Serverless Abstractions for Edge Computing in Large Low-Earth Orbit Satellite Networks

Tobias Pfandzelter
TU Berlin & ECDF
Berlin, Germany
tp@mcc.tu-berlin.de

ABSTRACT

Private and public actors are building massive LEO satellite communication networks. Researchers have proposed extending edge computing to satellites for global low-latency access to application services for, e.g., IoT or metaverses. Building applications for this LEO edge means managing services at large scale on highly dynamic infrastructure, in addition to the usual constraints of edge computing.

We seek to develop serverless abstractions for LEO edge applications. We introduce virtual testbed tooling that allows researchers, students, and practitioners to become familiar with the unique characteristics of the LEO edge and develop, test, and benchmark real software in a cost-efficient manner. Further, we develop abstractions for state and data management in geo-distributed edge-to-cloud environments. We then integrate these abstractions with a lightweight FaaS platform to allow building stateful yet scalable applications on the LEO edge. Finally, we propose applications for LEO edge computing to guide the evaluation of our design.

CCS CONCEPTS

• **Information systems** → **Computing platforms**; • **Software and its engineering** → *Development frameworks and environments*; • **Computer systems organization** → *Distributed architectures.*

KEYWORDS

LEO satellite networks, edge computing, serverless

ACM Reference Format:
Tobias Pfandzelter. 2023. Serverless Abstractions for Edge Computing in Large Low-Earth Orbit Satellite Networks. In *Middleware Demos, Posters and Doctoral Symposium '23: Proceedings of the 24th International Middleware Conference Demos, Posters and Doctoral Symposium (Middleware Demos, Posters and Doctoral Symposium '23), December 11–15, 2023, Bologna, Italy.* ACM, New York, NY, USA, 4 pages. https://doi.org/10.1145/3626564.3629088

1 INTRODUCTION

Large low-Earth orbit (LEO) satellite constellations, such as those in development by SpaceX, OneWeb, and public space agencies, can provide global Internet access with high bandwidth and low latency.

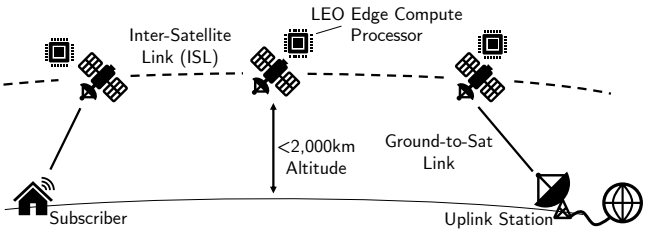

Figure 1: Subscribers access the Internet through the LEO network, connecting to the closest satellite, which in turn routes to an uplink station using ISLs. The LEO edge extends this with compute processors at each satellite, offering application services with lower latency and higher bandwidth.

Comprising thousands of satellites at altitudes below 2,000km connected with inter-satellite links (ISL), LEO satellite networks will extend the coverage of sixth generation (6G) mobile networks to remote and rural areas as well as ships and planes [5, 34].

A key consideration of this 6G integration is the extension of edge computing to LEO networks [36]. As shown in Figure 1, the *LEO edge* could provide low-latency, high-bandwidth application service access to a global subscriber base, supporting domains such as the Internet of Things (IoT), connected vehicles, and metaverses in AR/VR [6]. This is especially relevant as subscribers without sufficient access to terrestrial fiber are more likely to also lack cloud data centers in their proximity, given that cloud locations are predominantly located close to rich, urban areas [18].

From a software development perspective, building applications for terrestrial edge computing already poses significant challenges, from data management across geo-distributed locations to building scalable compute services on constrained edge hardware [4]. The LEO edge exacerbates this with two unique new challenges: First, the immense scale of LEO networks (the first generation SpaceX Starlink has more than 4,000 operational satellites [14]) means that applications services will need to be operated across thousands of geo-distributed, shared edge servers. Second, LEO satellites are highly mobile as a result of their low altitude. For example, a Starlink satellite (550km altitude) travels at speeds in excess of 27,000km/h, orbiting the Earth once every 95 minutes [7]. A subscriber on Earth will switch their uplink satellites every few minutes, meaning that LEO edge applications have to be migrated accordingly.

In this work, we propose using serverless abstractions to enable LEO edge computing. The main idea of serverless is to hide operational concerns from developers, providing a scalable, elastic, and flexible platform for applications [12, 13]. At the same time, operators benefit from being able to allocate their (possibly limited)

Middleware Demos, Posters and Doctoral Symposium '23, December 11–15, 2023, Bologna, Italy

resources at fine granularity. Serverless allows us to abstract away the dynamic and large-scale LEO edge while making the most out of the constrained resources. We make the following contributions:

- To better understand the characteristics of the LEO edge and to be able to evaluate LEO edge platform abstractions, we design a virtual testbed tool (§2).
- We propose a data management platform for geo-distributed edge computing with application-controlled replica placement (§3).
- We design a computing platform for the LEO edge based on the Function-as-a-Service (FaaS) paradigm (§4).
- Finally, we discuss possible LEO edge applications (§5).

2 LEO EDGE TESTBED

Research on and development of applications and software platforms for the LEO edge is hindered by two challenges: First, there is no existing LEO edge infrastructure to explore and test software on. And second, even if prototypical infrastructure is being developed [38], the barrier to entry is high, making development processes cumbersome and expensive.

Instead, researchers usually turn to virtual testbed tooling that allows running distributed software in emulation. Existing tooling for edge computing, e.g., *EmuFog* [17], *FogBed* [9], and *MockFog* [10, 11], is insufficient for LEO edge: First, LEO satellites are highly mobile, requiring us to change emulated network behavior at fine granularity, e.g., every second, which not all tools support. Second, satellite networks comprise thousands of satellites that could all serve as edge servers and which must thus all be emulated. Third, research on platforms and abstractions requires support of arbitrary software on the testbed, including isolation technologies such as Docker, which means that a testbed that itself relies on, e.g., containerization for isolation, is inherently too limited.

To meet all of these requirements, we propose *Celestial* [22, 24, 28], virtual testbed tooling for the LEO edge. Celestial combines a high-performance LEO satellite network simulation tool [15] with Firecracker microVMs [1]. With Firecracker, we are able to efficiently emulate many satellite edge servers on limited infrastructure, with support for scaling across multiple host servers. MicroVMs also allow us to run a full virtual machine for each edge server, including support for arbitrary Linux-based software. Further, we can leverage the homogenous nature of the LEO edge: As each LEO edge server must be identical, we implement an overlay file system in Celestial that allows us to give each virtual edge server access to a full-size file system without blocking more than a few hundred megabytes on the underlying hosts.

As most edge software serves a limited geographical area, e.g., clients in a specific country, we also introduce a 'bounding box' for experiments that only require emulating a subset of satellite servers. As satellites are highly mobile, we suspend servers that move outside this bounding box and restore them when moving back into the area of interest.

Future Work. Celestial is based on the standard Linux traffic control utility that allows us to emulate network delays and bandwidth constraints between network devices [8]. While this approach is used in most edge testbed tooling, our work on Celestial has revealed scalability issues when reaching thousands of emulated connections. We have thus developed a new approach for network emulation on Linux using extended Berkeley Packet Filters (eBPF) [2], and we plan to integrate this into Celestial in the future.

Further, we have developed a taxonomy of failure scenarios that could affect LEO edge software [26]. To allow developing resilient LEO edge software and platform abstractions, we plan to integrate failure emulation in Celestial.

3 GEO-DISTRIBUTED DATA STORE

The basic functionality of an edge network is low-latency, high-bandwidth access to local data replicas, e.g., content delivery networks (CDNs). For edge applications, regardless of whether they run on satellite or terrestrial edge infrastructure, managing data replication across geo-distributed edge nodes is challenging. For example, data replication requires long-running services at each node, managing data consistency, and coordination among nodes.

Instead of each application solving these issues anew, we propose a serverless geo-distributed data store. 'Serverless' here refers to the fact that the data store itself is run by some infrastructure provider, e.g., one that also provides the edge resources. Applications only configure replication policies and access their data using a scalable interface. *FReD* (Fog Replicated Data) [32] implements this using *keygroups*, logical data capsules. Keygroups are FReD's unit of replication, and applications simply specify at which location they want keygroup data replicated. As clients move, keygroups can thus move alongside by changing their replication policy.

Clients can access data in FReD using a simple key-value interface. We implement client-centric consistency guarantees using a local client library. Regardless of which FReD node the client accesses, the library caches seen data version information and ensures ordered reads and writes [3].

Future Work. Although our client library can detect version conflicts, it cannot automatically resolve them as correct conflict resolution depends on application logic. Instead, applications currently get all concurrent versions of data items for resolution, somewhat breaking the serverless abstractions. In future work, we plan to directly offer convergent and commutative replicated data types (CRDT) [35] in our library in order to let applications specify conflict resolution logic rather than implementing it themselves.

A further research area is coordination between nodes for control flow. FReD currently relies on a centralized naming service that provides a single source of truth about nodes, keygroups, and replication policies. Network partitions, which are likely in geo-distributed edge-to-cloud systems, can thus lead to node failures. We have instead proposed peer-to-peer coordination for FReD features, e.g., only replica nodes of a keygroup, which are likely to be geographically close, need to coordinate keygroup information [33].

4 SERVERLESS EDGE COMPUTE PLATFORM

Edge servers are more resource constrained than the cloud, with many single-node deployments across a geo-distributed area rather than a centralized data center. As a result, resource allocation at the edge between multiple tenants with varying resource requirements requires more fine-grained and elastic sharing. With scale-to-zero and allocation-per-invocation, the Function-as-a-Service (FaaS) paradigm is a good fit for addressing these challenges [19]: In FaaS,

developers compose their applications of small, stateless functions that are invoked by an underlying platform in response to external events. The platform is responsible for scaling services elastically with demand, allowing developers to focus on application logic rather than operational concerns. Operators can allocate resources dynamically and bin-pack tenant services more efficiently [12, 13].

Existing FaaS platforms focus on cloud-scale deployments, using complex dependencies such as Kubernetes [16]. We thus propose *tinyFaaS* [20], a lightweight FaaS platform designed specifically for single-node deployment at the edge. By removing dependencies such as Kubernetes and multi-node load balancers from the invocation hot-path, tinyFaaS can achieve a smaller footprint, increase throughput, and decrease request-response latency, which is of utmost importance for latency-constrained edge applications.

While a lightweight FaaS platform allows efficient deployment of stateless edge applications by multiple tenants, a key inhibitor to broad FaaS adoption is that stateful applications must still rely on external data stores. While manageable in the cloud (although not necessarily cost-effective [31]), incorporating a cloud database in an edge FaaS application removes the benefits of edge computing. A local edge database is more suitable, yet managing data replication across geo-distributed edge FaaS deployments is cumbersome. With *Enoki* [27] we integrate tinyFaaS with FReD, building a combined serverless platform for geo-distributed edge deployments that supports stateful applications. In Enoki, each node runs both a tinyFaaS instance and FReD node. When deploying a function, Enoki injects a database connection into the function handler and creates a FReD keygroup. If the function already exists on another node, the keygroup is simply replicated. Function code can thus rely on state using a key-value interface. Replication across edge nodes can lead to data staleness but ensures that all requests are made to a local copy of the data, without breaking FaaS transparency.

Future Work. The benefits of FaaS for the edge also apply to LEO edge computing. Here, an additional benefit of the high level of abstraction and separation of state and compute service is that service migration can be implemented easily [30]. Using 'virtual stationarity', i.e., migrating compute services to counteract the orbital movement of satellites, edge services can remain in proximity to the clients they serve without explicit support by the application.

Consequently, we plan to orchestrate Enoki deployments on the LEO edge for virtual stationarity, autonomously redeploying functions and keygroup replicas as satellites move. We will use Celestial testbeds to conduct our experiments.

We also plan to evaluate more sophisticated migration strategies: We have proposed an algorithm for service placement on LEO networks based on service level objectives using the unique 2D torus topologies of ISLs [23]. Such a placement for function and data replicas may uncover trade-offs between application service migration cost and the detriment of larger service access latency.

5 LEO EDGE APPLICATIONS

As one of the most widespread edge applications, CDNs serve local copies of web content, e.g., images or stylesheets, from points-of-presence (PoP) in proximity of clients rather than clients having to pull them from far-away origin locations. Traditional CDN topologies leverage the hierarchical nature of the Internet, placing PoPs in tier 2 or tier 3 regional networks [37]. In the flat hierarchy of LEO networks, where all global subscribers share an access network, it may instead make sense to place PoPs directly on satellites, serving content just one hop from clients. We have proposed to use simple migration techniques that counteract orbital movements to ensure web content is served only in a limited geographical area [21].

While applications of the LEO edge may be limited to clients without access to terrestrial fiber, an interesting future application of our research is a Mars compute cloud [25]: For sparse human settlement on Mars, satellite-based networking is more cost-efficient than networks on the ground given satellite networks' planet-wide coverage and high landing costs. A Mars satellite network could also serve as a local edge compute cluster, supporting, e.g., Mars-to-Mars communication, sensor networks, or industrial IoT.

Future Work. The applications of LEO edge computing must be kept in mind when evaluating how well serverless abstractions and compute platforms can solve its salient challenges. Similarly, we have previously argued that the quantitative evaluation of edge compute and data management platforms must be application-centric [29]. We plan to extend our survey of LEO edge applications in the future in order to define meaningful quality metrics and evaluation scenarios. We must also evaluate how applications that are latency-constrained yet not event-driven, e.g., live video communication, can be implemented with serverless abstractions.

6 CONCLUSION

The integration of large LEO satellite networks with edge computing raises new challenges for application developers, including high network dynamics and extraordinary scale and geo-distribution. We have proposed serverless abstractions for data and compute service management on the LEO edge, lowering complexity for developers and increasing flexibility and efficiency for operators. We have further introduced virtual LEO edge testbed tooling for researchers, students, and practitioners to build and evaluate software and platforms without access to physical satellite infrastructure. Finally, we have proposed possible applications of LEO edge computing that will guide the evaluation of our proposed abstractions.

ACKNOWLEDGMENTS

I thank my advisor Prof. Dr.-Ing. David Bermbach for his continuing support and guidance. This work was supported by the Bundesministerium für Bildung und Forschung (16KISK183) and the Deutsche Forschungsgemeinschaft (415899119).

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)*. USENIX Association, Berkeley, CA, USA, 419–434.

[2] Soeren Becker, Tobias Pfandzelter Pfandzelter, Nils Japke, David Bermbach, and Odej Kao. 2022. Network Emulation in Large-Scale Virtual Edge Testbeds: A Note of Caution and the Way Forward. In *Proceedings of the 2nd International Workshop on Testing Distributed Internet of Things Systems* (Asilomar, CA, USA) *(TDIS 2022)*. IEEE, New York, NY, USA, 1–7. https://doi.org/10.1109/IC2E55432.2022.00007

[3] David Bermbach, Jörn Kuhlenkamp, Bugra Derre, Markus Klems, and Stefan Tai. 2013. A Middleware Guaranteeing Client-Centric Consistency on Top of Eventually Consistent Datastores. In *Proceedings of the 1st IEEE International Conference on Cloud Engineering* (San Francisco, CA, USA) *(IC2E 2013)*. IEEE, New York, NY, USA, 114–123. https://doi.org/10.1109/IC2E.2013.32

[4] David Bermbach, Frank Pallas, David García Pérez, Pierluigi Plebani, Maya Anderson, Ronen Kat, and Stefan Tai. 2017. A Research Perspective on Fog Computing. In *Proceedings of the 2nd Workshop on IoT Systems Provisioning & Management for Context-Aware Smart Cities* (Malaga, Spain) *(ISYCC 2017)*. Springer, Cham, Switzerland, 198–210. https://doi.org/10.1007/978-3-319-91764-1_16

[5] Debopam Bhattacherjee, Waqar Aqeel, Ilker Nadi Bozkurt, Anthony Aguirre, Balakrishnan Chandrasekaran, Brighten P. 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* (Redmond, WA, USA) *(HotNets '18)*. Association for Computing Machinery, New York, NY, USA, 113–119. https://doi.org/10.1145/3286062.3286079

[6] 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* (Virtual Event, USA) *(HotNets '20)*. Association for Computing Machinery, New York, NY, USA, 197–204. https://doi.org/10.1145/3422604.3425937

[7] 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* (Orlando, FL, USA) *(CoNEXT '19)*. Association for Computing Machinery, New York, NY, USA, 341–354. https://doi.org/10.1145/3359989.3365407

[8] Martin A. Brown. 2006. *Traffic Control HOWTO*. Technical Report 102. The Linux Documentation Project. https://tldp.org/HOWTO/pdf/Traffic-Control-HOWTO.pdf

[9] 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* (Kansas City, MO, USA) *(ICC '18)*. IEEE, New York, NY, USA, 1–7. https://doi.org/10.1109/ICC.2018.8423003

[10] 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* 11, 1 (April 2021), 58–70. https://doi.org/10.1109/TCC.2021.3074988

[11] 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* (Prague, Czech Republic) *(ICFC 2019)*. IEEE, New York, NY, USA, 144–152. https://doi.org/10.1109/ICFC.2019.00026

[12] Scott Hendrickson, Stephen Sturdevant, Tyler Harter, Venkateshwaran Venkataramani, Andrea C. Arpaci-Dusseau, and Remzi H. Arpaci-Dusseau. 2016. Serverless Computation with OpenLambda. In *Proceedings of the 8th USENIX Workshop on Hot Topics in Cloud Computing* (Denver, CO, USA) *(HotCloud '16)*. USENIX Association, Berkeley, CA, USA.

[13] Eric Jonas, Johann Schleier-Smith, Vikram Sreekanti, Chia-Che Tsai, Anurag Khandelwal, Qifan Pu, Vaishaal Shankar, Joao Carreira, Karl Krauth, Neeraja Yadwadkar, et al. 2019. *Cloud Programming Simplified: A Berkeley View on Serverless Computing*. Technical Report 20193. EECS Department, University of California, Berkeley, Berkeley, CA, USA. https://www2.eecs.berkeley.edu/Pubs/TechRpts/2019/EECS-2019-3.html

[14] 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* (Virtual Event, USA) *(IMC '20)*. Association for Computing Machinery, New York, NY, USA, 214–229. https://doi.org/10.1145/3419394.3423635

[15] 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* (Montreal, QC, Canada) *(ICC)*. IEEE, New York, NY, USA, 1–6. https://doi.org/10.1109/ICC42927.2021.9500439

[16] Junfeng Li, Sameer G. Kulkarni, K. K. Ramakrishnan, and Dan Li. 2019. Understanding open source serverless platforms: Design considerations and performance. In *Proceedings of the 5th International Workshop on Serverless Computing* (Davis, CA, USA) *(WoSC '19)*. Association for Computing Machinery, New York, NY, USA, 37–42. https://doi.org/10.1145/3366623.3368139

[17] 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* (Santa Clara, CA, USA) *(FWC '17)*. IEEE, New York, NY, USA, 1–6. https://doi.org/10.1109/FWC.2017.8368525

[18] Nitinder Mohan, Lorenzo Corneo, Aleksandr Zavodovski, Suzan Bayhan, Walter Wong, and Jussi Kangasharju. 2020. Pruning Edge Research with Latency Shears. In *Proceedings of the 19th ACM Workshop Hot Topics in Networks* (Virtual Event, USA) *(HotNets '20)*. Association for Computing Machinery, New York, NY, USA, 182–189. https://doi.org/10.1145/3422604.3425943

[19] Tobias Pfandzelter and David Bermbach. 2019. IoT Data Processing in the Fog: Functions, Streams, or Batch Processing?. In *Proceedings of the 1st Workshop on Efficient Data Movement in Fog Computing* (Prague, Czech Republic) *(DaMove 2019)*. IEEE, New York, NY, USA, 201–206. https://doi.org/10.1109/ICFC.2019.00033

[20] Tobias Pfandzelter and David Bermbach. 2020. tinyFaaS: A Lightweight FaaS Platform for Edge Environments. In *Proceedings of the Second IEEE International Conference on Fog Computing* (Sydney, NSW, Australia) *(ICFC 2020)*. IEEE, New York, NY, USA, 17–24. https://doi.org/10.1109/ICFC49376.2020.00011

[21] 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* (Melbourne, Australia) *(CCGrid '21)*. IEEE, New York, NY, USA, 565–575. https://doi.org/10.1109/CCGrid51090.2021.00066

[22] Tobias Pfandzelter and David Bermbach. 2022. Celestial: Virtual Software System Testbeds for the LEO Edge. In *Proceedings of the 23rd ACM/IFIP International Middleware Conference* (Quebec, QC, Canada) *(Middleware '22)*. Association for Computing Machinery, New York, NY, USA, 69–81. https://doi.org/10.1145/3528535.3531517

[23] 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* (Taormina, Italy) *(ICFEC '22)*. IEEE, New York, NY, USA, 66–72. https://doi.org/10.1109/ICFEC54809.2022.00016

[24] Tobias Pfandzelter and David Bermbach. 2022. *Testing LEO Edge Software Systems with Celestial*. Technical Report 20221. TU Berlin & ECDF, Mobile Cloud Computing Research Group, Berlin, Germany.

[25] Tobias Pfandzelter and David Bermbach. 2023. Can Orbital Servers Provide Mars-Wide Edge Computing?. In *Proceedings of the 1st ACM MobiCom Workshop on Satellite Networking and Computing* (Madrid, Spain) *(SatCom '23)*. Association for Computing Machinery, New York, NY, USA, 7–12. https://doi.org/10.1145/3570361.3614239

[26] Tobias Pfandzelter and David Bermbach. 2023. Edge Computing in Low-Earth Orbit – What Could Possibly Go Wrong?. In *Proceedings of the the 1st ACM Workshop on LEO Networking and Communication 2023* (Madrid, Spain) *(LEONET '23)*. Association for Computing Machinery, New York, NY, USA, 19–24. https://doi.org/10.1145/3614204.3616106

[27] Tobias Pfandzelter and David Bermbach. 2023. Enoki: Stateful Distributed FaaS from Edge to Cloud. (Sept. 2023). arXiv:2309.03584

[28] Tobias Pfandzelter and David Bermbach. 2023. Evaluating LEO Edge Software in the Cloud with Celestial. In *Proceedings of the 11th IEEE International Conference on Cloud Engineering* (Boston, MA, USA) *(IC2E '23)*. IEEE, New York, NY, USA, 224–225. https://doi.org/10.1109/IC2E59103.2023.00034

[29] Tobias Pfandzelter and David Bermbach. 2023. Towards a Benchmark for Fog Data Processing. In *Proceedings of the 11th IEEE International Conference on Cloud Engineering* (Boston, MA, USA) *(IC2E '23)*. IEEE, New York, NY, USA, 92–98. https://doi.org/10.1109/IC2E59103.2023.00018

[30] 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* (Online, United Kingdom) *(EdgeSys '21)*. Association for Computing Machinery, New York, NY, USA, 43–48. https://doi.org/10.1145/3434770.3459736

[31] Tobias Pfandzelter, Sören Henning, Trever Schirmer, Wilhelm Hasselbring, and David Bermbach. 2022. Streaming vs. Functions: A Cost Perspective on Cloud Event Processing. In *Proceedings of the 10th IEEE International Conference on Cloud Engineering* (Asilomar, CA, USA) *(IC2E 2022)*. IEEE, New York, NY, USA, 67–78. https://doi.org/10.1109/IC2E55432.2022.00015

[32] Tobias Pfandzelter, Nils Japke, Trever Schirmer, Jonathan Hasenburg, and David Bermbach. 2023. Managing Data Replication and Distribution in the Fog with FReD. *Software: Practice and Experience* 53, 10 (Oct. 2023), 1958–1981. https://doi.org/10.1002/spe.3237

[33] Tobias Pfandzelter, Trever Schirmer, and David Bermbach. 2022. Towards Distributed Coordination for Fog Platforms. In *Proceedings of the 22nd IEEE/ACM International Symposium on Cluster, Cloud and Internet Computing, Posters* (Taormina, Italy) *(CCGrid 2021)*. IEEE, New York, NY, USA, 760–762. https://doi.org/10.1109/CCGrid54584.2022.00087

[34] Imadur Rahman, Sara Modarres Razavi, Olof Liberg, Christian Hoymann, Henning Wiemann, Claes Tidestav, Paul Schliwa-Bertling, Patrik Persson, and Dirk Gerstenberger. 2021. 5G evolution toward 5G advanced: An overview of 3GPP releases 17 and 18. *Ericsson Technology Review* 2021, 14 (Oct. 2021), 2–12. https://doi.org/10.23919/ETR.2021.9904665

[35] Marc Shapiro, Nuno Preguiça, Carlos Baquero, and Marek Zawirski. 2011. *A comprehensive study of Convergent and Commutative Replicated Data Types*. Technical Report 7506. Institut National de Recherche en Informatique et en Automatique (INRIA), Paris, France. https://hal.inria.fr/inria-00555588

[36] Weisong Shi and Schahram Dustdar. 2016. The Promise of Edge Computing. *Computer* 49, 5 (May 2016), 78–81. https://doi.org/10.1109/MC.2016.145

[37] Marten van Steen and Andrew S. Tanenbaum. 2023. *Distributed Systems*. Marten van Steen, Enschede, The Netherlands.

[38] Shangguang Wang, Qing Li, Mengwei Xu, Xiao Ma, Ao Zhou, and Qibo Sun. 2021. Tiansuan Constellation: An Open Research Platform. In *Proceedings of the 2021 IEEE International Conference on Edge Computing* (Chicago, IL, USA) *(EDGE)*. IEEE, New York, NY, USA, 94–101. https://doi.org/10.1109/EDGE53862.2021.00022