Toward Loosely Coupled Orchestration for the LEO Satellite Edge

Vaibhav Bhosale, Ketan Bhardwaj, Ada Gavrilovska
Georgia Institute of Technology

Abstract

Low Earth Orbit (LEO) satellites are envisioned to be capable of providing Internet services for billions of users who currently lack reliable Internet connectivity. This calls for a new LEO edge capable of providing edge computing benefits from space. This paper proposes an orchestration approach for the LEO edge that incorporates path models, temporal compensation and affinity chains as the primary scheduling constructs, and presents preliminary results that illustrate opportunities for achieving improved service availability and improved performance for a stateful (caching) edge function.

1 Introduction

By 2025 as many as 1,100 satellites could be launching each year. Just one project, SpaceX's ambitious Starlink, aims to fly 12,000 LEO satellites by 2027 [20, 21]. As per predictions from early 2019, Figure 1 shows the total number of planned launches yearly. In addition to this meteoric rise in launches, the Internet based on Low Earth Orbit (LEO) satellites is envisioned as being capable of providing Internet services to billions of users worldwide [34]. What this points to is that, within the next couple of years, the LEO satellites-based backhaul will support a significant fraction of Internet services. The more interesting aspect of these satellites is the availability of computational resources onboard the satellite [25], opening up an opportunity for a new form of edge – the LEO satellite edge.

Traditionally, LEO satellites are used in a “bent pipe” architecture where every satellite simply acts as a communication link and pushes off the processing to the terrestrial datacenters [35]. This has already been shown to help provide Internet to remote areas and settings where deploying fiber is challenging or not possible (such as for airplanes, ships, etc.) [5]. Treating LEO satellites as a "new edge" brings intuitive advantages such as reduced latency to serve requests and reduced down link utilization since all requests need not go to the data-center. In our experiments, we observe that a

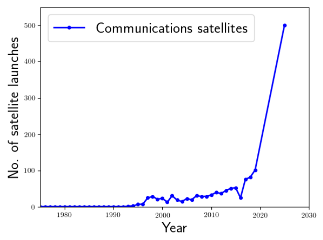

Figure 1: Satellites currently in orbit, along with plans announced for future years. Source: Union of Concerned Scientists; TR projections based on announced plans

simple LRU web cache (with size 4.5 GB) for the Wikipedia web service can achieve a cache hit rate upwards of 50%, thus providing half the latency to end users using a LEO satellite edge compared to one that uses LEO satellites with the bent pipe architecture. Similarly, an image analytics function on a LEO edge is shown to alleviate pressure on downlink capacity by 20x [26, 27]. Deploying such applications on the LEO edge would require orchestration support.

However, most of the currently commissioned satellites in space serve a single mission. There satellites typically capture images for weather forecasting, disaster management or monitoring regions on land for security. Some of these scenarios, such as collecting remote sensing data over Greenland (Ulloriaq) [15, 28], are so specialized that the satellite is idle for most of the duration of its orbit. This has led to development of an orchestration stack that only needs to deploy specialized and well-planned functions on satellites.

With recent developments to commercialize LEO satellites [11, 13] and the availability of a larger number of geographically distributed ground stations as a service [2], we argue that fixed-function deployment will be insufficient to harness the full potential of the LEO edge. This is particularly the case due to the costs and timescales associated with the satellite life cycles. We foresee the need for future capabilities that enable dynamic configuration and deployment of LEO edge functions [22]. Furthermore, the mobility and temporal visibility of a satellite raise the need for those functions to be selectively active in a target service area. Finally, to maximize the benefit from the LEO edge, which could play a role in helping LEO satellites achieve economic feasibility, it will be critical to enable capabilities for its on-demand, multi-tenant and dynamic operation. In short, the LEO edge will require orchestration capabilities, much like what is required (and being developed) for the terrestrial edge [32, 37].

A good starting point in the design space for LEO edge orchestration are current terrestrial orchestration systems. However, the current design of the orchestration systems is hinged on a *tight coupling* between the orchestrator and the infrastructure it manages, making it unsuitable for the LEO edge due to its inherent characteristics. First, the time-of-sight of a particular LEO satellite is severely restricted due to the continuous motion of the satellite with respect to the Earth. With a satellite visible only for a period of $7-30$ minutes depending on its altitude [36], it is imperative that applications be rescheduled onto the next incoming satellite to prevent the risk of reduced availability of the edge functions. Second, the lack of accuracy in predicting the position of LEO satellites is well known and continues to be an active area of research in aerospace engineering [24, 38]. This presents difficulties in determining the position of, and thus in predicting the trajectory of, the LEO satellites and further compounds the problem of a limited time-of-sight.

Addressing the above-mentioned problems is already nontrivial in terms of selecting the right LEO edge nodes and determining a schedule for application orchestration. A solution focused on these two aspects alone, would only be sufficient for stateless applications like geographical image analytics. For stateful applications, like a web cache, another key aspect that needs to be handled is graceful state transfer. If the state is rebuilt every $7-10$ minutes, there would not be much benefit that can be extracted from the edge functions on LEO satellites. Without access to state, the idea of localization, a key reason for edge benefits, cannot be realized in the LEO edge. This highlights a clear need to improve the state handling in orchestration to deliver tangible benefits from LEO satellite edge.

In response, we present Krios[1]– an orchestration system for the LEO edge. Krios provides fundamental constructs and directives which can be incorporated on top of any current orchestration framework, with a few additional modules to address the requirements of the LEO edge. Krios achieves its goals through the key idea of *loose coupling* between edge function orchestration and the underlying infrastructure.

Loose coupling in Krios is achieved through three new mechanisms as part of its scheduling constructs. First, incorporating satellite *path projection models* in orchestration enables Krios to predict the trajectory, and thus future positions, of LEO satellites with respect to regions in which they are visible (availability) with known bounds on the error, depending on models used. This allows Krios to select appropriate satellite nodes and (re)schedule the edge functions across the LEO edge without impacting their availability in the target region. Second, *temporal compensation* enables ahead-of-time hand-off, i.e. deployment and preemptive killing of edge functions on LEO satellites, masking the initialization times. Third, *cluster affinity chains* that span individual clusters allow Krios to choose the satellite among many which may be visible ahead of time, i.e. before those nodes join a given ground station cluster. This reduces the time complexity of choosing the satellite and makes deployment more predictable. These mechanisms are also leveraged to orchestrate the edge functions' state transfers across the LEO edge.

[1]derived from the Greek god of constellations, Crius

In this paper, we use Kubernetes [8] to demonstrate the benefits of Krios. It is, after all, the starting point for the terrestrial edge, as it leverages all the technology developments in the datacenter space and has gained acceptance in industry as the go-to orchestrator for web services in the cloud and edge functions at the edge [7].

Summarizing, the contributions of this paper are: (i) quantifying the limitations of terrestrial orchestration solutions when used in the LEO edge due to the tight coupling to infrastructure inherent in their design, and (ii) design of an end-to-end platform – Krios– that embodies the new system support needed to loosen this coupling and to make LEO edge orchestration feasible. Finally, we demonstrate the promise of the approach through our preliminary experimental evaluation.

2 Background

Case for Satellite Internet: LEO satellites have been in existence since the late 1990s when they were designed to support voice, data, facsimile, and paging. IRIDIUM was one of the major players, having launched 95 satellites between 1997 and 2002. But LEO satellites could not compete with the GSM services and hence, could not gain much traction. Since then, a number of factors have changed, making a case for the success of Internet services based on LEO satellites. The biggest of those is the decreasing cost of space access, aided by the commercialization of launch vehicles [10, 29]. While the lifecycle costs per satellite for Iridium were about US $ 5.7 billion [3], the estimated cost for a 4425 LEO constellation is about US $10 billion [18].

In addition to this, while the first use case of LEO satellites targeted the existing sets of mobile and Internet users, the focus has now shifted to the next billion users. This provides the benefit of manufacturing at scale and results in further reduction of operational and manufacturing costs. Further, there is a significant design difference in how the system was supposed to work earlier in comparison to today. Earlier, a user would need a specialized handheld device to communicate with a satellite. These devices with significantly powerful antennas (to directly connect to the satellites) led to higher

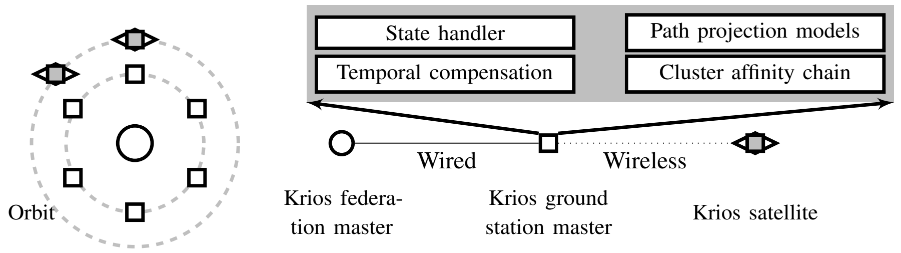

Figure 2: High level design of Krios components - geographically(left) and the network connections (right).

power consumption [34] than the competing GSM devices. In the newer design, devices connect first to ground stations using standard communication interfaces and the ground stations provide the connectivity to the satellites [19] and has been argued to be power efficient [34]. These factors make a strong case for the success of LEO satellites as a medium for providing Internet services.

Review of current satellite systems: Ground stations are a key component in the current satellite systems. End users' communication requests are passed to the satellites via the ground stations. As per the bent-pipe architecture, on receiving a request the satellite bounces it off to another specific ground station server. These ground servers are either connected to a datacenter that serves the request or pass it onto its next hop to the next visible satellite, creating an eventual zigzag pattern from the request flow [30, 31]. There are multiple ground stations depending on the geographical spread of users, and they are strategically positioned so that there is always at least one visible satellite that would be used to serve the requests.

Another important aspect of the system is support for inter-satellite communication links. The presence of these links can significantly reduce the time required to reach the datacenter due to the direct inter-satellite communication, compared to the zigzag pattern incorporating the ground servers. Among the current LEO providers, OneWeb's system will not have inter-satellite links, Starlink has plans to incorporate these links at a later time, though the satellites being launched currently do not have them, and Telesat will have these links when they start service in 2022 [25, 30, 31].

Potential role of LEO edge in current and emerging use cases: A LEO edge could play a role in satellite-supported application scenarios like remote sensing [39], weather forecasting, space analysis [23], etc. Most of the current scenarios continuously send the captured media content to ground systems. A LEO edge would be able to perform the initial processing and only transmit necessary metadata to the ground systems, utilizing low bandwidth and lesser resources on the ground systems [26, 27].

The proposed satellites have a high capacity capable of providing aggregate downlink capacity ranging from 17 to 23 Gbps [19]. Thus, a LEO edge will play a crucial role in adding coverage for sparsely populated areas and regions where setting up terrestrial systems is difficult. Further, it will be able to supplement communication during disasters when the terrestrial systems might have been impacted. A LEO edge can also be used in providing communication services in airplanes [4, 12] and ships.

An important aspect to consider is that the LEO edge would likely never be standalone; the terrestrial counterpart will continue to exist. Hence, to aid with economic feasibility, it is important to ensure that the developers' experiences remain similar irrespective of whether an application is being deployed in space or terrestrially.

3 Design Challenges and Feasibility

The goal of the design of Krios is to address the high dynamism inherent in the LEO satellite environment. Krios achieves this by federating ground stations as Krios-ground station masters responsible for tracking the mobility of LEO satellites and for triggering the necessary orchestration actions needed for application hand-off on Krios-satellites. This is done in a way that leverages opportunities afforded by the periodicity of the LEO satellites' motion, while also explicitly compensating for the inherent inaccuracy in the predicted satellite positions. Furthermore, Krios allows for current orchestration architectures such as Kubernetes to be seamlessly leveraged, while minimizing overheads related to the need for more frequent orchestration. Finally, unlike [27], Krios can work well for a constellation without inter-satellite links provided there are a sufficient number of ground stations available. Next, we briefly touch on the key design elements:

Achieving LEO edge orchestration with existing stacks: Using current orchestration stacks for a LEO edge will lead to considerable downtime. The reasons for this are tied to the underlying assumptions about the liveness of the infrastructure, the job lifetimes and the connectivity between infrastructure nodes. While generally failures in infrastructure nodes are dealt with in a reactive manner, Krios takes a preemptive approach, i.e. an edge application may need to be preempted and rescheduled even when the node it is running on is healthy. For instance, the number of times an application is initialized on a new LEO edge node is considerably higher than that on a terrestrial edge node since every time a satellite goes out of sight from a ground station, the application needs to be scheduled on a different node. An application that needs to be available 24 hours at a given location through a LEO edge in orbit at an altitude of 200 km would need to be initialized more than 200 times even without any true node or network failure. To put this in perspective, even 5 seconds of initialization time, commonly associated with Kubernetes pod startup time [1, 9], results in at least 16 minutes unavailability.

Incorporating mobility of the LEO edge: A major difference between the LEO edge relative to current terrestrial systems is that the satellite nodes are continuously in motion, leading to dynamic network connections with the ground stations. As stated earlier, an intuitive starting point for multi-tenant and dynamic operation over mobile satellites would be to use current terrestrial systems and to model a satellite going out of sight as a node failure, with the orchestrator handling the subsequent application hand-off to another node (in this case to a visible satellite).

In current terrestrial systems, individual node failures are fairly sporadic and random, whereas in a LEO edge scenario, due to the constant rotation, the failures are periodic and high frequency. For example, at the height of 500 km, every satellite will experience an out of sight failure every 10 minutes. In addition to this, it would be very difficult to identify true node failures in the system, making troubleshooting actual issues much harder. Similarly to [27], we observe that using path models and incorporating information generated via them into the scheduling constructs helps to solve this problem. However, adding them to the satellites would incur additional resource usage in the already resource constrained environment. This points to a more suitable design decision to run and cleanly incorporate information generated by path models into the scheduling constructs of Krios-ground station masters. The key insight here is that to leverage existing path models, one does not need to modify the orchestrator itself. Krios provides for use of pluggable path models to predict the location of satellites and, in that manner, their connectivity to a particular ground station. This enables Krios to drive scheduling decisions "just in time".

Leveraging periodicity in LEO edge mobility: With the mobility of the nodes incorporated in orchestration, the number of application handoffs will increase. In a LEO constellation, there are multiple satellites visible at any particular time depending on their different orbital radius, rotation in different rotational planes and also different visibility. Thus, it is important to add support for disambiguating the decision to hand-off an application to a particular satellite as per some pre-defined policy. For example, a policy could be as simple as hand-off to a satellite that has the longest visibility, which would reduce the need for recurring hand-offs and improve the overall network utilization.

By using a central master node, Krios is aware of the different satellites in the constellation and their physical specifications. Using this information, it builds an *affinity chain* as the list of satellites depending on the edge function or on a system-defined policy. Satellites in the same affinity chain would ensure that there would not be any specification mismatch for an application and also ensure that the new satellite would be in sight for a larger amount of time. Use of affinity chains allows Krios to make potential scheduling decisions ahead of time and to reduce the runtime scheduling overheads.

Compensating for error in mobility prediction of LEO satellites: Intuitively, the use of path models to drive orchestration seems straightforward. However, the reality is that it is difficult to accurately predict a satellite's position and thus its connectivity at a particular time. The fundamental reason is that the path models (or even real measurements through high power antennas) generally have an error component associated with their prediction of a position of a particular LEO satellite.

The satellite positioning error implies that a satellite may become out-of-reach of a ground station sooner than what is predicted by path models. If using current terrestrial systems, the orchestrator would consider this a potential node failure, and wait some predefined time before triggering a rescheduling operation. In Kubernetes, for example, this defaults to 300 seconds [14], which, as shown in §4, can lead to up to 85% unavailability of a LEO edge. To avoid this, Krios incorporates a preemptive hand-off of the application before the satellite actually moves out of sight. With this, Krios achieves a "just-ahead-of-time" (re)scheduling response by incorporating the error associated with the path model it is running and the initialization time for an application.

Krios is designed to contend with the worst possible scenarios and uses the combined error bounds in the predicted availability time to trigger early preemptive edge applications hand-off, referred to as *temporal compensation*. Empirically, this preemption time for rescheduling in Krios $\triangle t$ can be written as:

$$\triangle t = \triangle t_{error} + t_{initialize} + t_{RTT}$$

where $\triangle t_{error}$ is the time error due to inaccuracy of path model, $t_{initialize}$ the time to initialize the application on the new node, and t_{RTT} the round trip time for network communication between satellite and ground station.

We posit that there are many assumptions that need to be reconsidered for specific LEO edge scenarios, and this discussion presents initial steps toward highlighting and addressing some of them.

4 Preliminary Evaluation

We run two sets of experiments. The first set of experiments uses Kubernetes to demonstrate the benefits of using path models and temporal compensation. We use the stateless Nginx service for this experiment. The second set of experiments shows the benefit of preemptive state transfer in improving the utility of even simple edge functions. For this, we run the Least Recently Used (LRU) eviction policy for an in-memory web cache and calculate the hit ratio. We use the Simplified General Perturbations 4 (SGP4) [33] model to evaluate the orbital position of individual satellites relative to the Earth-centered coordinate axis, also accounting for Earth's rotation.

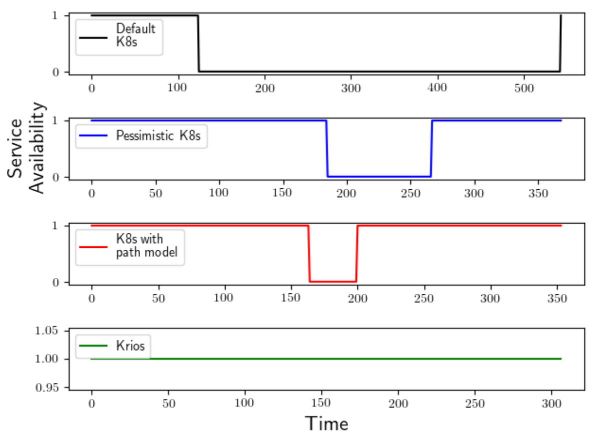

Figure 3: Downtime for the four scenarios - default Kubernetes, pessimistic Kubernetes, Kubernetes with just path awareness and Krios handoff.

Experimental Setup: We run these tests on five virtual machines, using one of them as the master (Krios-ground station master) and the rest as the worker nodes (Krios-satellites). The VMs are hosted on local x86 servers interconnected by 1 Gbps Ethernet. To model the satellites going out of sight, we periodically reboot the worker machines.

Results: In our preliminary experiments, we seek answers to the following questions:

Q. How much downtime is expected with and without Krios?

Figure 3 shows the service availability around the time a satellite goes out of sight. We compare four scenarios here: default Kubernetes, modeling the loss of connection as node failure; a slightly modified Kubernetes pessimistic to failure, i.e., with lower rescheduling timeout upon a node failure; a path-aware Kubernetes which initiates application handoff right before the loss of a connection; and a Krios application handoff incorporating path-awareness and temporal compensation before the loss of connection. The graph show the service downtime observed due to satellites moving out-of-sight of a ground station.

With the default configuration of Kubernetes, we see a downtime of about 350-400 seconds during every handoff. For a satellite at height 200 km which is visible for 7 minutes, this implies an overall 85% downtime. To be fair and conservative for Krios, we configured Kubernetes to be pessimistic and aggressively reactive to failure. Specifically, we change (i) node-monitor-period to 2 seconds (default 5 seconds [6]), (ii) tolerations.tolerationSeconds to 2 seconds (default 300 seconds [14]), and (iii) node-monitor-grace-period to 16 seconds (default 40 seconds [6]). These changes lead to more frequent monitoring of the nodes and a faster response to a failure. The downside to these changes is higher network utilization which in itself is not practical for LEO satellites as downlink and uplink are both expensive, and a reactive Kubernetes highly prone to false positives due to monitoring failures or oversight.

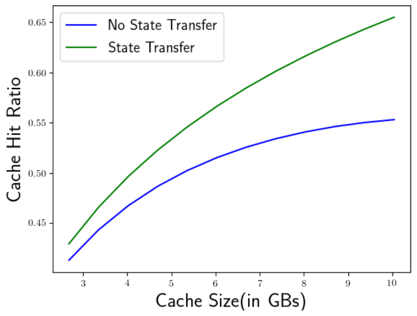

Figure 4: Cache Hits for different cache sizes showcasing benefits of state transfer from the outgoing satellite to the incoming satellite.

Even with an aggressive eviction on top of default Kubernetes, we observe a downtime of about 70-80 seconds. For a satellite at height 200 km with a 7 minute visiblity, this implies an overall 16-18% downtime. For the path aware Kubernetes which initiates the application handoff right before the satellite goes out sight, we still see about 40 seconds downtime, or an overall 10% downtime for a 200 km altitude orbit. For the Krios scenario, there is no observed downtime.

These results demonstrate the value of incorporating path projection models and temporal compensation in an orchestrator such as Kubernetes (even when configured in aggressive, pessimistic mode) for determining the right (re)scheduling policy for applications even with a fast network connection.

Q. How much benefit do stateful functions gain with Krios?

Figure 4 shows the cache hit ratio for varying cache sizes comparing scenarios with and without state transfer from the outgoing to the incoming satellite, when running a web cache for a Wikipedia service. We use the Wikipedia daily page counts dataset [17] assuming each page to have size 3*kb* [16].

Utilising Krios state transfer helps improve the cache hit ratio by about 5-12%. A cache hit eliminates the entire backhaul time from the satellite to the destination datacenter via ground station, and the associated bandwidth utilization. Use of state transfer amplifies these benefits.

5 Summary

We propose Krios as an orchestration service which enables utilizing the LEO edge to its full potential. Krios achieves this by incorporating path projection models, temporal compensation, and developing affinity chains of plausible candidates for (re)scheduling operations. More generally, we conclude that considering LEO satellites as edge, akin to the terrestrial edge, does have benefits. However, designing systems and services for a LEO edge requires more than just migrating the ones developed for the terrestrial edge. There remain many interesting tradeoffs to be explored and new mechanisms to be developed toward a solution for efficient use of a LEO edge. This paper is a step in that direction.

6 Discussion

In this paper, we focus on a few specific design aspects for LEO edge orchestration. However, there are assumptions in different areas that need to be revisited. These include the reliability of the network, the availability of power resources, the complexity of application design, the impact of soft hardware failures on orchestrators, and data compliance.

The network assumptions made in terrestrial systems do not hold in the LEO edge scenario. The large RTT and increased link error rate impact the reliability of the network. Further, LEO satellite networks are considerably more dynamic with regard to connectivity and latency. Thus, this requires new developments in improving the reliability of the network and in routing mechanisms. Power availability is a crucial assumption made in terrestrial systems. A LEO edge orchestrator needs to be aware of the power availability on different nodes while making scheduling decisions.

The stateful function we described is a rather simple one. The state to be maintained is fairly simple. More complex functions like machine learning inference would have a far more complicated state that needs to be considered and designed accordingly. There will be a need to reconsider how to structure these functions in order to benefit from a LEO edge.

Hardware faults such as flipping of RAM bits are higher in satellites due to the severe radiation effects in space. Thus, this requires new development in both the hardware and software stack to be able to handle these failures, in addition to the the discussion of failures presented in this paper. Handling these different types of failure scenarios may expose different tradeoffs that a solution must co-optimize for.

Data compliance has been one of the pressing issues in recent times. While there are no borders in space, the applications deployed would be serving people on the ground and hence there would be policies that might be applicable. This calls for policy changes to govern the compliance in space and also solutions to incorporate clean multi-tenancy to remain compliant.

This work highly depends on the latest events occurring in the industry concerning LEO satellites. As long as the industry players continue on that path, the relevance of this research could increae.

References

[1] 1000 Nodes and Beyond: Updates to Kubernetes Performance and Scalability In 1.2. https://kubernetes.io/blog/2016/03/1000-nodes-and-beyond-updates-to-kubernetes-performance-and-scalability-in-12/.

[2] Amazon AWS Ground Station. https://aws.amazon.com/ground-station/.

[3] Communications Satellite Constellations. http://web.mit.edu/deweck/www/research_files/comsats_2004_001_v10/Unit2%20Design%20Exploration/unit2_summary.htm.

[4] High-Capacity Satellite System. https://www.viasat.com/products/high-capacity-satellites.

[5] Iridium Global Network. https://www.iridium.com/network/globalnetwork/.

[6] Kube-Controller-Manager. https://kubernetes.io/docs/reference/command-line-tools-reference/kube-controller-manager/.

[7] KubeEdge. https://kubeedge.io/en/.

[8] Kubernetes. https://kubernetes.io/.

[9] Kubernetes Performance Measurements and Roadmap. https://kubernetes.io/blog/2015/09/kubernetes-performance-measurements-and/.

[10] Launch Costs to Low Earth Orbit, 1980-2100. https://www.futuretimeline.net/data-trends/6.htm.

[11] OneWeb LEO Satellite based Broadband. https://www.oneweb.world/.

[12] OneWeb Set to Revolutionize In-Flight Connectivity. https://www.oneweb.world/media-center/oneweb-set-to-revolutionize-in-flight-connectivity.

[13] SpaceX Starlink Satellite Constellation for Broadband. https://www.spacex.com/news/2019/05/24/starlink-mission.

[14] Taints and Tolerations. https://kubernetes.io/docs/concepts/configuration/taint-and-toleration/.

[15] Ulloriaq. https://gomspace.com/gomx-4.aspx.

[16] Wikipedia Statistics. https://stats.wikimedia.org/EN/TablesArticlesBytesPerArticle.htm.

[17] Wikistats Pageview Files. https://dumps.wikimedia.org/other/pagecounts-ez/.

[18] With Block 5, SpaceX to Increase Launch Cadence and Lower Prices. https://www.nasaspaceflight.com/2018/05/block-5-spacex-increase-launch-cadence-lower-prices/.

[19] Application for Approval for Orbital Deployment and Operating Authority for the SpaceX NGSO Satellite System. https://cdn.arstechnica.net/wp-content/uploads/2016/11/spacex-Legal-Narrative.pdf, 2016.

[20] FCC Selected Application for Space Exploration Holdings, LLC. SAT-LOA2016111500118, 2016.

[21] FCC Selected Application for Space Exploration Holdings, LLC. SAT-LOA2017030100027, 2017.

[22] Ketan Bhardwaj, Ming-Wei Shih, Pragya Agarwal, Ada Gavrilovska, Taesoo Kim, and Karsten Schwan. Fast, Scalable and Secure Onloading of Edge Functions Using AirBox. In *2016 IEEE/ACM Symposium on Edge Computing (SEC)*, pages 14–27. IEEE, 2016.

[23] Matthew C Chan and John P Stott. Deep-CEE I: Fishing for Galaxy Clusters With Deep Neural Nets. *Monthly Notices of the Royal Astronomical Society*, 490(4):5770–5787, Oct 2019.

[24] Junyu Chen, Jianli Du, and Jizhang Sang. Improved Orbit Prediction of LEO objects With Calibrated Atmospheric Mass Density Model. *Journal of Spatial Science*, 64(1):97–110, 2019.

[25] Inigo del Portillo, Bruce G Cameron, and Edward F Crawley. A Technical Comparison of Three Low Earth Orbit Satellite Constellation Systems to Provide Global Broadband. *Acta Astronautica*, 159:123–135, 2019.

[26] Bradley Denby and Brandon Lucia. Orbital Edge Computing: Machine Inference in Space. *IEEE Computer Architecture Letters*, 18(1):59–62, 2019.

[27] Bradley Denby and Brandon Lucia. Orbital Edge Computing: Nanosatellite Constellations as a New Class of Computer System. 2020.

[28] J. A. Fraire, G. Nies, H. Hermanns, K. Bay, and M. Bisgaard. Battery-Aware Contact Plan Design for LEO Satellite Constellations: The Ulloriaq Case Study. *IEEE Global Communications Conference (GLOBECOM)*, pages 1–7, 2018.

[29] Warren Frick and Carlos Niederstrasser. Small Launch Vehicles-A 2018 State of the Industry Survey. 2018.

[30] Mark Handley. Using Ground Relays for Low-Latency Wide-Area Routing in Megaconstellations. In *Proceedings of the 18th ACM Workshop on Hot Topics in Networks*, pages 125–132, 2019.

[31] Mark Handley. Using Ground Relays With Starlink. `https://www.youtube.com/watch?v=m05abdGSOxY`, 2019.

[32] Attila Hegyi, Hannu Flinck, Istvan Ketyko, Pekka Kuure, Csaba Nemes, and Lajos Pinter. Application Orchestration in Mobile Edge Cloud: Placing of IoT Applications to the Edge. In *2016 IEEE 1st International Workshops on Foundations and Applications of Self* Systems (FAS* W)*, pages 230–235. IEEE, 2016.

[33] Felix R. Hoots and Ronald L. Roehrich. Models For Propagation of Norad Element Sets.

[34] Farooq Khan. Mobile Internet from the Heavens. *arXiv preprint arXiv:1508.02383*, 2015.

[35] Wiley J Larson and James Richard Wertz. Space Mission Analysis and Design. Technical report, Torrance, CA (United States); Microcosm, Inc., 1992.

[36] Younes Seyedi and Seyed Mostafa Safavi. On the Analysis of Random Coverage Time in Mobile LEO Satellite Communications. *IEEE communications letters*, 16(5):612–615, 2012.

[37] Tarik Taleb, Konstantinos Samdanis, Badr Mada, Hannu Flinck, Sunny Dutta, and Dario Sabella. On Multi-Access Edge Computing: A Survey of the Emerging 5G Network Edge Cloud Architecture and Orchestration. *IEEE Communications Surveys & Tutorials*, 19(3):1657–1681, 2017.

[38] Zhaokui Wang, Zhendong Hou, and Yulin Zhang. Improvement of the Long-Term Orbit Prediction for LEO Navigation Satellites Using the Inner Formation Method. *IEEE Transactions on Aerospace and Electronic Systems*, 55(5):2532–2542, 2019.

[39] Q. Xu, H. Zhang, Y. Cheng, S. Zhang, and W. Zhang. Monitoring and Tracking the Green Tide in the Yellow Sea With Satellite Imagery and Trajectory Model. *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, 9(11):5172–5181, 2016.