Do We Need a Million Satellites in Orbit?

Constellation-as-a-Service with Modular Satellites: Challenges and Opportunities

Demi Lei and Ahmed Saeed
Georgia Institute of Technology, Atlanta, Georgia, USA
wlei36@gatech.edu,asaeed@cc.gatech.edu

Abstract

In this paper, we argue that deploying many mission-specific satellite mega-constellations incurs significant monetary and environmental costs. Instead, we propose launching a single or a small number of mega-constellations equipped with heterogeneous computing, communication, storage, and sensing capabilities, allowing them to offer a broad range of services to customers who no longer need to launch their own satellites. We argue that the hardware technology for building such platforms is already widely accessible. Thus, we highlight the algorithmic and systems challenges that the community needs to address to enable cost-efficient and secure constellation-as-a-service platforms. We also develop a simulator that allows for experimenting with different scheduling algorithms for constellation-as-a-service platforms.

CCS Concepts

• **Computer systems organization → Distributed architectures**; • **Security and privacy → Domain-specific security and privacy architectures**.

Keywords

Constellation as a Service, Satellite Scheduling

ACM Reference Format:

Demi Lei and Ahmed Saeed. 2024. Do We Need a Million Satellites in Orbit?In *The 2nd ACM Workshop on LEO Networking and Communication 2024 (LEO-NET 24), November 18–22, 2024, Washington D.C., DC, USA.* ACM, New York, NY, USA, 6 pages. https://doi.org/10.1145/3697253.3697262

1 Introduction

Mega-constellations, made up of thousands of Low Earth Orbit (LEO) satellites, are enabling innovations in many areas, including communications (e.g., Starlink and OneWeb), cloud computing [26], and earth monitoring (e.g., Planet Labs for earth imaging, Capella Space for SAR-based monitoring and tomorrow.io and PlanetiQ for radar imaging of the atmosphere). The rise of these mega constellations is driven by the decreasing cost of developing and launching satellites. In particular, reusable booster rockets from SpaceX along with low-cost CubeSat technologies create a cost-efficient pipeline for building these mega-constellations [10]. The current approach to operating mega-constellations is launching mission-specific custom-built satellites. The operator of a constellation can then choose to offer services on top of their platform. For example, Starlink and OneWeb offer communication services through their special user terminals. On the other hand, Planet Labs, Capella Space, tomorrow.io, and PlanetiQ sell data collected by their satellites.

With a million satellites planned to be in orbit [15], the downsides of special-purpose satellites have to be considered. Space launches are a source of air pollution as well as light pollution [22, 23, 30, 31]. Further, the growth in the number of satellites increases the risk of accidents caused by space debris [5]. Moreover, a mission-specific constellation is likely to be underutilized. In particular, LEO satellites orbit the Earth once every 90-120 minutes, meaning that a single satellite necessarily spends a significant portion of its time over areas that might not be of significance to its mission. For example, communication satellites will fly over swaths of oceans and deserts, serving no customers. The same applies to imaging satellites flying over areas that are of no interest to any of their customers.

In this paper, we explore the challenges and opportunities associated with constellation-as-a-service (CaaS) offerings [1, 19]. In particular, we argue for the deployment of a small number of mega-constellations made up of satellites with heterogeneous equipment payloads, allowing them to replace several mission-specific mega-constellations. The development of such satellites is enabled by the proliferation of modular satellites (e.g., CubeSats), making it easier to equip individual satellites with a wide range of consumer-grade electronics. Such a platform will maximize the utilization of individual satellites by spatially multiplexing their usage, allowing the same satellite to be used by multiple users as it orbits. Further, it will reduce the need for launching custom-built mission-specific satellites, reducing the downside effects of high-density mega-constellation deployments. Finally, by lowering the barrier to

LEO-NET 24, November 18–22, 2024, Washington D.C., DC, USA

ACM ISBN 979-8-4007-1280-7/24/11
https://doi.org/10.1145/3697253.3697262

entry, such a platform will encourage innovation, much like cloud computing has done for the tech industry at large.

We highlight the different service models that such a platform can offer, ranging from the existing models (e.g., leasing data or end-to-end communication) to a cloud-like model that allows for leasing virtual satellites. Further, we identify the different constraints imposed by shared satellites, highlighting the added challenges in a constellation-as-a-service platform compared to even LEO-based cloud computing platforms [26]. We present a straw man architecture of the platform to highlight the systems and algorithmic challenges that we need to overcome to make constellation-as-a-service platforms a viable product. Finally, we develop a simulator that allows researcher to experiment with different resource scheduling algorithms for constellation-as-a-service platforms following our proposed architecture and service models. Our objective, in this paper, is to outline the motivation, scope, and tools for tackling the fundamental algorithmic and systems challenges in building constellation-as-service platforms.

2 Background

LEO Satellites Primer. A LEO satellite constellation comprises tens to thousands of satellites orbiting the Earth at an altitude in the range of 200-1600 km [11]. The projected success of the current reincarnation of LEO satellite constellation deployment is driven by the reduced cost of building and launching such satellites [10, 12, 24]. Several private companies are building and deploying new satellite constellations, enabling new applications through their new sources of data or global communication platforms. For example, Starlink and OneWeb offer broadband connectivity. Capella Space offers radar-based imaging. tomorrow.io offers services for monitoring weather phenomena. Private deployments are growing in tandem with growing government deployments. Indeed, current filings with the ITU show plans for having nearly a million satellites in orbit, indicating a planned increase of over 100× in the number of satellites in orbit [15].

Downsides of Mega Constellations. Large-scale constellation deployment comes at the expense of exacerbating the space debris problem. Space debris refers to artificial objects in space that no longer serve a functional purpose. Space debris increases the risk of satellites falling to Earth. With the increased deployment, the number of incidents of satellite failure is also bound to increase, increasing the chances of hazardous fragments surviving reentries. The US FAA predicts that by 2035, "the total number of hazardous fragments surviving reentries each year is expected to reach 28,000" [5]. Further, space debris increases the chance of collision while launching new satellites in orbit [9, 27].

The growing density of LEO satellite deployments is impacting ground-based and space-borne telescopes for space observation [23]. Satellites reflect and scatter sunlight, making them visible to the naked eye at dark sites, complicating ground-based astronomy. Furthermore, rocket launches can ruin space telescope exposures, and the increase in satellite deployments will only exacerbate the problem. A single launch can deliver around 50 satellites to orbit, requiring at tens of thousands of launches to deploy a million satellites, leading to deleterious effects on the environment [22, 30, 31]. Finally, constellations have to compete for the scarce spectrum resource, which provides a hard limit on the aggregate communication capacity between Earth and LEO constellations [13, 18].

Modular Satellites Primer. There has been an explosion in the interest in developing modular satellites. A prime example is the growing popularity of CubeSats (i.e., satellites with a standardized size and form factor starting with a 1U of 10cm cube and a weight of 2kg) has been coupled with a growth in their commercially available off-the-shelf (COTS) components.[1] Further, there have been several attempts to build modular satellites where the sensing payload is decoupled from the power bus, which allows for building extensible and modular satellites. For example, Slingshot by the Aerospace Corporation relies on an Ethernet switch to create a modular system that facilitates the addition of modules to a CubeSat.[2] Novawurks has its own proposal for building modular satellites as well.[3] The benefit of this technology is enabling the deployment of multi-purpose satellites to compose a heterogeneous mega-constellation that can support multiple operations simultaneously. In our view, such technology enables a transformation in satellite technology akin to moving from monolithic vertically-integrated proprietary mainframe architectures to the "horizontal marketplace" enabled by microprocessors and open-source operating systems. In particular, we envision that modular satellites will allow the deployment of new hardware and new software that interact with each other through standard interfaces.

3 Overview of Service Model

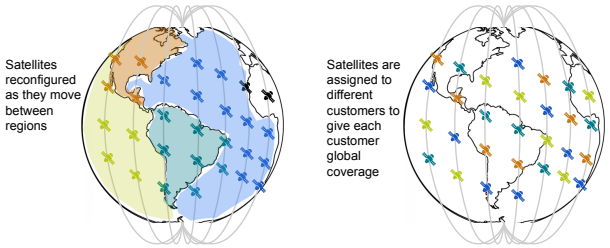
The users of a constellation-as-a-service platform leverage, potentially shared, resources to perform their missions. In this section, we provide an overview of the different types of resources available in a satellite. Then, we explore the different service models that can be offered and how demand can be expressed for those services.

Resources. A modular satellite can be equipped with a wide range of devices ranging for compute capacity including general purpose CPUs, GPUs, and FPGAs [25, 29]. Further, satellites are equipped with storage and communication devices,

[1] Example: https://www.cubesatshop.com/

[2] https://aerospace.org/article/slingshot-platform-fast-tracks-space-systems-using-modularity-and-open-standards

[3] https://www.novawurks.com/



(a) Regional Coverage **(b) Constellation Slice**

Figure 1: Demand coverage models, assuming that each satellite has only a single customer.

including separate equipment for inter-satellite communication and communication with ground stations. Finally, a satellite can also be equipped with a wide range of sensing devices (e.g., RGB sensors, hyperspectral sensors, and radar). A CaaS user can require access to any combination of these resources. There are current several efforts exploring special cases of CaaS that primarily focus on computing resources [8, 17, 26, 32] and communication [13, 32]. We envision a general purpose CaaS platform.

Services. There are different modes of sharing offered by a CaaS platform, depending on the level of abstraction offered to the user. Like in cloud computing settings, different customers will have different requirements and expertise to help determine how much control they need over their constellation. We envision two levels of abstractions inspired by the Software-as-a-Service (SaaS) and Infrastructure-as-a-Service (IaaS) models supported by modern cloud computing platforms.

1) Service subscription. Like with SaaS, at this level of abstraction, customers rely on the platform to manage all hardware and software running on its satellites to provide the services required by the satellite. This is the model followed by current mission-specific constellations. For example, Starlink manages its satellites to provide communication services to its customers and Planet Labs manages its satellites to provide Earth imaging data to its customers. Such model can be trivially implemented by CaaS platforms.

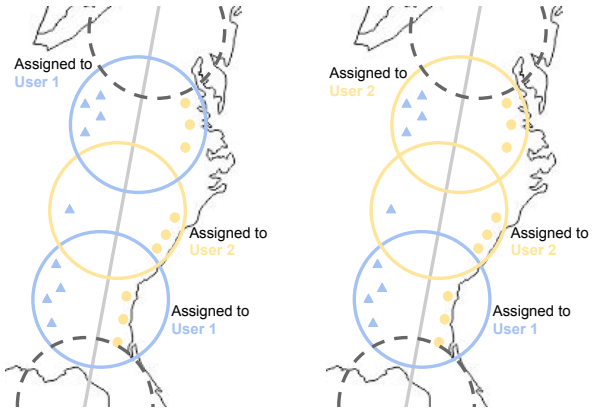
2) Virtual satellites. Like with IaaS, customers are offered a virtual constellation with compute, communication, sensing, and storage capabilities. Customers have the flexibility to implement their own application logic, deciding when their data should be communicated to terrestrial systems. Further, they are given the ability to orchestrate the usage of their satellites as needed by their application. Like with IaaS platforms, in this model, customers have to attach the compute, sensing, inter-satellite communication, and storage capabilities they expect in their individual satellites as well as the aggregate communication capacity they need to ground stations. A CaaS platform can co-locate multiple customers with demand for virtual satellites and service subscriptions on the same hardware.

Demand. To further help hide the complexity of managing a virtual constellation, a constellation-as-a-service platform can allow its customers to express their demand in terms of its intended geographic region of operation. There are two modes possible modes: *1) Regional coverage.* Under this model, the platform ensures that the virtual constellation created for the user always covers the intended area. The platform is responsible for migrating the user's code and data to ensure continuous coverage of the specific region. *2) Constellation slice.* A user can require a global constellation with some density. Figure 1 illustrates the difference between the two modes.

4 Platform Constraints

The deployment of the first CaaS platform might be several years away. However, we believe that the challenges to its realization are mostly on the software side: both architectural and algorithmic. Our belief stems from the fact that building the hardware for such programmable satellites, at a relatively low cost, is not only feasible but a routine operation for modern CubeSat developers. With the decreasing overhead of deployment and the growing role of satellite-based platforms (e.g., in communication, environment protection and monitoring, mining, etc), we argue that *the* bottleneck for the development and deployment of such a platform is ensuring its efficiency (i.e., making it a cost-efficient alternative to mission-specific constellations) and security (i.e., ensuring the same level of privacy and security a customer gets from operating their own constellation). Ensuring the efficiency and security of a shared satellite platform faces similar challenges to operating shared computing platforms (e.g., clouds and super computers) and mobility platforms (e.g., ride sharing platforms and autonomous drones platforms). We believe the experience the systems community has built, operating those platforms, will prove useful for operating CaaS platforms. However, satellite platforms face the following four unique challenges.

Sensor configuration. The output of remote sensing equipment depends on its configuration. For example, an operator might change the attitude of the satellite to change or expand the coverage area of a satellite sensor (e.g., hyperspectral or RBG cameras). Two customers might have different configurations planned for the same sensor, requiring the allocation of at least two different sensors in the same satellite if the configuration is limited to the sensor (e.g., exposure time of an RBG sensor) or two different satellites if the configuration is satellite-wide (e.g., satellite attitude). Configurable sensors opens the door to many possible side-channel attacks. For example, a malicious user can infer the operation of other customers by deploying lightweight tasks on a large number of satellites to estimate the configuration of different satellites (e.g., attitude, location, power draw, etc). Using that information, the malicious user can estimate the areas of operation, specific sensors, and even the configurations used by other customers (e.g.,



(a) High throughput **(b) Max-min fair**

	User 1 ▲	User 2 ●	Total
Maximize throughput	8	3	11
Max-Min	4	6	10

(c) Summary of allocations.

Figure 2: Allocation of three satellites to two users on the east coast of the US based on two allocation policies

from power draw and attitude). Thus, scheduling decisions made by a Constellation-as-a-Service platform will have to ensure the obfuscation of any customer-specific configuration. Further, the API provided by the OS to access sensory data should also ensure that sensor configuration cannot be detected. For instance, the API should correct distortions due to non-zero zenith angles in visual data. Alternatively, the API could support only acquiring cropped versions of any images collected by a satellite, hiding the overall area covered by the sensor. Such obfuscation can reduce the accuracy of any inference made by malicious users about the activity of other users.

Power management. Satellites typically have power constraints stemming from limitations on the power they can harvest from the sun. These limitations depend on their altitude, solar panel size, and battery capacity [14]. It's typical for satellites to have different modes of operation based on available power, with each mode determining the set of active equipment (e.g., sensing, computing, and communication). Thus, the capacity of a satellite to handle the demands of multiple customers will also depend on the availability of power to handle their combined load. A platform should take into account compute, networking, and storage capacity, like typical cloud platforms. However, it should also take into account power constraints, ensuring that a satellite has enough power to support the needs of its tenants. Further, continuous monitoring of power levels will be required to enable live-migration of services from satellites low on power, to others that are sufficiently charged.

Fairness. A shared platform needs to ensure fairness between its customers in events of contention. Although some strict prioritization between customers can exist, a constellation-as-a-service platform should ensure that no user starves or achieves considerably lower performance when competing for resources with another user of similar priority. Figure 2 shows two allocation schemes of three satellites to two different customers, each requesting regional coverage. In this example, we assume that each customer requires the satellite to be configured differently. Thus, no single satellite can be used to perform tasks for the two customers simultaneously.[4] The figure shows two allocation schemes: high throughput, where we maximize the number of tasks fulfilled, and max-min fair, where the minimum allocation is maximized. This example highlights a scenario where total throughput is reduced by only 9% to improve the smallest throughput by 30%. Clearly, fair resource allocation policies will interfere with other objectives like utilization maximization. Achieving fairness in CaaS platforms will require combining mobility, coverage, and capacity constraints, requiring an extension of prior work on achieving fairness in cloud computing [16] and drone fleets [6].

Constellation design. Does every satellite need to have every piece of equipment? Like most shared platforms, overprovisioning in a CaaS platform incurs significant costs. In CaaS platforms the cost of the equipment itself is overshadowed by the cost of putting it in orbit and the power budget needed to operate it in orbit. Unlike most shared platforms, capacity overprovisioning may not necessarily translate to an increase in useful capacity. In particular, power budget constraints can limit the number of sensors that can be used simultaneously. Further, the equipment needed on a particular satellite should be constrained by its orbit (i.e., the areas it will be flying over). Thus, capacity planning and constellation design in CaaS platforms will dictate the orbital parameters and the equipment of individual satellites, subject to cost, power, and usage constraints.

5 Platform Architecture

Figure 3 shows the architecture of a CaaS platform. The platform has two key components that tackle the challenges discussed in the previous section: a centralized scheduler and a per-satellite hypervisor. Users submit their demands to a central scheduler, specifying their required resources and region of operation. The scheduler allocates resources that meet customer demand, taking into account resource availability, policy, power, and privacy constraints. Further, the scheduler defines the configuration of each of the satellites to meet the demand of customers while ensuring isolation of resource utilization and private access to data. Configurations, along

[4]Note that other constraints including power, communication, and storage constraints can prevent a single satellite from serving two customers simultaneously. For example, each customer might require a different sensor but the power budget allows for only a single sensor to operate.

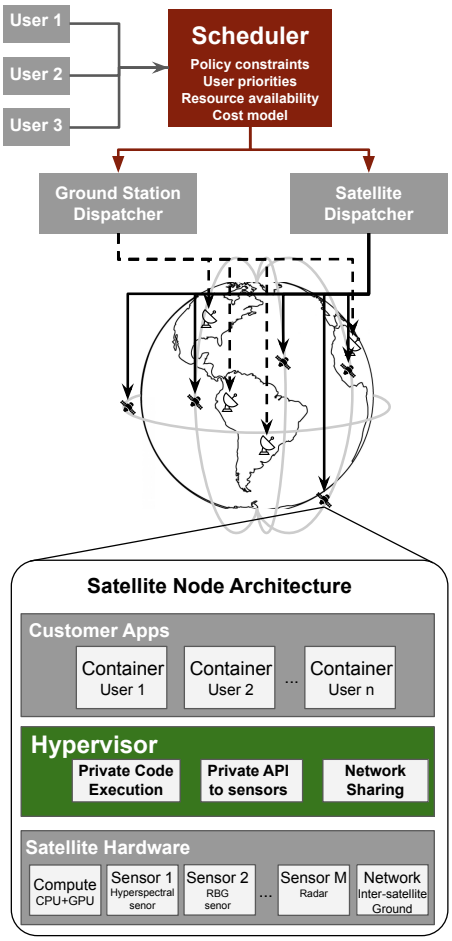


Figure 3: Platform Architecture with the two key components highlighted.

with the workload submitted by the customer, are sent to the satellite and ground station dispatchers. Dispatchers are responsible for configuring satellite and ground stations with the parameters set by the scheduler. Further, dispatchers are responsible for migrating tasks between satellites to ensure that a customer maintains coverage over a particular region. Note that the platform can treat subscription-based services as just another customer.

A satellite node provides a programmable platform for data collection and code execution. An application interacts with satellite hardware through a hypervisor that's responsible for isolating different customers. Further, the hypervisor is responsible for ensuring private access to sensor data and the protection of customer data stored on the satellite. In particular, the hypervisor provides an API for reading, processing, and storing sensor data. The goal of the API is to prevent access to raw data or direct manipulation of hardware configuration. Rather, raw data is preprocessed to prevent side channel attacks discussed above. It should also provide sandboxed execution environments [7, 20] that balance isolation, performance, and resource efficiency.

The objective of our design is to sandbox individual customers, allowing them to interact with satellite hardware only through sanctioned APIs to minimize the chances of misuse. Even when customers are offered a virtual constellation, satellite configuration and application execution and orchestration to the platform should be handled by the CaaS platform through user-supplied configurations that describe user intent, not low-level satellite parameters. This intent-based APIprevents customers from configuring a satellite or its sensors directly. Intent-based configuration has been demonstrating its success in improving network reliability by automating reaction to failure and lowering the chances of wrong configuration [21, 28]. Intent-based configuration is CaaS platforms will allow customers to express their requirements without providing them direct access to any of the infrastructure. Further, it will enable the automated handling of constellation dynamics. This approach sacrifices platform flexibility and programmability to ensure isolation between different customers.

6 A CaaS Scheduling Simulator

We develop a simulator to enable researchers to experiment with scheduling algorithms for constellation-as-a-service platforms. The simulator allows its users to define a physical constellation using a combination of CSV and YAML files: satellite TLEs, ground station locations and capacity, and resources available on individual satellites and their capacity (e.g., number of CPUs and network bandwidth). Users can use the same format to define one or more virtual constellations to represent virtual constellation demand. Coverage area demand is defined through lat-long coordinates, along with resources needed for the covered area (e.g., sensors and compute resources). The simulator parses this input and checks its validity, then, passes the information to an optimization engine. Our implementation of the optimization engine leverages Google's OR tools [4], but can be easily extended to accommodate user-defined heuristics or other optimization libraries. The output of the engine is a bipartite graph, representing the assignment of physical satellites to specific customers. The simulator processes the graph to produce a visualization in Cesium [3]. Our objective is to enable researchers to focus on experimenting with new scheduling algorithms. The simulator provides a library of physical and virtual constellations based on deployments of real constellations obtained from CelesTrak [2].

7 Conclusion

Constellation-as-a-service (CaaS) platforms is an idea whose time has come, enabled by an increasing demand for satellite-based services and COTS components that run conventional

software. Enabling the adoption of CaaS will require making those platforms efficient and secure. In this paper, we highlight four specific challenges unique to CaaS platforms compared to other shared platforms: sensor configuration, power management, fairness, and constellation design. Tackling these challenges will require algorithmic and systems innovations. We present a lightweight simulator of the performance of CaaS platforms under different scheduling policies, constrained by satellite equipment, constellation design, and user demand.

Acknowledgment. We thank the anonymous reviewers for their invaluable feedback. This work was funded in part by NSF grant CNS-2345827.

References

[1] 2023. Satellogic: Constellation-as-a-Service. Available at https://satellogic.com/products/constellation-as-a-service/ (2024/07/10).

[2] 2024. CelesTrak: NORAD GP Element Sets Current Data. Available at https://celestrak.org/NORAD/elements/ (2024/07/10).

[3] 2024. Cesium: The Platform for 3D Geospatial. Available at https://cesium.com (2024/07/10).

[4] 2024. Google OR-Tools. Available at https://developers.google.com/optimization/ (2024/07/10).

[5] The US Federal Aviation Administration. 2023. *Risk Associated with Reentry Disposal of Satellites from Proposed Large Constellations.* Technical Report. Available at https://www.faa.gov/sites/faa.gov/files/Report_to_Congress_Reentry_Disposal_of_Satellites.pdf (2024/07/10).

[6] Arjun Balasingam, Karthik Gopalakrishnan, Radhika Mittal, Venkat Arun, Ahmed Saeed, Mohammad Alizadeh, Hamsa Balakrishnan, and Hari Balakrishnan. 2021. Throughput-fairness tradeoffs in mobility platforms. In *Proceedings of the 19th Annual International Conference on Mobile Systems, Applications, and Services (MobiSys)*. 363–375.

[7] Andrew Baumann, Marcus Peinado, and Galen Hunt. 2015. Shielding applications from an untrusted cloud with haven. *ACM Transactions on Computer Systems (TOCS)* 33, 3 (2015), 1–26.

[8] Vaibhav Bhosale, Ketan Bhardwaj, and Ada Gavrilovska. 2020. Toward Loosely Coupled Orchestration for the {LEO} Satellite Edge. In *3rd USENIX Workshop on Hot Topics in Edge Computing (HotEdge 20)*.

[9] Aaron C Boley and Michael Byers. 2021. Satellite mega-constellations create risks in Low Earth Orbit, the atmosphere and on Earth. *Scientific Reports* 11, 1 (2021), 10642.

[10] Brian Wang. 2019. Starlink Satellites Could Cost $250,000 Each and Falcon 9 Costs Less than $30 Million. Available at https://www.nextbigfuture.com/2019/12/spacex-starlink-satellites-cost-well-below-500000-each-and-falcon-9-launches-less-than-30-million.html (2024/07/10). (2019).

[11] Christopher S. Allen et al. 2018. Chapter 4 - Spaceflight environment. In *Space Safety and Human Performance*, Tommaso Sgobba and Barbara Kanki and Jean-François Clervoy and Gro Mjeldheim Sandal (Ed.). 87–138.

[12] Chris Daehnick, Isabelle Klinghoffer, Ben Maritz, , and Bill Wiseman. 2020. Large LEO satellite constellations: Will it be different this time? Available at https://www.mckinsey.com/industries/aerospace-and-defense/our-insights/large-leo-satellite-constellations-will-it-be-different-this-time (2024/07/10).

[13] Lixin Liu et al. 2024. Democratizing Direct-to-Cell Low Earth Orbit Satellite Networks. In *21st USENIX Symposium on Networked Systems Design and Implementation (NSDI 24)*. USENIX Association.

[14] Vaibhav Bhosale et al. 2023. Don't Let Your LEO Edge Fade at Night. In *HotInfra'23*.

[15] Andrew Falle, Ewan Wright, Aaron Boley, and Michael Byers. 2023. One million (paper) satellites. *Science* 382, 6667 (2023), 150–152.

[16] Ali Ghodsi, Matei Zaharia, Benjamin Hindman, Andy Konwinski, Scott Shenker, and Ion Stoica. 2011. Dominant resource fairness: Fair allocation of multiple resource types. In *8th USENIX symposium on networked systems design and implementation (NSDI '11)*.

[17] Jared Michael Greene, Mohammed Faraz Admani, Jacob Nelson Glueck, Sergii Ziuzin, Francesco De Paolis, Dhruv Dawar, and Christopher Yu. 2023. System and method of providing access to compute resources distributed across a group of satellites. US Patent App. 17/955,401.

[18] Yvon Henri. 2016. ITU: Orbit/Spectrum International Regulatory Framework Challenges in the 21st century. *6th Nandasiri Jasentuliyana Keynote Lecture.([Online]. Available:)* (2016).

[19] Brent Horine. 2021. Creating a Marketplace for a Constellation as a Service. *Small Satellite Conference* (2021).

[20] Dayeol Lee, David Kohlbrenner, Shweta Shinde, Krste Asanović, and Dawn Song. 2020. Keystone: An open framework for architecting trusted execution environments. In *Proceedings of the Fifteenth European Conference on Computer Systems*. 1–16.

[21] Aris Leivadeas and Matthias Falkner. 2022. A survey on intent-based networking. *IEEE Communications Surveys & Tutorials* 25, 1 (2022), 625–655.

[22] Christopher M Maloney and et al. 2022. The climate and ozone impacts of black carbon emissions from global rocket launches. *Journal of Geophysical Research: Atmospheres* 127, 12 (2022), e2021JD036373.

[23] Jonathan C McDowell. 2020. The low earth orbit satellite population and impacts of the SpaceX Starlink constellation. *The Astrophysical Journal Letters* 892, 2 (2020), L36.

[24] Michael Baylor. 2018. With Block 5, SpaceX to Increase Launch Cadence and Lower Prices. Available at https://www.nasaspaceflight.com/2018/05/block-5-spacex-increase-launch-cadence-lower-prices/ (2024/07/10).

[25] Kathryn O'Donnell, Meghan Weber, Joy Fasnacht, Jeff Maynard, Margaret Cote, and Shayn Hawthorne. 2023. Extension of cloud computing to small satellites. (2023).

[26] Tobias Pfandzelter, Jonathan Hasenburg, and David Bermbach. 2021. Towards a computing platform for the LEO edge. In *Proceedings of the 4th International Workshop on Edge Systems, Analytics and Networking*. 43–48.

[27] Jonas Radtke, Christopher Kebschull, and Enrico Stoll. 2017. Interactions of the space debris environment with mega constellations—Using the example of the OneWeb constellation. *Acta Astronautica* 131 (2017), 55–68.

[28] Sivaramakrishnan Ramanathan, Ying Zhang, Mohab Gawish, Yogesh Mundada, Zhaodong Wang, Sangki Yun, Eric Lippert, Walid Taha, Minlan Yu, and Jelena Mirkovic. 2023. Practical intent-driven routing configuration synthesis. In *20th USENIX Symposium on Networked Systems Design and Implementation (NSDI '23)*. 629–644.

[29] Emilio Rapuano, Gabriele Meoni, Tommaso Pacini, Gianmarco Dinelli, Gianluca Furano, Gianluca Giuffrida, and Luca Fanucci. 2021. An fpga-based hardware accelerator for cnns inference on board satellites: benchmarking with myriad 2-based solution for the cloudscout case study. *Remote Sensing* 13, 8 (2021), 1518.

[30] Martin Ross. 1996. Local Effects on of Solid Rocket Motor Exhaust on Stratospheric Ozone. *Journal of spacecraft and rockets* 33, 1 (1996), 144–153.

[31] Robert G Ryan, Eloise A Marais, Chloe J Balhatchet, and Sebastian D Eastham. 2022. Impact of rocket launch and space debris air pollutant emissions on stratospheric ozone and global climate. *Earth's Future* 10, 6 (2022), e2021EF002612.

[32] Shangguang Wang and Qing Li. 2023. Satellite computing: Vision and challenges. *IEEE Internet of Things Journal* (2023).