Evaluating LEO Edge Software in the Cloud with CELESTIAL

Tobias Pfandzelter, David Bermbach
Technische Universität Berlin & Einstein Center Digital Future
Mobile Cloud Computing Research Group
{tp,db}@mcc.tu-berlin.de

***Abstract*—CELESTIAL is a toolkit for building LEO edge testbeds on cloud VMs. This allows testing and benchmarking LEO edge software systems such as applications and platforms without access to real LEO satellite infrastructure. CELESTIAL scales to emulate thousands of satellites servers on few cloud VMs using microVM technology. Its low footprint makes it cost-efficient for research and educational purposes without sacrificing flexibility and isolation. We show considerations for designing LEO edge software and demonstrate evaluating it on a CELESTIAL testbed.**

***Index Terms*—leo edge, edge testbed, software evaluation**

I. INTRODUCTION

Low-Earth orbit (LEO) satellite constellations will provide high-bandwidth broadband Internet access to a global subscriber base [1], [2]. These constellations comprise thousands of satellites orbiting Earth at high speeds, as shown in Fig. 1. Researchers have proposed integrating edge computing with LEO networks by deploying servers onto communication satellites in order to enable in-network data processing with low latency for a new class of Internet user [3]–[5].

While it can take many years and significant research and industry investment until the LEO edge becomes widely available, we must already evaluate if and how it may be used to support edge applications. To that end, we need virtual testbeds that allow us to test *real* software systems in a *scalable* manner with *cost-efficient* cloud resources.

II. CELESTIAL OVERVIEW

CELESTIAL is an open-source toolkit for running virtual LEO edge testbeds in the cloud [6], [7]. At its core is a coordinator that accurately simulates the satellite network and calculates machine positions and network characteristics. Host cloud VMs run low-footprint *Firecracker* microVMs [8] to emulate each satellite server and ground station. This allows emulating limited resources with more isolation than containers yet lower resource requirements than full VMs. Concurrently emulating hundreds of satellites on a single cloud VM is easily possibly and, if needed, emulation can scale out across multiple cloud VMs. CELESTIAL employs low-overhead eBPF-based network emulation to efficiently inject artificial delays and bandwidth limitation between satellites [9]. Software running in a CELESTIAL testbed can use an API to retrieve current constellation information in order to optimize for satellite server location or network capacity.

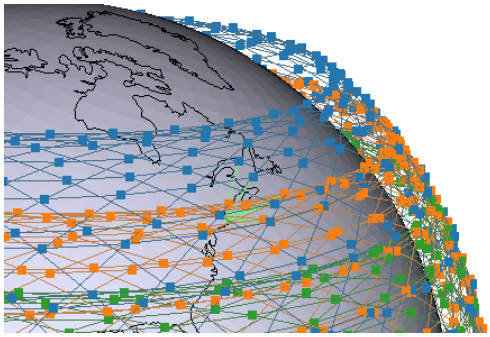

Fig. 1. The proposed Amazon Project Kuiper constellation comprises 3,236 satellites in three different orbits (orange, green, blue) [2]. A ground station in Boston, MA, USA and its uplink connections are shown in light green.

Users start CELESTIAL hosts as cloud VMs and provide simulation parameters and machine images for satellite servers. Simulation parameters include constellation design, satellite server hardware capabilities, and ground station location. CELESTIAL can generate animations to visualize characteristics for users. Machine images are full microVM disk images generated from a chosen Linux base system and application files. CELESTIAL includes tooling to generate such images from existing files, but users may choose, e.g., to enable additional Linux kernel features. This also allows flexibility and support for a wide array of software, including running container engines within CELESTIAL servers.

Finally, users can also provide a bounding box that limits emulation to a specified geographical area, as edge software is often only relevant for a single location. Instead of emulating thousands of satellite servers at the same time, only satellites that currently move over that location are started for software system evaluation, saving resources. LEO satellite mobility requires frequently starting and suspending servers as they move in and out of the bounding box, which CELESTIAL does automatically.

III. LEO EDGE SOFTWARE DESIGN

The key differences between terrestrial and LEO edge computing for software system design are *scale* and *mobility* [5].

Scale: LEO edge software systems must efficiently scale across thousands of identical compute nodes, i.e., the satellite servers. While LEO edge software can leverage the powerful satellite network for communication among nodes, it should

limit such communication in order to save costs and resources. Ideally, individual services are stateless in order to limit communication overhead needed for synchronization [5]. If necessary, a smaller number stateful nodes could be evenly distributed within the network [10].

Mobility: As a result of their low orbit, LEO satellites travel at high speeds in relation to Earth, e.g., 27,000km/h at 550km altitude [3]. A satellite flying over one geographic area can be on the other side of Earth within 30 minutes. This also means that ground stations on Earth change their uplink satellites frequently, on the order of minutes. LEO edge application services serving a specific geographic location have to anticipate and counteract this mobility, e.g., by frequently reading the state of the LEO constellation. When software is deployed in containers, those containers have to be migrated proactively between satellite servers. For stateless software, where multiple replicas can exist without conflicts, additional service copies can be started on other satellite servers that will reach the served location in the near future.

IV. Using Celestial

Celestial allows evaluating both mobility support and scalability of LEO edge software systems. A software system evaluation on Celestial encompasses the following tasks:

1) Planning: Evaluating software system requires defining quality metrics and evaluation scenarios, which have to be implemented with software deployment methods and load generators. While Celestial is merely a testbed and does not provide load generation tools, it is designed for broad compatibility by using Linux VMs.

2) Integration: Celestial provides APIs for reading the state of the emulated satellite constellation. Application need to integrate these APIs if this information is necessary, e.g., through a middleware.

3) Building Artifacts: Firecracker microVM images are necessary to run software on Celestial. Building these images requires software installation scripts that can be executed with our tooling. Further, the tested scenario has to be implemented in a Celestial configuration file, specifying, e.g., constellation parameters and ground station location.

4) Infrastructure Setup: To run a testbed, first start the necessary infrastructure. We show an example cloud configuration in Fig. 2: A coordinator server runs the network simulation and sends instructions to host servers running the `celestial.bin` host-side service. MicroVMs are distributed across two cloud hosts, where Celestial configures Firecracker and the network plane according to characteristics determined in live simulation.

5) Execution: Software execution begins as soon as Celestial is started and microVMs boot. While the evaluation is running, read-only access to microVM disk images and output is possible to monitor progress. Ending the emulation run also destroys all running microVMs.

6) Result Collection: Results can be collected from microVM disk images and terminal output at the end of the simulation. For example, a load generator may log service access latency to a local file that can be copied from the Celestial host.

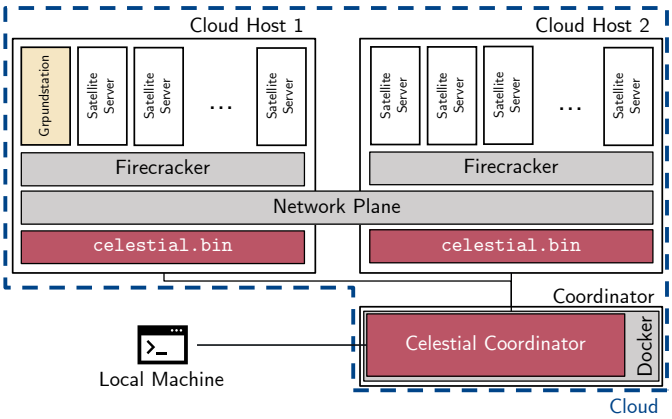

Fig. 2. Overview of a Celestial testbed in the cloud: a coordinator runs the network simulation and instructs the machines that host the celestial microVMs. On those hosts, a `celestial.bin` server configures Firecracker and the network plane accordingly.

7) Result Analysis: Finally, results can be analyzed using any suitable methodology. When evaluation service quality it is useful to compare results to the changing network characteristics of the LEO constellation in order to determine whether, e.g., access latency was caused by the service or the network.

V. Conclusion & Getting Started

Celestial is an open-source tool for evaluating LEO edge software systems with cost-efficient, scalable testbeds in the cloud. We provide extensive documentation online, including a quick start guide: https://openfogstack.github.io/celestial.

References

[1] D. Bhattacherjee, W. Aqeel, I. N. Bozkurt, A. Aguirre, B. Chandrasekaran, B. P. Godfrey, G. Laughlin, B. Maggs, and A. Singla, "Gearing up for the 21st century space race," in *Proc. HotNets '18*, Nov. 2018, pp. 113–119.

[2] S. Kassing, D. Bhattacherjee, A. B. Águas, J. E. Saethre, and A. Singla, "Exploring the "internet from space" with hypatia," in *Proc. IMC '20*, Oct. 2020, pp. 214–229.

[3] D. Bhattacherjee and A. Singla, "Network topology design at 27,000 km/hour," in *Proc. CoNEXT '19*, Dec. 2019, pp. 341–354.

[4] T. Pfandzelter and D. Bermbach, "Edge (of the earth) replication: Optimizing content delivery in large leo satellite communication networks," in *Proc. CCGrid '21*, May 2021, pp. 565–575.

[5] T. Pfandzelter, J. Hasenburg, and D. Bermbach, "Towards a computing platform for the leo edge," in *Proc. EdgeSys '21*, Apr. 2021, pp. 43–48.

[6] T. Pfandzelter and D. Bermbach, "Celestial: Virtual software system testbeds for the leo edge," in *Proc. MIDDLEWARE '22*, Nov. 2022, pp. 69–81.

[7] ——, "Testing leo edge software systems with Celestial," TU Berlin & ECDF, Mobile Cloud Computing Research Group, Tech. Rep. MCC.2022.1, Apr. 2022.

[8] A. Agache, M. Brooker, A. Iordache, A. Liguori, R. Neugebauer, P. Piwonka, and D.-M. Popa, "Firecracker: Lightweight virtualization for serverless applications," in *Proc. OSDI '20*, 2020, pp. 419–434.

[9] S. Becker, T. P. Pfandzelter, N. Japke, D. Bermbach, and O. Kao, "Network emulation in large-scale virtual edge testbeds: A note of caution and the way forward," in *Proc. TDIS '22*, Sep. 2022, pp. 1–7.

[10] T. Pfandzelter and D. Bermbach, "Qos-aware resource placement for leo satellite edge computing," in *Proc. ICFEC '22*, May 2022, pp. 66–72.