO satellite. Workflow SLOs may result in special requirements for the connections to these data sources, i.e., network QoS SLOs. This entails that a workflow DAG must capture not only executable nodes, but also data source nodes and support SLOs on their outgoing links. The "EO Sat" at the bottom of Fig. 2 represents an EO satellite node as a data source with its outgoing link providing EO data and imposing a max latency SLO of 175 ms to the `ObjectDetection` function. This metadata gives the HyperDrive scheduler all the required information to make a suitable placement of the workflow's tasks.

B. HyperDrive Scheduling Model

Let a Serverless workflow be a DAG $\mathcal{W} = (\mathcal{F}, \mathcal{E})$, where each node in the DAG represents a function in the Set $\mathcal{F}$ and each edge in Set $\mathcal{E}$ represents the invocation of the next task. Let the network graph be $\mathcal{G} = (\mathcal{N}, \mathcal{L})$, where $\mathcal{N}$ is a Set of nodes and $\mathcal{L}$ the communication latency between the nodes. The scheduling goal to minimize the latency in the Serverless workflow $\mathcal{W}$ execution in the 3D Continuum, effectively mapping the workflow $\mathcal{W}$ onto the network graph $\mathcal{G}$. To achieve this, we consider the following constraints:

Resource Capacity: This constraint ensures that every node has enough resources to process the scheduled function, maintaining system stability and performance. Additionally, this constraint helps balance the system load across the nodes, optimizing the overall utilization of available resources. Therefore, the total resource demand D_i of function i on each node n in $\mathcal{N}$ must not exceed its availability resources R_n:

$$\sum_{i \in \mathcal{F}} D_i \leq R_n \quad \forall n \in \mathcal{N} \tag{1}$$

Network SLOs: This constraint ensures that data transfer between functions occurs within acceptable timeframes, ensuring that functions perform as expected. This means that communication between functions must meet performance criteria defined by the user to minimize delays. Thus, the SLOs latency S_{ij} must be met for each function invocation pair (i, j) in functions $\mathcal{F}$. The latency L_{nm} of the path between nodes n, m in $\mathcal{N}$ must not exceed the SLO S_ij:

$$L_{nm} \leq S_{ij} \quad \forall (i,j) \in \mathcal{F}, \forall (n,m) \in \mathcal{N} \tag{2}$$

Temperature: Managing thermal conditions not only protects the physical integrity of the nodes but also maintains optimal performance and longevity, specially in space where extreme temperature variations are common. Therefore, the temperature of each node n in $\mathcal{N}$ must not exceed its maximum allowed temperature T_{max} , considering the maximum temperature caused by the satellite exposure to the sun and the temperature sum increase due to the execution of the each function T_{exc}:

$$T^n_{\text{orb}} + \sum_{i \in \mathcal{F}} T^{in}_{\text{exc}} \leq T^n_{\text{max}} \quad \forall n \in \mathcal{N} \tag{3}$$

The scheduler goal is to minimize the total latency in the workflow execution by summing the latency L_{nm} between nodes n, m in $\mathcal{N}$ for each function invocation i, j in $\mathcal{E}$, where variables x_{in} and x_{jm} is a binary that indicates function placement to node. The optimization problem can be defined as follows:

$$\begin{aligned} \min_{x} \quad & \sum_{(i,j) \in \mathcal{E}} \sum_{n,m \in \mathcal{N}} L_{nm} x_{in} x_{jm} \\ \text{s.t.} \quad & \sum_{i \in \mathcal{F}} D_i \leq R_n \quad \forall n \in \mathcal{N} \\ & L_{nm} \leq S_{ij} \quad \forall (i,j) \in \mathcal{F}, \forall (n,m) \in \mathcal{N} \\ & T^n_{\text{orb}}(t_i) + \sum_{i \in \mathcal{F}} T^{in}_{\text{exc}} \leq T^n_{\text{max}} \quad \forall n \in \mathcal{N} \\ & x \in \{0,1\} \quad \forall i \in \mathcal{F}, \forall n \in \mathcal{N} \end{aligned} \tag{4}$$

The HyperDrive scheduling optimization model addresses key constraints of resource capacity, network SLOs, and temperature to guarantee efficient and reliable execution of Serverless workflows in the 3D Continuum. Minimizing total latency while adhering to these constraints enables the scheduler to make placement decisions across diverse environments, ensuring optimal performance and system stability. The consideration of satellite costs during scheduling is currently out of scope, since there are currently no pricing models for satellite nodes available.

C. Heuristic Scheduling Algorithms for the 3D Continuum

Given the high computational complexity of the aforementioned optimization problem, heuristics are needed to allow implementing the HyperDrive scheduling model for the 3D Continuum. We now examine the heuristic scheduling algorithms that approximate the aforementioned optimization problem. To this end we rely on an MCDM approach consisting of a sequence of filters that remove nodes that are not capable of hosting the task and scoring algorithms that determine the best suitable node among the eligible ones.

1) Vicinity Selection: Since the 3D Continuum may consist of tens of thousands of nodes, we need to perform a preselection of nodes before we can address the constraints of the optimization problem. To this end, HyperDrive contacts the orchestrator to select a set of candidate nodes that are located in the vicinity of the desired location specified by the task or in the vicinity of its predecessor task. The definition of the term "vicinity" can be configured independently for each part of the 3D Continuum. For example, for the Cloud any data center node within a radius of 500 km of the desired location may be selected, while the radius could be 200 km for Edge nodes, and 2,000 km for satellites. Akin to the vicinity, the total size of the candidates set and its composition can be configured as well, e.g., 500 total nodes consisting of 40% Cloud nodes, 40% Edge nodes, and 10% Space nodes.

2) Resource Checking: After selecting the set of candidate nodes, HyperDrive first filters out all nodes that do not meet the resource requirements of the task. Specifically, it checks the CPU architecture, CPU cores, memory, GPU (if present), local storage, and minimum battery charge (if the node has a battery) requested by the task.

3) Network SLOs Enforcement: HyperDrive uses a combination of filtering and scoring to ensure that the network QoS SLOs constraints for the incoming links of the task are fulfilled and the nodes with the best network properties are preferred.

For filtering we use Algorithm 1. It iterates through all network SLOs for incoming links, originating from predecessor tasks and external data sources (if any) and queries the network QoS values for the lowest latency path between the candidate node and the node hosting the predecessor task or the data source. If the network SLO requirements are not met, the node is discarded.

For scoring we iterate through the aforementioned network paths again to determine the highest latency value We assign the highest score, i.e., 100, to the node with the lowest latency and zero to the node with the highest latency; all nodes in between are assigned proportional scores in the target interval.

Algorithm 1 Network SLOs Filter.

Input: t: Task to be scheduled;
cn: Candidate node;
$W = (V_W, E_W)$: Workflow DAG;
$N = (V_N, E_N)$: Network graph;
$S_t = \{(v,s) \forall v \in V_W \ s.t. \ (v,t) \in E_W \wedge s \neq \varnothing\}$: Network SLOs for incoming links of t;
Output: $true$ if cn can host t, otherwise $false$;

for all $(v,s) \in S_t$ **do**
  $u \leftarrow$ GETHOSTNODE(v, W, N)
  $q \leftarrow$ QUERYNETWORKQOS(u, cn, N)
  if LATENCY$(q) >$ MAXLATENCY(s) **then**
    return $false$
  end if
  if BANDWIDTH$(q) <$ MINBANDWIDTH(s) **then**
    return $false$
  end if
  if JITTER$(q) >$ MAXJITTER(s) **then**
    return $false$
  end if
  if PACKETDROP$(q) >$ MAXPACKETDROP(s) **then**
    return $false$
  end if
end for
return $true$

4) Temperature Optimization: The algorithm to enforce the temperature constraint is geared specifically towards the Space part of the 3D continuum to prevent satellites from overheating due to excessive workload when in the sunlight. Since a satellite that is close to overheating will reduce its computational power to prevent damage. Thus, HyperDrive aims to prefer satellites, where the new task will not cause a problematic temperature. This decision involves a complex estimate based on the current temperature of a satellite's compute unit, the expected duration of the task on the satellite's hardware, the required CPU and, possibly, GPU resources, the heat generated by these resources over the duration of the task, and the highest environmental temperature (based on in-orbit sunlight exposure) expected for the duration of the task. This is encapsulated in the scoring logic of Algorithm 2.

The algorithm first tries to get a duration estimate d_t for the task. This can be supplied by the user or through preceding profiling (on hardware similar to the satellite's) or the maximum response time SLO of the task can be used. If none of these values are available the score is calculated based on the current temperature of the satellite. If d_t value is available, it is used in conjunction with the requested resources to estimate the computation-based temperature increase

Algorithm 2 Temperature Optimization Scoring.

Input: t: Task to be scheduled;
cpu_t: CPU cores requested by t;
gpu_t: GPU cores requested by t;
n: Node to be scored;
$temp^n_{max}$: Maximum operating temperature for n;
$temp^n_{rec}$: Recommended high temperature for n;
Output: Score for the node n in the range $[0; 100]$;

if NODETYPE$(n) \neq$ "$satellite$" **then**
  return 100
end if
$d_t \leftarrow$ GETEXPECTEDDURATION(t)
if $d_t == nil$ **then**
  ▷ If d_t is unknown use the current temperature to compute the score.
  $temp_{curr} \leftarrow$ GETCURRTEMP(n)
  return CALCSCORE$(temp_{curr}, temp^n_{rec}, temp^n_{max})$
end if

$temp_{inc} \leftarrow$ ESTIMATECOMPTEMPINCREASE(n, d_t, cpu_t, gpu_t)
$temp^{orb}_{max} \leftarrow$ ESTIMATEMAXORBITTEMP(n, d_t)
$temp^t_{max} \leftarrow temp^{orb}_{max} + temp_{inc}$
return CALCSCORE$(temp^t_{max}, temp^n_{rec}, temp^n_{max})$

function ESTIMATEDURATION(t)
  $d_t \leftarrow$ GETEXPECTEDDURATION(t)
  if $d_t \neq nil$ **then**
    return d_t
  end if
  return MAXRESPONSETIMESLO(t)
end function

▷ Estimates the temperature increase due to computation
function ESTIMATECOMPTEMPINCREASE(n, d_t, cpu_t, gpu_t)
  $temp_{inc} \leftarrow$ CPUTEMPINCREASE(n, cpu_t, d_t)
  $temp_{inc} \leftarrow temp_{inc} +$ GPUTEMPINCREASE(n, gpu_t, d_t)
  return $temp_{inc}$
end function

function CALCSCORE$(temp_{exp}, temp_{rec}, temp_{max})$
  if $temp_{exp} \leq temp_{rec}$ **then**
    return 100
  end if
  if $temp_{exp} > temp_{max}$ **then**
    return 0
  end if
  $range \leftarrow temp_{max} - temp_{rec}$
  $over_{rec} \leftarrow temp_{exp} - temp_{rec}$
  return $\lfloor \left(1 - \frac{over_{rec}}{range}\right) * 100 \rfloor$
end function

$temp_{inc}$. Subsequently, we determine the maximum expected environmental temperature $temp^{orb}_{max}$ during the orbit(s) within the duration of the task. The sum of these two temperatures is the maximum expected temperature for the satellite during the execution of the task and is used for computing the node's score. If the expected temperature is below the recommended temperature or above the maximum temperature, 100 or zero are returned respectively. Otherwise, a score is computed based on how much the temperature will go into the range between recommended and maximum temperature.

5) Multi Commit: Finally, all scores are accumulated for each node and the nodes are sorted by their scores. The HyperDrive scheduler, then, contacts the orchestrator to assign the task to the highest scored available node using a multi-commit approach [19]. Since multiple schedulers may be active, the orchestrator checks if the required resources are still available on the selected node. If that is the case, the task

is committed to the node, a success message is returned to the and the scheduler updates the information in the DAG of the workflow instance. If the orchestrator reports that the required resources are no longer available, the result is a scheduling conflict, which most distributed schedulers resolve by rerunning the scheduling pipeline. To avoid doing this, HyperDrive tries committing the task to the second best node and, if that fails too, to the third best node, before triggering a rescheduling of the task. This multi-commit technique has been shown to decrease the number of scheduling conflicts by a factor of 10 with respect to immediately rescheduling the task [19].

V. Implementation & Experiments Design

To evaluate the HyperDrive scheduler we focus on the quality of the scheduling decisions and its scalability. Since HyperDrive is, to the best of our knowledge, the first serverless scheduler specifically designed for the 3D Continuum, we compare it against three theoretical scheduling approaches: Greedy First-fit, Round-robin and Random scheduling.

A. Implementation

The prototype of the HyperDrive scheduler is implemented in Python as available as open-source[4]. Since it is not feasible to run experiments on a low earth orbit (LEO) satellite mega constellation, we have connected our scheduler to a modified version of the StarryNet satellite constellation simulator [58]. The connection to the simulator is fully abstracted as an orchestrator interface, so that the simulator can be easily swapped. StarryNet normally executes Docker containers for all nodes. However, since we are interested in benchmarking the scheduling algorithms, we have replaced the containers with an in-memory nodes manager that tracks the available resources.

We have implemented the 3D Continuum-specific scheduling heuristics described in Section IV-C. StarryNet precomputes latencies between adjacent nodes for the entire duration of an experiment. For each new time index, we use these latencies to update our network graph for network SLOs enforcement. Due to the absence of real satellite hardware information, we rely on reasonable estimates for the temperature optimizations.

B. Experiments Design

With our experiments we evaluate two critical aspects of the HyperDrive scheduler: (i) scheduling quality with respect to latency and satellite temperature management and (ii) scalability.

For assessing the scheduling quality we examine two major quality objectives. The primary objective is the latency achieved between the individual tasks of a serverless workflow and the E2E latency. The secondary objective is the intelligent selection of satellite nodes with respect to their temperature situation, i.e., satellites should be chosen, which will not overheat and reduce computational power while processing a task.

To set up the experiment we use TLE data, obtained on July 2, 2024 from CelesTrack[5], describing the orbits of 6,192 nodes of the Starlink[6] LEO satellite constellation. We deploy our wildfire detection use case, whose workflow is shown in Fig. 2. We assign the `Ingest` function to a drone flying over a region of California, USA that is prone to wildfires and trigger the scheduling of the remaining functions as the simulation progresses. Since the StarryNet only supports satellite and ground station nodes, we model the drone as a ground station node. Since we evaluate the scheduling at the time when the second function needs to be placed, we do not require any movement from the drone, hence modeling it as a ground station does not limit our evaluation scenario. All experiments are run using Python 3.12 on Ubuntu 20.04 LTS on a Windows Subsystem for Linux 2 VM with 8 vCPUs and 8 GB of RAM. The VM is hosted on a laptop running Windows 10 22H2 on a Whiskey Lake-U generation Intel Core i7 processor.

TABLE I: Infrastructure Sizes used for Evaluation.

Satellites	Edge Nodes	Cloud Nodes	Total Nodes
1,008	100	10	**1,118**
2,016	200	20	**2,236**
3,024	300	30	**3,354**
4,032	400	40	**4,472**

We benchmark HyperDrive against the following theoretical schedulers, which we use as baselines:

- Greedy First-fit
- Round-robin
- Random selection

For evaluating the scalability we want to examine how HyperDrive scales with respect to the infrastructure size. To this end, we benchmark the placement of wildfire detection workflow on increasing infrastructure sizes. For Cloud and Edge nodes we simulate nodes in the region the workflow is deployed in, while for satellites we simulate an entire constellation with the current 72 orbital planes of Starlink and an equal number of satellites per plane. Specifically, we use the infrastructure sizes described in Table I – node that the numbers in this table refer to our simulation only, which is limited by the resources of our host machine.

We execute five iterations of every scheduler's placement of the wildfire detection workflow on each of the four infrastructure sizes. We examine the achieved E2E latencies and temperature characteristics to evaluate the scheduling quality of all four schedulers and HyperDrive's processing time per task to assess its scalability.

VI. Experimental Results

A. Scheduling Quality

To evaluate the scheduling quality we examine the network latencies achieved by the placements and the temperatures of the selected satellites (if any).

Fig. 4 shows the mean network E2E latencies achieved by the four schedulers across all 20 experiment iterations, i.e., five iterations for each of the four infrastructure sizes. For clarity, the shown latencies are the sum of the network latencies only,

[4]https://github.com/polaris-slo-cloud/hyper-drive

[5]https://celestrak.org/NORAD/elements/

[6]https://www.starlink.com

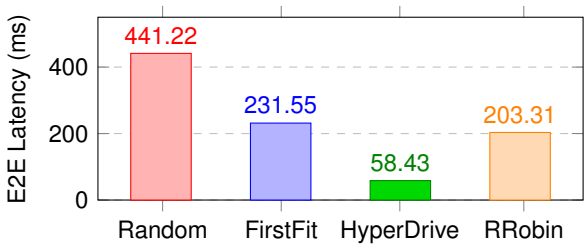

Fig. 4: Wildfire Detection Workflow Mean E2E Latency per Scheduler.

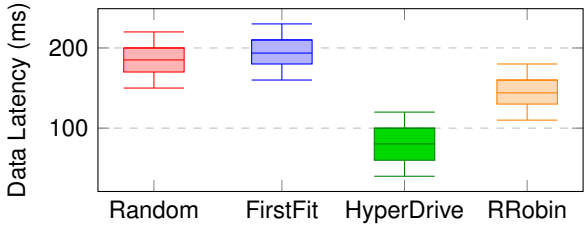

Fig. 5: Data Latency per Scheduler

without function execution times. The E2E network latency SLO, without function execution times, across all four functions of the wildfire workflow is 350 ms. While all schedulers, except for the Random scheduler, meet the E2E network latency SLO, HyperDrive clearly has the lowest latency, because it actively optimizes for it. HyperDrive's E2E latency is 71% lower than Round-robin's, which is the second best. While Greedy First-fit and Round-robin meet the E2E network SLO, they violate individual function network SLOs in about 33% of the cases for Greedy First-fit and in 30% of the cases for Round-robin. HyperDrive fulfills all function network SLOs.

Apart from inter-function network SLOs, the wildfire detection workflow also defines a network SLO for an EO satellite data source. The `object-det` function requires a maximum latency of 175 ms to the respective EO satellite. Fig. 5 shows the EO data latencies achieved by the schedulers. Random and Greedy First-fit violate the SLO. HyperDrive and Round-robin fulfill it on average, albeit Round-robin violates the SLO in 35% of the cases. HyperDrive always fulfills it, because its filtering does not allow scheduling on nodes that would violate the SLO.

The secondary optimization objective after the network latency, is satellite temperature measurement to avoid overheating. Fig. 6 shows a heat map for the three scheduled functions that documents cases when the functions are scheduled on satellites and their temperature exceeds the recommended operating temperature. HyperDrive places 34 of the total 60 function instances (56.7%) across all iterations on satellites and never exceeds the recommended temperature. The Random scheduler places 56 of 60 function instances (93.3%) on satellites and exceeds the recommended temperature in 23 (41%) of these cases; in three cases it even exceeds the maximum operating temperature. Round-robin schedules all 60 function instances on satellites and exceeds the recommended temperature in one third of the cases; in four cases it exceeds the maximum

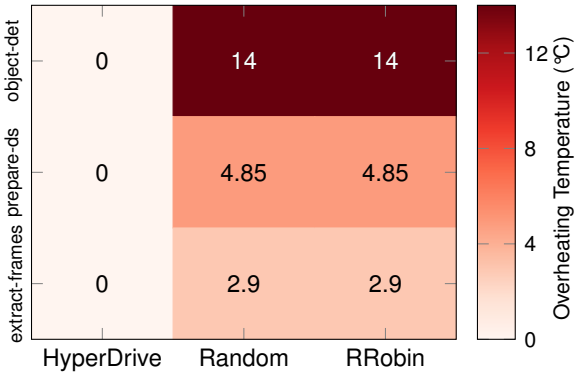

Fig. 6: Scheduling Overheating Map.

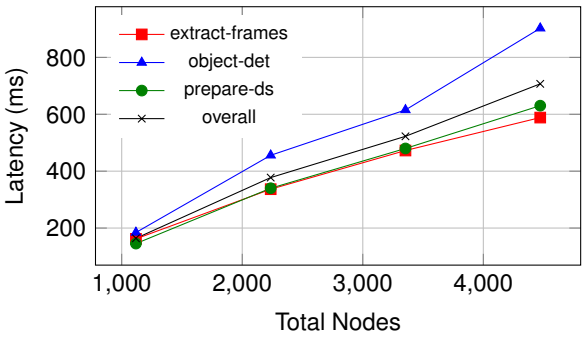

Fig. 7: HyperDrive Scheduling Latency Across Infrastructure Sizes.

operating temperature. It should be noted that as the number of Edge nodes increased in the two larger infrastructure sizes, HyperDrive selected more Edge nodes instead of satellites, due to their favorable network latencies; for the smaller two infrastructure sizes 86.7% of the nodes were satellites, while for the larger two only 33.3% were satellites.

B. Scalability

The goal of the scalability evaluation is to see how HyperDrive's performance evolves as the infrastructure size increases. Fig. 7 shows the mean scheduling latency for each of the three serverless functions as well as the overall average. Since the prototype implementation is not connected to a real orchestrator and manually performs the vicinity selection with a linear search, we disregard the nominal scheduling latency values and focus on how they evolve with increasing infrastructure sizes.

It is evident that HyperDrive's performance scales linearly with the infrastructure size. The `object-det` function has a steeper incline than the others or the overall average, because needs to check twice as many network SLOs as the others, because it has a data source network SLO. Nevertheless, its increase remains linear.

C. Discussion

As previously seen, HyperDrive is the only scheduler specifically designed for the challenges of the 3D Continuum. HyperDrive excels at choosing between satellite and terrestrial

nodes, depending on what benefits the network SLOs the most. As more nodes are available the quality of its scheduling decisions improves, e.g., the mean network latency between functions drops by 73% in the larger two infrastructures compared to the smaller two infrastructures.

Larger infrastructures yield better scheduling results, but they also increase processing time. Increased processing time, however, does not offset the benefits of optimized scheduling, because transfer and processing times of EO data are orders of magnitude greater than the scheduling duration. Additionally, scheduling on LEO satellites typically does not require high scheduling throughput, due to the type of applications that are expected to be deployed, e.g., federated learning in space or at the Edge [59, 60], advanced automotive use cases [61], monitoring applications [62], or disaster relief [26].

Finding multiple shortest paths through a large network graph is the biggest concern to the performance of HyperDrive. While HyperDrive scales linearly with the infrastructure size, the path finding time can be reduced by using a hypergraph to reduce the number of links and by computing paths between regions instead of single nodes. Additionally, the paths can be periodically precomputed and cached by the orchestrator. This will be addressed by our future work.

Currently we assume the absence of congestion on the network routes, but as satellite usage increases, this will be considered in future work. Additionally, a dense constellation can provide multiple routes [35] between two nodes and prioritization can be employed for disaster response applications.

We evaluated HyperDrive in simulations. However, its scheduling algorithms can be transitioned to a physical system. To this end they must be connected to a real-world orchestrator, which supplies metadata about real satellites (as well as Edge and Cloud nodes) and which can deploy functions on these nodes.

VII. Related Work

We now discuss other work that is related to ours and compare HyperDrive to it.

A. Edge Cloud Continuum & Orbital Edge Computing

Several research studies [6, 7, 63, 64, 65] have proposed a paradigm known as the Edge-Cloud continuum (ECC). This paradigm involves integrating computing resources in different layers, composed of Edge devices such as sensors and wearables that produce data processed by low-resource Edge nodes close that are close to the Edge devices and high-resource Cloud servers. ECC aims to enable seamless integration between all the layers, allowing for efficient task distribution and improved application performance. By utilizing the advantages of both Edge and Cloud resources, ECC enables heterogeneous environments to adjust to computational needs and connectivity conditions. In this paper, HyperDrive proposes to expand the ECC to orbit by seamlessly integrating satellites as Edge nodes, thus creating a 3D Edge-Cloud-Space Continuum.

Lately, the extensive effort to expand on-orbit capability lead further research to explore the implementation of core networks in space, offering several benefits such as enhancing mobile coverage in remote areas, facilitating direct device-satellite connections, and satellite computing [30, 66, 67, 68].

LEO satellites, like terrestrial Edge nodes, have limited computing capacity and like Edge nodes, satellites can be near data sources, such as Earth observation satellites. Therefore, the increase in LEO satellites in orbit allows data to be processed directly in orbit, near the data source, enabling Orbital Edge Computing (OEC) [69, 70]. Research [71, 72, 73, 74, 75] enables federated learning by leveraging their distributed localized data processing capabilities, enhancing real-time data analysis and decision-making in space applications.

The Tiansuan [41, 76] constellation leverages a cloud-native design to enhance onboard services, resources, and the development and management of satellite equipment. Tiansuan's cloud-native approach provides advantages in application deployment, scalability, and cost-effectiveness compared to traditional satellite designs, allowing for seamless integration of computing, and networking. Tisuan's platform is composed of six different layers: Physical, Virtual Resource, Operating System, Container Service, Collaborative Orchestration, and Function Application. MobileViT [77] propose a three-layer architecture to enable Satellite Internet of Things for Smart Agriculture. The infrastructure layer is composed of IoT devices such as sensors, drones and satellites. The capacity layer contains computing communication and caching while the application layer represents the different use cases such as Smart Agriculture, Smart Grid and Smart Port.

B. Space-as-a-Service

Research identifies emerging services in space [78, 79] such as *Constellation-as-a-Service*, *Satellite-as-a-Service* and *Payload-as-a-Service*.

a) Constellation-as-a-Service: Mission MP42 [80] by NanoAvionics and Satellogic [81] aims to offer a satellite constellation service to IoT/M2M operators. Constellation-as-a-service allows businesses to deploy and manage their satellite network without launching their own spacecraft. This business model promises customized services such as dedicated satellites, dedicated rocket launches, in-country operation centers, access to a global ground station network, and dedicated platforms, including a private Cloud for image cataloging, processing, and storage.

b) Satellite-as-a-Service: It proposes a shared multi-tenant satellite concept [82, 83, 84]. The shared-access model allows multiple missions to be hosted on a single satellite, enabling users to share platform and payload capabilities. The Satellite as a Service model includes ground segment validation using continuous integration and hardware simulators to ensure the reliability and safety of user-uploaded software. This approach uses existing satellite infrastructure and modern software tools such as CI/CD to create a flexible and cost-effective platform for space technology development.

c) Payload-as-a-Service: It emerged after a shift from analog to digital satellite payload, that enabled satellites to serve multiple clients. Digital payload made it possible to customize satellite computing for specific purposes such as machine learning and earth observation [85]. Consequently, it enabled an alternative to expensive space infrastructure - *Payload-as-a-Service*. In this business model, a commercial operator owns and manages the satellite system, providing data (payload) to customers on demand. The service providers manage the satellite bus, integration, launch, and operations. On the other hand, clients access the data and may even operate the payload by starting/stopping data collection and monitoring. [79, 86, 87].

All of these approaches focus mostly on satellites only, with very little or no involvement of terrestrial compute nodes. HyperDrive, proposes an unified computing continuum that spans seamlessly across Edge, Cloud, and Space nodes. As such HyperDrive goes further than the aforementioned concepts. But the HyperDrive scheduler can also complement the Constellation-as-a-Service and Satellite-as-a-Service, because both allow customers to run their own workloads on satellites and, thus, require a scheduling mechanism.

C. Satellite Edge Task Scheduling & Offloading

In [88], an efficient framework is proposed for offloading inference tasks by partitioning Deep Neural Network (DNN) models into multiple satellites, including one high Earth orbit (HEO) satellite and multiple LEO satellites. The approach divides inference tasks, with the task owner executing the initial portion of the DNN and offloading the remaining portion to other satellites. FedLEO [74] proposes a distributed scheduling mechanism for LEO satellite constellations to overcome bandwidth limitations and intermittent connectivity. FedLEO leverages Satellite Edge Computing (SEC) to improve training efficiency by adding horizontal communication pathways among satellites and optimally scheduling interactions with ground stations. Unlike HyperDrive, FedLEO and task offloading approach considers data processing only satellites SEC, thus excluding Edge nodes in the ground for task placement.

An Orbital Edge (OE) [14] platform leverages ISL for satellite processing, reducing latency and leveraging distributed computational capabilities. It offloads computing tasks from ground nodes to a single satellite and its neighboring satellites. While the OE platform relies on satellite-ground communication links, which may cause data transfer delays, HyperDrive addresses each computing layer's challenges separately to create a unified Edge Cloud and Space Continuum platform.

VIII. Conclusion

We presented HyperDrive, a novel Serverless platform that is specifically designed to enable a seamless execution of Serverless workflows across the 3D Continuum. We discussed the unique challenges of the 3D Continuum, such as the short availability windows of the fast moving LEO satellites, solar power supply, and the possibility of overheating while facing the sun. The HyperDrive scheduler enables the optimized placement of Serverless functions in the 3D Continuum by considering network SLOs, workflow and data source dependencies, and thermal conditions of satellites. HyperDrive is, to the best of our knowledge, the first Serverless scheduler for the 3D Continuum. We evaluate it against three theoretical baseline schedulers by scheduling a wildfire disaster response workflow with strict network SLOs and EO satellite data dependencies. HyperDrive achieves 71% lower E2E network latency than the best baseline and shows linear performance scalability with the infrastructure size.

As future work we plan to continue our realization of the HyperDrive Serverless platform for the 3D Continuum. An important goal is to further to improve the coordination of function execution and satellite orbits, by placing functions on satellites that will be in the ideal position for a low-latency handoff to the next node when the function completes. We also intend to reduce scheduling complexity for large infrastructure sizes by using hypergraphs for inter-node path computations. Additionally, we envision a lightweight framework for Serverless-native development of the next generation EO applications for the 3D Continuum.

Acknowledgment

This work is partially funded by the Austrian Research Promotion Agency (FFG) under the project RapidREC (Project No. 903884).

This research received funding from the EU's Horizon Europe Research and Innovation Program under Grant Agreement No. 101070186. EU website for TEADAL: https://teadal.eu.

References

[1] Orbit.ing-now.com, "Low earth orbit," 2024, accessed: 2024-06-30. [Online]. Available: https://orbit.ing-now.com/low-earth-orbit/

[2] D. Bhattacherjee and A. Singla, "Network topology design at 27,000 km/hour," in *Proceedings of the 15th International Conference on Emerging Networking Experiments And Technologies*, ser. CoNEXT '19. New York, NY, USA: ACM, 2019, p. 341–354.

[3] D. Shepardson, "Fcc chair wants more competition to spacex's starlink unit," 2024. [Online]. Available: https://www.reuters.com/technology/space/fcc-chair-wants-more-competition-spacexs-starlink-unit-2024-09-11/

[4] C. Henry, "Fcc oks lower orbit for some starlink satellites," *Space News*, 2019. [Online]. Available: https://spacenews.com/fcc-oks-lower-orbit-for-some-starlink-satellites/

[5] Federal Communications Commission, "Kuiper systems, llc – application for authority to deploy and operate a ka-band non-geostationary satellite orbit system – order and authorization." [Online]. Available: https://docs.fcc.gov/public/attachments/FCC-20-102A1.pdf

[6] S. Nastic, A. Morichetta, T. Pusztai, S. Dustdar, X. Ding, D. Vij, and Y. Xiong, "Sloc: Service level objectives for next generation cloud computing," *IEEE Internet Computing*, vol. 24, no. 3, pp. 39–50, 2020.

[7] S. Nastic, P. Raith, A. Furutanpey, T. Pusztai, and S. Dustdar, "A serverless computing fabric for edge & cloud," in *2022 IEEE 4th International Conference on Cognitive Machine Intelligence (CogMI)*, 2022, pp. 1–12.

[8] A. Brogi, S. Forti, C. Guerrero, and I. Lera, "How to place your apps in the fog: State of the art and open challenges," *Software: Practice and Experience*, vol. 50, no. 5, 2020.

[9] M. Ghobaei-Arani and M. Ghorbian, *Scheduling Mechanisms in Serverless Computing*. Cham: Springer International Publishing, 2023, pp. 243–273.

[10] The Kubernetes Authors, "Scheduling framework — kubernetes," 2024-02-19. [Online]. Available: https://kubernetes.io/docs/concepts/scheduling-eviction/scheduling-framework/

[11] T. Rausch, A. Rashed, and S. Dustdar, "Optimized container scheduling for data-intensive serverless edge computing," *Future Generation Computer Systems*, vol. 114, pp. 259–271, 2021.

[12] D. Balla, C. Simon, and M. Maliosz, "Adaptive scaling of kubernetes pods," in *NOMS 2020 - 2020 IEEE/IFIP Network Operations and Management Symposium*. IEEE, 20.04.2020 - 24.04.2020, pp. 1–5.

[13] Q. Tang, R. Xie, Z. Fang, T. Huang, T. Chen, R. Zhang, and F. R. Yu, "Joint service deployment and task scheduling for satellite edge computing: A two-timescale hierarchical approach," *IEEE Journal on Selected Areas in Communications*, vol. 42, no. 5, pp. 1063–1079, 2024.

[14] P. Cassará, A. Gotta, M. Marchese, and F. Patrone, "Orbital edge offloading on mega-leo satellite constellations for equal access to computing," *IEEE Communications Magazine*, vol. 60, no. 4, pp. 32–36, 2022.

[15] X. Ma, M. Xu, Q. Li, Y. Li, A. Zhou, and S. Wang, *Visions of Edge Computing in 6G*. Springer Nature Singapore, 2024, pp. 179–202.

[16] Y. Cheng and Z. Zhou, "Autonomous resource scheduling for real-time and stream processing," in *2018 IEEE SmartWorld, Ubiquitous Intelligence & Computing, Advanced & Trusted Computing, Scalable Computing & Communications, Cloud & Big Data Computing, Internet of People and Smart City Innovation (SmartWorld/SCALCOM/UIC/ATC/CBDCom/IOP/SCI)*, 2018, pp. 1181–1184.

[17] O. Skarlat, M. Nardelli, S. Schulte, and S. Dustdar, "Towards qos-aware fog service placement," in *2017 IEEE 1st International Conference on Fog and Edge Computing (ICFEC)*, 2017, pp. 89–96.

[18] T. Pusztai, S. Nastic, A. Morichetta, V. C. Pujol, P. Raith, S. Dustdar, D. Vij, Y. Xiong, and Z. Zhang, "Polaris scheduler: Slo- and topology-aware microservices scheduling at the edge," in *2022 IEEE/ACM 15th International Conference on Utility and Cloud Computing (UCC)*, 2022, pp. 61–70.

[19] T. Pusztai, S. Nastic, P. Raith, S. Dustdar, D. Vij, and Y. Xiong, "Vela: A 3-phase distributed scheduler for the edge-cloud continuum," in *2023 IEEE International Conference on Cloud Engineering (IC2E)*. Los Alamitos, CA, USA: IEEE Computer Society, sep 2023, pp. 161–172.

[20] Kernel.org, "Energy-aware scheduling," 2024, accessed: 2024-06-30. [Online]. Available: https://docs.kernel.org/scheduler/sched-energy.html

[21] C. Yao, W. Liu, W. Tang, and S. Hu, "Eais: Energy-aware adaptive scheduling for cnn inference on high-performance gpus," *Future Generation Computer Systems*, vol. 130, pp. 253–268, 2022.

[22] X. Zhang, J. Liu, R. Zhang, Y. Huang, J. Tong, N. Xin, L. Liu, and Z. Xiong, "Energy-efficient computation peer offloading in satellite edge computing networks," *IEEE Transactions on Mobile Computing*, vol. 23, no. 4, pp. 3077–3091, 2024.

[23] C. Li, Y. Zhang, R. Xie, X. Hao, and T. Huang, "Integrating edge computing into low earth orbit satellite networks: Architecture and prototype," *IEEE Access*, vol. 9, pp. 39 126–39 137, 2021.

[24] N.-N. Dao, Q.-V. Pham, D.-T. Do, and S. Dustdar, "The sky is the edge—toward mobile coverage from the sky," *IEEE Internet Computing*, vol. 25, no. 2, pp. 101–108, 2021.

[25] G. Mateo-Garcia, J. Veitch-Michaelis, C. Purcell, N. Longepe, S. Reid, A. Anlind, F. Bruhn, J. Parr, and P. Mathieu, "In-orbit demonstration of a re-trainable machine learning payload for processing optical imagery," *Scientific Reports*, vol. 13, no. 1, p. 10391, Jun 2023.

[26] C. van Arsdale, "A breakthrough in wildfire detection: How a new constellation of satellites can detect smaller wildfires earlier," 2024. [Online]. Available: https://blog.google/outreach-initiatives/sustainability/google-ai-wildfire-detection/

[27] European Space Agency, "European data relay satellite system (edrs) overview," 2024. [Online]. Available: https://connectivity.esa.int/european-data-relay-satellite-system-edrs-overview

[28] ——, "Sentinel-2 operations." [Online]. Available: https://www.esa.int/Enabling_Support/Operations/Sentinel-2_operations

[29] Airbus, "Airbus built sentinel-2c satellite successfully launched," 2024. [Online]. Available: https://www.airbus.com/en/newsroom/press-releases/2024-09-airbus-built-sentinel-2c-satellite-successfully-launched

[30] D. Vasisht, J. Shenoy, and R. Chandra, "L2d2: low latency distributed downlink for leo satellites," in *Proceedings of the 2021 ACM SIGCOMM 2021 Conference*, ser. SIGCOMM '21. New York, NY, USA: ACM, 2021, p. 151–164.

[31] European Space Agency, "Sentinel online - glossary," 2024. [Online]. Available: https://sentinels.copernicus.eu/web/sentinel/technical-guide/sentinel-2-msi/glossary

[32] ——, "European space agency-funded projects reach new performance level in groundwork for optical leo to geo data relays," 2024. [Online]. Available: https://connectivity.esa.int/news/european-space-agencyfunded-projects-reach-new-performance-level-groundwork-optical-leo-geo-data-relays

[33] X. Chen, B. Hopkins, H. Wang, L. O'Neill, F. Afghah, A. Razi, P. Fulé, J. Coen, E. Rowell, and A. Watts, "Wildland fire detection and monitoring using a drone-collected rgb/ir image dataset," *IEEE Access*, vol. 10, pp. 121 301–121 317, 2022.

[34] D. Bhattacherjee, W. Aqeel, I. N. Bozkurt, A. Aguirre, B. Chandrasekaran, P. B. Godfrey, G. Laughlin, B. Maggs, and A. Singla, "Gearing up for the 21st century space race," in *Proceedings of the 17th ACM Workshop on Hot Topics in Networks*, ser. HotNets '18. New York, NY, USA: ACM, 2018, p. 113–119.

[35] M. Handley, "Delay is not an option: Low latency routing in space," in *Proceedings of the 17th ACM Workshop on Hot Topics in Networks*, ser. HotNets '18. New York, NY, USA: ACM, 2018.

[36] Q. Chen, G. Giambene, L. Yang, C. Fan, and X. Chen, "Analysis of inter-satellite link paths for leo mega-constellation networks," *IEEE Transactions on Vehicular Technology*, vol. 70, no. 3, pp. 2743–2755, 2021.

[37] N. Mohan, A. E. Ferguson, H. Cech, R. Bose, P. R. Renatin, M. K. Marina, and J. Ott, "A multifaceted look at starlink performance," in *Proceedings of the ACM on Web Conference 2024*, ser. WWW '24. New York, NY, USA: ACM, 2024, pp. 2723–2734.

[38] F. Davoli, C. Kourogiorgas, M. Marchese, A. Panagopoulos, and F. Patrone, "Small satellites and cubesats: Survey of structures, architectures, and protocols," *International Journal of Satellite Communications and Networking*, vol. 37, 09 2018.

[39] National Aeronautics and Space Administration, "State-of-the-art small spacecraft technology." [Online]. Available: https://www.nasa.gov/wp-content/uploads/2024/03/soa-2023.pdf

[40] M. M. Finckenor and K. K. de Groh, "A researcher's guide to: Space environmental effects." [Online]. Available: https://www.nasa.gov/science-research/for-researchers/a-researchers-guide-to-space-environmental-effects/

[41] C. Wang, Y. Zhang, Q. Li, A. Zhou, and S. Wang, "Satellite computing: A case study of cloud-native satellites," in *2023 IEEE International Conference on Edge Computing and Communications (EDGE)*. Los Alamitos, CA, USA: IEEE Computer Society, 2023, pp. 262–270.

[42] S. Wang and Q. Li, "Satellite computing: Vision and challenges," *IEEE Internet of Things Journal*, vol. 10, no. 24, pp. 22 514–22 529, 2023.

[43] T. Pfandzelter, J. Hasenburg, and D. Bermbach, "Towards a computing platform for the leo edge," in *Proceedings of the 4th International Workshop on Edge Systems, Analytics and Networking*, ser. EdgeSys '21. New York, NY, USA: ACM, 2021, p. 43–48.

[44] T. Pfandzelter, "Serverless abstractions for edge computing in large low-earth orbit satellite networks," in *Proceedings of the 24th International Middleware Conference: Demos, Posters and Doctoral Symposium*, ser. Middleware '23. New York, NY, USA: ACM, 2023, p. 3–6.

[45] T. Pusztai, A. Morichetta, V. C. Pujol, S. Dustdar, S. Nastic, X. Ding, D. Vij, and Y. Xiong, "A novel middleware for efficiently implementing complex cloud-native slos," in *2021 IEEE 14th International Conference on Cloud Computing (CLOUD)*, 2021, pp. 410–420.

[46] ——, "Slo script: A novel language for implementing complex cloud-native elasticity-driven slos," in *2021 IEEE International Conference on Web Services (ICWS)*, 2021, pp. 21–31.

[47] T. Pfandzelter and D. Bermbach, "Qos-aware resource placement for leo satellite edge computing," in *2022 IEEE 6th International Conference on Fog and Edge Computing (ICFEC)*, 2022, pp. 66–72.

[48] S. Nastic, "Self-provisioning infrastructures for the next generation serverless computing," *SN Computer Science*, vol. 5, no. 6, pp. 678 – 693, 2024.

[49] K. Li and S. Nastic, "Attentionfunc: Balancing faas compute across edge-cloud continuum with reinforcement learning," in *The 13th International Conference on the Internet of Things (IoT 2023)*, 2023.

[50] C. Delimitrou, D. Sanchez, and C. Kozyrakis, "Tarcil: Reconciling scheduling speed and quality in large shared clusters," in *Proceedings of the Sixth ACM Symposium on Cloud Computing*, ser. SoCC '15. New York, NY, USA: ACM, 2015, pp. 97–110.

[51] C. Marcelino and S. Nastic, "Cwasi: A webassembly runtime shim for inter-function communication in the serverless edge-cloud continuum," in *Proceedings of the Eighth ACM/IEEE Symposium on Edge Computing*, ser. SEC '23. New York, NY, USA: ACM, 2024, p. 158–170.

[52] S. Shillaker and P. Pietzuch, "Faasm: Lightweight isolation for efficient stateful serverless computing," in *2020 USENIX Annual Technical Conference (USENIX ATC 20)*. USENIX Association, 2020.

[53] P. Raith, S. Nastic, and S. Dustdar, "Serverless edge computing—where we are and what lies ahead," *IEEE Internet Computing*, vol. 27, no. 3, pp. 50–64, 2023.

[54] C. Marcelino, J. Shahhoud, and S. Nastic, "Goldfish: Serverless actors with short-term memory state for the edge-cloud continuum," in *Proceedings of the 14th International Conference on the Internet of Things*, ser. IoT '24. New York, NY, USA: ACM, 2024.

[55] V. Goronjic and S. Nastic, "Miso: A crdt-based middleware for stateful objects in the serverless edge-cloud continuum," in *TThe 12th IEEE International Conference on Cloud Engineering (IC2E 2024)*, 2024.

[56] S. Sehic, F. Li, S. Nastic, and S. Dustdar, "A programming model for context-aware applications in large-scale pervasive systems," in *Proceedings of the IEEE 8th International Conference on Wireless and Mobile Computing, Networking and Communications (WiMob 2012)*. IEEE Computer Society, 2012, pp. 142–149.

[57] S. Nastic, G. Copil, H.-L. Truong, and S. Dustdar, "Governing elastic iot cloud systems under uncertainty," in *Proceedings of the IEEE 7th International Conference on Cloud Computing Technology and Science (CloudCom 2015)*. IEEE Computer Society, 2015, pp. 131–138.

[58] Z. Lai, H. Li, Y. Deng, Q. Wu, J. Liu, Y. Li, J. Li, L. Liu, W. Liu, and J. Wu, "StarryNet: Empowering researchers to evaluate futuristic integrated space and terrestrial networks," in *20th USENIX Symposium on Networked Systems Design and Implementation (NSDI 23)*. Boston, MA: USENIX Association, 2023, pp. 1309–1324.

[59] B. Matthiesen, N. Razmi, I. Leyva-Mayorga, A. Dekorsy, and P. Popovski, "Federated learning in satellite constellations," *IEEE Network*, vol. 38, no. 2, pp. 232–239, 2024.

[60] R. Gajanin, A. Danilenka, A. Morichetta, and S. Nastic, "Towards adaptive asynchronous federated learning for human activity recognition," in *Proceedings of the 14th International Conference on the Internet of Things (IoT 2024)*. New York, NY, USA: ACM, 2024.

[61] H. W. Zaglauer, "Intelligent satellite payloads as enablers for 6g," ETSI Conference on Non-Terrestrial Networks, A Native Component of 6G, 2024. [Online]. Available: https://docbox.etsi.org/Workshop/2024/04_ETSI_6G_NTN/SESSION%2006/S6_03_Zaglauer.pdf

[62] Z. Zhai, L. Zeng, T. Ouyang, S. Yu, Q. Huang, and X. Chen, "Seco: Multi-satellite edge computing enabled wide-area and real-time earth observation missions," in *IEEE INFOCOM 2024 - IEEE Conference on Computer Communications*, 2024, pp. 2548–2557.

[63] S. Dustdar and I. Murturi, *Towards IoT Processes on the Edge*. Cham: Springer International Publishing, 2021, pp. 167–178.

[64] V. Pujol, P. Donta, A. Morichetta, I. Murturi, and S. Dustdar, "Edge intelligence—research opportunities for distributed computing continuum systems," *IEEE Internet Computing*, vol. 27, no. 04, pp. 53–74, jul 2023.

[65] V. Casamayor Pujol, P. K. Donta, A. Morichetta, I. Murturi, and S. Dustdar, "Distributed computing continuum systems – opportunities and research challenges," in *Service-Oriented Computing – ICSOC 2022 Workshops*, J. Troya, R. Mirandola, E. Navarro, A. Delgado, S. Segura, G. Ortiz, C. Pautasso, C. Zirpins, P. Fernández, and A. Ruiz-Cortés, Eds. Cham: Springer Nature Switzerland, 2023, pp. 405–407.

[66] D. Zhou, M. Sheng, J. Li, and Z. Han, "Aerospace integrated networks innovation for empowering 6g: A survey and future challenges," *IEEE Communications Surveys & Tutorials*, vol. 25, no. 2, pp. 975–1019, 2023.

[67] M. M. Azari, S. Solanki, S. Chatzinotas, O. Kodheli, H. Sallouha, A. Colpaert, J. F. Mendoza Montoya, S. Pollin, A. Haqiqatnejad, A. Mostaani, E. Lagunas, and B. Ottersten, "Evolution of non-terrestrial networks from 5g to 6g: A survey," *IEEE Communications Surveys & Tutorials*, vol. 24, no. 4, pp. 2633–2672, 2022.

[68] R. Xing, X. Ma, A. Zhou, S. Dustdar, and S. Wang, "From earth to space: A first deployment of 5g core network on satellite," *China Communications*, vol. 20, no. 4, pp. 315–325, 2023.

[69] B. Denby and B. Lucia, "Orbital edge computing: Nanosatellite constellations as a new class of computer system," in *Proceedings of the Twenty-Fifth International Conference on Architectural Support for Programming Languages and Operating Systems*, ser. ASPLOS '20. New York, NY, USA: ACM, 2020, p. 939–954.

[70] D. Bhattacherjee, S. Kassing, M. Licciardello, and A. Singla, "In-orbit computing: An outlandish thought experiment?" in *Proceedings of the 19th ACM Workshop on Hot Topics in Networks*, ser. HotNets '20. New York, NY, USA: ACM, 2020, p. 197–204.

[71] M. R. Jabbarpour, B. Javadi, P. Leong, R. N. Calheiros, D. Boland, and C. Butler, "Performance analysis of federated learning in orbital edge computing," in *Proceedings of the IEEE/ACM 16th International Conference on Utility and Cloud Computing*, ser. UCC '23. New York, NY, USA: ACM, 2024.

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

[73] 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," in *2023 IEEE 34th Annual International Symposium on Personal, Indoor and Mobile Radio Communications (PIMRC)*, 2023.

[74] M. Elmahallawy and T. Luo, "Optimizing federated learning in leo satellite constellations via intra-plane model propagation and sink satellite scheduling," in *ICC 2023 - IEEE International Conference on Communications*, 2023, pp. 3444–3449.

[75] A. Furutanpey, Q. Zhang, P. Raith, T. Pfandzelter, S. Wang, and S. Dustdar, "Fool: Addressing the downlink bottleneck in satellite computing with neural feature compression," 2024. [Online]. Available: https://arxiv.org/abs/2403.16677

[76] S. Wang and Q. Li, "Satellite computing: Vision and challenges," *IEEE Internet of Things Journal*, vol. 10, no. 24, pp. 22 514–22 529, 2023.

[77] J. Liu, W. Jiang, H. Han, M. He, and W. Gu, "Satellite internet of things for smart agriculture applications: A case study of computer vision," in *2023 20th Annual IEEE International Conference on Sensing, Communication, and Networking (SECON)*, 2023, pp. 66–71.

[78] E. Kulu, "Satellite constellations - 2021 industry survey and trends," in *35th Annual Small Satellite Conference*, 08 2021.

[79] A. Hein and C. Bruce Rosete, "Space-as-a-service: A framework and taxonomy of -as-a-service concepts for space," in *Proceedings of the International Astronautical Congress 2022*, September 2022.

[80] NanoAvionics, "Startical to test its technology on a nanoavionics-built satellite, paving the way for the first space-based air traffic service constellation," 2024, accessed: 2024-06-16. [Online]. Available: https://nanoavionics.com/news/startical-to-test-its-technology-on-a-nanoavionics-built-satellite-paving-the-way-for-the-first-space-based-air-traffic-service-constellation/

[81] Satellogic, "Constellation-as-a-service," 2024, accessed: 2024-06-16. [Online]. Available: https://satellogic.com/products/constellation-as-a-service/

[82] Exodus Orbitals, "Satellite-as-a-service: A new approach for space industry," 2024, accessed: 2024-06-16. [Online]. Available: https://www.exodusorbitals.com/files/whitepaper.pdf

[83] T. M. Kebedew, V. N. Ha, E. Lagunas, J. Grotz, and S. Chatzinotas, "Qoe-aware cost-minimizing capacity renting for satellite-as-a-service enabled multiple-beam satcom systems," *IEEE Transactions on Communications*, vol. 72, no. 3, pp. 1773–1789, 2024.

[84] W. Zhang, Y. Xue, J. Wu, and X. Xu, "Satellite as a service: a hybrid resource management framework for space-terrestrial integrated networks," in *2020 IEEE 11th International Conference on Software Engineering and Service Science (ICSESS)*, 2020, pp. 171–174.

[85] GMV, "Evolution of payload management systems for communications satellites: New challenges," 2024, accessed: 2024-06-16. [Online]. Available: https://www.gmv.com/en/media/blog/space/evolution-payload-management-systems-communications-satellites-new-challenges

[86] P. Senior, S. Eckersley, V. Irwin, B. Stern, A. Haslehurst, A. Cawthorne, A. da Silva Curiel, and M. Sweeting, "Can we use low cost small satellites to observe space debris missed by ground systems," in *Proceedings of the 8th European Conference on Space Debris, Darmstadt, Germany*, 2021, pp. 20–23.

[87] M. Höyhtyä, S. Boumard, A. Yastrebova, P. Järvensivu, M. Kiviranta, and A. Anttonen, "Sustainable satellite communications in the 6g era: A european view for multilayer systems and space safety," *IEEE Access*, vol. 10, pp. 99 973–100 005, 2022.

[88] J. Guan, Q. Zhang, I. Murturi, P. K. Donta, S. Dustdar, and S. Wang, "Collaborative inference in dnn-based satellite systems with dynamic task streams," 2023. [Online]. Available: https://arxiv.org/abs/2311.06073