Mobile Cloud Computing

EINSTEIN CENTER Digital Future

Technical Report

No. MCC.2022.1

Cite this report as:
Tobias Pfandzelter, David Bermbach *Testing LEO Edge Software Systems with Celestial*. Technical Report MCC.2022.1. TU Berlin & ECDF, Mobile Cloud Computing Research Group, 2022.

Abstract:
LEO satellite constellations to provide global broadband Internet access, researchers have proposed to embed compute services within satellite constellations to provide computing services on the *LEO edge*. While the LEO edge is merely theoretical at the moment, providers are expected to rapidly develop their satellite technologies to keep the upper hand in the new space race.

In this paper, we answer the question of how researchers can explore the possibilities of LEO edge computing and evaluate arbitrary software systems in an accurate runtime environment and with cost-efficient scalability. To that end, we present CELESTIAL, a virtual testbed for the LEO edge based on microVMs. CELESTIAL can efficiently emulate individual satellites and their movement as well as ground station servers with realistic network conditions and in an application-agnostic manner, which we show empirically. Additionally, we explore opportunities and implications of deploying a real-time remote sensing application and a web application on LEO edge infrastructure in two case studies on CELESTIAL.

Testing LEO Edge Software Systems with Celestial*

Tobias Pfandzelter
TU Berlin & ECDF
Mobile Cloud Computing Research Group
Berlin, Germany
tp@mcc.tu-berlin.de

David Bermbach
TU Berlin & ECDF
Mobile Cloud Computing Research Group
Berlin, Germany
db@mcc.tu-berlin.de

ABSTRACT

As private space companies such as SpaceX and Telesat are building large LEO satellite constellations to provide global broadband Internet access, researchers have proposed to embed compute services within satellite constellations to provide computing services on the *LEO edge*. While the LEO edge is merely theoretical at the moment, providers are expected to rapidly develop their satellite technologies to keep the upper hand in the new space race.

In this paper, we answer the question of how researchers can explore the possibilities of LEO edge computing and evaluate arbitrary software systems in an accurate runtime environment and with cost-efficient scalability. To that end, we present Celestial, a virtual testbed for the LEO edge based on microVMs. Celestial can efficiently emulate individual satellites and their movement as well as ground station servers with realistic network conditions and in an application-agnostic manner, which we show empirically. Additionally, we explore opportunities and implications of deploying a real-time remote sensing application and a web application on LEO edge infrastructure in two case studies on Celestial.

1 INTRODUCTION

Private aerospace and Internet companies, such as SpaceX[1], OneWeb[2], and Telesat[3], are launching tens of thousands of satellites to provide global broadband Internet access. Thanks to their low-Earth orbit (LEO) and free-space laser links, consumers can expect low-latency, high-bandwidth Internet access anywhere on Earth. New LEO satellite networks challenge not only the old satellite-based Internet access but terrestrial fiber as well [8, 58, 60].

Fueled by that development, it has also been proposed to implement computing resources within these LEO constellations, e.g., [9, 57, 70], to facilitate LEO edge computing, as is common with terrestrial multi-access edge computing (MEC) [23]. Edge resources located on communication satellites, which act as radio uplinks for LEO Internet subscribers, could provide low-latency application access from which especially rural areas and clients without a nearby cloud data center would benefit [9, 11, 54, 57].

LEO edge computing introduces novel challenges in application management: The number of potential satellite servers, with, e.g., the proposed Starlink constellation comprising more than 40,000 satellites at completion [31, 32, 43], will require new approaches to server management. Further, LEO satellites move at speeds in excess of 27,000km/h and ground equipment frequently needs to reconnect to new satellites, resulting in an ever-changing network topology. Finally, LEO edge software is subject to constrained compute resources and the harsh environment of space.

To solve these challenges, researchers will need to develop new middleware systems for application state management, request routing, and service offloading. As LEO edge computing currently exists only as a concept and any actual infrastructure is years away from implementation, emulated testbeds are needed to go beyond the capabilities of simulation and to actually test and benchmark such middleware prototypes. Emulating LEO edge infrastructure in the cloud, however, is non-trivial, as the number of satellite servers raises scalability and cost concerns, and the highly dynamic satellite network and environmental effects must be accurately reflected in the testbed.

In this paper, we thus raise the question of how we can evaluate *arbitrary software systems* for the LEO edge *as accurately as possible* and *in a cost-efficient manner*. To answer this question, we make the following contributions:

- We present LEO edge computing and the challenges of building and evaluating LEO edge software without access to actual infrastructure (§2).
- We introduce Celestial, a novel LEO edge emulation tool based on microVMs and discuss how it addresses these challenges (§3).
- We evaluate these claims by deploying a latency-sensitive edge application on Celestial (§4).
- In a case study on Celestial, we evaluate different deployments of a distributed real-time data analysis service in the context of remote sensor networks to assess the opportunities and implications of the LEO edge environment (§5).
- In a further case study, we evaluate different deployments of a distributed web application to assess the implications of the LEO edge environment (Section 6).
- We discuss threats to validity for our work and derive avenues for future work on Celestial and the LEO edge (§7).

We make our implementation of Celestial available as open-source[4] to help future researchers validate their own applications and platforms. Our hope is that this will make the field of LEO edge computing more accessible and provide a starting point for systems research in this area.

2 BACKGROUND & RELATED WORK

In this section, we give an overview of the state of the art in large LEO satellite communication networks and describe the opportunities and challenges of the novel LEO edge computing paradigm.

*This technical report extends a paper that has been published at the 23rd International Middleware Conference [55]. The main extension is an additional case study with a web application.

[1] https://www.starlink.com/
[2] https://www.oneweb.world/
[3] https://www.telesat.com/

[4] https://github.com/OpenFogStack/celestial

Furthermore, we discuss what it takes to build and test LEO edge software systems without access to LEO edge infrastructure, and where current simulators and testbeds fall short.

2.1 Large LEO Satellite Communication Networks

Satellite Internet access using geostationary orbits at altitudes in excess of 35,000km have been in operation for decades. Yet, their high communication delays and low bandwidth make them infeasible for most applications [15]. Advances in radio and laser technology [44], and progressively lower satellite launch costs in the last few years [42], have now enabled private companies such as SpaceX, Telesat, OneWeb, and Amazon to deploy high-bandwidth, low-latency satellite communication networks using thousands of satellites at altitudes between 500 and 1,500km, called the low-Earth orbit (LEO). Rather than only relaying radio signals from ground stations, these new satellites can use inter-satellite laser links (ISL) for communication between adjacent satellites. Given the vacuum in space, these ISLs can benefit from a ~47% faster light propagation than in fiber cables [10, 31]. Point-to-point communication between two ground stations over the satellite network can thus incur less communication delay than with terrestrial fiber connections. As installing user equipment is also cheaper than installing fiber optic cables, the new large LEO satellite communication networks are expected to challenge not only traditional satellite Internet access but also terrestrial fiber [8, 10].

The core of a LEO satellite communication network is the actual satellite constellation. Such a constellation comprises shells of satellites, each shell at a different altitude and with different orbital parameters. Each of these shells consists of a number of orbital planes, evenly spaced around the equator. Within each plane are satellites that follow the same orbit, evenly spaced around that plane [56, 67].

Figure 1 shows an overview of the satellites in the planned phase I Starlink constellation, which comprises five such shells. The first shell has 1,584 satellites at altitudes of 550km and is split into 72 planes of 22 satellites each. Each of these orbital planes is inclined at an angle of 53° to the Earth's equatorial plane. Additional shells at higher altitudes provide greater areas of coverage at the expense of additional network delay [31, 43]. The ISLs of such a constellation are likely to be arranged in a *+GRID* pattern, where each satellite keeps a link to its predecessor and successor within its plane as well as to one neighbor each in the two closest adjacent planes. This allows any two ground stations to connect directly to each other over the satellite network [10].

2.2 Bringing Compute to the LEO Edge

The edge computing paradigm is the answer to a growing demand and need to process data close to its origin rather than in distant cloud data centers [7, 69]. In fog computing [12] and the terrestrial MEC architecture [23], compute resources are embedded within the access network, i.e., close to clients, the *network edge*. Here, edge applications run on virtualization infrastructure, which makes them available with high bandwidth and low latency, with lower network costs, and with decreased privacy and security risks [52, 61].

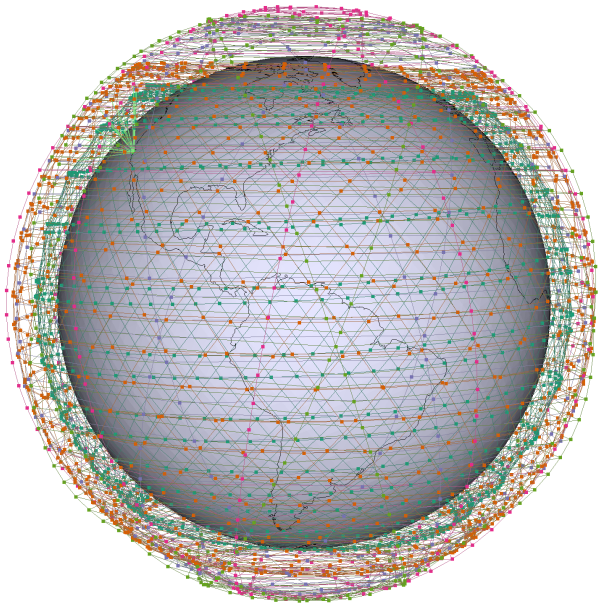


Figure 1: Overview of the planned phase I Starlink constellation with five shells of 1,584 satellites at 550km (turquoise), 1,600 at 1110km (orange), 400 at 1130km (blue), 375 at 1275km (pink), and 450 at 1325km (green) altitude. Colored lines illustrate ISLs, bright green lines show possible ground-to-satellite links for a ground station [43].

In large LEO satellite communication networks, the network edge is the satellite constellation itself – the last provider-operated hop before packets reach user equipment. Uplinks for groups of ground stations converge in a satellite, making it an efficient location for shared computing resources close to clients [54]. Possible applications for LEO edge computing include CDN replication to reduce bandwidth usage in the network and access latency for clients, meetup servers for multi-user interaction such as low-latency gaming or video conferencing, or processing space-native data [9, 11, 54] – a detailed overview of the architecture of such an application is given in our case-study in §5. Given the current skew of cloud and cloudlet data centers being placed close to metropolitan areas, LEO edge computing could help meet the needs of rural areas through the global coverage of LEO satellite networks.

A number of challenges still lie ahead on the path to global LEO edge computing: First, satellite constellations themselves are still being built out and do not yet provide uninterrupted global coverage. As a result, there are no system traces from real world LEO satellite constellations. Second, none of the communication satellites that are already in operation at the moment provide sufficient additional compute resources that could be sold to customers, as satellites are limited by payload restrictions and launch costs. Although multi-tenant satellites, e.g., the *F6* fractionated satellite [14] or the *Tiansuan* satellites [68], have been researched, to the best of our knowledge, no operational compute platform for LEO satellite networks exists at the moment. Third, the deployment and

operation of such resources in LEO introduce additional engineering challenges in power consumption, waste-heat management, or radiation hardening [9, 51, 66]. Fourth, how to abstract from the highly mobile and unique infrastructure using LEO edge computing platforms is still actively being researched, e.g., [11, 57].

2.3 Open Challenges in Building and Testing LEO Edge Software Systems on Earth

A key part of systems research for the LEO edge is the evaluation of software systems in a realistic environment, e.g., through functional testing and benchmarking. While algorithms can be evaluated in simulation, studying the behavior of concrete software systems requires a realistic runtime environment. We believe that enabling researchers and practitioners to quickly and affordably run virtual LEO edge computing infrastructure in the cloud will accelerate both systems research and the actual development and deployment of infrastructure as operators gauge industry and research interest. Yet, building such testbeds for large-scale, highly dynamic infrastructure requires addressing three specific research challenges: How can we ensure that a testbed offers an accurate representation of real LEO constellations? Which abstractions are necessary in the design of a testbed so that arbitrary software systems can be deployed on it? Finally, how can we achieve this with cost-efficiency at scale?

Accurate Emulation of LEO Constellations. A fundamental difference of the LEO edge compared to terrestrial edge or cloud computing is the high mobility of satellites in relation to Earth that result from their low orbits. A complete LEO edge infrastructure is constantly evolving, with network connections and characteristics between servers changing by the second. Furthermore, ground stations also frequently switch their uplink to their closest satellite as a result of this mobility. All applications and platforms developed for the LEO edge need to consider this and proactively replicate data and services. These characteristics of the network influence the performance of edge applications, which are often latency or bandwidth sensitive, and must thus also be part of a LEO edge computing testbed. This should be based on detailed models of Earth and space so that hand-off and data migration techniques can be evaluated as accurately as possible.

Satellite servers will likely use commercial, off-the-shelf compute hardware, e.g., as HPE does with their *Spaceborne Computers* onboard the International Space Station [22, 63]. Although the effects of the Van Allen radiation belts are negligible in lower orbits, servers will be subject to some single event upsets caused by intermittent galactic cosmic rays that can normally be absorbed by Earth's atmosphere and magnetic field [46]. HPE has shown that this can be remediated with standard software and hardware mechanisms, yet at the cost of temporary performance degradation or full shutdowns. Such failure will impact any software running on a satellite and developers will want to test their applications against these scenarios. Especially service orchestrators on top of LEO edge infrastructure need to adapt and react quickly.

Support for Arbitrary Software Systems. A LEO edge computing testbed should enable development and evaluation of any kind of LEO edge software and should not be restricted to certain programming languages, frameworks, or deployment models. The emulator should thus accurately reflect blank-slate servers, especially as we still face unknowns regarding middleware platforms and applications.

When considering the development of platforms, great care should be taken to ensure that a virtual testbed does not impose restrictions on the kind of technologies that can be evaluated within, e.g., a container-based approach where each satellite server is emulated with a single container would impose limitations on testing container-based platforms.

Cost-Efficient Scalability. Large LEO satellite constellations can comprise tens of thousands of satellites and serve millions of clients. Building and scaling applications and platforms for such infrastructure is a difficult challenge and a LEO edge testbed should be able to serve as a way to perform scalability evaluation. The testbed must thus also scale for large constellations and provide the means to emulate such infrastructure.

However, users should not be expected to also provide tens of thousands of physical computers for such an emulation. Rather, the testbed should be cost-efficient in a way where more than one satellite server can be emulated on one host. Further, it should also let the user evaluate only a part of the complete constellation, e.g., to support rapid prototyping and testing of a constellation subset over a certain geographical area or to save experiment costs.

2.4 Related Work

In the edge computing, where applications are widely distributed and physical infrastructure is often inaccessible, researchers commonly rely on virtual testbeds, network simulators, and edge simulators for testing.

Testbed tools such as *Fogbed* [17], *EmuFog* [49], or *MockFog* [33, 34] allow users to create and manipulate virtual infrastructure that mimics a distributed, heterogeneous edge-cloud continuum. Fogbed and EmuFog, however, do not support an emulation of highly dynamic network topologies as required for large LEO satellite constellations. Furthermore, applications are deployed as Docker containers, which makes it unsuitable for evaluating novel platforms as it limits supported software. MockFog supports dynamic network changes and uses a dedicated cloud virtual machine for each compute node, which imposes fewer restrictions on the kinds of software it supports, yet such flexibility comes at a price: With a dedicated cloud virtual machine for each satellite server, we cannot achieve a cost-efficient emulation for large LEO constellations. Similar concerns can also be raised for further IoT and edge computing testbed tooling [5, 6, 21, 37, 38].

On the other hand, edge simulators such as *iFogSim* [30] or *FogExplorer* [35, 36] are more cost-efficient and allow evaluating larger topologies over a longer time period. These could be extended to also simulate the "movement" of (LEO) edge nodes, yet would still not allow users to run their actual application in experiments, thus limiting accuracy of results to the assumptions of the underlying simulation model.

Finally, network simulators such as *ns–3* [16] let users explore network-level effects such as congestion or routing in large-scale networks. Kassing et al. have presented *Hypatia* [43], a network simulator for large LEO satellite constellations based on *ns-3* that allows researchers to measure LEO satellite network characteristics

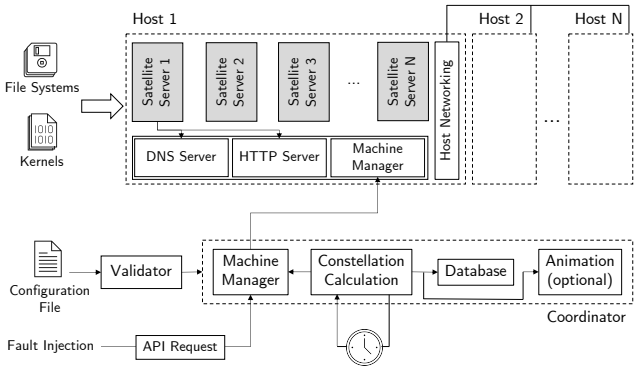


Figure 2: Celestial's coordinator calculates satellite positions and updates machines and network links on Celestial's hosts.

on a packet-level. Similarly, *SILLEO-SCNS* [45], which Celestial's Constellation Calculation is based on, also facilitates studying and visualizing such networks. While important in their own right, these network simulators target a different use case, namely the evaluation of network measurement, and cannot be used to evaluate software systems.

In fact, to the best of our knowledge, there is no testbed tooling which allows users to evaluate LEO edge software and, consequently, systems researchers have until now not been able to study the impact of the LEO edge environment on real applications and platforms.

3 EMULATING THE LEO EDGE WITH CELESTIAL

We present Celestial, a novel emulation tool for edge computing on large LEO satellite constellations. An overview of its architecture is shown in Fig. 2. Celestial can run on an arbitrary number of standard Linux servers, e.g., in the cloud, and comprises two main components: A central coordinator computes satellite orbital paths and networking characteristics. This information is sent to Celestial servers that host a microVM for each satellite server and ground station. Celestial servers also manipulate network connections between microVMs to accurately reflect satellite movement, available links, as well as their delays and bandwidth. In this section, we describe how we address the open research challenges with Celestial.

3.1 Accurate Emulation of LEO Constellations

At the heart of Celestial, the *Constellation Calculation* component updates the state of the satellite network periodically, including the positions of satellites and ground stations, network link distance and delays, and shortest paths between nodes. This is based in large parts on the *SILLEO-SCNS* network simulator [45] for large LEO satellite constellations, which we extend with support for the *SGP4* simplified perturbations models. Additionally, we use more efficient implementations of Dijkstra's algorithm [19] and the Floyd-Warshall algorithm [25] to calculate the shortest network paths within the constellation and their end-to-end latency.

SGP4 is the state-of-the-art in calculating the positions of satellites and takes perturbations caused by atmospheric drag, the Earth's shape, and gravitational effects from Moon or Sun into account [39]. The model input parameters can be obtained from the database of NORAD two-line element sets (*TLE*)[5] for satellites already in orbit or calculated based on simple parameters such as a satellite shell's inclination and altitude. To limit side effects and ensure repeatable testing, all parameters are passed within a single configuration file. This includes network parameters, such as ISL bandwidth, compute parameters that describe the allocated resources for satellite and ground station servers, orbital parameters for satellite shells to support different kinds of constellations, and ground station locations.

Celestial then uses satellite and ground station positions to calculate ground station uplinks and the satellite network topology. Here, ISL connectivity depends on the line of sight between two adjacent satellites, e.g., if a possible laser link drops below a certain altitude, the Earth's atmosphere may refract that laser, causing an intermittent loss of connectivity. Similarly, ground stations can only communicate with satellites that are above a configurable minimum elevation above the horizon. In our tests, these calculations could be completed within one second even on a standard laptop. The results are then transferred to the hosts, where network delays and bandwidth constraints between the satellite servers are emulated using `tc`[6] and `tc-netem`[7]. Emulated network delays are injected with a 0.1ms accuracy and any latency between hosts is taken into account, yet this only works if this latency is low enough, e.g., hosts are located in the same datacenter. Additionally, emulated servers can still reach the Internet through the host, e.g., to store experiment data in some central location.

We isolate microVMs in dedicated `cgroups` to gain more finely grained control over the CPU cycles a server process is allowed to use, making the emulation of severely constrained satellite servers possible. Through an API, users can change machine parameters at runtime and even terminate and reboot machines to model faults, e.g., caused by radiation.

Celestial also provides an optional animation component that visualizes the state of the constellation during the emulation run, e.g., Fig. 1 was generated by this component. We believe that this can be a great help in understanding the characteristics of satellite mobility and networking as well as the effects on their software systems, especially for developers who are new to satellite networks

In Celestial, we take all effects into account that are currently conjectured to have a significant impact on software systems running on the LEO edge, namely dynamic network delays, bandwidth restrictions, constrained compute and storage resources, and service interruption through environmental effects such as radiation [9, 57]. When additional effects are studied and traces from real LEO edge deployments become available, researchers will be able to quickly incorporate the necessary changes in Celestial: First, the core Constellation Calculation component that dictates the effects that are emulated in the rest of the system comprises only 971 lines of Python source code that can be easily customized and extended. The

[5] https://celestrak.com/NORAD/elements/
[6] https://man7.org/linux/man-pages/man8/tc.8.html
[7] https://man7.org/linux/man-pages/man8/tc-netem.8.html

separation from Machine Managers allows the resulting networking and machine parameters to be sent to host machines without modification. Second, `tc-netem` offers advanced network emulation features that are not currently used in Celestial, such as packet loss or duplication, delay distributions, packet corruption, or packet reordering. If such characteristics will be useful for LEO edge testbeds in the future, they can be added with only small changes to the Celestial codebase. Third, we forego the use of complex dependencies such as orchestrators (e.g., Kubernetes or Mesos) and overlay network tooling (e.g., Flannel or Calico) to reduce the complexity of Celestial. While we rely on some of their underlying technologies, such as the `firecracker-containerd` plugin[8], not including such complex software simplifies prototyping new features and reduces maintenance efforts of our implementation, especially considering the required changes to such dependencies.

3.2 Support for Arbitrary Software Systems

All satellite and ground station servers are emulated with *Firecracker* [1] microVMs that run on the Celestial hosts, managed by the *Machine Manager* components. Firecracker microVMs provide a sub-second boot time and support for microVM suspensions, and can use configurable Linux kernels and root filesystems. Celestial thus gives users the control over kernel features, installed software, and programming and deployment models for their applications. The process of compiling filesystems, starting (cloud) hosts, and uploading the necessary files can be automated using common orchestration tools such as *Ansible*[9], yet it is highly user-specific, so we do not include it in Celestial. Crucially, satellite servers are thus provided as a blank-slate and users may even set up container technologies such as Docker within their emulated servers. This is of considerable importance for systems researchers, as it also facilitates the evaluation of container-based application orchestration using tools such as Kubernetes or FaaS platforms [11, 53].

To aid the development of applications and platforms on Celestial, it includes two additional components: First, each Celestial host provides a local DNS server that can resolve microVM network addresses with a custom DNS record. Applications can simply query the `A` records for, e.g., `878.0.celestial` to get the network addresses of satellite 878 in the first shell. Applications thus do not have to be aware of the underlying IP address space calculation for Celestial's virtual network interfaces.

Second, Celestial hosts run an HTTP server that provides information on satellite positions, network paths between satellites, constellation information, and more to the emulated satellite servers. This information is sourced from a central database on the Celestial coordinator that is updated by the Constellation Calculation. Application developers can leverage this API to quickly test their applications on different kinds of LEO constellations without having to implement a custom model of satellite movement and network behavior. In a real LEO edge computing scenario, we expect such information to be available from a central source provided by the satellite network operator or from public sources such as a TLE database. However, users are free to use only a subset or none of these APIs, e.g., when testing their own models.

[8] https://github.com/firecracker-microvm/firecracker-containerd
[9] https://www.ansible.com/

3.3 Cost-Efficient Scalability

To emulate arbitrarily large LEO and complex LEO satellite constellations with thousands of satellite servers, Celestial supports horizontal scale-out across many hosts over which microVMs are distributed. For this, users may simply instantiate necessary cloud infrastructure that can be terminated once experiments and tests are completed. Celestial automatically creates an overlay network using *WireGuard* [20], thus connecting hosts and offering routing between microVMs. Further, the use of microVMs allows for high over-provisioning as well as collocation of many emulated satellite servers on few physical servers. In Celestial, we use the fact that all satellite servers are identical to our advantage and de-duplicate microVM root filesystems using a common immutable disk image in addition to an overlay for each microVM, allowing us to save on storage space and improve performance.

Celestial can emulate large and complex LEO satellite constellations with thousands of satellite servers, yet not every test or evaluation requires emulating each individual satellite server at the same time: As an optional feature, we introduce a configurable bounding box, a geographical area on Earth to which emulated satellite servers are limited. As satellites are mobile, they can quickly move in and out of this bounding box and may only stay relevant for evaluation for a few minutes. To free up resources, satellite microVMs are suspended when they move out of the bounding box and re-activated when they come back into it. The underlying idea is that in edge computing, clients will want to use edge servers that are in their proximity, and distant edge servers will thus not need emulation. To give an example, in §4, we use the bounding box to only emulate satellites over North Africa, where our clients are located, to save resources. Celestial also helps the user configure their bounding box in a manner that makes sure that available resources meet the demand from the emulation based on per-microVM resources and bounding box area. Note that this bounding box does not affect network path calculation, as the shortest network path between two ground stations may not follow the line of sight. This allows algorithms, applications, and platforms for large LEO satellite constellations to be tested cost-efficiently on a minimal subset of servers before moving to an emulation of the entire constellation.

4 DEPLOYING AN EDGE APPLICATION ON CELESTIAL

To evaluate Celestial's performance, accuracy, and scalability, we first deploy an example LEO edge application, namely a multi-user interaction between users in West Africa as presented by Bhattacherjee et al. [9]. In this example, which we illustrate in Fig. 3, three users located in Accra, Ghana; Abuja, Nigeria; and Yaoundé, Cameroon require a common meetup-server for their application. While their nearest available cloud data center is located in Johannesburg, South Africa, using a satellite server reduces the RTT for the most distant of the three users from 46ms to only 16ms over SpaceX' phase I Starlink network.

We implement this as a WebRTC video conference where each participant sends high-definition video at 2.6Mb/s and receives a video stream from the other participants. An intermediary bridge server, our meetup service, duplicates each user's stream for all

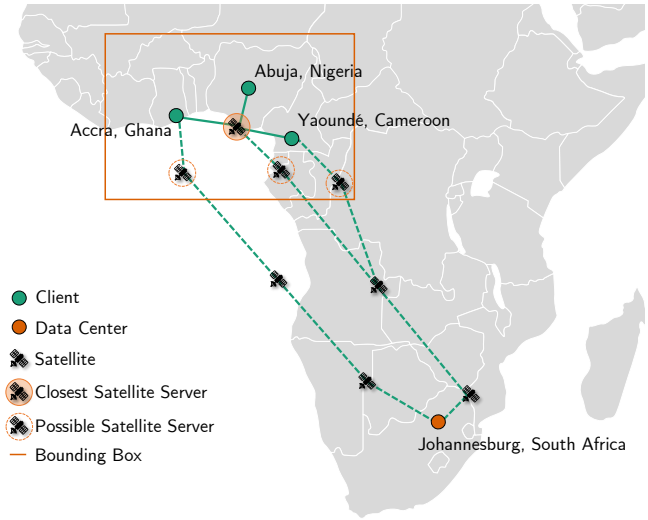


Figure 3: Three clients in Accra, Ghana; Abuja, Nigeria; and Yaoundé, Cameroon, require a common meetup-server for an interactive application. This meetup server may be located in a satellite server or the nearest cloud datacenter in Johannesburg, South Africa. Dashed lines indicate the additional hops needed to reach the cloud datacenter [9].

other users rather than each user sending multiple copies of their stream at the same time, which would put a considerable strain on their bandwidth. We can now compare two scenarios: First, we run the video bridge on the datacenter, which we assume to have an antenna to access the satellite network [65], which is the best-case scenario for latency to our clients. Second, we deploy a small tracking service to this datacenter that periodically checks the satellites in reach of our clients and instructs them to use the optimal satellite server based on combined latency as a video bridge server. As the video bridge for our real-time video conferencing use-case can be considered stateless, we do not take any migration costs into account. We discuss the issue of state management on the LEO edge in §7.7.

4.1 Experiment Setup

We assume both laser propagation in a vacuum and propagation of ground-to-satellite RF links at speed of light c [31, 64] and a 10Gb/s bandwidth connection for ISLs and radio links. We allocate four CPU cores and 4GB memory for each client and the tracking service, while each satellite server as well as the cloud video bridge are allocated two CPU cores and 512MB memory. To limit resource usage and the number of satellite servers the tracking service must consider, we draw a bounding box as shown in Fig. 3. To minimize the impact of clock drift in delay measurements between our clients, we schedule them to run on the same Celestial host and instruct them to use a shared *PTP* clock. In a preliminary baseline evaluation, we find that our clients and bridge server incur a 1.37ms median processing delay (3.86ms standard deviation) that we must take into account when comparing measured end-to-end latency and expected network distance. We expect our client measurement software, packet duplication, packet forwarding, and clock drift to cause this jitter.

Our experiments are conducted on three Google Cloud Platform `N2-highcpu` instances with 32 cores and 32GB memory each, placed in the *europe-west3-c* zone, connected in a private virtual network, and using an installation of Ubuntu 18.04. We measure a network delay of 0.2ms between those machines, yet this delay is already taken into account by Celestial when emulating microVM network distance. Although larger instances are available that may allow us to host our testbed on only a single cloud server, our goal is to show the horizontal scalability of Celestial. While Celestial estimates 137 required CPU cores given satellite density and bounding box size, we use only 96 CPU cores to test its over-provisioning capabilities. As our application actually uses resources on only one satellite server at a time, we expect only a small load on our Celestial servers. In addition, we run our Celestial Coordinator on a GCP `C2` instance with 16 cores and 64GB memory and choose an update interval of two seconds. Each experiment is ten minutes long and repeated three times to validate its reproducibility.

4.2 Results

We start by giving an overview of the results of our experiments by presenting the results of a randomly selected run. We can then validate the accuracy of the network conditions in our Celestial testbed by comparing it to the simulated network, analyze the reproducibility of our experiments on Celestial, and finally check resource efficiency.

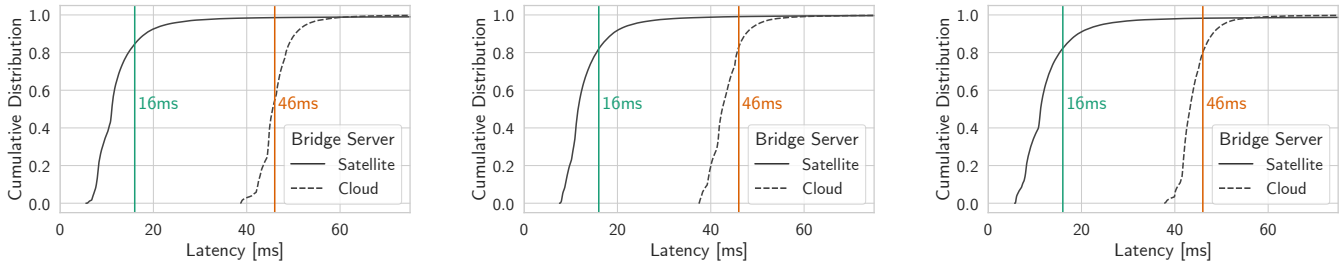
We show the cumulative distributions of measured end-to-end latency between client pairs in Fig. 4. For at least 80% of the duration of the video conference, end-to-end latency is below this maximum RTT of 16ms for the satellite servers and 46ms for the cloud server, confirming our expectation that satellite servers can provide a considerable QoS improvement for clients in latency-sensitive applications.

While available bandwidth is an unlikely factor for increased latency in the remaining 20% given the small size of our video stream, processing delay may add several milliseconds. Additionally, we cannot assume an optimal server to be selected at all times as, in some instances, the satellite constellation may change noticeably during the five-second update interval used in our application, e.g., with a particular uplink satellite becoming unavailable.

We further observe that only satellites in the two shells with the lowest altitude and highest density are ever selected as satellite servers, as they are more likely to have an optimal connection to all three clients at the same time. In deploying LEO edge infrastructure, it might thus make sense to start with the densest and lowest altitude shell.

Accuracy. To investigate whether the measured latency over time is accurate, we compare *expected* network latency as calculated by our tracking server, which includes the 1.37ms median processing delay. As an example, we show this comparison for the path from Abuja to Accra using the cloud server over the course of one evaluation in Fig. 5.

Both curves follow the same general trend and changes in calculated network latency are reflected in measured end-to-end latency. Again, we note that the tracking server only chooses a new server



(a) Accra to Abuja (b) Accra to Yaoundé (c) Abuja to Yaoundé

Figure 4: Our results show that the satellite servers can offer better QoS over the cloud data center, with only up to 15ms end-to-end latency for our users for 80% of the duration of their video conference.

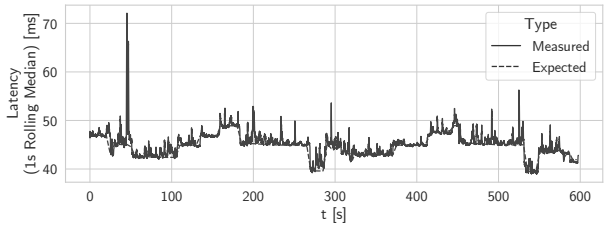


Figure 5: Measured and expected end-to-end latency from Abuja to Accra using the Johannesburg cloud datacenter. The expected value includes both simulated network distance and median processing delay of 1.37ms.

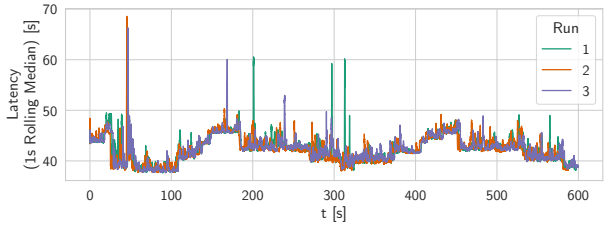


Figure 6: Measured end-to-end latency from Yaoundé to Abuja using the Johannesburg cloud datacenter across three repetitions of the experiment

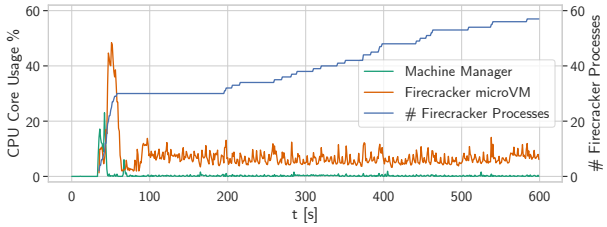


Figure 7: CPU usage on one Celestial host over the course of one experiment. In total, 32 CPU cores are available on the host.

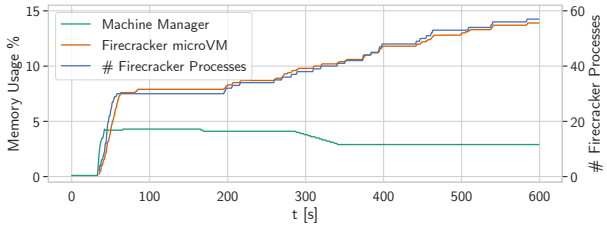


Figure 8: Memory usage on one Celestial host over the course of one experiment. In total, the host has 32GB available memory.

at the coarse interval of five seconds and intermittent changes in infrastructure that can cause spikes in end-to-end latency are thus not reproduced in expected network distance. Additionally, the jitter in our measurements is likely caused by processing as we observe it in our baseline measurements as well. We thus conclude that simulated network distances are accurately reflected in the emulated testbed.

Reproducibility. Furthermore, we can investigate whether our results are reproducible by looking at the results from the three repetitions we carry out for each experiment. As an example, we plot the measured end-to-end latency from Yaoundé to Abuja using the cloud server across the three repetitions in Fig. 6.

We can observe that results for all three runs follow the same trends and even the spike in measured latency after the first minute of the experiment can be reproduced. Since users can provide an arbitrary but firm starting point for their testbed emulation, Celestial offers a repeatable environment that enables reproducible tests and benchmarks.

Efficiency. Finally, in a separate test we trace the CPU and memory usage on our Celestial hosts for a glimpse into the resource efficiency of our testbed. Specifically, we look at the host under the highest load, which is that on which all our clients run for accurate time synchronization, in addition to a third of all satellite servers. We show its CPU and memory utilization in Figs. 7 and 8, respectively.

At the beginning, we see a spike in CPU usage caused by Celestial's Machine Manager as it starts setting up the host and network environment, followed by an even larger spike as Firecracker microVMs are starting to boot up at the beginning of the emulation run. CPU usage then decreases to below 5% as the clients prepare for the experiment, e.g., by synchronizing clocks and reading the workload traces. Then, during the experiment, we see total CPU usage from microVMs on the order of 10%. Considering that three clients and one tracking server with four allocated CPU cores each, and one active bridge server with two allocated cores run a demanding workload, and at least 25 additional satellite servers idle, we can conclude that Celestial is efficient on CPU resources as it can benefit from over-provisioning. Additionally, note that Celestial's Machine Manager itself consumes few CPU resources after the initial start up, an average of 0.2% with a slightly higher load every two seconds as the constellation is updated.

On the other hand, Celestial's Machine Manager uses up to 4.5% of the host's available memory from the start of the simulation, although that number decreases after the demanding initial setup. Firecracker microVM memory usage increases linearly with the number of booted microVMs, regardless of whether they are suspended or not, as each keeps a `virtio` memory device that blocks a fixed portion of the host's memory for the VM. As microVMs are only suspended when their corresponding satellites move out of the bounding box, their memory is not released. While this has not been an issue in any of our experiments, because microVM memory usage stays below 20% even on hosts with comparatively little available memory, Firecracker microVMs can also be configured to use ballooning to allow the host to reclaim unused memory from the VMs.

Finally, we also note the cost of our testbed: For our three hosts and one coordinator, a 10-minute experiment with an additional five minutes for setup and data collection yields a total cost of $3.30 on Google Cloud Platform. For comparison, creating 4,409 `f1-micro` virtual machine instances, with one for each satellite server, costs at least $539.66 for 15 minutes [28].

5 CASE STUDY: REAL-TIME OCEAN ENVIRONMENT ALERTS WITH REMOTE SENSORS

Using Celestial, we can empirically investigate the potential of the LEO edge for applications that may benefit from running at the network edge. One class of such applications are real-time monitoring services, e.g., in the context of industrial IoT or environment observation. The National Oceanic and Atmospheric Administration's (NOAA) *Deep-ocean Assessment and Reporting of Tsunamis* (DART) project uses remote sensing on stationary buoys located in the Pacific to detect early tsunami warning signs [27, 50]. Given their remote locations, these sensors cannot communicate over terrestrial networks, but instead use the Iridium LEO satellite constellation. Sensor data is sent to the Pacific Tsunami Warning Center on Ford Island, Hawaii, where it is processed centrally. This system's architecture is illustrated in Fig. 9.

We use a system inspired by DART to evaluate if and how LEO edge computing can assist real-time ocean environment monitoring for use cases such as Tsunami warning. In our experiments, 100

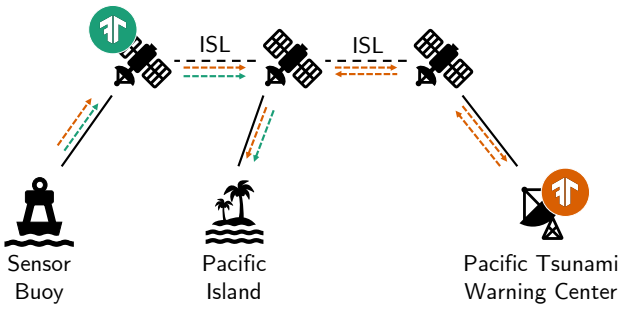


Figure 9: DART stations are remote buoys equipped with environment sensors, such as pressure recorders, that send their measurement data over the Iridium satellite constellation. Sensor data is processed with an LSTM neural network in a central processing location (orange) or on the LEO edge (green). Inferred data is sent to geographically close island ground stations and ships.

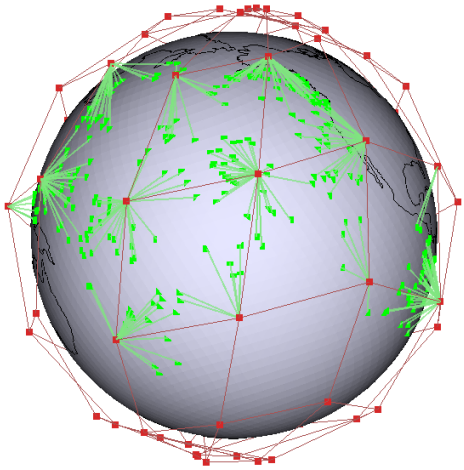


Figure 10: In our experiments, 100 data buoys in the Pacific Ocean send sensor data over the Iridium satellite network. Sensor readings are used for inference with a LSTM neural network and results are forwarded to 200 ships and islands. Ground stations and their connections are shown in bright green, satellites in red. As a result of the 180° arc of ascending nodes in the Iridium constellation, no ISLs exist between satellites of the first and last orbital plane of the constellation, as these satellites move in opposite directions.

data buoys in the Pacific Ocean transmit sensor readings over the Iridium satellite network. The Iridium satellite network has only one shell, 66 satellites in 6 planes at a 780km altitude, in a polar orbit (90° inclination) and spaced evenly only around half the globe (180° arc of ascending nodes) so that the other half is covered by satellites descending their orbit [18]. As a result of this spacing, the Iridium constellation cannot provide ISLs between the first and last orbital planes, which is reflected in our testbed. The readings,

grouped by sensor location and type, are used to predict weather and environmental events with a long short-term memory (LSTM) neural network [40]. Results are then distributed to ships and islands in the vicinity of the sensor, using a total of 200 locations. We show this topology in Fig. 10. We compare two different deployments of the inference service: First, we deploy it in a central ground station server at the location of the Pacific Tsunami Warning Center on Ford Island, Hawaii. Second, we deploy the inference service on each of the Iridium satellites, facilitating device-to-device communication.

5.1 Experiment Setup

All components of our experiments run on a Celestial testbed. Sensor data is sent at a one-second interval over UDP. Again, we use shared PTP clocks to minimize the impact of clock drift. Both sensors and data sinks are equipped with one CPU core and 1024MB memory. The inference service uses a TensorFlow stacked LSTM network. In the satellite deployment, satellite servers each have one CPU core and 1024MB memory, whereas we equip the ground station server with eight CPU cores and 8192MB memory in the datacenter deployment. Satellite link bandwidth is set at 88Kb/s for sensors and data sinks as recommended by Iridium for remote sensing applications [**?**]. ISLs and links to the central processing ground station are set at 100Mb/s.

All experiments are conducted on a cluster of four GCP `N2-highcpu` instances with 32 cores and 32GB memory each, placed in the *europe-west3-c* zone and using an installation of Ubuntu 18.04. In addition, the Coordinator is hosted on a GCP `C2` instance with 16 cores and 64GB memory, with an update interval of 5 seconds. Each experiment is 15 minutes long with an additional five-minute startup phase for the system to stabilize. We repeat each experiment three times and present results for the median runs.

5.2 Results

In Fig. 11, we show the mean end-to-end latency for the real-time ocean environment alert system. Overall, the satellite server deployment leads to a better performance compared to the centralized deployment. As a result of the shorter communication distances, end-to-end latency is reduced from between 22ms and 183ms to between 13ms and 90ms. Note that processing latency is similar between both deployments, at an average of 2ms. Further, Fig. 11a shows that ground stations that require data from the same sensors observe similar delays, a result of the device-to-device architecture.

The effect of the lack of ISLs between first and last orbital plane of the Iridium constellation can be seen in both deployments: Over the course of the experiment, locations in the West Pacific region (Asia and Oceania) tend to connect to uplink satellites from the last orbital plane, while those in the Americas connect to the first. Consequently, any requests between those locations must be routed through satellites near the poles, increasing the communication delay. In the central processing deployment, this leads to a higher observed latency in this area over the course of the experiment, as all sensor data are routed in this manner. In higher latitudes, which are nearer to the North Pole, this effect is not as pronounced. With satellite servers, however, data is only routed across shorter geographical distances. Increased communication delays can thus only be observed when sensor station and data sink connect to opposite planes of the constellation, which happens less frequently when both are geographically close.

5.3 Opportunities and Implications for LEO Edge Applications

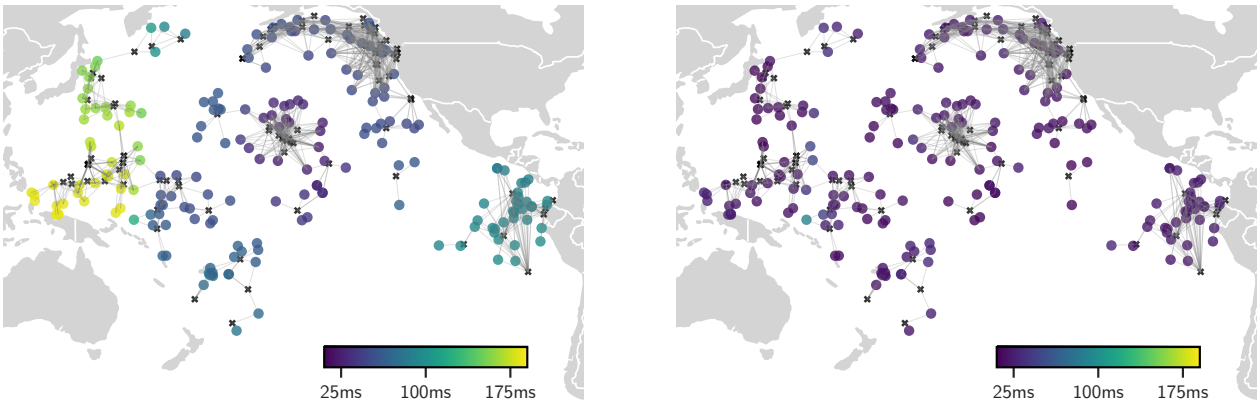
The results of our experiments show that LEO edge computing can improve QoS for remote-sensing applications where communication delay is the limiting factor. Without sending all data to a central processing facility, such as a cloud datacenter, and instead processing it on the communication path, significant latency improvements can be achieved. For this machine-to-machine communication use-case, on-device processing could not yield such an improvement: In case of processing on data buoys, sensor data would need to be sent back to buoys of the same group, incurring additional delays. Processing directly at the data consumers, i.e., island or ship ground stations, requires performing the same computation on several machines.

Please note that our experiment assumed rather small data volumes being sent: Since communication delay is the main cost factor in end-to-end latency, larger data volumes will increase the latency difference further. Also, larger data volumes will at some point make it inefficient to transmit all data for centralized processing – similar to bandwidth limits in terrestrial edge computing [52]. Similar to terrestrial edge computing, however, centralized processing may be more efficient for scenarios with low data volumes that can tolerate the higher end-to-end latency.

6 CASE STUDY: LEO EDGE DEPLOYMENT FOR A WEB APPLICATION

Using Celestial, we can empirically investigate the potential of the LEO edge for web applications, which can typically benefit from running at the network edge to provide lower response times for clients. As an example web application, we use *Twissandra*, a Twitter clone that uses a stateless Python web service backed by a Cassandra database, as shown in Figure 12 [24, 29]. Users interact with Twissandra in a browser by reading and posting tweets, while the web service queries the database and generates web pages with server-side rendering. Cassandra is a decentralized and distributed database that can be deployed across multiple hosts and it is accessed over the *CQL native binary protocol* [47].

In our experiments, clients in 120 European cities interact with Twissandra over the proposed Kuiper network, as shown in Figure 13. The Kuiper constellation comprises three shells: 1,156 satellites in 34 planes at a 630km altitude and 51.9° inclination; 1,296 satellites in 36 planes at a 610km altitude and 42° inclination; and 784 satellites in 28 planes at a 590km altitude and 33° inclination [43]. We compare two different deployments of Twissandra: First, both the web service and Cassandra are deployed in a single datacenter, located centrally in Frankfurt, Germany. Second, we deploy only Cassandra in that datacenter and run an instance of the Twissandra web service on each satellite server in the first shell of the Kuiper constellation. Clients in Europe are not in reach of the other two shells during large parts of the day given their low orbit inclination.



(a) Deployment with Central Processing Location

(b) Satellite Server Deployment on Iridium Constellation

Figure 11: Mean observed end-to-end latency for two different deployments: Colored circles show data sinks, crosses show remote sensor locations, and gray lines indicate data paths. A greater communication distance to the central processing location also increases the communication delay (Fig. 11a), while a satellite server deployment results in uniform load distribution (Fig. 11b). In both deployments, the lack of ISLs between first and last orbital plane of the Iridium constellation increases the communication delay towards the West Pacific region.

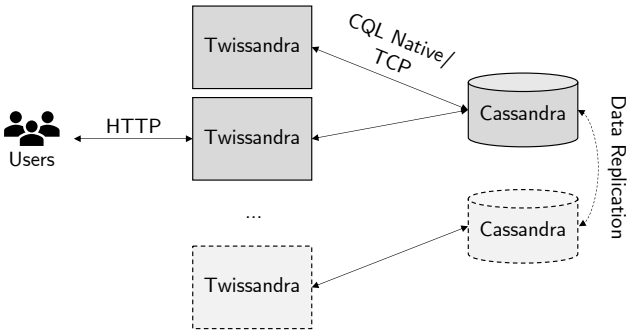


Figure 12: Clients access Twissandra through a stateless web service that is backed by a distributed Cassandra database.

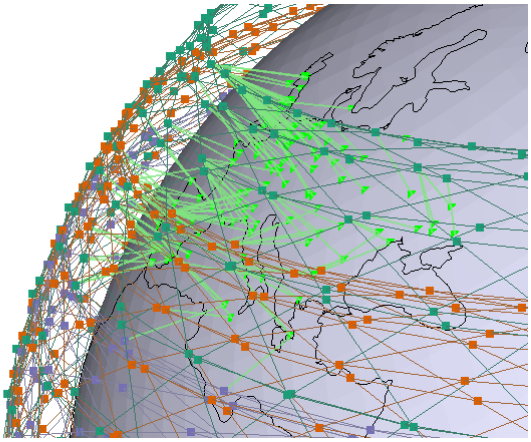


Figure 13: In our experiments, clients in 120 European cities (bright green) access the Kuiper network to interact with the Twissandra web application. The first shell with a 51.9° inclination (turquoise) can be accessed by clients in all of Europe except parts of Scandinavia, which we omit from our experiments. The shell at a 42° inclination (orange) is accessible only in more southern cities, while that at a 33° (purple) is inaccessible to almost all client locations.

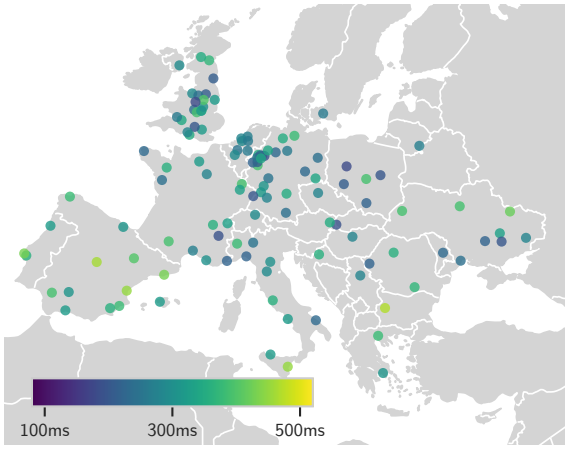
6.1 Experiment Setup

All components of our experiments run inside our Celestial testbed. Clients run an HTTP load generator that measures web service response times and are equipped with one CPU core and 512MB memory. In the satellite server deployment, clients run a local proxy that proxies requests to the nearest satellite server. In both deployments, Cassandra has eight CPU cores and 8192MB memory available. In the datacenter deployment, we equip the web service server with eight CPU cores and 8192MB memory, whereas each of the satellite servers in the satellite deployment has one CPU core and 1024MB memory.

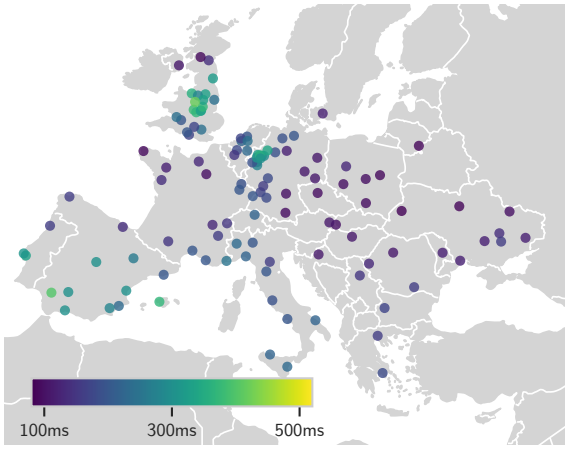
All experiments are conducted on a cluster of four GCP `N2-highcpu` instances with 32 cores and 32GB memory each, placed in the *europe-west3-c* zone and using an installation of Ubuntu 18.04. In addition, the Coordinator is hosted on a GCP `C2` instance with 16 cores and 64GB memory, with an update interval of 5 seconds. Each experiment is 15 minutes long with an additional five minute startup phase for the system to stabilize. We repeat each experiment three times and present results for the median runs.

6.2 Results

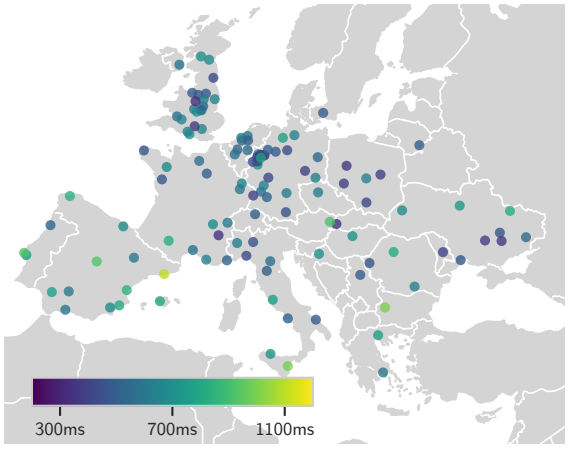
In Figure 14, we show the median response times for client requests to the Twissandra web application. Overall, `GET` requests for the public timeline take a median 300ms for the cloud deployment



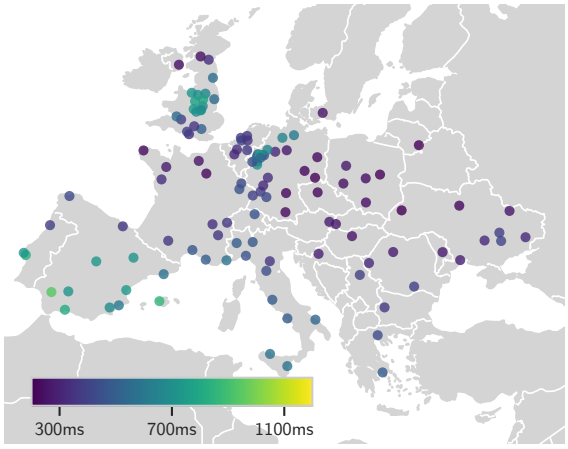
(a) Client GET Requests in Twissandra Datacenter Deployment



(b) Client GET Requests in Twissandra Satellite Deployment



(c) Client POST Requests in Twissandra Datacenter Deployment



(d) Client POST Requests in Twissandra Satellite Deployment

Figure 14: Median response times for client requests to Twissandra timeline for two different deployments: While the satellite deployment leads to a better overall performance, satellite servers can become a bottleneck when serving densely populated areas, as seen, e.g., for clients in the UK and west Germany.

and 180ms for the satellite deployment, a 40% improvement. Similarly, the average response time for a POST request to create a new tweet improves by 31% from 580ms to 400ms[10]. This improvement cannot necessarily be attributed to only the deployment topology of our web service. While the datacenter Twissandra server has more resources than the individual satellite servers, requests are naturally sharded across multiple satellite servers in the distributed deployment.

One consequence of this natural sharding is that satellite servers can become a bottleneck when serving areas that are densely populated with clients, as shown in Figures 14b and 14d. In the UK and west Germany, where multiple clients are located in close proximity and thus share common satellite uplinks, one satellite serving more clients leads to higher response times. On the other hand, clients in east Europe benefit from the satellite deployment: These areas are sparsely populated with clients and can thus use more resources on their nearest satellite server. In the centralized deployment, however, requests to the distant Frankfurt datacenter incur more network delays.

While the total network distance is identical for a client accessing the central database through the web service regardless of where that web service is deployed along the network path, distant clients measure a lower response delay when accessing a nearby datacenter. This can be explained in part by the overhead of opening TCP connections: The connection between the web service and database can be persisted, as the service will continuously query that database,

[10]Please note that these values are comparatively large for a simple web service as Twissandra is primarily implemented for educational and research purposes, consequently sacrificing some performance optimizations.

whereas clients incur the overhead of the 3-way handshake when starting a new session. When the distance of client to web server is smaller, as is the case when we deploy Twissandra on satellite servers, this overhead decreases as well. Additionally, in our satellite deployment, the longer leg of the transmission uses Cassandra's efficient CQL binary protocol format rather than HTTP.

6.3 Implications for LEO Edge Web Applications

Our experiments shed some light on the issues and opportunities involved in deploying web application on the LEO edge: Clients in sparsely populated areas can benefit from nearby web servers, while satellite servers can simultaneously become bottlenecks where many clients share a common uplink. For the LEO edge, this trade-off might actually be feasible if we expect satellite Internet to be more popular in sparsely populated rural areas, whereas densely populated urban areas can use terrestrial fiber. Alternatively, the skew in client density can be addressed by offloading client requests from the closest to nearby satellite servers as proposed by Bhattacherjee et al. [9].

Additionally, we may not be able to deploy all web services on all satellites as we have done in our evaluation. While this can lead to even more bottlenecks when less resources are available in total, we can also expect resource usage to average out across satellite servers as more clients share satellites. Placing services on satellite servers efficiently on the 2D torus network will be a future challenge in this field, akin to the facility location problem [3, 48].

We have also observed the impact of network communication overheads on the performance of our web application, as we have identified the TCP handshake, connection persistence, and the efficiency differences between the CQL native protocol as possible influences on the overall request delays. While the network stack has typically been less of an issue in traditional web applications, it will be a more important factor when planning a deployment on the LEO edge.

7 DISCUSSION & FUTURE WORK

In our evaluation of Celestial, we find that it fulfills our original requirements for a LEO edge testbed. In this section, we discuss threats to validity as well as limitations of our work and derive avenues for future work on Celestial and the LEO edge.

7.1 Limits to Scalability

In our experiments we show that Celestial is horizontally scalable across multiple servers and that we can run an application in a realistic environment of large LEO edge infrastructure. We use both over-provisioning of our hosts and a bounding box to achieve this in a cost-efficient manner, but there may be use-cases where neither is an option. When emulation of the entire infrastructure is needed and all satellite and ground station servers run at their full capacity, host infrastructure must be sized appropriately, e.g., requiring 36TB of memory when emulating a full constellation of 4,400 satellites with 8GB memory each.

While Celestial is designed to scale out across as many hosts as necessary, we must assume that such scale comes with caveats. The network connection between hosts could, for example, become a bottleneck if inter-satellite communication bandwidth exceeds available physical bandwidth. Some of these effects could be mitigated in Celestial by dynamically migrating satellite server microVMs across hosts to optimize communication and resource provisioning or by using a more advanced scheduler such as *FirePlace* [4].

7.2 Resource Isolation in microVM Collocation

Choosing microVMs over alternative technologies allows Celestial to provide an application-agnostic runtime environment while also achieving cost-efficiency through collocation of different machines on one physical server. Nevertheless, that collocation can also come at a price if resources are not appropriately available and microVMs start competing for, e.g., CPU cycles. While satellite servers are independent of each other, with each server placed on an individual satellite, microVMs on one host may affect each other's performance if their processes are scheduled on the same physical host CPU core. In practice, these effects cannot be easily circumvented without attaching microVM processes to host CPU cores strictly, which limits scalability as it inhibits over-provisioning of hosts. However, we see that this can only become a larger problem once resources required by satellite servers exceed the host's resources, which can be mitigated by scaling Celestial hosts vertically through additional CPU cores or memory, or horizontally by adding further host machines.

7.3 Impact of microVM Suspension

Celestial's bounding box allows for a smaller testbed footprint by suspending microVMs of satellite servers that move outside a specified area. In our example application, we have seen how this reduces the load on the Celestial hosts without impacting the application. There may be some cases where microVM suspension has unwanted side effects on the user's application. If the satellite server software is expected to change its state based on its location, e.g., a CDN service that needs to proactively replicate files before it is in reach of a ground station, it will not be able to do so while the machine is suspended and will potentially have to catch up to missed updates once it is activated again. To avert this, a user may increase the size of their bounding box, possibly to cover the entire earth so that no microVM is ever suspended. Note that this requires more host resources and hence increases cost.

7.4 Assumptions on Hardware Architecture

To work on commodity hardware, Celestial assumes satellite servers to have the same hardware architecture as common terrestrial servers, i.e., the *x86-64* architecture supported by Firecracker. In reality, we cannot currently know the hardware which will be used on the LEO edge. Specific hardware features such as special instructions or real-time guarantees can thus not be emulated on Celestial at the moment and would require full hardware virtualization or at least emulation of a subset of instructions. Still, even if future LEO edge operators were to go the unlikely path of developing a unique compute infrastructure for satellite servers, Celestial could still provide a realistic environment to evaluate algorithms and programming models.

7.5 Emulating External Factors

Although Celestial can emulate satellite server degradation, e.g., caused by radiation, there are many other external factors that can impact satellite constellations. To give one example, satellites may not always follow completely deterministic orbits. Starlink's satellites are equipped with ion thrusters to adjust their orbits, which can be necessary to dodge space debris or other satellites [2, 26]. This has effects on the network as well, as physical distances change or ISLs can become unavailable during such maneuvers.

Adverse weather conditions can have an effect on the radio link between a ground station and satellites, with radio dishes overheating [13] and rain causing refraction of radio waves [59]. Ground station equipment may also be mobile, e.g., if installed on a plane or car, which must be taken into account when selecting uplink satellites. Such factors may impact LEO edge applications, and it can be helpful to test their effects ahead of application deployment.

The separation of Constellation Calculation and Machine Managers in Celestial will allow researchers to incorporate new models quickly, as only the calculation component must be extended. The resulting networking and machine parameters can then be sent to the host machines without modification. Additionally, `tc` and `tc-netem` offer advanced network emulation features that can be used in Celestial in the future with only small changes to its codebase, such as packet loss or duplication, delay distributions, packet corruption, or packet reordering. Whether these emulated characteristics will be useful for LEO edge testbeds depends on future research on LEO satellite network measurements.

While no efficiently emulated testbed can ever be fully accurate with respect to the environment it emulates, we believe that the current feature set of Celestial accurately reflects what is known about the LEO edge today. As future research on LEO satellite networks will likely yield additional insights on the effects of external factors, Celestial can easily be adapted or extended in all possible directions thanks to its small and simple codebase.

7.6 Lack of Validation Data

With Celestial, one of our goals is to emulate the LEO edge accurately, yet the only method to verify this is to compare our measurements with simulation results. What the research community is still missing are realistic data traces for large LEO satellite constellations, as network operators guard their data to keep a competitive edge. The recent *SatNetLab* proposal for a research platform for global satellite-based networks [62] may change this in the future, and we hope to adapt Celestial's network calculation as new data becomes available.

7.7 Future Work on LEO Edge Systems

With its scale and dynamic topology, the LEO edge poses significant challenges for applications. To simplify the deployment of services, systems researchers should address these challenges with application platforms that abstract from the underlying infrastructure. The number of satellite servers will require new coordination approaches to guarantee service consistency. Further, the servers' limited resources must be allocated efficiently, while they are possibly even shared by multiple tenants [57].

The high mobility of LEO satellites introduces the additional challenge of state management: Clients will frequently need to connect to a new satellite edge server, and any server-side state must be migrated accordingly. Bhattacherjee et al. [9] have proposed the concept of "virtual stationarity", where such state is migrated between satellites based on their locations relative to Earth, so that data appears to be in the same location from a client perspective. Further, such state management requires routing the clients' requests to the correct satellite server, which is made more difficult by the dynamic network topology.

Celestial itself does not include any strategies for state management, request routing, or service management, as it is intended as a testbed on which future systems implementing such strategies can be evaluated.

7.8 Feasibility of the LEO Edge

Finally, Celestial's utility depends heavily on the future of the LEO edge. While considerable resources are committed to the development of large LEO satellite networks to provide global Internet coverage, LEO edge computing has so far only received limited attention from research and industry. If current trends in this field do not come to fruition and satellite network operators do not see sufficient financial incentives in operating LEO compute infrastructure, Celestial can only be used for theoretical evaluation.

Nevertheless, we also see this as an opportunity to discover those incentives: Celestial makes it possible to easily evaluate possible LEO edge applications and to bootstrap the development of LEO edge infrastructure.

8 CONCLUSION

In this paper, we have motivated the need for a testbed for the LEO edge that enables systems researchers, application developers, and platform designers to test and evaluate their LEO edge computing software on Earth. With Celestial, we have answered how such a testbed can be built for accuracy, in an application-agnostic manner, and with cost-efficiency and scalability in mind. In support of those claims, we have used Celestial to deploy a LEO edge application from the existing body of research. Additionally, we have shown in a case study how we use Celestial to empirically evaluate the potential of the LEO edge for remote sensing applications and for web applications.

Finally, we have laid out interesting avenues for future work: With Celestial, researchers will now be able to build and test platforms and systems for the LEO edge that address the challenges of state management, resource allocation, or request routing. To further improve the accuracy of emulated testbeds, experiences and data of real LEO edge deployments are needed. We hope that Celestial, which we have published as open-source, can be of great use in this further research on and development of LEO edge computing.

ACKNOWLEDGMENTS

We thank our anonymous reviewers, our anonymous shepherd, and our colleague Dr. Jonathan Hasenburg for their valuable feedback on this work. Funded by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) – 415899119. This

work is supported by the Google Cloud Research Credits program (GCP202443755) and by the AWS Cloud Credit for Research program.

REFERENCES

[1] Alexandru Agache, Marc Brooker, Alexandra Iordache, Anthony Liguori, Rolf Neugebauer, Phil Piwonka, and Diana-Maria Popa. 2020. Firecracker: Lightweight Virtualization for Serverless Applications. In *Proceedings of the 17th USENIX Symposium on Networked Systems Design and Implementation (NSDI '20)*. 419–434.

[2] Salvatore Alfano, Daniel Oltrogge, Holger Krag, Klaus Merz, and Robert Hall. 2021. Risk Assessment of Recent High-Interest Conjunctions. *Acta Astronautica* 184 (2021), 241–250.

[3] Babatunde Azeez, Hogil Kim, Yuho Jin, and Eun Jung Kim. 2006. I/O Node Placement for Performance and Reliability in Torus Networks. In *International Conference on Parallel and Distributed Computing and Systems (PCDS2006), IASTED*.

[4] Bharathan Balaji, Christopher Kakovitch, and Balakrishnan Narayanaswamy. 2021. FirePlace: Placing Firecraker Virtual Machines with Hindsight Imitation. In *Proceedings of Machine Learning and Systems (MLSys 2021)*. 652–663.

[5] Daniel Balasubramanian, Abhishek Dubey, William R. Otte, William Emfinger, Pranav S. Kumar, and Gabor Karsai. 2014. A Rapid Testing Framework for a Mobile Cloud. In *Proceedings of the 2014 25th IEEE International Symposium on Rapid System Prototyping*. 128–134.

[6] Ilja Behnke, Lauritz Thamsen, and Odej Kao. 2019. Héctor: A Framework for Testing IoT Applications Across Heterogeneous Edge and Cloud Testbeds. In *Proceedings of the 12th IEEE/ACM International Conference on Utility and Cloud Computing Companion (UCC '19)*. 15–20.

[7] David Bermbach, Frank Pallas, David García Pérez, Pierluigi Plebani, Maya Anderson, Ronen Kat, and Stefan Tai. 2018. A Research Perspective on Fog Computing. In *Proceedings of the 2nd Workshop on IoT Systems Provisioning and Management for Context-Aware Smart Cities (ISYCC 2018)*. 198–210.

[8] Debopam Bhattacherjee, Waqar Aqeel, Ilker Nadi Bozkurt, Anthony Aguirre, Balakrishnan Chandrasekaran, P Brighten Godfrey, Gregory Laughlin, Bruce Maggs, and Ankit Singla. 2018. Gearing up for the 21st Century Space Race. In *Proceedings of the 17th ACM Workshop Hot Topics in Networks (HotNets '18)*. 113–119.

[9] Debopam Bhattacherjee, Simon Kassing, Melissa Licciardello, and Ankit Singla. 2020. In-orbit Computing: An Outlandish thought Experiment?. In *Proceedings of the 19th ACM Workshop Hot Topics in Networks (HotNets '20)*. 197–204.

[10] Debopam Bhattacherjee and Ankit Singla. 2019. Network Topology Design at 27,000 km/hour. In *Proceedings of the 15th International Conference on Emerging Network Experiments And Technologies (CoNEXT '19)*. 341–354.

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

[12] Flavio Bonomi, Rodolfo Milito, Jiang Zhu, and Sateesh Addepalli. 2012. Fog Computing and Its Role in the Internet of Things. In *Proceedings of the 1st Edition of the MCC Workshop on Mobile Cloud Computing (MCC '12)*. 13–16.

[13] Jon Brodkin. 2021. Starlink dishes go into "thermal shutdown" once they hit 122° Fahrenheit. https://arstechnica.com/information-technology/2021/06/starlink-dish-overheats-in-arizona-sun-knocking-user-offline-for-7-hours/. Accessed: 2021-6-20.

[14] Owen Brown, Paul Eremenko, and Paul Collopy. 2009. Value-Centric Design Methodologies for Fractionated Spacecraft: Progress Summary from Phase I of the DARPA System F6 Program. In *Proceedings of the AIAA SPACE 2009 Conference & Exposition*.

[15] Arthur C. Clarke. 1945. Exta-Terrestrial Relays – Can Rocket Stations Give World-wide Radio Coverage? *Wireless World* LI, 10 (1945), 305–308.

[16] ns–3 Consortium. 2011. ns–3 Discrete Event Simulator. https://www.nsnam.org/. Accessed: 2021-9-3.

[17] Antonio Coutinho, Fabiola Greve, Cassio Prazeres, and Joao Cardoso. 2018. Fogbed: A Rapid-Prototyping Emulation Environment for Fog Computing. In *Proceedings of the 2018 IEEE International Conference on Communications (ICC)*. 1–7.

[18] Olivier de Weck, Uriel Scialom, and A Siddiqui. 2004. Optimal Reconfiguration of Satellite Constellations with the Auction Algorithm. In *Proceedings of the 22nd AIAA International Communications Satellite Systems Conference & Exhibit 2004 (ICSSC)*.

[19] Edsger W. Dijkstra. 1959. A Note on Two Problems in Connexion with Graphs. *Numer. Math.* 1, 1 (1959), 269–271.

[20] Jason A. Donenfeld. 2017. Wireguard: Next Generation Kernel Network Tunnel. In *Proceedings of the 2017 Network and Distributed System Security (NDSS)*. 1–12.

[21] Scott Eisele, Geoffrey Pettet, Abhishek Dubey, and Gabor Karsai. 2017. Towards an Architecture for Evaluating and Analyzing Decentralized Fog Applications. In *Proceedings of the 2017 IEEE Fog World Congress (FWC)*. 1–6.

[22] Hewlett Packard Enterprise. 2021. HPE Spaceborne Computer. https://www.hpe.com/us/en/compute/hpc/supercomputing/spaceborne.html. Accessed: 2021-11-8.

[23] ETSI, GS MEC. 2022. *V3.1.1: Multi-Access Edge Computing (MEC); Framework and Reference Architecture.* Technical Report. Sophia Antipolis CEDEX, France: European Telecommunications Standards Institute.

[24] Eric Florenzano, Tyler Hobbs, Eric Evans, and Jon Hermes. 2019. Twissandra. https://github.com/twissandra/twissandra.

[25] Robert W. Floyd. 1962. Algorithm 97: Shortest Path. *Commun. ACM* 5, 6 (1962), 345.

[26] Jeff Foust. 2019. SpaceX's space-Internet woes: Despite technical glitches, the company plans to launch the first of nearly 12,000 satellites in 2019. *IEEE Spectrum* 56, 1 (2019), 50–51.

[27] Frank I. Gonzalez, Hank M. Milburn, Eddie N. Bernard, and Jean C. Newman. 1998. Deep-Ocean Assessment and Reporting of Tsunamis (DART): Brief Overview and Status Report. In *Proceedings of the International Workshop on Tsunami Disaster Mitigation*. 19–22.

[28] Google Cloud. 2022. Compute Engine Pricing. https://cloud.google.com/compute/all-pricing. Accessed: 2022-03-12.

[29] Rachid Guerraoui, Matej Pavlovic, and Dragos-Adrian Seredinschi. 2016. Incremental Consistency Guarantees for Replicated Objects. In *12th USENIX Symposium on Operating Systems Design and Implementation (OSDI 16)*. 169–184.

[30] Harshit Gupta, Amir Vahid Dastjerdi, Soumya K Ghosh, and Rajkumar Buyya. 2017. iFogSim: A Toolkit for Modeling and Simulation of Resource Management Techniques in the Internet of Things, Edge and Fog Computing Environments. *Software: Practice and Experience* 47, 9 (2017), 1275–1296.

[31] Mark Handley. 2018. Delay is Not an Option: Low Latency Routing in Space. In *Proceedings of the 17th ACM Workshop Hot Topics in Networks (HotNets '18)*. 85–91.

[32] Mark Handley. 2019. Using Ground Relays for Low-Latency Wide-Area Routing in Megaconstellations. In *Proceedings of the 18th ACM Workshop Hot Topics in Networks (HotNets '19)*. 125–132.

[33] Jonathan Hasenburg, Martin Grambow, and David Bermbach. 2021. MockFog 2.0: Automated Execution of Fog Application Experiments in the Cloud. *IEEE Transactions on Cloud Computing* (2021).

[34] Jonathan Hasenburg, Martin Grambow, Elias Grünewald, Sascha Huk, and David Bermbach. 2019. MockFog: Emulating Fog Computing Infrastructure in the Cloud. In *Proceedings of the First IEEE International Conference on Fog Computing 2019 (ICFC 2019)*. 144–152.

[35] Jonathan Hasenburg, Sebastian Werner, and David Bermbach. 2018. FogExplorer. In *Proceedings of the 19th International Middleware Conference, Demos, and Posters (Middleware 2018)*. 1–2.

[36] Jonathan Hasenburg, Sebastian Werner, and David Bermbach. 2018. Supporting the Evaluation of Fog-based IoT Applications During the Design Phase. In *Proceedings of the 5th Workshop on Middleware and Applications for the Internet of Things (M4IoT 2018)*. 1–8.

[37] Raoufeh Hashemian, Niklas Carlsson, Diwakar Krishnamurthy, and Martin Arlitt. 2019. WoTbench: A Benchmarking Framework for the Web of Things. In *Proceedings of the 9th International Conference on the Internet of Things (IoT 2019)*. 1–4.

[38] Raoufehsadat Hashemian, Niklas Carlsson, Diwakar Krishnamurthy, and Martin Arlitt. 2020. Contention Aware Web of Things Emulation Testbed. In *Proceedings of the ACM/SPEC International Conference on Performance Engineering (ICPE '20)*. 246–256.

[39] Felix R. Hoots and Ronald L. Roehrich. 1980. *Models for Propagation of NORAD Element Sets.* Technical Report. Aerospace Defense Command Peterson AFB CO Office of Astrodynamics.

[40] Rong Hu, Fangxin Fang, Christopher C. Pain, and Ionel Michael Navon. 2019. Rapid Spatio-Temporal Flood Prediction and Uncertainty Quantification Using a Deep Learning Method. *Journal of Hydrology* 575 (2019), 911–920.

[41] Jiridiumcertus Iridium Communications Inc. [n. d.]. Iridium Certus®100. https://www.iridium.com/services/iridium-certus-100/. Accessed: 2022-03-04.

[42] Harry Jones. 2018. The Recent Large Reduction in Space Launch Cost. In *Proceedings of the 48th International Conference on Environmental Systems*.

[43] Simon Kassing, Debopam Bhattacherjee, André Baptista Águas, Jens Eirik Saethre, and Ankit Singla. 2020. Exploring the "Internet from Space" with Hypatia. In *Proceedings of the ACM Internet Measurement Conference (IMC '20)*. 214–229.

[44] Hemani Kaushal and Georges Kaddoum. 2017. Optical Communication in Space: Challenges and Mitigation Techniques. *IEEE Communications Surveys Tutorials* 19, 1 (2017), 57–96.

[45] Benjamin Kempton and Anton Riedl. 2021. Network Simulator for Large Low Earth Orbit Satellite Networks. In *Proceedings of the 2021 IEEE International Conference on Communications (ICC)*. 1–6.

[46] Steve Koontz, Robert Suggs, John Alred, Erica Worthy, Courtney Steagall, William Hartman, Benjamin Gingras, William Schmidl, and Paul Boeder. 2018. The International Space Station Space Radiation Environment: Avionics systems performance in low-Earth orbit Single Event Effects (SEE) environments. In *Proceedings of the 48th International Conference on Environmental Systems*.

[47] Avinash Lakshman and Prashant Malik. 2010. Cassandra: A Decentralized Structured Storage System. *Operating Systems Review* 44, 2 (2010), 35–40.

[48] Marilynn Livingston and Quentin F Stout. 1990. Perfect dominating sets. *Congressus Numerantium* 79 (1990), 187–203.

[49] Ruben Mayer, Leon Graser, Harshit Gupta, Enrique Saurez, and Umakishore Ramachandran. 2017. EmuFog: Extensible and Scalable Emulation of Large-Scale Fog Computing Infrastructures. In *Proceedings of the 2017 IEEE Fog World Congress (FWC)*. 1–6.

[50] Christian Meinig, Scott E. Stalin, Alex I. Nakamura, and Hugh B. Milburn. 2005. *Real-Time Deep-Ocean Tsunami Measuring, Monitoring, and Reporting System: The NOAA DART II Description and Disclosure.* Technical Report. NOAA, Pacific Marine Environmental Laboratory (PMEL).

[51] Joseph Nedeau, Dan King, Denise Lanza, Ken Hunt, and Lester Byington. 1998. 32-bit Radiation-Hardened Computers for Space. In *Proceedings of the 1998 IEEE Aerospace Conference Proceedings (Cat. No.98TH8339)*. 241–253 vol.2.

[52] Frank Pallas, Philip Raschke, and David Bermbach. 2020. Fog Computing as Privacy Enabler. *IEEE Internet Computing* 24, 4 (2020), 15–21.

[53] Tobias Pfandzelter and David Bermbach. 2020. tinyFaaS: A Lightweight FaaS Platform for Edge Environments. In *Proceedings of the 2020 IEEE International Conference Fog Computing (ICFC 2020)*. 17–24.

[54] Tobias Pfandzelter and David Bermbach. 2021. Edge (of the Earth) Replication: Optimizing Content Delivery in Large LEO Satellite Communication Networks. In *Proceedings of the 21st IEEE/ACM International Symposium on Cluster, Cloud and Internet Computing (CCGrid 2021)*. 565–575.

[55] Tobias Pfandzelter and David Bermbach. 2022. Celestial: Virtual Software System Testbeds for the LEO Edge. In *Proceedings of the 23rd International Middleware Conference (Middleware '22)*.

[56] Tobias Pfandzelter and David Bermbach. 2022. QoS-Aware Resource Placement for LEO Satellite Edge Computing. In *Proceedings of the 6th IEEE International Conference on Fog and Edge Computing (ICFEC 2022)*.

[57] 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 (EdgeSys 2021)*. 43–48.

[58] Tereza Pultarova. 2015. Telecommunications – Space tycoons go head to head over mega satellite Network [News Briefing]. *Engineering & Technology* 10, 2 (2015), 20–20.

[59] Ahmad Safaai-Jazi, Haroon Ajaz, and Warren L. Stutzman. 1995. Empirical Models for Rain Fade Time on Ku- and Ka-Band Satellite Links. *IEEE Transactions on Antennas Propagation* 43, 12 (1995), 1411–1415.

[60] Michael Sheetz. 2021. Elon Musk blasts Jeff Bezos' Amazon, alleging effort to 'hamstring' SpaceX's Starlink satellite internet. https://www.cnbc.com/2021/01/26/elon-musk-blasts-jeff-bezos-amazon-competitor-to-spacexs-starlink-.html. Accessed: 2021-2-24.

[61] Weisong Shi and Schahram Dustdar. 2016. The Promise of Edge Computing. *Computer* 49, 5 (2016), 78–81.

[62] Ankit Singla. 2021. SatNetLab: a call to arms for the next global Internet testbed. *ACM SIGCOMM Computer Communication Review* 51, 2 (2021), 28–30.

[63] Amelia Williamson Smith. 2019. Supercomputing in Space: HPE's Spaceborne Computer Returns After a Successful 1.5-Year Mission on the ISS National Lab. https://www.issnationallab.org/iss360/supercomputing-in-space-hpes-spaceborne-computer-returns/. Accessed: 2021-11-8.

[64] Reginald L. Smith-Rose. 1950. The Speed of Radio Waves and Its Importance in Some Applications. *Proceedings of the Institute of Radio Engineers* 38, 1 (1950), 16–20.

[65] Jennifer Sokolowsky. 2020. Azure Space Partners Bring Deep Expertise to New Venture. https://news.microsoft.com/transform/azure-space-partners-bring-deep-expertise-to-new-venture/. Accessed: 2022-03-08.

[66] Peter Stauning, Pål Davidsen, and Mathias Cyamukungu. 2004. Detection of Radiation-Induced Anomalies in the Memory Circuits of the Ørsted LEO Satellite. In *Proceedings of the 35th COSPAR Scientific Assembly*. 3974.

[67] Chia-Jiu Wang. 1993. Structural Properties of a Low Earth Orbit Satellite Constellation – the Walker Delta Network. In *Proceedings of MILCOM '93 - IEEE Military Communications Conference*. 968–972 vol.3.

[68] Shangguang Wang, Qing Li, Mengwei Xu, Xiao Ma, Ao Zhou, and Qibo Sun. 2021. Tiansuan Constellation: An Open Research Platform. *preprint* (2021).

[69] Ben Zhang, Nitesh Mor, John Kolb, Douglas S Chan, Ken Lutz, Eric Allman, John Wawrzynek, Edward Lee, and John Kubiatowicz. 2015. The Cloud is Not Enough: Saving IoT from the Cloud. In *Proceedings of the 7th USENIX Workshop Hot Topics in Cloud Computing (HotCloud '15)*.

[70] Zhenjiang Zhang, Wenyu Zhang, and Fan-Hsun Tseng. 2019. Satellite Mobile Edge Computing: Improving QoS of High-Speed Satellite-Terrestrial Networks Using Edge Computing Techniques. *IEEE Networks* 33, 1 (2019), 70–76.