Università di Genova

PhD Program in Science and Technology for Electronic and Telecommunication Engineering

Design, Emulation, and Control of Edge Computing Systems for the Space Cloud in LEO Satellite Networks

Camilo José Rojas Milla

Thesis submitted for the degree of *Doctor of Philosophy* (38° cycle)

Feb 2026

Prof. Mario Marchese	Tutor
Prof. Juan Fraire	Tutor
Prof. Fabio Patrone	Co-tutor
Prof. Maurizio Valle	Head of the PhD program

Università di Genova
DITEN DIPARTIMENTO
DI INGEGNERIA NAVALE, ELETTRICA,
ELETTRONICA E DELLE TELECOMUNICAZIONI

Design, Emulation, and Control of Edge Computing Systems for the Space Cloud in LEO Satellite Networks

Camilo José Rojas Milla

⋈

Committee:

Reviewers

Kanglian Zhao, Full Professor, Nanjing University, China
Claudio Sacchi, Associate professor, University of Trento, Italy

Tutors

Mario Marchese, Full Professor, University of Genoa, Italy
Juan Fraire, Full Professor, Inria-INSA Lyon, France and University of Cordova, Argentina
Fabio Patrone, Assist Professor, University of Genoa, Italy

University of Genoa

Camilo José Rojas Milla

Design, Emulation, and Control of Edge Computing Systems for the Space Cloud in LEO Satellite Networks

xiv+90 p.

To Belen and Family.

Abstract

The advent of mega-constellations in Low Earth Orbit (LEO) has transformed satellite communications, enabling global connectivity and paving the way for integrating computational services in orbit. This thesis explores the concept of a *Space Cloud*, in which in-orbit infrastructures provide distributed computing capabilities beyond their traditional role as communication relays. While Multi-access Edge Computing (MEC) technologies have proven effective in terrestrial networks, their adaptation to dynamic and latency-sensitive space environments raises unresolved challenges in computing resource orchestration, including the *Edge Server Activation* (ESA) and *Edge Server Placement* (ESP) problems defined in this work. We address these challenges through models, software tools, and algorithms that couple orbital propagation with real-time network emulation to evaluate control and placement strategies under realistic conditions. Central to our contributions is *MeteorNet*, an open-source, continuous-time platform that integrates orbital dynamics, network virtualization, and containerized execution to reproduce operational network behavior using Docker, Mininet, and Software-Defined Networking (SDN) controllers. MeteorNet dynamically adapts link states and propagation delays to satellite motion and line-of-sight conditions, generating high-fidelity synthetic datasets for training and validating learning-based algorithms in space systems, where real data are scarce. For ESA, we develop two adaptive controllers—*Fuzzy Logic* and *Reinforcement Learning*—that implement decentralized policies based on each node's current state and historical performance data to minimize activation time while maintaining low task failure rates, outperforming Access and Bernoulli baselines across traffic regimes. For ESP, we introduce the *Farthest Point Sampling* (FPS) method, a topology-aware deployment strategy for space systems. We compare FPS with alternative space, terrestrial, and hybrid deployments, showing that it can reduce mean round-trip time by 15–20% and that, at sufficient infrastructure budgets, space-based configurations can surpass terrestrial clouds. Experiments conducted with *MeteorNet*, together with the results obtained for the MEC resource orchestration problems, confirm that intelligent orchestration and informed placement substantially outperform reactive or naive baselines in complex, dynamic constellations. Overall, this thesis lays the groundwork for scalable, autonomous, and latency-aware edge computing in satellite constellations by unifying mixed-integer formulations, learning-based ESA controllers, topology-aware ESP strategies, and a real-time emulation framework to advance the Space Cloud paradigm.

Keywords: Edge Computing, Satellite Constellations, Server Placement, Resource Orchestration.

Acknowledgements

I would like to express my sincere gratitude to Professor Mario Marchese for his guidance, continuous support, and insightful feedback throughout this research.

I am deeply grateful to Dr. Juan A. Fraire for his patience, generosity with his time, and dedicated mentorship. His thoughtful advice and steady encouragement were essential to the completion of this work.

I would also like to thank Professor Fabio Patrone for his assistance and support during my Ph.D. studies, and for the valuable exchange of ideas and thoughts.

I would also like to thank Professor Maurizio Valle, Head of the Ph.D. Program in Science and Technology for Electronic and Telecommunication Engineering, for his leadership and for fostering an excellent research environment.

To all my colleagues at the SCNL lab, thank you for the collaboration, discussions, and support that made the daily work both productive and enjoyable.

To my friends, colleagues, and mentors at the SPEL lab in Chile, thank you for always being present for collaboration and the exchange of ideas, regardless of distance.

To my family, I am grateful for your constant support and for being close in spirit during these years living abroad.

Finally, my deepest thanks go to my partner in life, Belén González. Her presence throughout this process has been invaluable. I will always be grateful for her help, her unconditional love, and the strength with which we have navigated both the ups and downs together. I am thankful for the family we have built and for the future ahead of us. An honorary mention also goes to our cats, Nipori and Flash, for their critical emotional support `(=^.^=)`.

Camilo J. Rojas
Genoa, 2026

Contents

List of Figures

List of Tables

Notation

Sets and Indices

$\mathcal{V}$	Set of constellation nodes (space servers / satellites), $\mathcal{V} = \{v_j \mid 1 \leq j \leq V\}$
$\mathcal{U}$	Set of user locations (ground segment), $\mathcal{U} = \{u_i \mid 1 \leq i \leq I\}$
$\mathcal{S}$	Set (or subset) of satellite nodes that may host MEC servers, $\mathcal{S} = \{s_n \mid 1 \leq n \leq N\}$
$\mathcal{G}$	Set of terrestrial gateways, $\mathcal{G} = \{g_m \mid 1 \leq m \leq M\}$
$\mathcal{L}$	Set of inter-satellite links (ISLs) used to build the constellation graph
T	Set of discrete time instants for emulation, $T = \{0, \Delta t, \ldots, \lvert T \rvert\}$
i, j, n, m, t	Indices for users (i), satellites/nodes (j), satellite servers (n), gateways (m), and time (t)

Orbit Topology and Kinematics

V, U, N, G	Counts of satellites (V), user locations (U), satellite servers (N), and gateways (G)
$\mathcal{P}$	Set of orbital planes in constellation
i_o	Orbital inclination angle
P_j^t	Satellite v_j position in TEME coordinates at time t
P_i^t	Ground user u_i position in TEME coordinates at time t
$d_{i,j}^t$	Shortest-path delay proxy between u_i and v_j at time t
$\theta_{i,j}^t$	Elevation angle of v_j as seen from u_i at time t
θ_c	Minimum elevation angle to declare line of sight (LoS) available
$\mathrm{LoS}_{i,j}^t$	Line-of-sight availability indicator between u_i and v_j at time t

Network Modeling

Δt	Duration of one emulation step (sampling period)
$h_{i,j}^t$	Hop count on the shortest path between u_i and v_j at time t
T_l	Per-link interface delay (forwarding component per hop)
T_p	Per-task processing time at the serving node
$T_{F_{i,j}}^t$	Forwarding delay between u_i and v_j at time t
δ_0	Fixed per-link processing/switching overhead added to propagation
c	Speed of light in vacuum (used in propagation delay model)
$\mathrm{CAP}_{i,j}^t$	Abstracted link capacity class for the u_i–v_j direct channel at time t
d_g	Additional latency from gateway to terrestrial cloud
H_j	Computational capacity of node v_j (e.g., lines/s or equivalent compute units)
λ_i^t	Task generation rate at user u_i at time t
Pr	Task size
K	Maximum number of servers to place in ESP

Decision Variables and States

$x_{i,j}^t$	Binary routing/offloading decision: 1 if u_i offloads to node v_j at time t, 0 otherwise
y_j^t	Binary activation state of the server at node v_j at time t (1 active, 0 inactive)
y_j	Binary placement decision for ESP: 1 if a server (or gateway) is placed at node v_j, 0 otherwise
S_j^t	Local state vector at node v_j and time t, $S_j^t = [B_j^t, C_j^t, D_j^t, K^t]$
α, β	Objective weights in ECO optimization (context-dependent; not to be confused with inclination α and elevation β)

Performance Metrics

$L_{i,j}^t$	End-to-end latency for a task from u_i served at v_j at time t
L_i^t	Latency of tasks generated by u_i at time t (after assignment)
L_{th}	Latency threshold for success; tasks with latency above L_{th} are failures
b_i^t	Binary success indicator for u_i at time t (1 if $L_i^t > L_{\text{th}}$, else 0)
D_j^t	Offload factor: number (or rate) of tasks redirected from node v_j to others at time t
C_j^t	Instantaneous CPU utilization at node v_j at time t
$\overline{C}$	Average utilization ratio across nodes and time
B_j^t	Number (or rate) of failed tasks associated with node v_j at time t
$\overline{B}$	Average failure rate across nodes and time
K^t	Number of active servers at time t
$\overline{K}$	Average activation ratio across time
$TL_{i,j}^t$	Total latency (transmit–process–return) for a task from u_i via v_j at time t
$D_{i,j}^t$	Path-dependent delay component ($d_{i,j}^t$ for space, $d_{i,j}^t + d_g$ for ground via gateway)
R_i^t	Round-Trip Time (RTT) for user u_i at time t
$\overline{R}$	Average RTT across users and time, load-weighted by λ_i^t

Controllers and Learning (FL/RL)

f_j^t	Continuous output of the Fuzzy Logic (FL) controller prior to hysteresis thresholding
P_d	Input fuzzy sets for variable $d \in \{1, 2, 3, 4\}$, typically $P_d = \{L_d, M_d, H_d\}$
Q	Output fuzzy sets for controller output, $Q = \{L_f, M_f, H_f\}$
$\mu_{(\cdot)}$	Membership functions for fuzzy sets (triangular for M, trapezoidal for L, H)
R_k	Linguistic rules for FL inference (e.g., $R_1, \ldots, R_6$)
T, J	Numerical operators used in FL implication/aggregation, ($T = J =$ min)
π_θ	Actor policy network parameterized by θ in the RL controller
V_θ	Critic value network parameterized by θ in the RL controller
r_j^t	Per-step reward assigned to node v_j at time t
r^*	Reward threshold used as a convergence criterion during RL training
ω_1, ω_{11}	Reward shaping weights, concerning B_j^t, in the RL formulation
ω_2	Reward shaping weights, concerning y_j^t, in the RL formulation
ω_3	Reward shaping weights, concerning D_j^t, in the RL formulation
ω_4	Reward shaping weights, concerning K^t, in the RL formulation

Acronyms

3GPP	Third Generation Partnership Project
5G	Fifth Generation Mobile Network
6G	Sixth Generation Mobile Network
AI	Artificial Intelligence
AO	Along Orbit (server placement distribution)
API	Application Programming Interface
CAP	Channel Capacity
CPU	Central Processing Unit
COTS	Commercial Off-The-Shelf
CP	Cross Plane (server placement distribution)
DB	Database
ECI	Earth-Centered Inertial
ESA	Edge Server Activation
ESP	Edge Server Placement
FL	Fuzzy Logic
FPS	Farthest Point Sample (distribution strategy)
GEO	Geostationary Earth Orbit
GMAT	General Mission Analysis Tool
GUI	Graphical User Interface
ILP	Integer Linear Programming
IoTs	Internet of Things
ISLs	Inter-Satellite Links
KPIs	Key Performance Indicators
LEO	Low Earth Orbit
LoS	Line of Sight

MEC	Multi-access Edge Computing
MI	Million Instructions
MIPS	Million Instructions per Second
ML	Machine Learning
MongoDB	Document-oriented NoSQL Database
NP	Nondeterministic Polynomial-time
NTNs	Non-Terrestrial Network
OF	OpenFlow
ONOS	Open Network Operating System
OS	Operating System
P-POD	Poly Picosatellite Orbital Deployer
QoS	Quality of Service
RL	Reinforcement Learning
RTT	Round Trip Time
SDN	Software-Defined Networking
SGP4	Simplified General Perturbations Model 4
SPD	Shortest Path Delay
STK	Systems Tool Kit
TCP/IP	Transmission Control Protocol / Internet Protocol
TEME	True Equator Mean Equinox
TLE	Two-Line Element
VM	Virtual Machine

CHAPTER 1

Introduction

The rapid evolution of satellite technology has redefined the landscape of global communications. Once the exclusive domain of state-funded space agencies and large telecommunications corporations, access to space has been democratized by advances in miniaturization, modularity, and cost-effective launch systems. Thousands of satellites now orbit the Earth in coordinated Low Earth Orbit (LEO) constellations, forming the backbone of a new generation of Non-Terrestrial Networks (NTNs). These systems promise to provide continuous, low-latency connectivity across the globe, extending network access to regions beyond the reach of terrestrial infrastructure. As satellite constellations grow in scale and capability, so does global connectivity. The challenge is no longer simply to connect, but to transform these orbital networks from passive data relays into intelligent, distributed computing systems capable of processing information directly in space.

1.1 Problem Definition and Motivation

Global LEO satellite constellations for telecommunications are now a common reality and have triggered a transformative shift in both the space industry and the communications sector. The adoption of standardized satellite platforms, Commercial Off-The-Shelf (COTS) components, and modular launch systems such as the Poly Picosatellite Orbital Deployer (P-POD) has significantly reduced the barriers to designing and deploying space missions. As a result, an increasing number of actors—ranging from established commercial operators to research institutions and emerging startups—can now deploy and operate sophisticated missions once monopolized by national agencies and large telecommunications companies.

Recent advancements in NTNs have been primarily driven by private companies deploying large-scale LEO constellations to provide global Internet coverage. Prominent examples include SpaceX's *Starlink* and *OneWeb*, which together account for thousands of operational satellites, with planned expansions into the tens of thousands. Forthcom-

ing constellations are expected to employ Inter-Satellite Links (ISLs) to connect nodes across and within orbital planes, extending coverage and providing continuous connections to remote or underserved areas where terrestrial infrastructure deployment would be prohibitively expensive or technically infeasible.

While these developments have established LEO constellations as critical enablers of global connectivity, they also present unique opportunities to extend network functionality beyond simple data relay. In particular, the integration of Multi-access Edge Computing (MEC) into LEO constellations has emerged as a promising paradigm to reduce end-to-end latency, improve bandwidth utilization, and enable time-sensitive applications directly in orbit. This *Space Cloud* vision reframes satellite constellations as distributed computation platforms capable of delivering processing services closer to users than terrestrial cloud infrastructures, thereby reducing dependence on gateway-based offloading for latency-sensitive workloads.

However, realizing the Space Cloud concept is far from trivial. Deploying edge-compute resources in orbit introduces several challenges absent from terrestrial networks, including routing in highly dynamic topologies, intermittent connectivity, strict energy budgets, and complex link-layer characteristics. The efficient allocation and orchestration of computational resources across a constellation require careful consideration of where servers should be placed, when they should be activated, and how tasks should be routed between satellites and ground segments. Poorly designed strategies may lead to excessive propagation delays, underutilized resources, or unnecessary energy consumption, ultimately undermining the benefits of in-orbit processing.

The problem is further aggravated by the lack of practical experimentation environments for evaluating these strategies at scale. Access to operational satellite constellations for testing is prohibitively costly and often restricted by commercial or regulatory constraints. This motivates the development of high-fidelity, real-time emulation platforms that can replicate the behavior of space networks while enabling reproducible experimentation with orchestration algorithms, network configurations, and server placement strategies.

In this context, this thesis addresses the core problem of **how to design, emulate, and control edge computing systems for LEO satellite networks**. Specifically, we focus on mechanisms that enable efficient task processing under constellation dynamics, balancing latency and resource usage under realistic networking constraints. By combining practical emulation with intelligent orchestration mechanisms, this work seeks to provide

actionable insights into the deployment and management of Space Cloud architectures for next-generation NTNs.

1.2 Research Objectives and Hypothesis

The integration of MEC into Low Earth Orbit (LEO) satellite constellations represents a paradigm shift in how satellite networks are conceived and operated. By enabling in-orbit processing, the *Space Cloud* concept has the potential to reduce latency, optimize bandwidth utilization, and support emerging time-sensitive services such as real-time environmental monitoring, autonomous vehicle coordination, and low-latency IoT applications. Nevertheless, achieving these benefits requires overcoming several technical challenges related to the orchestration, placement, and activation of edge servers in highly dynamic non-terrestrial environments.

The central objective of this thesis is to design, emulate, and evaluate control strategies for MEC-enabled LEO constellations that minimize latency and energy consumption while maintaining high task completion rates. To meet this goal, we identify the following specific objectives:

1. **Model the MEC-LEO satellite environment** by defining the relevant network, computational, and orbital parameters that affect task processing and data routing in a Space Cloud architecture.

2. **Develop a high-fidelity emulation platform** capable of replicating real-time orbital dynamics, network conditions, and containerized application execution to support reproducible experimentation at scale.

3. **Formulate the Edge Server Activation and Edge Server Placement problems** for space systems, defining their respective optimization functions and constraints.

4. **Implement intelligent server activation algorithms**—including Fuzzy Logic (FL) and Reinforcement Learning (RL) controllers—to dynamically manage server activation, task offloading, and resource allocation across the constellation.

5. **Formulate and analyze MEC server placement strategies** for LEO constellations, comparing terrestrial, in-orbit, and hybrid configurations under different traffic patterns and spatial distributions.

6. **Generate synthetic datasets for machine learning research** in Non-Terrestrial Networks, enabling the training and benchmarking of orchestration models where operational data are unavailable.

7. **Evaluate system performance** through key performance indicators (KPIs) such as activation ratio, round-trip time, task completion ratio, routing path length, and energy efficiency.

The central **hypothesis** of this thesis is that *smart orchestration algorithms applied to the server activation and placement optimization problems can significantly improve the performance of MEC-LEO constellations*. Specifically, we hypothesize that:

> Intelligent controllers can reduce MEC server active time without compromising task completion rates.
>
> Optimized placement strategies can minimize routing path length and overall network latency in dynamic satellite topologies.
>
> Emulation-driven experimentation can produce reliable insights that generalize to real-world system deployments.

By addressing these objectives and testing this hypothesis, the thesis aims to provide both theoretical foundations and practical tools to guide the deployment of future Space Cloud systems in NTNs.

1.3 Methodology and Thesis Structure

The research methodology combines system modeling, real-time emulation, algorithmic design, and performance evaluation. Given the operational constraints and limited accessibility of real satellite constellations, the work relies on an experimental software framework capable of accurately representing the behavior of MEC-enabled LEO systems. The methodological approach is structured into the following phases:

1. **Problem analysis and system modeling:** A detailed study of the operational environment of LEO satellite constellations is conducted, incorporating orbital mechanics, network topology evolution, link characteristics, and computational capabilities. The model formalizes the task offloading process, MEC server activation dynamics, and communication delays, providing the foundation for algorithm design.

2. **Emulation platform development:** Building on the *MeteorNet* framework, the methodology integrates continuous-time orbital propagation, network virtualization via Mininet, and containerized execution using Docker. The platform supports realistic replication of both network and computing resources, including the ability to generate high-fidelity synthetic datasets for machine learning research in Non-Terrestrial Networks.

3. **Design of orchestration and placement strategies:** Intelligent control algorithms, including Fuzzy Logic (FL) and Reinforcement Learning (RL) controllers, are developed to dynamically manage MEC server activation and task offloading. Complementary optimization models are formulated to determine efficient MEC server placement in space, terrestrial, and hybrid configurations.

4. **Scenario definition and performance evaluation:** Representative constellation scenarios are defined to test the proposed methods under varying traffic patterns, spatial distributions, and resource constraints. KPIs such as round-trip time, task completion ratio, average routing path length, and MEC server active time are measured to evaluate system performance.

5. **Comparative analysis and validation:** The performance of the proposed strategies is compared against baseline approaches, including static placement and reactive orchestration policies. Sensitivity analysis is conducted to assess robustness under different operating conditions and to validate the scalability of the solutions.

Thesis Structure

The remainder of this thesis is organized as follows:

Chapter 1: Introduction — Presents the research problem, motivation, objectives, hypothesis, methodology, and structure of the thesis.

Chapter 2: Background and State of the Art — Reviews key concepts in NTNs, MEC, and satellite constellation architectures, along with relevant prior work in orchestration strategies, server placement optimization, and emulation tools.

Chapter 3: Methods and Problem Formulations — Describes the modeling of MEC-LEO constellations, including system parameters, task processing flows, and the formalization of the server placement and activation problems.

Chapter 4: Constellation and Edge Computing Emulation — Details the design and implementation of the *MeteorNet* platform, including the software architecture, class-diagram and modular components layers for orbital propagation, network virtualization, containerized execution, and dataset generation.

Chapter 5: Evaluation and Results — Defines the experimental scenarios, metrics, and methodology used for testing, and presents the results of server placement strategies and orchestration algorithms under various operating conditions.

Chapter 6: Conclusions and Future Work — Summarizes the main contributions, discusses the implications of the results, outlines limitations, and proposes directions for future research in the deployment of Space Cloud architectures.

1.4 Summary

This chapter has presented the motivation, objectives, and methodological foundations of the research. It has framed the problem of efficient orchestration and server placement within NTNs and established the need for realistic emulation and intelligent control mechanisms to enable in-orbit computing. The following chapter, *Background and State of the Art*, expands upon these ideas by reviewing the technological evolution of LEO constellations, the integration of MEC into space systems, and existing research efforts in orchestration, server placement, and emulation frameworks that support this thesis.

CHAPTER 2

Background and State of the Art

This chapter provides the technological and conceptual background for the Space Cloud vision studied in this thesis. It reviews the evolution of Low Earth Orbit (LEO) satellite constellations from early mobile-satellite systems to today's mega-constellations, and it frames their role within the ongoing Terrestrial–Non-Terrestrial Network (T-NT) integration trends towards 5G-Advanced and 6G. It then introduces the standardization path for Non-Terrestrial Networks (NTNs), emphasizing the European vision for integrated connectivity and the policy and programmatic context that motivates this work. Finally, it summarizes existing Multi-access Edge Computing (MEC) approaches, AI-driven network management trends, and NTN experimentation tools, motivating the need for the methods and real-time emulation framework proposed in this thesis.

2.1 LEO Constellation Evolution

LEO constellations have become a cornerstone of global connectivity because they reduce propagation delay relative to geostationary systems while enabling wide-area coverage through many coordinated satellites [1]. Although today's mega-constellations are often associated with broadband Internet access, the historical development of LEO constellations spans multiple application waves, including mobile telephony, messaging, maritime and aeronautical services, Earth observation, and machine-type communications. This historical trajectory is essential to contextualize why LEO is now considered a natural substrate for integrated communications and computing in 5G-Advanced and 6G-oriented architectures.

2.1.1 Early LEO Systems and the Broadband Transition

Early LEO constellations were primarily designed to deliver global mobile-satellite services, targeting voice and low-rate data through handheld or vehicular terminals. Representative systems pioneered in the 1990s and early 2000s demonstrated global coverage

using many satellites and frequent handovers, establishing technical foundations that remain relevant today, including multi-beam payloads, mobility management, and large-scale network operations. These systems also highlighted the operational complexity of LEO, where continuously changing geometry affects coverage, routing, and service continuity, and where constellation-scale management becomes a first-class system requirement. As the space industry matured, the adoption of standardized satellite platforms and Commercial Off-The-Shelf (COTS) components enabled faster design cycles and lower mission costs [2]. In parallel, improvements in launch availability and cost reduction accelerated constellation deployment, while advances in digital payloads and onboard processing increased the functional ambition of satellites beyond simple bent-pipe relays. This industrial and technological evolution set the stage for broadband LEO networks, where satellites provide high-throughput access and backhaul, and where Internet-like service expectations require stronger guarantees on latency, availability, and scalability.

2.1.2 Mega-Constellations and ISL Networking

Recent commercial ventures such as SpaceX's Starlink and OneWeb demonstrated the feasibility of broadband LEO at scale, with thousands of satellites and aggressive expansion plans [3]. A critical enabler of this new generation is the progressive adoption of Inter-Satellite Links (ISLs), which reduce reliance on ground gateways, improve routing flexibility, and support continuous connectivity over remote regions [4]. ISLs effectively transform constellations into dynamic, space-borne mesh networks, where routing decisions must continuously adapt to time-varying topology and link conditions. From a systems perspective, this shift implies that LEO constellations increasingly resemble large-scale, time-evolving networks whose performance depends on joint optimization across communication, routing, and resource management layers. This observation directly motivates the Space Cloud paradigm explored in this thesis, because once a constellation is treated as a networked platform, onboard computing becomes a natural complement to onboard forwarding. In particular, the same ISL-enabled connectivity that improves communication coverage also enables distributed task placement and offloading across satellites, which is essential for low-latency services that cannot rely exclusively on terrestrial backhaul.

2.1.3 LEO Constellations Toward 6G

Current trends towards 6G emphasize ubiquitous connectivity, resilience, and seamless integration of heterogeneous segments, including terrestrial, aerial, and space networks. In this context, LEO mega-constellations are increasingly viewed as an integral component of future end-to-end architectures rather than as standalone overlay systems. From the standardization viewpoint, 3GPP has already introduced normative work to integrate NR with NTN components, establishing a technical baseline that subsequent releases will extend towards tighter integration and more advanced capabilities [5, 6]. From the policy and programmatic viewpoint, Europe has launched the Secure Connectivity Programme, commonly referred to as IRIS^2, as a multi-orbit system to strengthen resilience and autonomy for secure communications while complementing commercial and governmental connectivity needs [7, 8, 9]. IRIS^2 is relevant to this thesis because it illustrates the institutional commitment to large-scale space connectivity in Europe, and because its long-term roadmap is explicitly aligned with the broader trajectory of integrated networks envisioned for the 2030 horizon. At the global level, the ITU IMT-2030 framework and technology-trend reports describe 6G as a system-of-systems that integrates non-terrestrial components to improve coverage, reliability, and service continuity at planetary scale [10, 11]. Therefore, the Space Cloud vision addressed in this thesis should be interpreted within this broader 6G direction, where communication and computing resources across space and Earth are orchestrated jointly to meet latency, availability, and efficiency requirements.

2.2 NTNs from 5G-Advanced to 6G

The integration of satellite systems into terrestrial mobile networks has led to the concept of NTNs, standardized within 3GPP and progressively expanded to support new service requirements and deployment scenarios [5]. NTNs aim to provide seamless service continuity between ground and space segments, supporting mobility, coverage extension, and network resilience, which are key requirements for time-sensitive and mission-critical applications [12]. In practice, this integration requires harmonized solutions for radio access, mobility management, timing, and end-to-end network control, and it must operate under unique constraints including long round-trip times, Doppler effects, and time-varying connectivity.

2.2.1 3GPP NTN Standardization Baseline

3GPP has established a formal technical path for NR-based NTN support, with channel models, deployment assumptions, and system requirements that enable satellite components to be incorporated into the 5G system architecture [6]. This standardization baseline is fundamental for research on Space Cloud architectures because it defines the operational envelope within which integrated services will be engineered and evaluated. Consequently, the MEC orchestration problems studied in this thesis can be interpreted as higher-layer resource-management challenges that build upon the standardized connectivity substrate provided by NTN-capable 5G systems.

2.2.2 European T–NT Vision for 6G

A distinctive aspect of the European landscape is the strong coupling between research, standardization, and industrial policy around 6G, including explicit attention to non-terrestrial components as part of a unified connectivity fabric [13, 14]. Flagship projects such as Hexa-X emphasize the co-design of connectivity, computing, and intelligence, creating a natural conceptual bridge to the Space Cloud paradigm [15]. Within this vision, satellite networks are expected to evolve from isolated systems into tightly integrated segments that interoperate with terrestrial networks and host service functions when terrestrial infrastructure is unavailable or suboptimal.

2.2.3 IRIS^2 and Integrated Architectures

IRIS^2 represents a concrete European programmatic commitment to non-terrestrial infrastructure at scale, targeting secure and resilient connectivity [7, 9]. This institutional context further motivates the need for autonomous, scalable, and energy-aware orchestration mechanisms for space-based computing resources.

2.3 MEC Computing in Space

Multi-access Edge Computing (MEC) was originally introduced in terrestrial 5G networks to bring computing capabilities closer to end users, thereby reducing latency and improving application performance [16]. As the vision evolves towards 6G, MEC is increasingly regarded not merely as a deployment option, but as a foundational architectural principle for supporting AI-native, low-latency, and context-aware services.

Its extension to NTNs and LEO constellations introduces additional challenges, including rapidly changing network topologies, intermittent connectivity, limited onboard resources, and strict energy constraints [17]. Nevertheless, recent advances in onboard processing, virtualization, and inter-satellite networking have enabled the execution of computational tasks directly in orbit, reducing service time and alleviating reliance on terrestrial backhaul [18].

Beyond basic task offloading, very recent research has started to investigate the joint use of MEC and Artificial Intelligence (AI) to manage, monitor, and optimize 6G-oriented NTNs. In particular, AI-driven control loops are being proposed to enable self-configuration, self-optimization, and self-healing capabilities across heterogeneous terrestrial and non-terrestrial segments [19, 20]. These approaches align with the emerging 6G paradigm in which intelligence is embedded natively into the network architecture rather than treated as an external management function.

A key enabler of this trend is the progressive disaggregation of the Next-Generation Radio Access Network (NG-RAN), as promoted by O-RAN and cloud-native 6G architectures. Disaggregated RAN architectures expose fine-grained control and monitoring interfaces that facilitate AI-based optimization of radio, networking, and computing resources [21]. Recent studies have begun to explore how similar principles can be extended to NTN scenarios, enabling AI-assisted orchestration of satellite-based MEC resources in coordination with terrestrial edge and core components [22, 23].

In this context, MEC-in-space can be interpreted as a distributed, AI-controllable computing fabric that complements the disaggregated NG-RAN by providing elastic computational capacity beyond the terrestrial footprint. Such an interpretation is particularly relevant for LEO constellations, where topology-aware and latency-sensitive decisions must be taken under strict timing constraints. This directly motivates the Space Cloud vision adopted in this thesis, in which onboard MEC resources are activated, placed, and coordinated using intelligent control mechanisms that account for orbital dynamics and network state.

2.4 Orchestration Strategies for MEC in LEO Constellations

Effective MEC deployment in LEO constellations requires orchestration mechanisms that determine when and where to activate servers and how to offload tasks between nodes.

Recent research increasingly focuses on distributed and autonomous approaches, including fuzzy logic and reinforcement learning, to cope with the dynamic nature of orbital topologies and service demand [24, 25]. This thesis contributes to this line of work by formalizing the Edge Server Activation and Edge Server Placement problems and evaluating intelligent controllers under realistic emulation.

2.5 Emulation Frameworks for NTNs

High-fidelity emulation frameworks are indispensable for evaluating MEC orchestration in NTNs, as real deployments are costly and inaccessible [26]. Continuous-time emulation enables realistic assessment of interactions between routing, orchestration, and application execution, motivating the real-time emulation approach adopted in this thesis.

2.6 Summary

This chapter reviewed the evolution of LEO satellite constellations and framed NTN integration as a key trajectory from 5G-Advanced towards 6G. It highlighted recent advances in MEC and AI-driven network management, emphasizing the relevance of disaggregated NG-RAN concepts for space-based computing. These foundations motivate the methodological framework introduced in the next chapter for designing and evaluating Space Cloud orchestration strategies.

CHAPTER 3

Methods and Problem Formulations

In this chapter, we formally define two Multi-access Edge Computing (MEC) challenges for intelligent Low Earth Orbit (LEO) constellations: 1) the Edge Server Activation (ESA) problem and 2) the Edge Server Placement (ESP) problem. We introduce a common notation and propose Mixed-Integer Linear Programming (MILP) objectives for both problems. Because the resulting formulations are NP-hard, we design heuristics and learning-based strategies to obtain high-quality solutions with practical runtimes within the margins of real-time processing systems. While ESA decides, online and autonomously, when a satellite server should be active, ESP determines where to deploy a limited number of servers across space and terrestrial infrastructure.

Beyond formal problem definitions, this chapter adopts a systemic view of the *Space Cloud* vision. The term Space Cloud refers not merely to have MEC on satellites, but an integrated connectivity-compute environment in which (i) *physical connectivity* (user links, ISLs, and gateways) provides the base structure, (ii) *virtualization and MEC* expose compute and networking primitives as controllable resources, (iii) *software-defined control* as SDN-inspired concepts, enables programmable routing and service chaining across a time-varying topology, and (iv) *AI-driven orchestration* supports autonomous decisions under uncertainty and scale. Within this environment, ESA and ESP capture two complementary control levers: shaping the temporal availability of compute (activation) and shaping the *spatial distribution* of compute (placement), both under constellation dynamics and workload variability.

We are promoting the terms Space Cloud and *Hybrid Cloud*, not yet standardized in the NTN literature, and position them relative to the conventional cloud–edge–terminal architecture. The term "cloud" is used in the cloud-computing sense as a pool of virtualized compute resources accessed over a network, and the term "edge" is used in the MEC sense as compute terminals closer to users than a centralized data center. In NTNs, a backhaul/transport function remains necessary in all cases: tasks and results must still traverse access links and ISLs, and in some architectures may additionally traverse

gateway links to reach terrestrial resources. Therefore, a Space Cloud removes the need for terrestrial cloud offloading for the considered service, it avoids the space-to-ground transport segment associated with gateway-based processing while still relying on in-space transport (ISLs) and user access links. Hybrid Cloud, in turn, does not redefine MEC itself; instead, it denotes a two-domain compute paradigm in which both in-space MEC servers and terrestrial cloud resources are simultaneously available, and the system selects the execution domain that best satisfies latency and congestion objectives.

3.1 Common Notation

We consider a MEC-LEO constellation divided into space and user segments and, for ESP, an additional ground-segment proxy to represent terrestrial cloud services. Symbols are reused across sections to maintain consistency and enable cross-comparison of methods.

3.1.1 Space Segment

Let V be the number of satellites in the constellation and $\mathcal{V} = \{v_j \mid 1 \leq j \leq V\}$ the set of satellites, distributed over $\mathcal{P}$ orbital planes. All planes share the same mean inclination (i_o), while their mean anomalies (M_o) are spaced to provide global coverage (see Appendix C). Neighboring satellites connect via Inter-Satellite Links (ISLs) within each plane and across adjacent planes, with up to four links per satellite, forming a grid-like topology. Figure 3.1 illustrates a constellation topology, similar to Starlink, in Earth-Centered Inertial (ECI) coordinates and linked with neighbor ISLs. Each node v_j can host an active–inactive server, with activation variable $y_j^t \in \{0, 1\}$ and computational capacity H_j.

The time-varying geometry of LEO implies that both link availability and shortest paths evolve continuously. From a control perspective, satellites behave as mobile infrastructure nodes: the same compute resource may be close to a user at one time and many ISL hops away at another. This coupling between topology and compute motivates why activation and placement cannot be treated as purely static provisioning problems, but instead must be formulated in a way that explicitly accounts for dynamic routing distances and time-varying load.

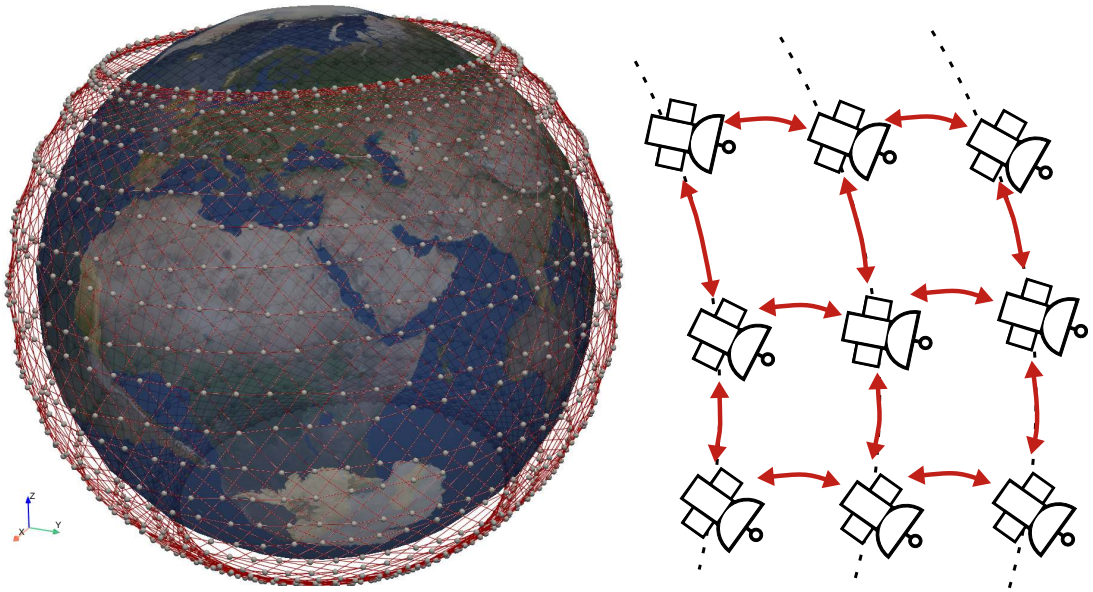


Figure 3.1: Visualization of 1,584 Starlink satellites with ISLs.

3.1.2 User Segment

Users are located at U ground positions, with $\mathcal{U} = \{u_i \mid 1 \leq i \leq U\}$. Each site hosts $|\mathcal{D}|$ devices that generate tasks according to Poisson stochastic processes, simulating user demand over a discrete horizon $T = \{0, \Delta t, \ldots, |T|\}$. Devices neither access the terrestrial cloud directly nor process tasks locally and thus rely on the constellation to compute at the space edge or forward to ground through gateways in the ESP problem. Users attach to the nearest satellite in Line-of-Sight (LoS) and hand over to the next-nearest when they lose communication.

Importantly, the task flow from user to compute resource is mediated by the connectivity fabric: access links determine the entry satellite, ISLs determine the multi-hop path to a compute node, and gateways provide a transport path to terrestrial cloud resources. Therefore, even when the application is purely computing-oriented, the relevant performance quantities (latency, failures, congestion) are determined by a communication link pattern, which motivates the integrated metrics used in the following sections.

3.2 Edge Server Activation

We orchestrate onboard servers at satellites $v_j \in \mathcal{V}$, activating or deactivating them based on workload and topology dynamics, with the dual objective of limiting resource usage and preventing task failures.

In the Space Cloud view, ESA corresponds to the run-time control of the virtualized cloud. Activation can be interpreted as a coarse-grained power-management and capacity-allocation primitive: it determines how much compute is available in orbit at each time step, and where it is available relative to the evolving ISL topology. Because LEO dynamics can change the effective service area of a given satellite over short time scales, ESA must balance local responsiveness with global stability, avoiding oscillations that might reduce reliability despite activating more servers.

3.2.1 Problem Formulation

Let $L^t_{i,j}$ denote the end-to-end latency from task transmission to result reception for a task generated by u_i at time t and processed at v_j. We define

$$L^t_{i,j} = 2 \cdot d^t_{i,j} + T^t_{F_{i,j}} + T_p. \tag{3.1}$$

Here, $d^t_{i,j}$ is the Shortest Path Delay (SPD), $T^t_{F_{i,j}}$ accounts for queuing and per-hop transmission, and T_p is the processing time. The forwarding delay depends on hop count $h^t_{i,j}$ and per-link delay T_l, i.e.,

$$T^t_{F_{i,j}} = 2 \cdot h^t_{i,j} \cdot T_l. \tag{3.2}$$

Figures 3.2 and 3.3 show the latency profile and schema for a task from u_i served at v_j.

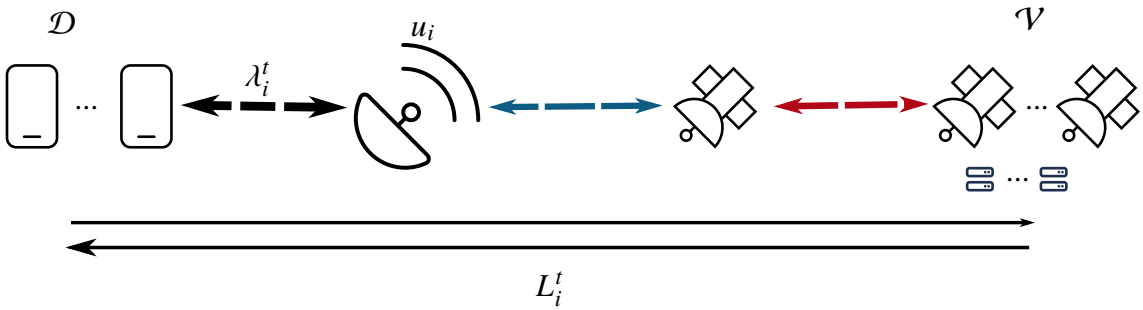


Figure 3.2: Constellation communication schema.

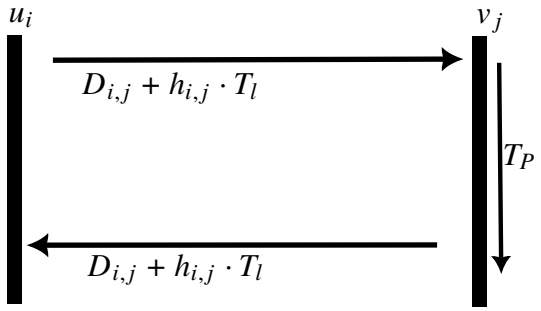


Figure 3.3: Task latency profile.

Let $x_{i,j}^t \in \{0, 1\}$ indicate whether u_i offloads to v_j at time t. The latency of a task from u_i is thus

$$L_i^t = 2 \cdot \sum_{v_j \in \mathcal{V}} x_{i,j}^t \cdot \left(d_{i,j}^t + h_{i,j}^t \cdot T_l \right) + T_p. \tag{3.3}$$

Capacity requirements are given by

$$C_j^t = \frac{\sum_{u_i \in \mathcal{U}} x_{i,j}^t \cdot \lambda_i^t \cdot \mathrm{Pr}}{H_j \cdot \Delta t} < 1 \quad \forall v_j \in \mathcal{V}, \tag{3.4}$$

and their time–space average is

$$\overline{C} = \frac{\Delta t \cdot \sum_{t \in T} \sum_{v_j \in \mathcal{V}} C_j^t}{|T| \cdot V}. \tag{3.5}$$

Here, Pr is the task size, H_j the capacity at v_j, and λ_i^t the task rate at u_i. We refer to $\overline{C}$ as the *Utilization Ratio*. Tasks exceeding the threshold L_{th} are failures, counted in the variable

$$B_j^t = \sum_{u_i \in \mathcal{U}} \lambda_i^t \cdot b_i^t \cdot x_{i,j}^t, \tag{3.6}$$

and the normalized *Failure Rate* is

$$\overline{B} = \frac{\Delta t \cdot \sum_{t \in T} \sum_{v_j \in \mathcal{V}} B_j^t}{\sum_{t \in T} \sum_{u_i \in \mathcal{U}} \lambda_i^t}. \tag{3.7}$$

Here, $b_i^t = 1$ if $L_i^t > L_{th}$ and 0 otherwise. Lastly, let $y_j^t \in \{0, 1\}$ denote the activation of the server at v_j, so the number of active servers is

$$K^t = \sum_{v_j \in \mathcal{V}} y_j^t. \tag{3.8}$$

Its normalized average is

$$\overline{K} = \frac{\Delta t \cdot \sum_{t \in T} K^t}{|T| \cdot V} \in [0, 1], \tag{3.9}$$

which we call the *Activation Ratio*. The ESA objective minimizes a weighted sum of the Activation Ratio $\overline{K}$ and the Failure Rate $\overline{B}$ under assignment and capacity constraints:

$$\begin{aligned} \min_{x,y} \quad & \alpha \cdot \overline{K} + \beta \cdot \overline{B} \\ \text{s.t.} \quad & \sum_{v_j \in \mathcal{V}} x_{i,j}^t = 1 \quad \forall u_i \in \mathcal{U},\ \forall t \in T, \\ & x_{i,j}^t \le y_j^t \qquad \forall u_i \in \mathcal{U},\ \forall v_j \in \mathcal{V},\ \forall t \in T, \\ & C_j^t < 1 \qquad \forall v_j \in \mathcal{V},\ \forall t \in T. \end{aligned} \tag{3.10}$$

Coefficients α and β tune the trade-off between energy efficiency and reliability.

3.2.2 ESA Methods

We represent the local state at server v_j and time t as

$$S_j^t = [B_j^t, C_j^t, D_j^t, K^t], \tag{3.11}$$

where B_j^t, C_j^t, and K^t follow (3.6), (3.5), and (3.8), and D_j^t counts tasks offloaded from v_j to other nodes. The decision $y_j^t \in \{0, 1\}$ indicates the activation state, and the goal is a distributed per-satellite policy that minimizes (3.10) given S_j^{t-1}. We evaluate four controllers: a deterministic Access Controller, a stochastic Bernoulli Controller, a heuristic Fuzzy-Logic (FL) controller, and a learning-based Reinforcement Learning (RL) actor–critic policy.

At a high level, the four controllers represent a range of modeling complexity and adaptivity. Deterministic thresholding is lightweight and easy to verify, but it is brittle under non-stationary workloads. Randomized baselines provide a reference for what can be achieved without state-awareness. Fuzzy logic introduces structured adaptivity with explicit rules, offering interpretability at the cost of manual tuning. Finally, RL provides data-driven policies that can exploit subtle correlations between topology-induced congestion and compute load, but requires training, careful reward design, and robustness checks to avoid unstable behaviors. This section details the substance of each controller and clarifies their respective pros and cons.

Access Controller

The Access Controller uses thresholds on the observed failure rate and queue length to toggle activation. It reacts to workload shifts but lacks context awareness, often over-activating under volatile conditions.

Substance. The Access Controller implements a reactive, rule-based control loop: when locally observed indicators exceed fixed thresholds, the server is activated; when indicators fall below lower thresholds, it is deactivated (with hysteresis implied by distinct on/off triggers). This controller treats the constellation as an exogenous environment and does not explicitly reason about time-varying topology, future handovers, or the global activation level.

Pros and cons. Its main advantage is operational simplicity: it is inexpensive to compute, straightforward to implement onboard, and easy to validate. However, it is sensitive to threshold selection and may oscillate when workload is bursty or when topology changes cause transient rerouting. Moreover, because it uses only local parameters, it may activate servers in locations that are topologically disadvantaged, which can increase overall activation.

Bernoulli Controller

The Bernoulli baseline samples $y_j^t \sim \text{Bernoulli}(p)$ independently of the state. It provides a randomized reference that highlights the advantage of stateful and adaptive policies.

Substance. The Bernoulli controller is intentionally non-adaptive: it implements a fixed probability of being active at each time step, which yields a stationary activation process across satellites.

Pros and cons. While not a practical control mechanism, it is a useful experimental baseline. It makes explicit the performance gap between mere capacity availability and intelligent capacity allocation. Its limitation is intrinsic: it cannot exploit context such as congestion, routing distance, or demand spikes, and therefore may either waste energy or induce excessive failures.

Fuzzy Logic Controller

Our FL controller automates activation through linguistic rules and membership functions, using only local information per satellite and avoiding a global network view. It maps the

state in (3.11) to a continuous score $f_j^t \in [0, 1]$ and then applies hysteresis to produce y_j^t:

$$y_j^t = \begin{cases} 0, & \text{if } f_j^t \le \frac{1}{3}, \\ 1, & \text{if } f_j^t \ge \frac{2}{3}, \\ y_j^{t-1}, & \text{if } \frac{1}{3} < f_j^t < \frac{2}{3}. \end{cases} \tag{3.12}$$

The fuzzy reasoning proceeds in three phases (see Appendix A) [27]. *Fuzzification*: inputs are mapped to sets $P_d = \{L_d, M_d, H_d\}$ with trapezoidal (L, H) and triangular (M) membership functions tuned by experiment-driven expert knowledge. *Inference*: rules in Table 3.1 combine memberships via a multilinear implication with $T = J = \min$ [28] to obtain output memberships $Q = \{L_f, M_f, H_f\}$. *Defuzzification*: the scalar f_j^t is computed by area–centroid weighting,

$$f_j^t = \frac{\sum_{q \in Q^*} \mu_q \cdot q}{\sum_{q \in Q^*} \mu_q}. \tag{3.13}$$

Controller quality depends on membership ranges and rule design, which we tune iteratively to balance $\overline{K}$ and $\overline{B}$ as in (3.10) [29].

Substance. The FL controller can be interpreted as a compact, interpretable surrogate for a non-linear policy. Instead of hard thresholds on raw metrics, it reasons over linguistic abstractions (low/medium/high) and combines them into activation pressure through explicit rules. The use of hysteresis in (3.12) is critical in LEO because it avoids frequent toggling caused by short-lived changes in routing distances.

Pros and cons. FL offers an appealing compromise between simplicity and adaptivity. It is lightweight, does not require training data, and yields transparent decision logic that can be inspected and modified. However, it relies on expert-designed membership functions and rules, and therefore may require substantial tuning when the constellation scale, traffic varies, or ISL characteristics change. In addition, FL remains largely reactive: while it can smooth decisions, it does not explicitly optimize long-term objectives beyond what is programmed in the rule base.

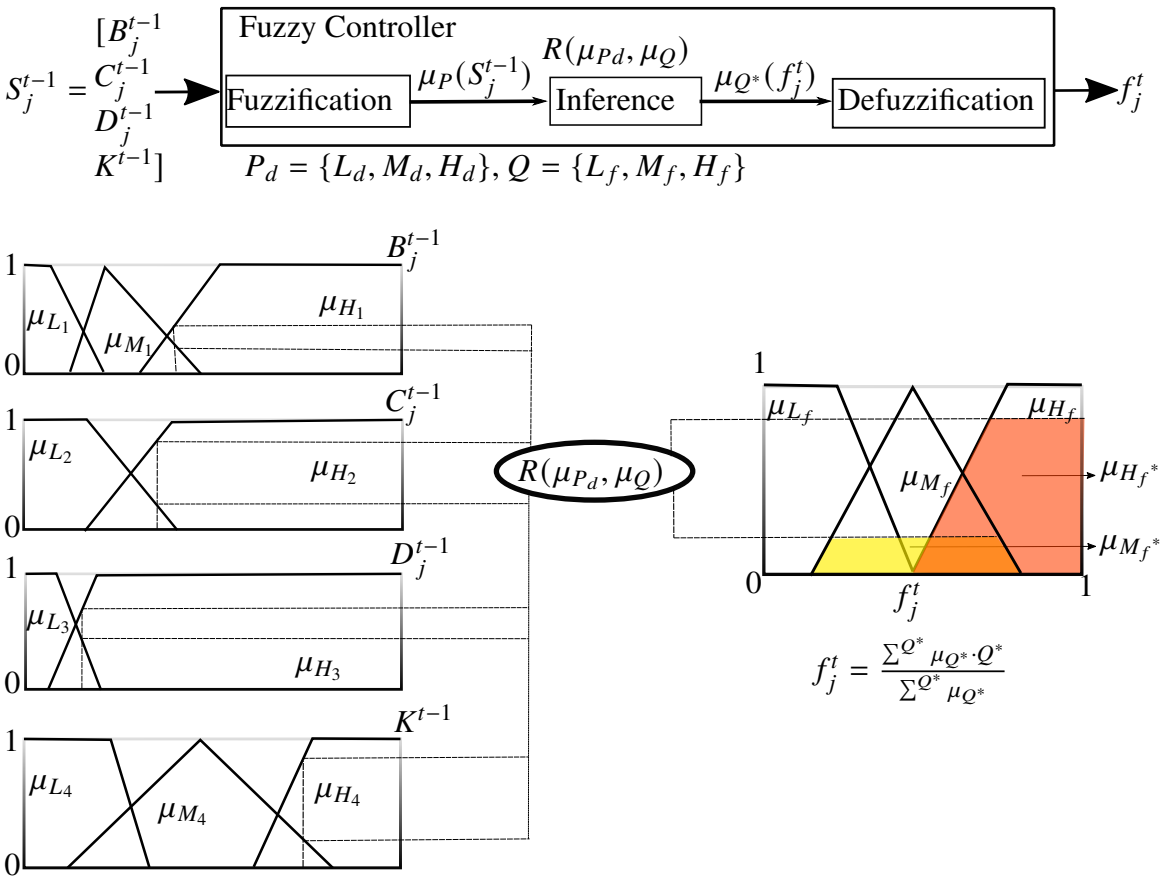


Figure 3.4: Fuzzy-Logic controller diagram (top) and membership functions (bottom).

Table 3.1: Linguistic rules for the Fuzzy-Logic controller.

R_1:	if B_j^{t-1} is H_1			then f_j^t is H_f
R_2:	if B_j^{t-1} is M_1	and D_j^{t-1} is H_3		then f_j^t is H_f
R_3:	if C_j^{t-1} is L_2	and D_j^{t-1} is H_3		then f_j^t is M_f
R_4:	if B_j^{t-1} is L_1	and C_j^{t-1} is H_2		then f_j^t is M_f
R_5:	if C_j^{t-1} is M_2	and K^{t-1} is H_4		then f_j^t is L_f
R_6:	if D_j^{t-1} is L_3	and C_j^{t-1} is L_2	and B_j^{t-1} is L_1	then f_j^t is L_f

Reinforcement Learning Controller

The RL controller learns activation policies via an actor–critic architecture (see Appendix B) that jointly optimizes a stochastic policy and a value function. The **actor** $\pi_\theta(S_j^{t-1})$ outputs a probability distribution over actions $\{0, 1\}$, and the **critic** $V_\theta(S_j^{t-1})$ estimates the expected return, sharing parameters θ for efficient training. Figure 3.5 depicts

the model: a shared backbone with ReLU layers feeds an actor head (softmax over actions) and a critic head (scalar value). Training is episodic, with rewards aligned to (3.10). The per-step reward is

$$\begin{aligned} r_j^t = \omega_1 \cdot \exp(-\omega_{11} \cdot B_j^t) \\ + \ \omega_2 \cdot (1 - y_j^t) \\ + \ \omega_3 \cdot C_j^t \\ + \ \omega_4 \cdot (1 - K^t), \end{aligned} \tag{3.14}$$

where the ω coefficients weight reliability, activation, utilization, and global activity. Policy gradients increase the likelihood of advantageous actions, and the value loss fits returns via mean squared error, with joint updates of θ per episode. This process yields context-aware activation patterns that reduce failures while conserving resources.

Substance. In a LEO Space Cloud, activation is a sequential decision problem: turning a server on influences future routing, future congestion, and the future availability of nearby compute due to orbital motion. The actor–critic structure is well suited to this setting because the critic provides a learned baseline that stabilizes policy gradients, while the actor represents a stochastic policy that can trade off exploration and exploitation during training. The state vector in (3.11) captures both local service quality (failures, utilization) and coupling signals (offloaded tasks, global activity), enabling the learned policy to respond to congestion that may be caused by topology shifts rather than only by local demand spikes.

Pros and cons. RL can outperform handcrafted controllers when the environment is complex and non-linear, as in large constellations with time-varying shortest paths and correlated traffic. It can also adapt to patterns that are difficult to encode as explicit rules, like anticipating congestion based on rising offload counts. However, RL introduces operational costs: it requires extensive training data, careful reward shaping, and robustness validation to ensure stability under distribution shifts. In addition, interpretability is lower than in FL, so practical deployment benefits from complementary analyses to increase trust in the learned policy.

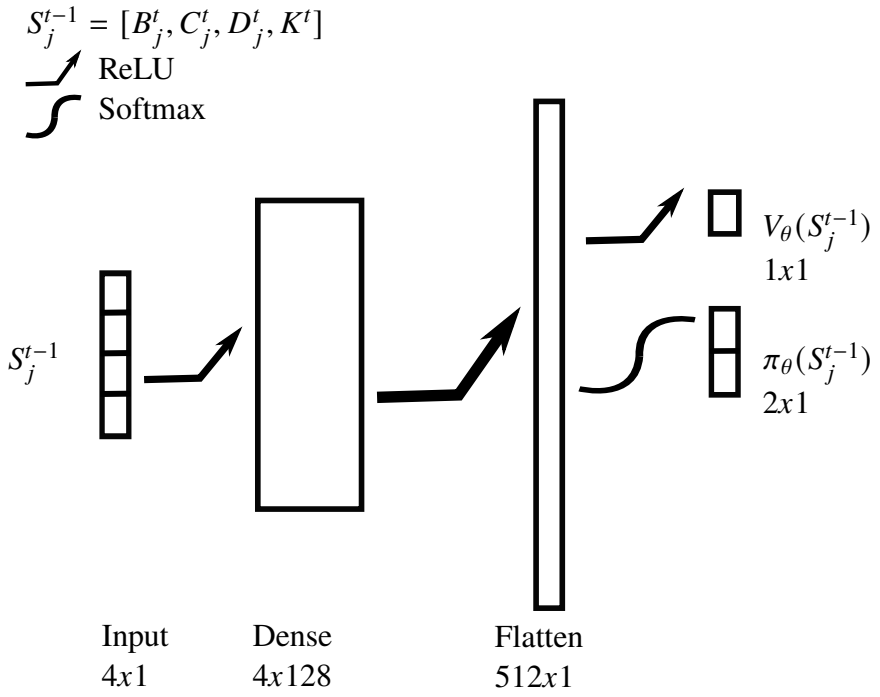


Figure 3.5: Actor–critic model diagram.

Algorithm 1 Pseudocode for training the RL actor–critic model.

π_θ, V_θ ▷ Actor–critic networks
Env ← init_constellation() ▷ Environment class
$r \leftarrow 0$ ▷ Reward variable per episode
r^* ▷ Reward threshold to converge
while $r < r^*$ **do** ▷ Iterate through episodes
    Env.start() ▷ Start dynamics
    $r \leftarrow 0$
    for $t \in T$ **do**
        for $v_j \in \mathcal{V}$ **do**
            $S_j^{t-1} \leftarrow$ Env.get_state$(t-1)$
            $y_j^t \leftarrow \{0,1\}$.random_choice$(p = \pi_\theta(S_j^{t-1}))$
            $r_j^t \leftarrow$ Env.get_reward(t)
            $r \leftarrow r + r_j^t$
        end for
    end for
    $\mathcal{L}_\pi \leftarrow \mathbb{E}\big[-\log(\pi_\theta(S_j^{t-1})[y_j^t]) \cdot (V_\theta(S_j^{t-1}) - r_j^t)\big]$
    $\mathcal{L}_V \leftarrow \mathbb{E}\big[(r_j^t - V_\theta(S_j^{t-1}))^2\big]$
    $\theta \leftarrow$ update_gradient(loss $= \mathcal{L}_\pi + \mathcal{L}_V, \theta$)
end while

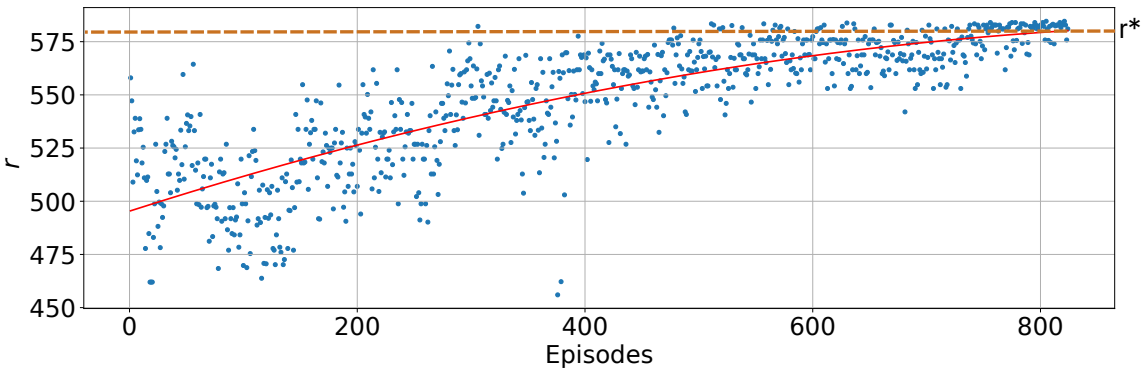


Figure 3.6: Episode rewards during training.

3.3 Edge Server Placement

For the ESP problem, we compare the performance of space, terrestrial, and hybrid deployments, as sketched in Figure 3.7. Nodes are partitioned into user, space, and ground segments, and tasks are served either in orbit or via gateways to terrestrial data centers.

From the Space Cloud ecosystem perspective, ESP configures the *infrastructure layer* upon which orchestration and routing act. Placement determines the set of compute-capable nodes available to the system, while routing and activation determine how that capacity is used at runtime. Therefore, ESP is naturally coupled to the connectivity environment: a placement that is optimal under one ISL configuration or routing policy may be suboptimal under another. In this thesis, we fix the routing abstraction through SPD, and we focus on how space-only, ground-only, and mixed deployments alter latency merics.

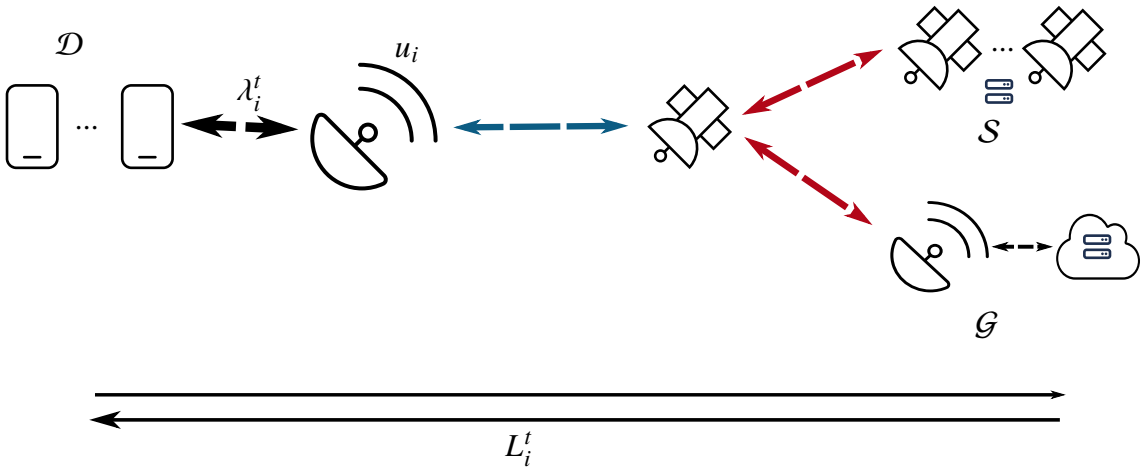


Figure 3.7: Cloud models.

3.3.1 Terrestrial Segment

For ESP, tasks may be processed on Earth by proxy gateways. Thus, the server set is $\mathcal{V} = \mathcal{S} \cup \mathcal{G} = \{v_j \mid 1 \leq j \leq V = N + G\}$, with satellites $\mathcal{S} = \{s_n \mid 1 \leq n \leq N\}$ and gateways $\mathcal{G} = \{g_m \mid 1 \leq m \leq G\}$.

Gateways play a dual role in the integrated ecosystem. Physically, they anchor the constellation to terrestrial infrastructure and provide a transport path to ground compute resources. Architecturally, they act as boundary nodes between a time-varying space mesh and a stable terrestrial cloud, enabling a two-domain compute paradigm in which tasks can be served either in orbit or on the ground depending on latency and load.

3.3.2 Problem Formulation

We compare processing in space versus forwarding to the terrestrial cloud by measuring task latency as defined in Equation 3.1. If tasks are served via a gateway, we add an extra delay term d_g to the SPD $d_{i,j}^t$, i.e.,

$$\begin{aligned} D_{i,j}^t &= d_{i,j}^t && \forall v_j \in \mathcal{S}, \\ D_{i,j}^t &= d_{i,j}^t + d_g && \forall v_j \in \mathcal{G}. \end{aligned} \tag{3.15}$$

Subtracting T_p from Equation 3.3 and replacing the forwarding factor from Equation 3.2, the Round-Trip Time (RTT) for u_i is

$$R_i^t = 2 \cdot \sum_{v_j \in \mathcal{V}} x_{i,j}^t \cdot \left(D_{i,j}^t + h_{i,j}^t \cdot T_l \right). \tag{3.16}$$

The load-weighted average RTT is

$$\overline{R} = \frac{\sum_{t \in T} \sum_{u_i \in \mathcal{U}} \lambda_i^t \cdot R_i^t}{\sum_{t \in T} \sum_{u_i \in \mathcal{U}} \lambda_i^t}. \tag{3.17}$$

We minimize $\overline{R}$ under a budget of K servers placed in space or accessed via ground gateways:

$$\begin{aligned}
\min_{x,y} \quad & \overline{R} \\
\text{s.t.} \quad & \sum_{v_j \in \mathcal{V}} y_j = K, \\
& \sum_{v_j \in \mathcal{V}} x_{i,j}^t = 1 \quad \forall u_i \in \mathcal{U},\ \forall t \in T, \\
& x_{i,j}^t \le y_j \qquad \forall u_i \in \mathcal{U},\ \forall v_j \in \mathcal{V},\ \forall t \in T, \\
& y_j = 0 \qquad \forall v_j \in \mathcal{G} \quad \text{(Space Cloud only)}, \\
& y_j = 0 \qquad \forall v_j \in \mathcal{S} \quad \text{(Terrestrial Cloud only)}.
\end{aligned} \tag{3.18}$$

Variable $y_j \in \{0, 1\}$ indicates a server at node v_j or a gateway's existence. The last two constraints can be toggled to enforce space-only or terrestrial-only scenarios, while the hybrid case removes both. The objective is a MILP and NP-hard due to the coupling between combinatorial placement and routing. By analyzing $\overline{R}$ and the distribution of R_i^t across user locations, we identify when space, terrestrial, or hybrid deployments minimize RTT for a static deployment of K servers or ground proxies.

3.3.3 Cloud Placement Strategies

We study three types of placements: **Terrestrial Clouds**, **Space Clouds**, and **Hybrid Clouds**, where server location determines routing and latency trade-offs. We define the regimes through their execution domain and the associated transport segments:

Space Cloud (in-orbit execution domain). Tasks are executed on servers hosted onboard satellites (*in-space MEC*). Transport still includes the user access link and,

when needed, multi-hop ISL routing. What is avoided in this regime is terrestrial cloud offloading, i.e., tasks do not traverse the gateway path to reach a ground data center for execution.

Terrestrial Cloud (ground execution domain). Tasks are executed on Earth through gateways (*gateway-to-cloud offloading*). Transport includes access, ISLs to a gateway, and an additional gateway-to-cloud component captured by d_g.

Hybrid Cloud (dual execution domain). Both in-orbit execution and terrestrial execution are enabled concurrently. The system can route tasks either to an onboard server or to a gateway-backed terrestrial cloud, depending on which option yields lower RTT (or better congestion conditions) at each time.

With these definitions, Hybrid Cloud allows simultaneous availability of two compute tiers—an in-space MEC tier and a terrestrial cloud tier—connected by the NTN transport layer. This is analogous to a classical hybrid cloud in terrestrial computing, where private edge resources and public cloud resources coexist and workloads can be shifted between them.

Terrestrial Cloud

Tasks traverse the constellation to gateways and are processed in the terrestrial cloud through distributed data centers.

This regime resembles a space-assisted access/backhaul network: satellites provide connectivity and routing, while compute is anchored to Earth. Its advantage is stable and scalable terrestrial compute capacity; its main limitation is that RTT becomes sensitive to gateway reachability and to the additional gateway-to-cloud component represented by d_g, which can dominate for latency-critical services.

Space Cloud

Servers are onboard satellites, routing tasks through ISLs to an in-orbit compute node. We consider static placement at launch, with relay-only satellites forwarding to the nearest server. We evaluate three distributions: **Cross Plane** (CP) spreads servers across planes to exploit inter-plane ISLs and distribute load, potentially increasing path length for some users. **Along Orbit** (AO) co-locates servers within a plane, simplifying intra-plane routing and deployment but limiting global reach. **Farthest Point Sampling** (FPS) greedily

maximizes the minimum graph distance to selected servers over averaged snapshots to improve coverage and reduce ISL hops.

In this thesis, Space Cloud denotes an in-orbit cloud tier: a pool of virtualized compute resources distributed across satellites and accessed over the constellation network. The term "cloud" emphasizes that services are not tied to a single satellite, but to a resource pool reachable through connectivity and managed through orchestration. Unlike terrestrial edge nodes, however, the effective proximity between users and servers changes over time because the topology is time varying. Therefore, Space Cloud performance depends on (i) where servers are placed on the constellation graph, (ii) how routing translates placement into hop-dependent delay, and (iii) whether sufficient servers are active at the right times (ESA).

Algorithm 2 FPS algorithm.

Initialize:
$\mathcal{V}$ Set of server nodes
$Q = \{\}$ Set of selected nodes, initialized as empty
q Node to select
Q.add($\mathcal{V}$.pop_random())
while $|Q| < K$ **do**
    $d_{\max} \leftarrow -1$
    for $v_j \in \mathcal{V}$ **do**
        $d_{j,Q}$ SPD from v_j to all selected nodes in Q
        $d_{\min} \leftarrow \min(d_{j,Q})$ Delay between v_j and its closest node in Q
        if $d_{\min} > d_{\max}$ **then**
            $d_{\max} \leftarrow d_{\min}$
            $q \leftarrow v_j$
        end if
    end for
    Q.add($\mathcal{V}$.pop(q))
end while

Hybrid Cloud

Hybrid deployments combine gateways and onboard servers, routing tasks to the lowest-latency option and mitigating congestion in either domain. In our setup, gateways follow the KSAT layout and space servers follow FPS, and as K grows, nodes are added across both domains to balance performance.

Hybrid Cloud denotes a two-tier space–ground compute architecture. The first tier is the Space Cloud tier: onboard servers reachable through ISLs. The second tier is the

terrestrial cloud tier: reachable through gateways with additional gateway-to-cloud delay d_g. The hybrid regime is characterized by a domain-selection decision, when for each task and time step, the system can choose whether to execute in orbit or on the ground. This mechanism is valuable because it mitigates two different sources of performance degradation: (1) topology-induced path extension in space, when the nearest onboard server is many ISL hops away, and (2) capacity-induced congestion in either domain, like overloaded onboard servers or overloaded gateway paths. Thus, Hybrid Cloud is not simply "MEC in NTN," but rather a specific architectural choice to jointly provision and exploit both in-space MEC and terrestrial cloud resources as a unified service provider.

Implications in Server Placement Optimization

These server placements instantiate the general ESP optimization, which selects K locations in space or on Earth to minimize average RTT. Exhaustive search is infeasible at constellation scale, so the proposed strategies provide scalable approximations with strong empirical performance.

In addition, the three regimes highlight distinct optimization implications. Space-only placement must prioritize constellation-wide coverage and low ISL hop counts to mitigate topology dependence. Terrestrial-only placement must prioritize gateway reachability and stable access to ground compute, implicitly accepting the additional gateway-to-cloud delay d_g. Hybrid placement must balance both, because adding an onboard server reduces dependence on gateways while adding a gateway reduces RTT for users whose shortest in-space route is temporarily long. These trade-offs motivate why ESP is evaluated comparatively rather than as a single best configuration: the preferred regime depends on the target latency distribution, gateway geography, and the extent to which the system is expected to operate autonomously in orbit.

3.4 Summary

This chapter defined the ESA and ESP problems, established common notation and metrics, and introduced optimization objectives together with heuristic and learning-based controllers. It also positioned Space Cloud and Hybrid Cloud as architectural regimes within an integrated connectivity-and-compute ecosystem, where physical links, MEC virtualization, programmable control, and AI-driven orchestration interact under time-varying LEO dynamics. The next chapter, *Constellation and Edge Computing Emulation*,

details the emulation framework, including orbital propagation, network virtualization, and containerized execution, which we use to evaluate orchestration and placement strategies under realistic operating conditions.

CHAPTER 4

Constellation and Edge Computing Emulation

This chapter describes the design and implementation of *MeteorNet*, a continuous-time emulation platform developed to assess edge computing systems in Low Earth Orbit (LEO) constellations. MeteorNet supports the design and evaluation of MEC architectures under realistic space-communication conditions by integrating orbital propagation, network virtualization, and containerized execution. Unlike discrete-time simulators that emphasize scalability and repeatability but rely on abstracted models, continuous-time emulation offers real-world fidelity and supports the integration of production-grade networking and industry-standard system code.

4.1 Emulation Tools Review

MeteorNet extends prior research on constellation emulation by integrating kernel-level virtualization and network-stack emulation with orbital-mechanics models. Previous work introduced early versions of this framework to explore task orchestration and edge computing in Non-Terrestrial Networks (NTN) [30]. The current implementation includes real-time orbit propagation, containerized computing environments, and Software-Defined Networking (SDN) controllers for adaptive link management and server orchestration. Table 4.1 compares MeteorNet with existing tools used in orbital analysis, network simulation, and constellation emulation. Conventional testbeds such as PlanetLab [31] and Emulab [32] focus on terrestrial networks and lack orbital modeling or active maintenance. Commercial orbital simulators like STK [33] and open-source alternatives like GMAT [34] provide accurate orbital computation but do not emulate communication protocols or system-level networking. Other frameworks, such as Hypatia [35] and StarPerf [36], extend performance analysis to satellite constellations but depend on external components and have limited update cycles. Among network emulators, Mininet [37] remains the reference for SDN research but operates on static topologies. MeteorNet closes these gaps

by merging real-time orbital dynamics, TCP/IP emulation, and SDN orchestration into a unified, open-source environment suitable for reproducible large-scale experiments.

Table 4.1: Comparison of emulation tools.

Type	Tools	(i) Orbital Propagation	(ii) Networking Environment	(iii) Maintenance Support	(iv) Open Source
Constellation platforms	PlanetLab	X	✓	X (2019)	✓
	Emulab	✓	✓	X (2019)	X
Space and orbital analysis	STK	✓	X	✓	X
	GMAT	✓	X	✓	✓
	SNS3	GEO	✓	✓	✓
	Hypatia	✓	✓	X (2020)	✓
	StarPerf	third-party	✓	✓	✓
Network emulators	Mininet	X	✓	✓	✓
	DieCast	X	✓	X (2007)	X
	Etalon	X	✓	X	X
MeteorNet		✓	✓	✓	✓

4.2 MeteorNet Software Architecture

MeteorNet is an open-source* continuous-time emulation environment that supports all stages of a LEO constellation mission, from research to operational testing. It integrates five main components:

1. **Orbital Dynamics Function**, using the SGP4 model [39] to propagate satellite trajectories and perturbations.
2. **Contact Tables**, maintaining link states and delays for handover management.
3. **Network Emulation**, built with Mininet [37], OpenFlow [40], and the ONOS controller [41].
4. **Containerized Services**, leveraging Docker to isolate application software and emulate real execution environments [42].
5. **Performance Monitoring**, with telemetry stored in a MongoDB database for offline analysis [43].

The platform, implemented primarily in Python, integrates native-kernel libraries and Docker APIs to ensure performance and flexibility for user-defined algorithms. Figure 4.1 presents the core software architecture of MeteorNet through an automatically generated UML class diagram. Because the diagram is extracted from the codebase, it reflects implementation-level classes and dependencies, whereas the separation into “layers” is

*Public repository at https://gitlab.com/camilo.rojas/meteornet [38]

an architectural design pattern used to structure the emulation workflow and to clarify responsibilities. As a result, the four layers described below are not necessarily rendered as explicit visual blocks in the UML figure, even though they guide how the platform is composed and executed.

The layered view separates (i) orbital dynamics and connectivity inputs, (ii) node instantiation and containerized execution, (iii) packet-level network emulation and forwarding-state programming, and (iv) edge-computing control and measurement. Within the scope of this thesis, the *control* plane refers specifically to edge-computing orchestration (i.e., activating or deactivating edge servers and collecting their performance signals) rather than to optimizing routing or SDN policy design. Routing is treated as an enabling substrate that must be emulated with sufficient fidelity to expose realistic latency, path inflation, and congestion effects to the MEC algorithms evaluated in Chapters 3 and 5.

At the orbital layer, the `Orbit` class (SGP4-based) computes satellite states, handles visibility events, and generates contact tables with time-indexed link availability and delay information. This layer provides the time-varying "ground truth" used to parameterize the network topology and its link delays.

At the execution layer, `Orbit` instantiates `Node` classes for satellites, users, and gateways, which specialize into `SatelliteNode`, `UserNode`, and `GatewayNode`. These nodes embed Docker hosts to run containerized software, enabling the execution of real application code and Linux networking stacks. Nodes are grouped into `OrbitalPlane`, which computes ISL neighbors for the constellation, while `ContainerNet`—a specialization of `Mininet`—manages virtual interfaces and link creation.

At the network layer, Mininet interfaces are assembled into a graph structure through `GraphNetwork`, which manages routing and forwarding behavior according to the contact tables. Forwarding state is programmed through an SDN control process based on ONOS, which runs as an external controller process rather than as a Python class within the codebase. Accordingly, ONOS is not depicted as a class element in the UML diagram; instead, the UML captures the components that interface with the SDN controller to install OpenFlow rules and keep forwarding state consistent with the evolving network graph. In this architecture, SDN mechanisms are used to realize predictable routing in the emulated constellation and to prevent forwarding loops under multi-path connectivity.

At the control and measurement layer, `EdgeServer` runs in Docker containers and processes user workloads—generated by `UserNode` and `TaskGenerator`—through `TaskProcessing`. The `EdgeControl` class implements the edge-computing control

logic considered in this thesis, namely activation of onboard servers via `FuzzyControl` and `RLControl`. Performance metrics (latency, utilization, and failures) are logged by `SatDataBase` into a MongoDB-based component (`MongoDB`), enabling reproducible experiments and post-run analysis. This layer intentionally focuses on MEC control and measurement; SDN routing control is used to maintain correct and loop-free forwarding, but it is not an optimization target of the orchestration algorithms studied here.

In the current experimental setup, the SDN control process is executed on a centralized node with direct access to the emulated switch tables, simplifying orchestration experiments and reducing control-plane variability. In operational NTNs, SDN control-plane placement becomes a first-class design dimension: controller location induces control-plane latency, and intermittent controller–node connectivity can lead to stale forwarding state or inconsistent network views. While these aspects are outside the scope of the present thesis, the architecture described above provides a baseline that can be extended to study distributed control, controller placement, and resilient SDN operation under time-varying LEO connectivity.

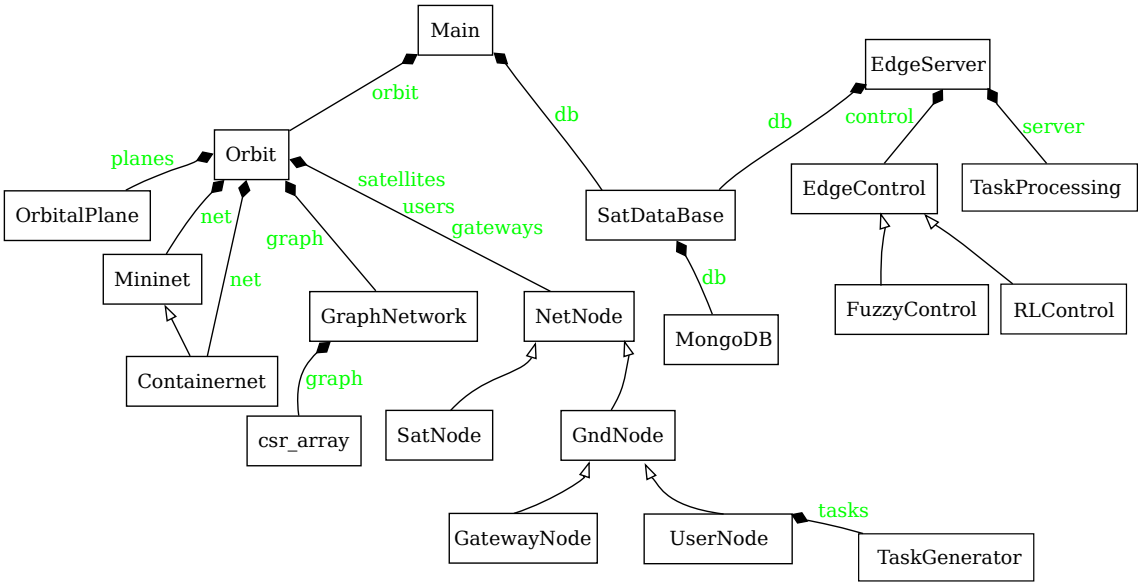


Figure 4.1: MeteorNet UML class diagram.

Workflow MeteorNet requires as input a set of Two-Line Element (TLE) files, ground-node coordinates, and emulation duration. The execution proceeds through five stages: (A) orbit propagation of satellite nodes; (B) computation of satellite–ground visibility tables; (C) update of link parameters according to topology dynamics; (D) deployment of dockerized services; and (E) real-time measurement of throughput, delay, and computation

metrics. This workflow enables reproducible, continuous-time emulation that captures the interplay between orbital motion and network performance.

4.2.1 Orbital Dynamics

We propagate satellite trajectories with the Simplified General Perturbations Model 4 (SGP4) using Two-Line Element (TLE) sets (see Appendix C) [39]. At time t, the orbital propagator yields each satellite's position $P_j^t \in \mathbb{R}^3$ in the True Equator Mean Equinox (TEME) frame. Ground nodes are initialized from geodetic latitude–longitude–altitude, converted to TEME as $P_i^{t_0}$, and updated over time using Earth rotation to obtain P_i^t. Given P_i^t and P_j^t, we compute the slant range $|P_i^t - P_j^t|$ and elevation angle $\theta_{i,j}^t$ for each ground–satellite pair (u_i, v_j) in Line of Sight (LoS). A link is considered available and in LoS if the elevation exceeds a configurable threshold $\theta_c = 30°$, as expressed in Equation (4.1). For available links, the one-way propagation delay $d_{i,j}^t$ and the link capacity $CAP_{i,j}^t$ are derived from range according to Equations (4.2) and (4.3). These per-pair, per-time quantities are stored in *contact tables* that serve as caches to drive network dynamics efficiently at runtime.

$$LoS_{i,j}^t = \begin{cases} \text{True}, & \text{if } \theta_{i,j}^t \geq \theta_c, \\ \text{False}, & \text{if } \theta_{i,j}^t < \theta_c. \end{cases} \tag{4.1}$$

$$d_{i,j}^t = \frac{|P_i^t - P_j^t|}{c} + \delta_0, \tag{4.2}$$

$$CAP_{i,j}^t = \begin{cases} 200 \text{ Mb/s}, & 0 < |P_i^t - P_j^t| \leq 500 \text{ km}, \\ 80 \text{ Mb/s}, & 500 < |P_i^t - P_j^t| \leq 1000 \text{ km}, \\ 60 \text{ Mb/s}, & 1000 < |P_i^t - P_j^t| \leq 2000 \text{ km}, \\ 20 \text{ Mb/s}, & 2000 < |P_i^t - P_j^t| \leq 3000 \text{ km}, \\ 10 \text{ Mb/s}, & 3000 < |P_i^t - P_j^t| \leq 4000 \text{ km}, \\ \varnothing, & |P_i^t - P_j^t| > 4000 \text{ km}. \end{cases} \tag{4.3}$$

Here, $c \approx 3.0 \times 10^5$ km/s is the speed of light in vacuum, and δ_0 captures non-propagation processing overhead per link. Contact tables list, for each timestamp, the on/off state dictated by Equation (4.1) together with the corresponding $d_{i,j}^t$ and $CAP_{i,j}^t$,

and can be precomputed and reused across scenarios to reduce runtime overhead in large-scale emulations.

4.2.2 Contact Table Computation

MeteorNet employs Mininet [37] to emulate a virtual network that mirrors real communication environments through the use of native Linux-kernel, switching, and application code [44]. Ground nodes can operate in either *single-link* or *multi-link* mode. In single-link mode, each ground node maintains a connection only with the nearest visible satellite until LoS is lost. In multi-link mode, the node can simultaneously sustain multiple active links with all satellites currently in LoS. The link states of all nodes are summarized in data structures known as *contact tables*, which record, for each timestamp, the connectivity, delay, and capacity between every node pair according to the orbital geometry described earlier. Each node maintains its own contact table, with rows representing discrete time steps and columns listing potential links. When a link becomes unavailable (e.g., $\theta^t_{i,j} < \theta_c$), the corresponding entry is set to zero; otherwise, it stores the current inter-node distance or equivalent delay. Because contact tables can be precomputed from orbital propagation and reused across emulation runs, they substantially reduce computational overhead and allow large-scale constellations to be simulated efficiently. These structures act as caches that dynamically update network topology as satellites move, providing a consistent mechanism to enable or disable links and adjust their parameters in real time. Through this approach, MeteorNet continuously evolves the virtual network to reflect realistic satellite motion and connectivity changes.

Table 4.2 illustrates an example contact table for a ground node u_{11} that alternates connections among three satellites (v_{10}, v_9, v_8) over time. Each cell value corresponds to the link distance (in km) when active or zero when no LoS is available. At $t = 0$ s, u_{11} has links with v_{10} and v_9, preferring the closer v_{10}. At $t = 400$ s, it hands over to v_9, and by $t = 1000$ s it connects to v_8 as other satellites move out of view.

Table 4.2: Example of a user node's (u_i) contact table.

t (s)	v_{10}	v_9	v_8
0	1672	3405	0
100	2188	2808	0
200	2769	2246	0
300	3376	1759	0
400	0	1431	0
500	0	1385	0
600	0	1644	3660
700	0	2096	3072
800	0	2639	2516
900	0	3225	2023
1000	0	0	1660

4.2.3 Network Emulation

MeteorNet uses Mininet to instantiate a virtual network of OpenFlow switches and hosts, controlled via an external SDN controller based on ONOS [41]. Links are created as software interfaces with fixed capacities and time-varying delays derived from node distances and signal-propagation speeds. Figure 4.2 illustrates the conceptual composition of an emulated connection and the modules involved in the virtual network.

In MeteorNet, a node in the emulated topology is represented by a networking element—an Open vSwitch instance—and, when application logic is required, an associated host container. User sites, gateways, and edge-server satellites are therefore modeled as a switch–host pair: the switch provides OpenFlow-controlled connectivity, and the Docker container provides the execution environment for traffic sources/sinks and computation services. Relay-only satellites, which forward traffic but do not execute application workloads, are modeled as a single OpenFlow switch without an attached service container. This split clarifies why satellites and links may be depicted as Open vSwitch elements in Figure 4.2, while containerized hosts are used to emulate real software stacks at nodes that generate, terminate, or process traffic.

The time-varying nature of LoS and ISL connectivity implies that each satellite switch establishes variable connections with neighboring satellites and visible user–gateway nodes over time. Multiple switch connections could otherwise lead to forwarding loops due to redundant paths. To circumvent this, MeteorNet integrates an ONOS controller and recomputes shortest paths on demand whenever a link state changes, editing corresponding OpenFlow rules to ensure consistent routing and loop avoidance. Alternative SDN

controllers may also be used, provided they support the OpenFlow switch protocol. In this role, ONOS monitors the emulated topology and programs forwarding tables, while the MEC orchestration logic remains confined to edge-server activation and performance measurement.

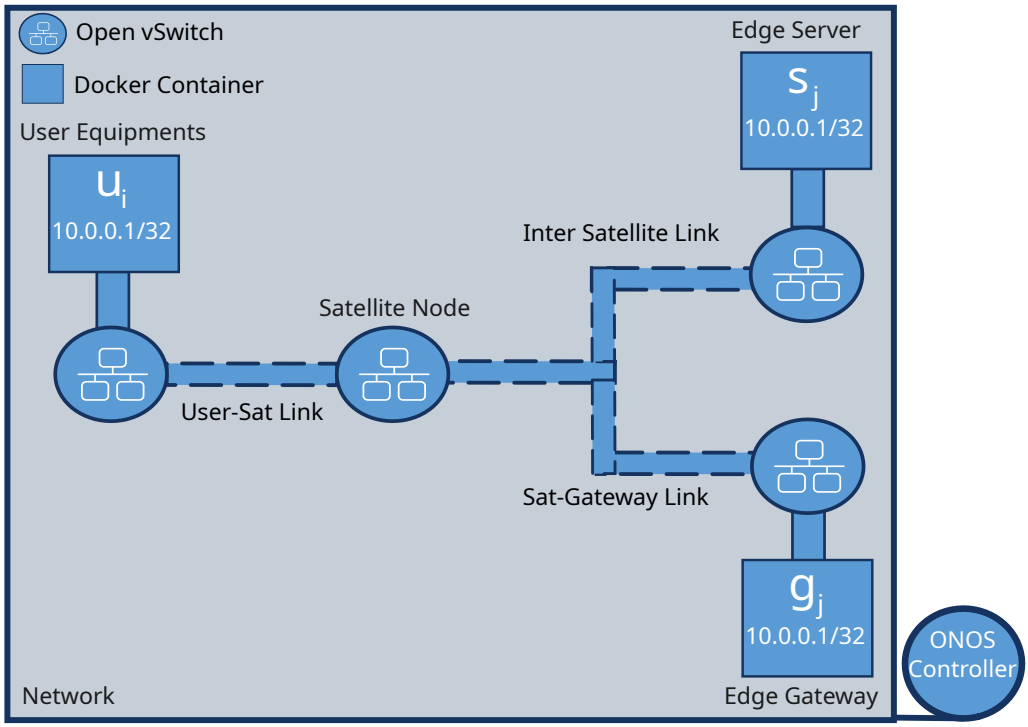


Figure 4.2: MeteorNet communication-links diagram.

4.2.4 Containerized Execution

Each user, gateway, and compute-capable satellite node runs an independent Docker container that encapsulates its software stack and allocated virtual resources. This ensures isolation between processes and enables direct execution of production-grade code, such as flight software prototypes, mission-control logic, or machine-learning workloads. In the MEC experiments of this thesis, containers primarily host workload generation (`TaskGenerator`), computation (`EdgeServer`/`TaskProcessing`), and telemetry exporters used for monitoring and logging.

Containerized services are coupled to the emulated network via virtual interfaces attached to the corresponding OpenFlow switch. Thus, application traffic enters the Mininet topology through the node's switch, traverses the time-varying constellation graph under SDN-programmed forwarding, and reaches the destination container for

processing. Containers can be dynamically instantiated or paused to reflect edge-server activation policies, enabling direct evaluation of ESA decisions under realistic network dynamics. Together with Mininet, this setup provides full-stack emulation from network forwarding to application execution, while the physical layer is approximated through propagation models, Equations (4.1)–(4.2). This integration allows networking behavior, orchestration policies, and computation mechanisms to be evaluated in a unified and reproducible environment.

4.2.5 Performance Monitoring and Metrics Database

MeteorNet continuously collects telemetry data, including throughput, delay, and CPU utilization, which are stored in a MongoDB database. Each emulation run records network states, server loads, and latency distributions, allowing post-processing for performance analysis and model validation. This monitoring layer is essential for generating high-fidelity synthetic datasets used to train and evaluate intelligent orchestration algorithms. The integration of real-time data collection and offline analysis provides a complete feedback loop between emulation, evaluation, and learning-based control strategies.

4.2.6 MEC Use Case

To demonstrate its capabilities, MeteorNet was used to emulate MEC orchestration in a LEO constellation where ground users offload computational tasks to space-based servers. Experiments compared Fuzzy Logic (FL) and Reinforcement Learning (RL) controllers, showing improved energy efficiency and reduced latency [45]. Further evaluations quantified latency trade-offs between terrestrial and in-orbit server placement, identifying operating regimes in which distributed space computing outperforms terrestrial clouds [46]. Beyond this specific use case, MeteorNet supports complex workflows such as IoT networks, disaster monitoring, and hardware-in-the-loop testing. These features make it a versatile platform bridging the gap between terrestrial emulation and operational space missions.

4.3 Emulation Testcase Validation

To validate the fidelity and scalability of the emulation stack, we define a controlled testcase that exercises the platform along the two dimensions that most directly affect

end-to-end MEC performance: (i) channel emulation correctness: propagation-delay reproduction over multi-hop paths and (ii) system-level robustness: under concurrent container execution (service workload, TCP dynamics, and SDN path setup). The testcase is intentionally small enough to be analytically tractable, yet sufficiently realistic to expose performance degradation modes when many containers and virtual links share the same host.

4.3.1 Scenario and Experimental Setup

We study a ring constellation with ten satellites $\{s_1, \ldots, s_{10}\}$ deployed in a single orbital plane and two ground stations $\{g_{11}, g_{12}\}$. The orbital parameters are: inclination 90°, Right Ascension of the Ascending Node (RAAN) 144°, eccentricity 0, and single-link mode for ground access. All satellites maintain inter-satellite connectivity with their two nearest neighbors, yielding a time-varying but sparse mesh.

The emulation runs on a laptop equipped with an Intel Core i7 CPU (4 cores, 8 threads) and 16 GB RAM. We evaluate two test applications that cover both pure network measurement and application-level MEC behavior:

1. **ICMP ping validation:** periodic ICMP echo requests from g_{11} and g_{12} to satellite s_2. We measure the Round-Trip Time (RTT) and compare it against the analytical minimum derived from the propagation-delay model (Equation (4.2)) and a per-hop switching overhead.

2. **MEC task offloading validation:** two task generators located at g_{11} and g_{12} offload tasks via TCP to a MEC server hosted on s_2. Each ground station emulates a population of 200 devices generating tasks according to a Poisson process with a mean rate of two tasks per minute per device. The MEC server queues tasks and serves them using parallel workers; the per-task processing time is approximately 250 ms on one CPU core. We measure the *task completion time* as the elapsed time between task transmission at the ground station and reception of the computed result. Because TCP introduces acknowledgment-driven dynamics and SDN requires time to establish valid forwarding paths after topology events, this metric is expected to exceed the analytical propagation-plus-processing lower bound.

In addition, we monitor per-interface link utilization along the routing paths between ground stations, access links, and inter-satellite links using sFlow telemetry.

4.3.2 Initial State and Analytical Delay Baseline

At the emulation start time ($t = 0\,\text{s}$), g_{11} and g_{12} are connected through satellites s_7 and s_9, respectively, and each satellite is connected to its two nearest neighbors. Tasks transmitted from g_{11} traverse the path s_7–s_6–…–s_2 (five hops), yielding an end-to-end path length of approximately 23,000 km. Tasks transmitted from g_{12} traverse the path s_9–s_8–…–s_2 (seven hops), with an end-to-end path length of approximately 32,200 km. Paths are selected by the spanning-tree logic implemented in the SDN control process, which ensures loop-free forwarding under the ring topology.

Using Equation (4.2) and assuming a per-link switching overhead of $\delta_0 = 4\,\text{ms}$, the minimum ping RTT for g_{11} can be approximated as

$$\text{RTT}_{g_{11}}^{\min} \approx 2 \cdot \left(\frac{23{,}000}{C} + 5 \cdot \delta_0 \right) \text{ms} \approx 172.72\,\text{ms}, \tag{4.4}$$

where C is the speed of light. Figure 4.3 depicts the initial routing configuration, and Figure 4.4 reports measured ping RTTs. At $t = 0$, the measured RTT is approximately 193 ms, which is consistent with the analytical bound once additional kernel, container, and emulation overheads are taken into account. This gap is informative: it quantifies the baseline cost of virtual switching, host scheduling, and measurement instrumentation under the chosen hardware constraints.

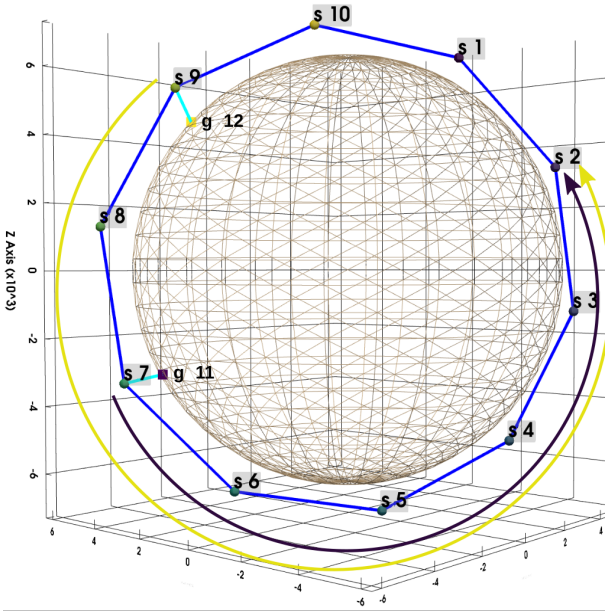


Figure 4.3: Network testcase at the start time ($t = 0$). The MEC server is hosted on s_2, on the far side of the ring constellation.

4.3.3 State Evolution and Delay Reproduction

During intervals in which a ground station remains connected to the same access satellite, both ping RTTs and task completion times evolve smoothly. Small increments are observed as the access distance grows within a contact window, which matches the expected behavior of a propagation-driven delay model. Across these windows, the mean ping RTTs remain close to the analytical predictions, indicating that the emulated link delays track the intended propagation-delay profile under changing geometry.

The task offloading experiment highlights the difference between *channel-delay fidelity* and *application-level completion time*. In an idealized case for g_{11}, the mean task completion time would be approximately the sum of the propagation RTT and the fixed processing time, i.e., around 172.72 ms + 250 ms ≈ 423 ms. In practice, Figure 4.4 shows substantially larger completion times, and the gap increases with the number of hops on the routing path. This behavior is expected in a full-stack emulation that includes TCP reliability mechanisms: acknowledgments, congestion control, and retransmissions introduce additional waiting time before the application-level request–reply cycle completes. Moreover, topology events can trigger transient path reconfiguration, during which flows may experience short-lived disruption before forwarding rules converge. Therefore, the measured completion-time inflation provides a realistic indication of how transport protocols and SDN dynamics compound over multi-hop LEO paths, especially when many containers share CPU time and network interfaces on a single host.

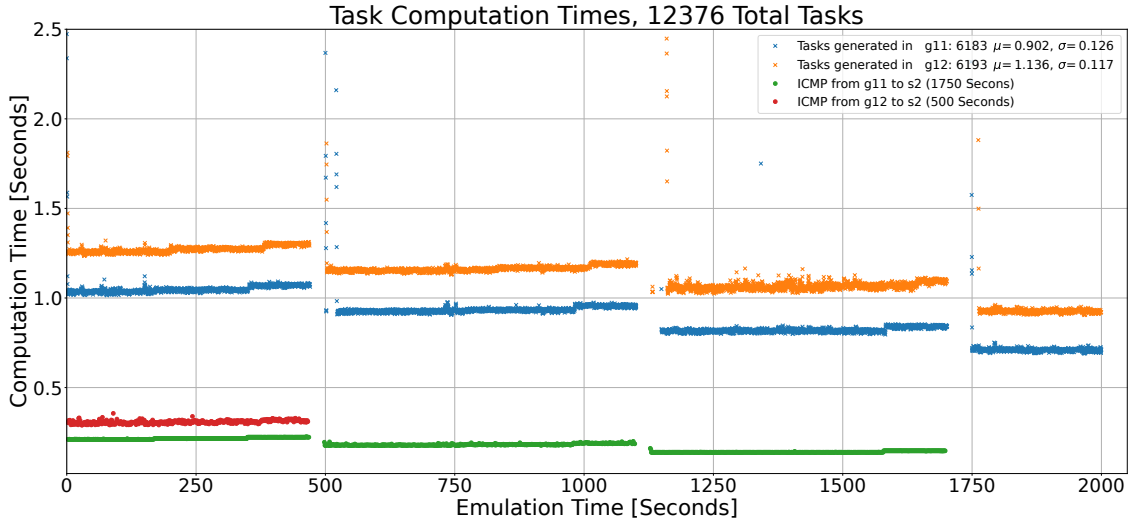


Figure 4.4: Task completion times for the testcase applications.

4.3.4 Interface Utilization and Scalability Signals

Figure 4.5 reports per-interface utilization measured via sFlow over approximately 30 minutes of emulation time. The traces reveal which links become bottlenecks under sustained traffic and reflect route changes across contact windows. In addition to validating that traffic concentrates along the expected shortest/loop-free paths, these measurements provide a practical way to diagnose scalability limits: when many containers and virtual links run on a single host, CPU contention and virtual switching overhead can manifest as elevated queueing, increased RTT variance, and more pronounced utilization spikes on specific interfaces. The availability of per-interface telemetry thus supports both debugging and the construction of synthetic datasets for learning-based predictors (e.g., interface utilization forecasting) under controlled constellation dynamics.

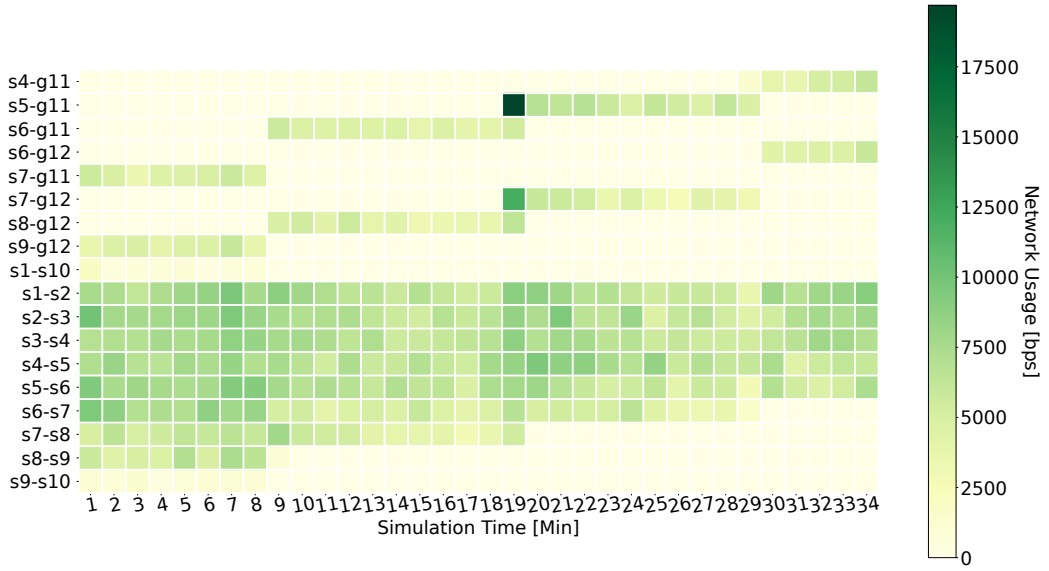


Figure 4.5: Per-interface network usage measured via sFlow during the testcase.

4.3.5 Discussion

Overall, the testcase confirms two complementary properties of the platform. First, propagation-driven delay trends are reproduced at the packet level with measured RTTs that remain close to the analytical lower bounds, up to a consistent virtualization overhead. Second, application-level metrics naturally deviate from purely analytical models once reliable transport, SDN path setup, and host resource contention are included. This is a desirable property for the present thesis: ESA/ESP strategies are evaluated under realistic

end-to-end behavior rather than under idealized delay-only assumptions, and performance degradations caused by scaling the number of containers become observable through completion-time inflation and interface-level telemetry. As a result, the testcase serves as both a correctness check for link-delay emulation and a practical indicator of how host-side resource limits influence fidelity when scaling experiments on commodity hardware.

4.4 Summary

This chapter has presented the design and implementation of MeteorNet, a continuous-time emulation framework for edge computing in Low Earth Orbit constellations. By integrating orbital propagation, SDN-based network virtualization, and containerized execution environments, MeteorNet enables realistic experimentation and the collection of high-fidelity performance data. Its modular design supports diverse scenarios, from dynamic link adaptation to autonomous edge orchestration, while remaining open and reproducible for future research. The next chapter applies this framework to evaluate the proposed orchestration and server placement strategies, providing quantitative analysis of latency, energy efficiency, and system reliability under realistic constellation conditions.

CHAPTER 5

Evaluation and Results

In this chapter, we evaluate the performance of the controllers and strategies applied to the problems studied: 1) Edge Server Activation (ESA) and 2) Edge Server Placement (ESP), within the broader context of Multi-access Edge Computing (MEC) resource orchestration in space systems. For each case, we design baseline constellations and evaluate key optimization metrics.

5.1 Edge Server Activation Results

5.1.1 Baseline Constellation Scenario

We evaluate ESA strategies on an emulated MEC–LEO constellation consisting of $V = 10$ satellite nodes ($v_j \in \mathcal{V}$) in the same orbital plane, interconnected through neighbors to form a ring topology, as shown in Figure 5.1. The constellation serves a variable number of user stations, ranging from $U = 3$ to $U = 12$, which act as switches for end-user devices. These user stations are arranged in a quasi-circumferential pattern on the Earth's surface, providing evenly distributed demand across the service area. User nodes represent a variable number of devices, with a total sum of $|\mathcal{D}|$ that varies from 300 to 1800 across the test cases, allowing assessment under different traffic conditions. The resulting traffic is summarized by the mean task generation rate $\overline{\lambda}$, which ranges between 10 and 60 tasks per second, representing scenarios from light to heavy load. All tasks have a constant program size of Pr = 200,000 million instructions (MI), and each server has a processing capacity of $H = 800{,}000$ million instructions per second (MIPS). Each emulation runs for $T = 1000$ seconds, divided into control windows of $\Delta t = 15$ seconds. The full set of parameters is summarized in Table 5.1.

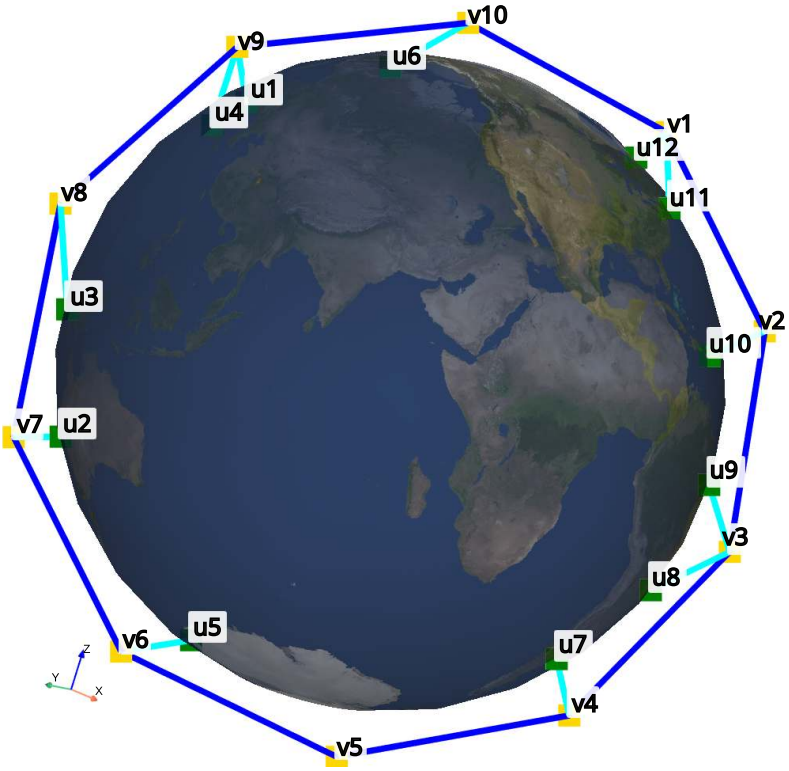


Figure 5.1: Baseline constellation for ESA with 10 satellites and 12 users.

Table 5.1: Constellation parameters for ESA test cases.

Parameter	**Value**	**Description**
V	10	Number of satellite nodes.
U	3, 6, 9, 11, 12	Number of ground nodes.
$\lvert\mathcal{D}\rvert$	300, 600, 900, 1100, 1650, 1800	Number of devices.
T	1000 [s]	Total emulation time.
Δt	15 [s]	Control window duration.
$\bar{\lambda}$	10, 20, 30, 36, 55, 60 [tasks/s]	Mean task rate over all devices.
Pr	200,000 [MI]	Constant program size for all tasks.
H	800,000 [MIPS]	Constant processing capacity for all servers.

We evaluate the performance of four ESA strategies—Access, Bernoulli, Fuzzy Logic (FL), and Reinforcement Learning (RL) controllers—using the metrics defined in Section 3.2: Activation Ratio ($\overline{K}$), Failure Rate ($\overline{B}$), and Utilization Ratio ($\overline{C}$). Figures 5.2, 5.4, and 5.3 summarize the comparative results.

5.1.2 Active Time Ratio

The Activation Ratio results in Figure 5.2 show that the Bernoulli controller maintains a nearly constant level of server activation independent of demand, reflecting its random nature. The Access controller activates more servers as load increases, producing higher $\overline{K}$ values across all scenarios. Its policy keeps roughly half of the servers active on average, yielding $\overline{K} \approx 0.5$ regardless of task rate. Both the FL and RL controllers achieve a balanced activation strategy, consistently maintaining fewer active servers than the Access controller while avoiding the inefficiency of Bernoulli randomization. Considering $\overline{K}$ across task rates, the FL and RL controllers achieve lower activation than the Access baseline for all loads, meaning they use fewer computational resources to process comparable demand. This is expected, since both FL and RL explicitly optimize the activation policy, whereas Access follows a greedier strategy and Bernoulli is random and input-independent. FL and RL trends are similar because both target the same objective with different mechanisms.

5.1.3 Task Failure Rate

Failure-rate trends highlight the limitations of both baselines. There is a clear trade-off between activation ($\overline{K}$) and failures ($\overline{B}$) for all controllers. As the active server ratio increases, the system can process more tasks successfully, thereby reducing $\overline{B}$. The Bernoulli controller's constant $\overline{K}$ leads to rising $\overline{B}$ once the offered load surpasses the available compute capacity. We observe a sharp increase in $\overline{B}$ for Bernoulli when $\overline{\lambda} > 20$ tasks/s, reaching a maximum of approximately 0.3 at $\overline{\lambda} = 60$. This confirms that a demand-agnostic policy overworks capacity at low load and under-provisions at high load. The Access controller reduces failures relative to Bernoulli by aggressively activating servers, but at the cost of higher resource usage. By contrast, the RL controller consistently achieves the lowest $\overline{B}$, adapting activation to demand and minimizing drops.

5.1.4 Utilization Ratio

Figure 5.3 shows that the RL controller maximizes processing efficiency by aligning activations with demand. The Access controller, while reliable, suffers from lower utilization as many servers remain active but idle. The Bernoulli controller underperforms in both utilization and reliability, confirming the inefficiency of purely stochastic orchestration. Both FL and RL maintain higher $\overline{C}$ than Access across all task rates, indicating better per-server productivity when active.

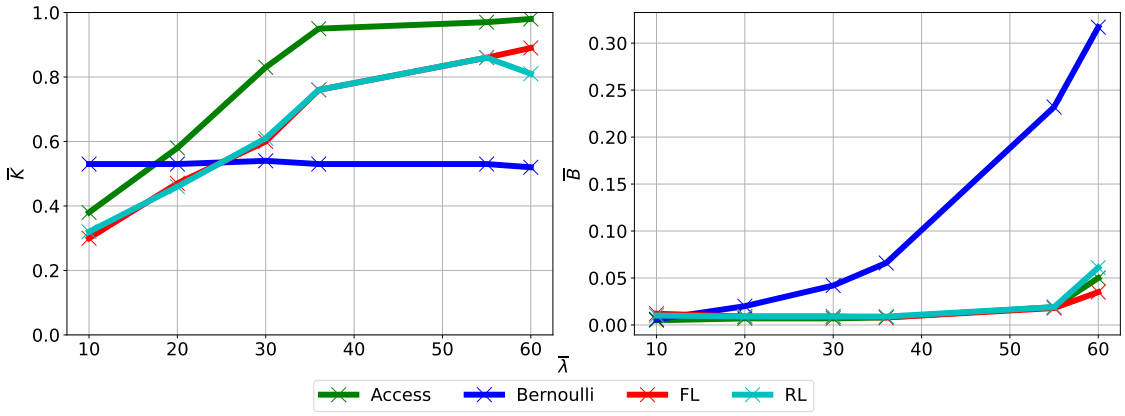


Figure 5.2: Activation Ratio $\overline{K}$ and Failure Rate $\overline{B}$ across increasing task rates $\overline{\lambda}$.

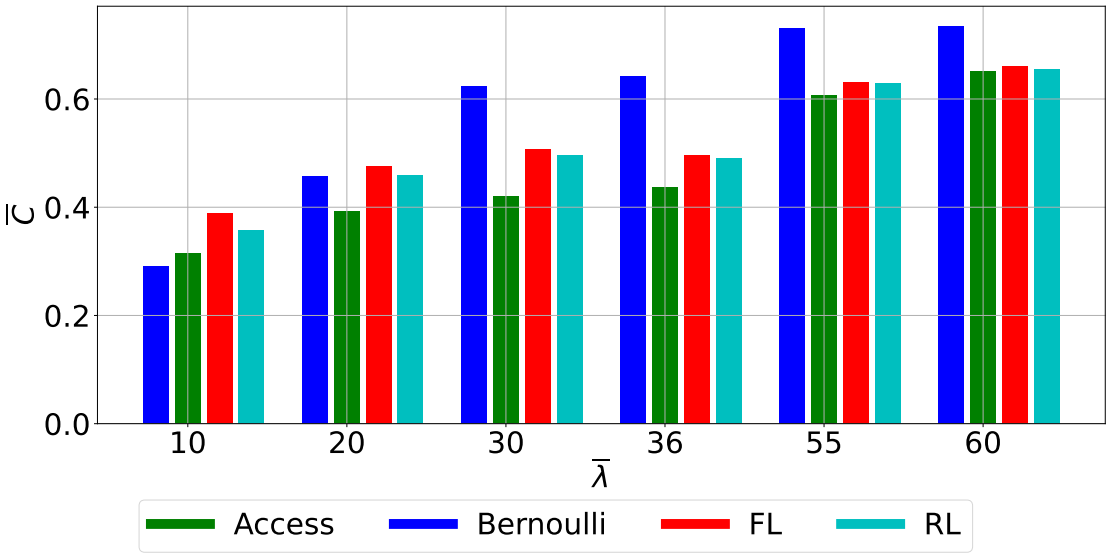


Figure 5.3: Utilization Ratio $\overline{C}$ across increasing task rates $\overline{\lambda}$.

Table 5.2: KPI summary across controllers and task rates.

	Access			Bernoulli			FL			RL		
$\overline{\lambda}$	$\overline{K}$	$\overline{B}$	$\overline{C}$	$\overline{K}$	$\overline{B}$	$\overline{C}$	$\overline{K}$	$\overline{B}$	$\overline{C}$	$\overline{K}$	$\overline{B}$	$\overline{C}$
10	0.38	0.005	0.31	0.53	0.006	0.29	0.30	0.012	0.38	0.32	0.010	0.36
20	0.58	0.007	0.39	0.53	0.020	0.46	0.47	0.009	0.48	0.46	0.009	0.46
30	0.83	0.007	0.42	0.54	0.042	0.62	0.60	0.009	0.51	0.61	0.009	0.50
36	0.95	0.008	0.43	0.53	0.066	0.64	0.76	0.008	0.50	0.76	0.009	0.49
55	0.97	0.019	0.60	0.53	0.232	0.73	0.86	0.018	0.63	0.86	0.019	0.63
60	0.98	0.050	0.65	0.52	0.317	0.74	0.89	0.035	0.66	0.81	0.061	0.66

5.1.5 Pareto Analysis and Qualitative Interpretation

To analyze the relation between K (Activation Ratio) and B (Failure Rate), Figure 5.4 presents a Pareto comparison across controllers for increasing $\overline{\lambda}$. Each point represents a single simulation outcome, and dotted lines indicate controller-wise averages. For $\overline{\lambda} < 20$, Bernoulli attains failure rates comparable to other controllers, but for higher loads its $\overline{B}$ rises quickly. Both FL and RL achieve lower $\overline{K}$ than Access at the expense of slightly higher $\overline{B}$ at moderate loads, which is often acceptable when energy efficiency is prioritized. For $\overline{\lambda} > 40$, demand exceeds total constellation capacity, causing Access to saturate and reach $\overline{K} \approx 1$, while FL and RL approach saturation more gradually. At $\overline{\lambda} = 60$, the three non-Bernoulli approaches converge in behavior, as all are capacity-limited. These results provide a practical lens to tune ESA policies to mission requirements. If modest increases in $\overline{B}$ are acceptable, FL and RL substantially reduce activation time and energy use. If low failure is paramount and energy budgets are less constrained, the Access controller offers lower $\overline{B}$ at the cost of resource efficiency. Comparing FL and RL, both yield comparable KPIs overall. FL tends to reduce active time at lower loads, whereas RL can achieve marginally lower failure in specific regimes. Beyond metrics, the trade-off is implementation. FL is lightweight, interpretable, and easy to deploy on constrained platforms but relies on expert-tuned rules. RL requires training but adapts to changing conditions and can explore richer policies, which is promising for future scaling.

The qualitative comparison in Table 5.3 summarizes the trade-offs among ESA controllers across three dimensions: resource consumption ($\overline{K}$), reliability ($\overline{B}$), and CPU utilization ($\overline{C}$). The Bernoulli controller performs poorly overall due to its demand-agnostic policy, which wastes capacity at low load and under-provisions at high load. The Access controller improves reliability by activating more servers, but does so at the expense of higher resource usage and lower per-server efficiency. In contrast, the intelligent controllers (FL and RL) achieve the best overall balance: they reduce active time while maintaining low failure rates and high utilization. Practically, FL is lightweight and interpretable—well suited to constrained platforms—whereas RL requires training but adapts more readily to changing conditions.

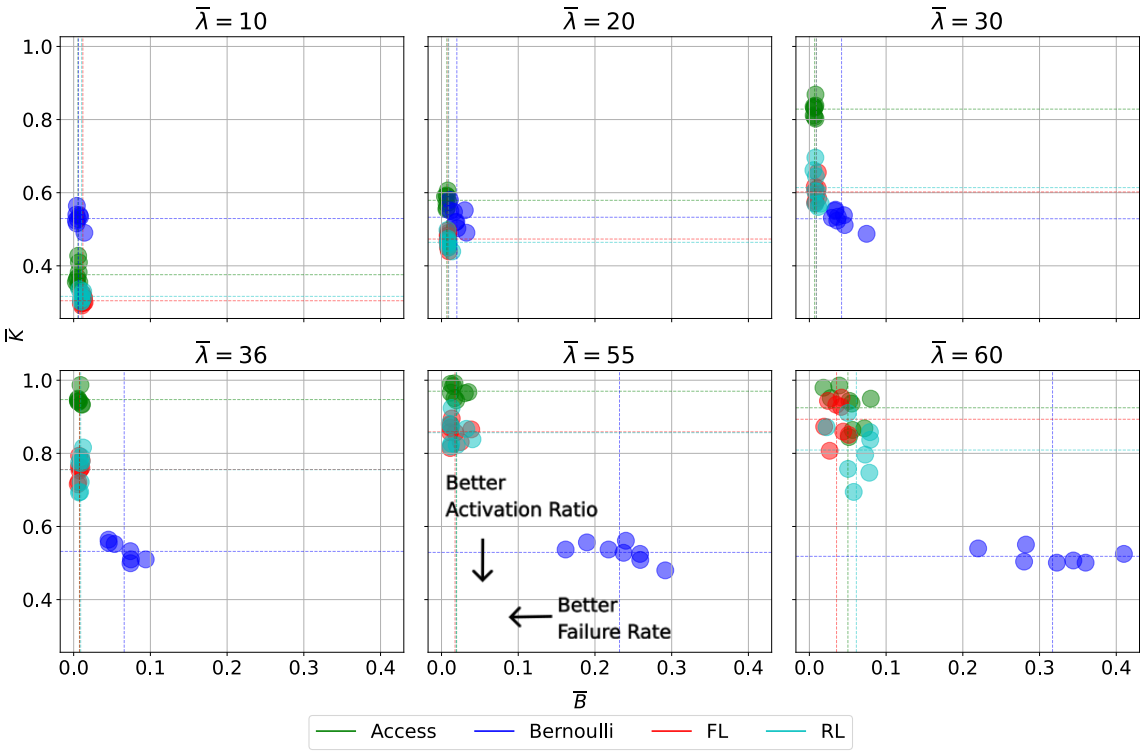


Figure 5.4: Pareto analysis between Activation Ratio $\overline{K}$ and Failure Rate $\overline{B}$ at various task rates $\overline{\lambda}$.

Table 5.3: Qualitative comparison of ESA strategies.

Metric	Bernoulli	Access	FL and RL
Resource Consumption ($\overline{K}$)	Medium	Poor	Good
Reliability ($\overline{B}$)	Poor	Medium	Good
CPU Utilization ($\overline{C}$)	Poor	Medium	Good

5.2 MEC Server Placement Results

5.2.1 Baseline Constellation Scenario

We now evaluate ESP results by comparing space, terrestrial, and hybrid server placements using round-trip time (RTT) measurements of user tasks generated from globally distributed locations. ESP test cases use a constellation of $|\mathcal{S}| = N = 198$ satellites ($s_n \in \mathcal{S}$) across $|\mathcal{P}| = 18$ orbital planes with mean inclination $i_o = 53°$. Additionally, we use up to $|\mathcal{G}| = G = 10$ gateways placed around the Earth—resembling *Kongsberg*

Satellite Services (KSAT) sites [47]—to account for terrestrial and hybrid infrastructures. Figures 5.5 and 5.6 show 3D and map views for the baseline scenario used for ESP. Table 5.4 summarizes the parameters of the reference scenario, while Table 5.5 lists gateway and user locations.

Table 5.4: Parameters and values for the ESP reference scenario.

Parameter	**Value**	**Description**
$\lvert\mathcal{S}\rvert = N$	198	Number of satellites.
$\lvert\mathcal{G}\rvert = G$	10	Number of gateways.
$\lvert\mathcal{U}\rvert = U$	9	Number of user locations (UEs).
$\lvert\mathcal{P}\rvert$	18	Number of orbital planes.
$\lvert\mathcal{D}\rvert$	10	Number of devices per user location.
i_o	53°	Orbital inclination.
θ_c	30°	Minimum elevation angle for LoS.
λ	2 tasks/min/device	Task generation rate per device.
T	90 minutes	Total emulation duration.
d_g	13 ms	Terrestrial cloud processing penalty.
T_p	250 ms	Task processing time.
T_l	1 ms	Per-link interface delay.

Table 5.5: Gateway and user locations.

Name / Location	**Latitude [°]**	**Longitude [°]**	**Name**	**Latitude [°]**	**Longitude [°]**	**Distance to g_1 [km]**
g_1 Puertollano	38.69	-4.11	u_1	40.35	-3.80	187
g_2 Awarua	-46.49	168.38	u_2	35.84	24.27	2522
g_3 Hokkaido	43.27	142.66	u_3	25.74	47.75	5018
g_4 Mauritius	-20.28	57.24	u_4	12.50	67.08	7520
g_5 Los Angeles	34.06	-118.28	u_5	-2.00	84.46	10025
g_6 Tolhuin	-54.51	-67.19	u_6	-16.30	102.14	12529
g_7 Singapore	1.29	103.84	u_7	-28.90	122.41	15030
g_8 Dubai	25.22	55.31	u_8	-37.73	147.24	17525
g_9 Panama	8.89	-79.01	u_9	-40.34	176.07	19837
g_{10} Nuuk	64.18	-51.75				

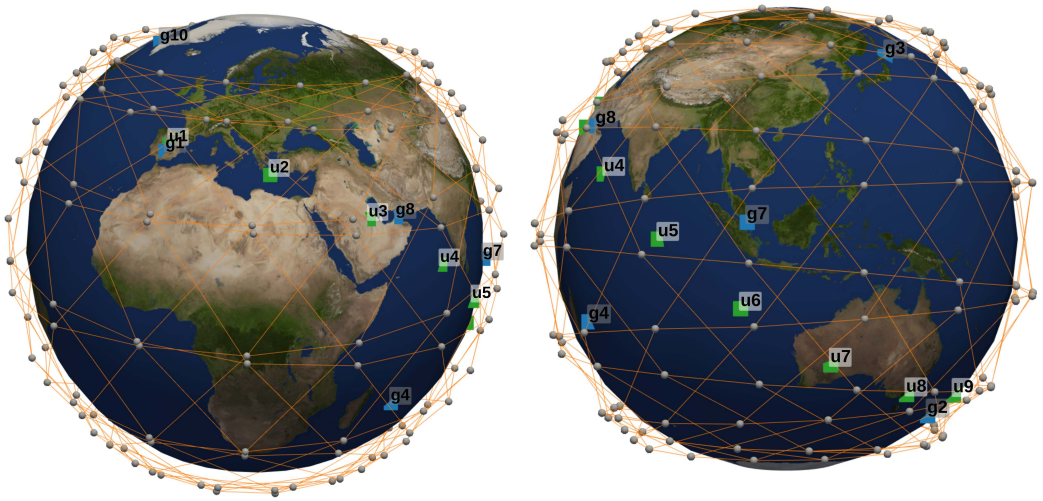


Figure 5.5: Baseline constellation for ESP with 198 satellites, 9 users, and 10 gateways.

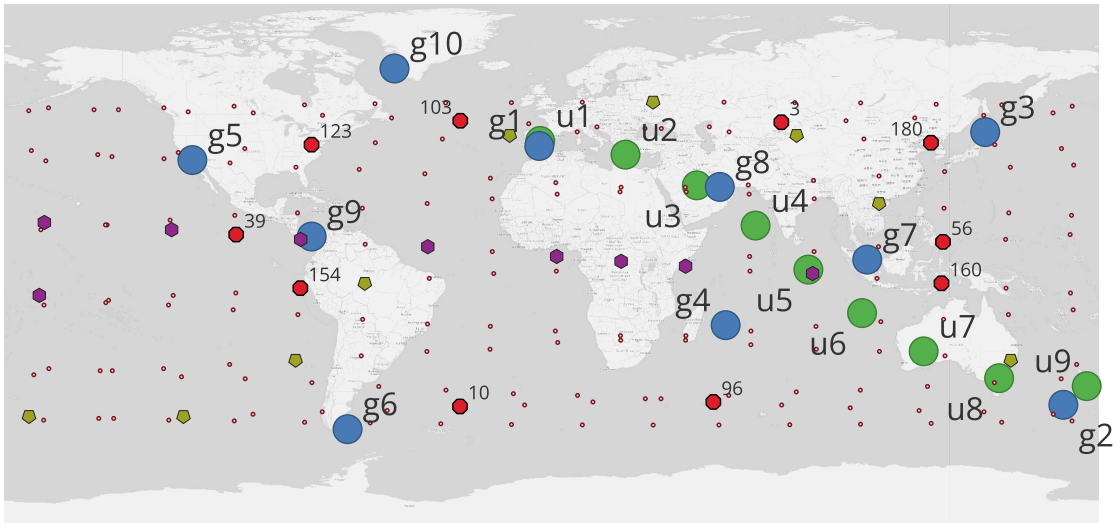


Figure 5.6: Space and terrestrial nodes on the map: ● ← $\mathcal{U}$, ● ← $\mathcal{G}$, ● ← FPS, ● ← CP, and ● ← AO.

The budget K represents the number of servers onboard satellites and/or gateways placed by each strategy and is a key comparison parameter representing the same total computational capacity. We compare the different strategies described in Chapter 3, ranging the budget from $1 \leq K \leq 10$. Table 5.6 outlines the node IDs used as edge servers for each cloud configuration. The base case is a terrestrial cloud with a single gateway located in Spain. UE locations are distributed progressively farther from this point, as shown in Table 5.5.

Table 5.6: Node IDs of servers across cloud configurations.

Distribution	K**=1**	K**=2**	K**=4**	K**=6**	K**=8**	K**=10**
Terrestrial	g: 1	g: 2	g: 3 4	g: 5 6	g: 7 8	g: 9 10
Space: FPS	s: 160	s: 56	s: 154 39	s: 96 103	s: 10 3	s: 123 180
Space: AO	s: 160	s: 159	s: 161 162	s: 158 163	s: 157 164	s: 156 165
Space: CP	s: 160	s: 6	s: 94 105	s: 50 138	s: 28 193	s: 72 116
Hybrid		s: 160 g: 1	s: 56 154	s: 39 g: 2	s: 96 103	s: 10 3

5.2.2 Round-Trip Time Analysis

Figure 5.7 shows box plots of R_i per user u_i, summarizing key statistics of task distributions. Results are organized by user locations in ascending order of geodesic distance from g_1, providing a clear perspective on performance variation. The yellow line indicates the median, box limits correspond to quartiles 1 and 3, and whiskers mark the minimum and maximum values. This ordering highlights the effect of distance on performance. Figure 5.9 summarizes mean RTT $\overline{R}$ and standard deviation for each distribution and K, complemented by Table 5.7, which highlights the best distribution in each case.

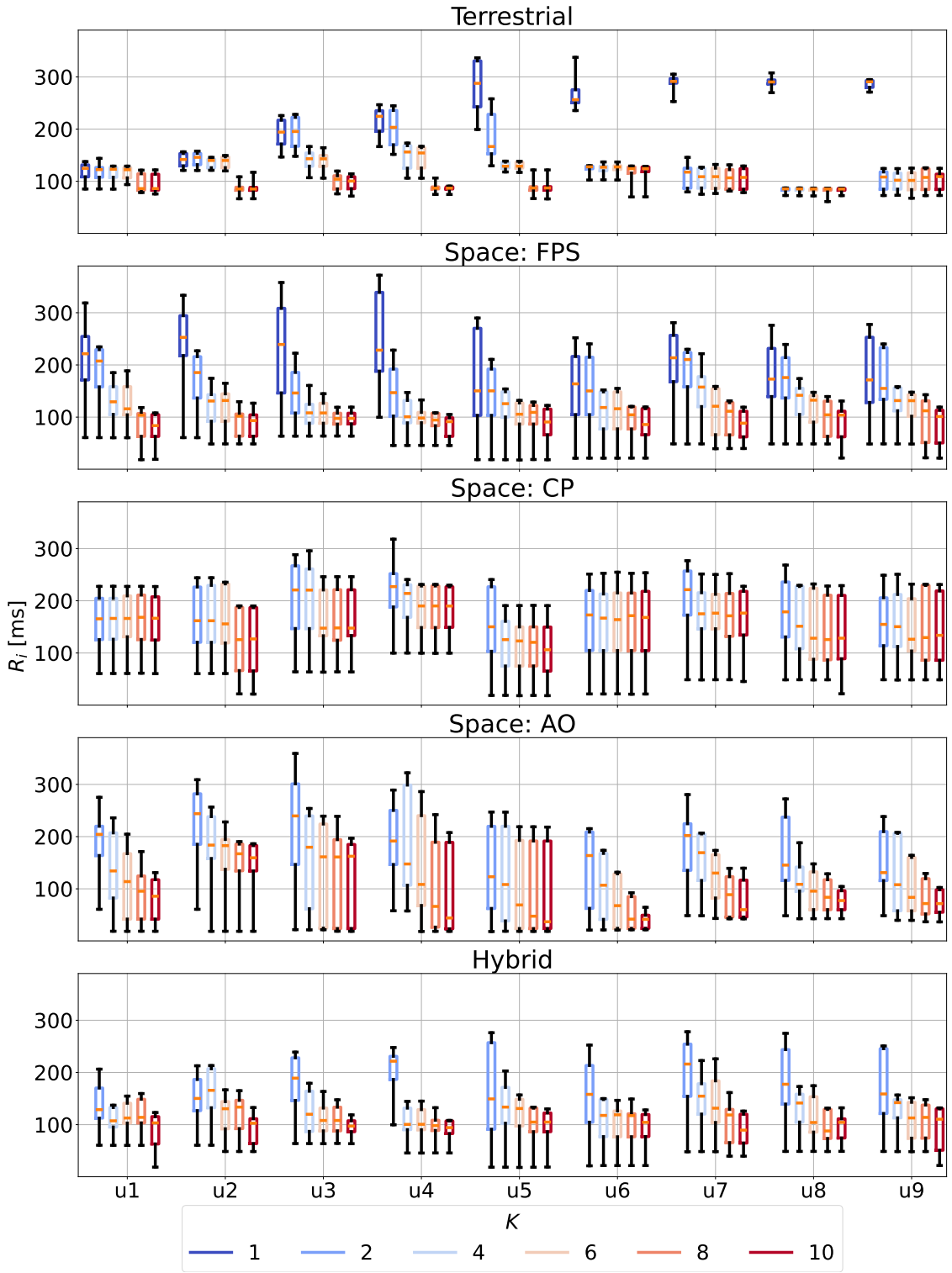


Figure 5.7: Box plots of R_i per user across values of K.

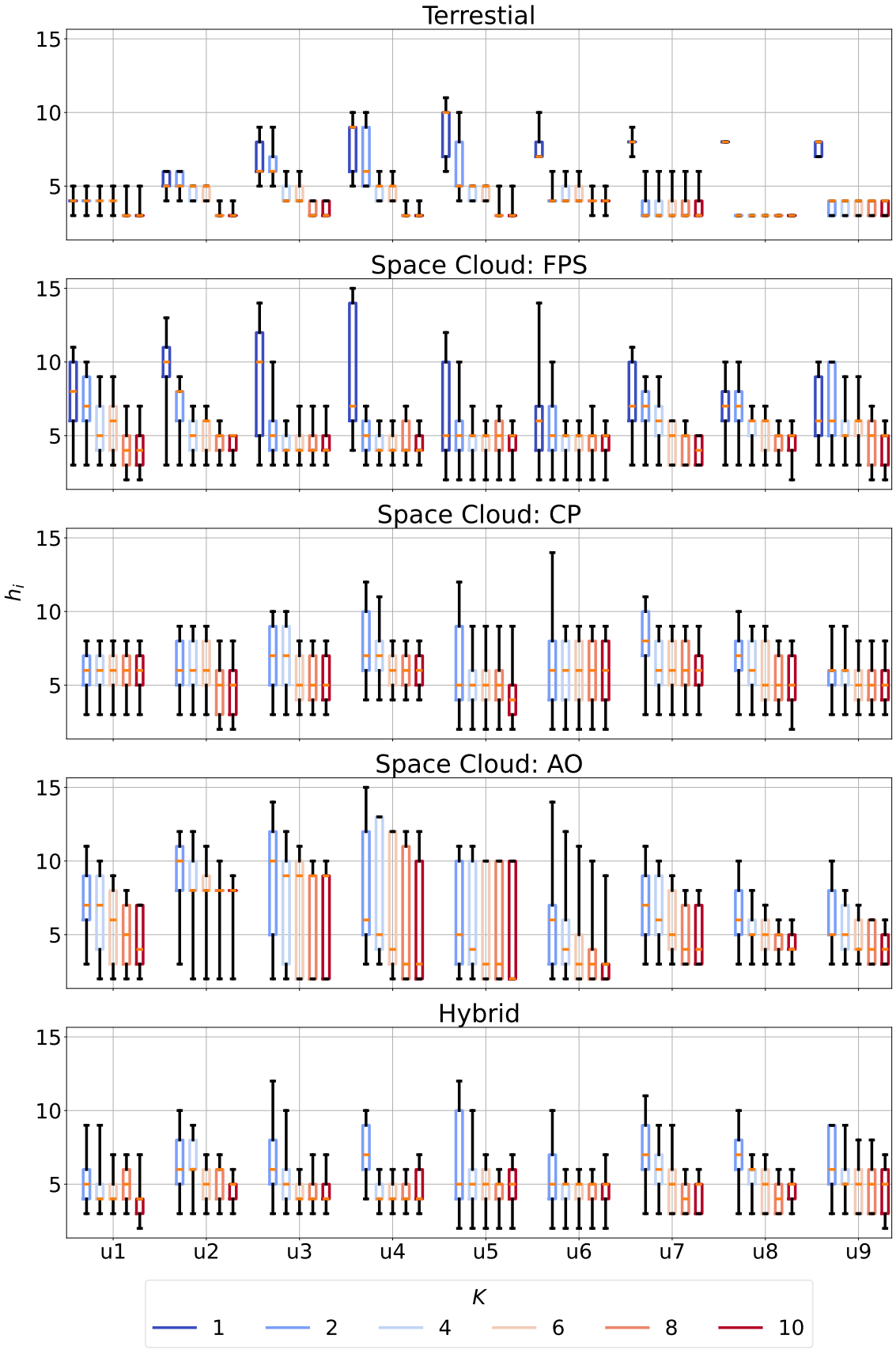


Figure 5.8: Box plots of hop counts per user across values of K.

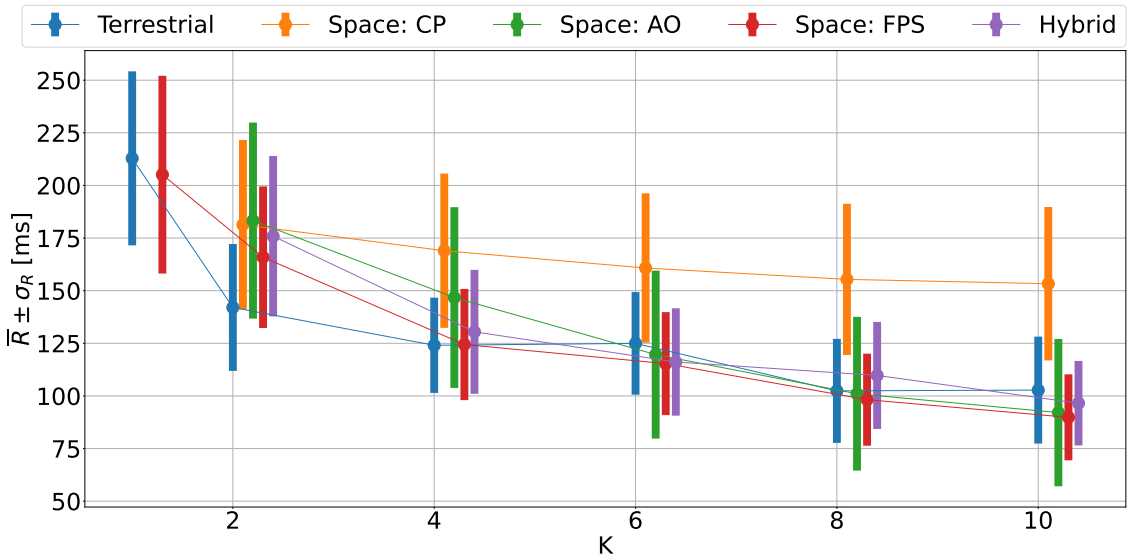


Figure 5.9: Mean RTT for the considered cloud configurations across values of K.

Table 5.7: Average RTTs and standard deviations for all edge server placements.

	Terrestrial		**Space: FPS**		**Space: AO**		**Space: CP**		**Hybrid**	
Servers	$\overline{R}$	σ_R	$\overline{R}$	σ_R	$\overline{R}$	σ_R	$\overline{R}$	σ_R	$\overline{R}$	σ_R
1	212.87	82.66	**205.13**	93.94	-	-	-	-	-	-
2	**142.03**	60.26	165.90	67.25	183.30	93.15	181.36	80.47	175.89	76.18
4	**124.06**	45.24	124.44	52.84	146.74	85.85	168.99	73.29	130.45	58.89
6	124.97	48.74	**115.36**	48.92	119.64	79.79	160.77	70.98	116.14	50.99
8	102.41	49.44	**98.23**	43.75	101.04	73.01	155.38	71.84	109.74	50.82
10	102.78	50.80	**89.84**	40.84	92.07	69.99	153.31	72.89	96.56	40.14

Single Server or Gateway

In a terrestrial cloud with one gateway, RTT increases with the user's distance from g_1 and with the number of ISL hops $h_{i,j}$ to the server. Variance is higher for mid-latitude users u_4, u_5, and u_6, likely due to denser node spacing and greater path variability. A space cloud with a single server shows more stable RTT across user locations. Placing the server near the average user distance balances delays. On average, the space cloud slightly outperforms the terrestrial cloud (205 ms vs. 213 ms), although terrestrial exhibits marginally lower variance for users near g_1.

Multiple Servers or Gateways

As K increases, all configurations reduce mean $\overline{R}$ and variability. Terrestrial clouds benefit users nearer gateways, while space clouds improve global reach. In general:

Cross Plane (CP) underperforms due to poor path efficiency.

Terrestrial clouds perform best up to $K = 4$.

FPS becomes the best performer for $K \geq 6$.

Along Orbit (AO) is competitive for small K but shows higher variance than FPS or terrestrial.

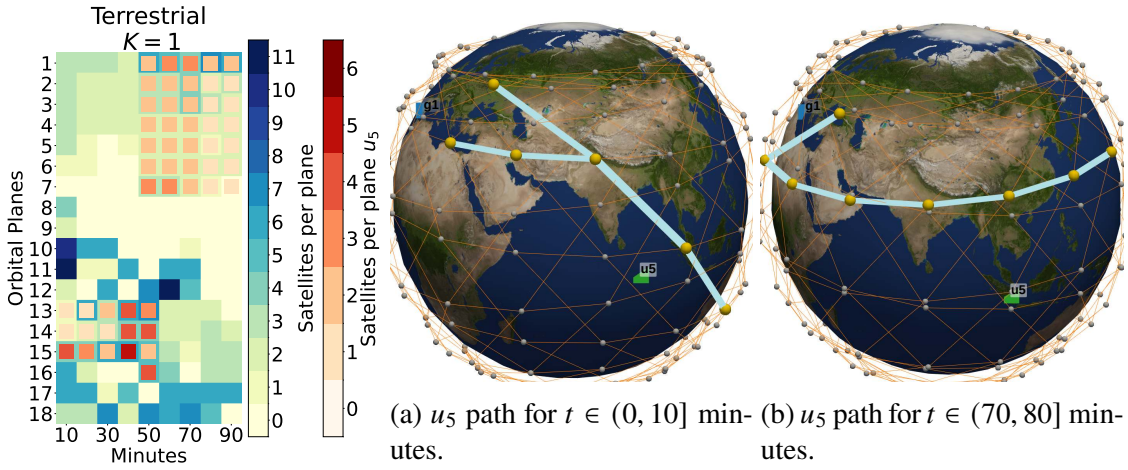
These results confirm the importance of topology-aware placement such as FPS.

Hybrid Distribution Performance

Hybrid configurations combining terrestrial and in-orbit servers provide consistent R_i across users. While not always yielding the lowest mean $\overline{R}$, they reduce variance and thus provide more uniform QoS for geographically dispersed demand. Notably, with $K = 6$, the hybrid configuration is competitive with the best FPS cases while improving fairness across locations.

5.2.3 Network Topology Analysis

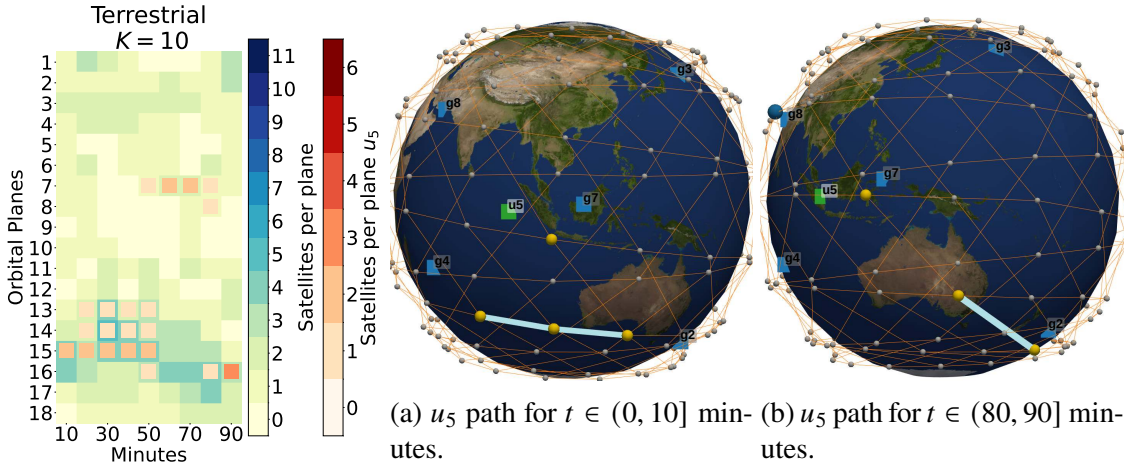
Figures 5.10, 5.11, 5.12, and 5.13 visualize satellite usage per orbital plane in 10-minute intervals for terrestrial and space (FPS) deployments with $K \in \{1, 10\}$. Color intensity indicates usage: green–blue for all user traffic and red for u_5 only. These plots show how placement affects ISL routing and satellite load.



(a) u_5 path for $t \in (0, 10]$ minutes.

(b) u_5 path for $t \in (70, 80]$ minutes.

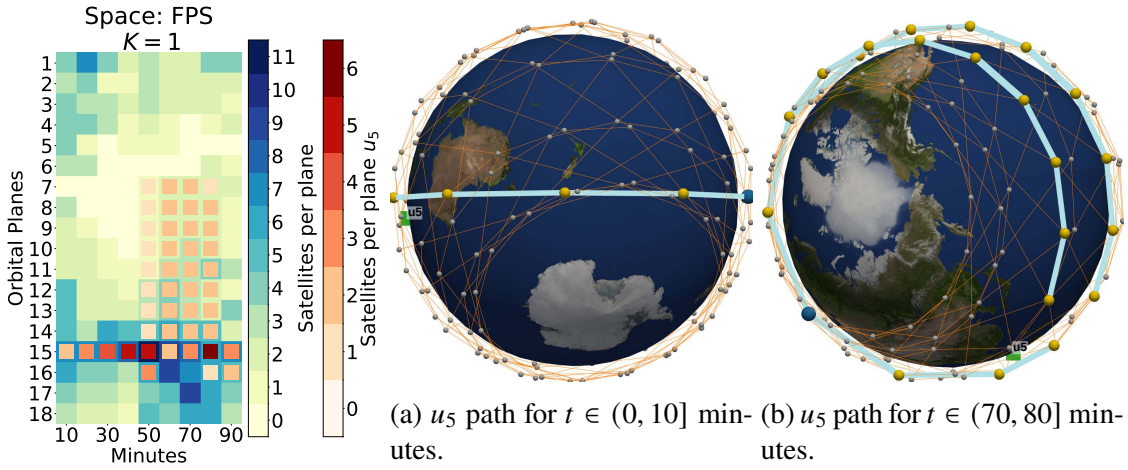
Figure 5.10: Network usage for the terrestrial cloud, $K = 1$.



(a) u_5 path for $t \in (0, 10]$ minutes.

(b) u_5 path for $t \in (80, 90]$ minutes.

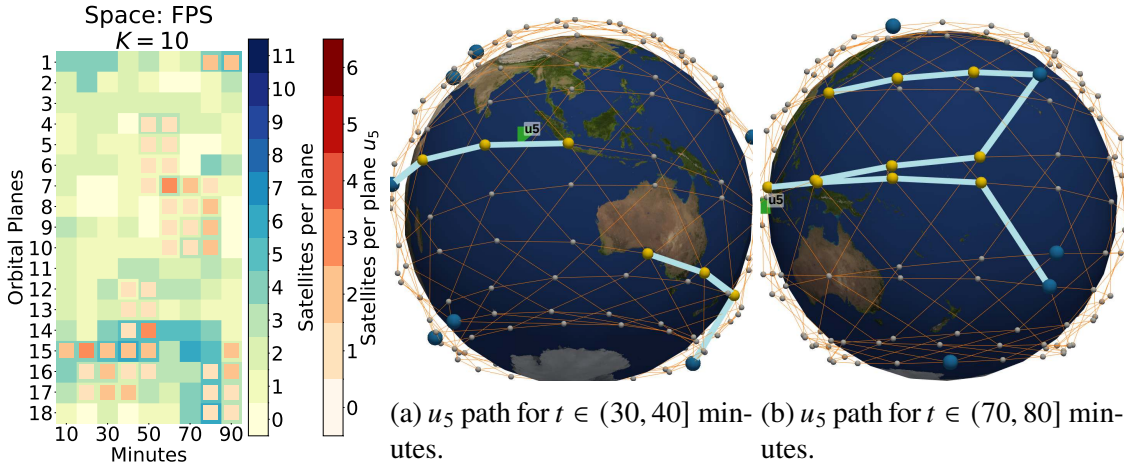
Figure 5.11: Network usage for the terrestrial cloud, $K = 10$.



(a) u_5 path for $t \in (0, 10]$ minutes.

(b) u_5 path for $t \in (70, 80]$ minutes.

Figure 5.12: Network usage for the space cloud (FPS), $K = 1$.



(a) u_5 path for $t \in (30, 40]$ minutes.

(b) u_5 path for $t \in (70, 80]$ minutes.

Figure 5.13: Network usage for the space cloud (FPS), $K = 10$.

With a single terrestrial gateway (Figure 5.10), u_5 traffic is routed through a narrow corridor of satellites, concentrating load on a few planes and creating longer paths. With $K = 10$ terrestrial gateways (Figure 5.11), traffic distributes more evenly, reducing congestion. For one space server (Figure 5.12), usage concentrates near the server, creating long paths for distant users. With $K = 10$ space servers via FPS (Figure 5.13), usage spreads across planes, reducing routing distances and balancing ISL load. In both terrestrial and space clouds, increasing K alleviates congestion, shortens paths, and benefits difficult locations such as near the equator.

5.2.4 Key Findings for ESP

The evaluation underscores several insights regarding edge-server distribution in LEO constellations.

Space clouds are viable and scalable. Even a single in-orbit server can outperform a terrestrial gateway for globally distributed users, supporting the feasibility of the Space Cloud paradigm.

Placement strategy matters. Farthest Point Sampling (FPS) consistently achieves the lowest average RTT for $K \geq 6$, demonstrating the value of topology-aware placement.

Hybrid architectures offer robustness. While not always minimizing mean RTT, they reduce variance and improve fairness across user locations.

Cross-plane distribution underperforms. Despite coverage benefits, CP often yields longer paths and higher RTT, making it less suitable for latency-sensitive services.

Increasing K mitigates congestion. More servers reduce hop counts and ISL load, improving scalability and QoS.

These findings validate the use of real-time emulation to assess design choices for future space–ground edge-computing infrastructures.

5.3 Summary

This chapter presented experimental results for ESA and ESP. Through controlled emulation, we analyzed FL and RL activation policies under dynamic conditions and compared terrestrial, space-based, and hybrid placements. Results show that intelligent control can reduce energy consumption and failure rates while maintaining service reliability, supporting autonomous orchestration as a viable approach for space-edge computing. Placement comparisons further revealed regimes in which hybrid strategies balance latency and accessibility. Building on these findings, the next chapter concludes this thesis by summarizing the main contributions, discussing implications for future NTNs, and outlining directions for the design and control of the Space Cloud.

CHAPTER 6

Conclusions and Future Work

This thesis has explored the design, control, and evaluation of edge computing architectures in Low Earth Orbit (LEO) satellite constellations, contributing to the broader vision of the emerging Space Cloud. Starting from the concept of Multi-access Edge Computing (MEC) resource orchestration applied to Non-Terrestrial Networks (NTNs), we investigated two fundamental challenges: (1) the dynamic activation of edge servers, the Edge Server Activation (ESA) problem, and (2) the optimal placement of servers across space and terrestrial domains, the Edge Server Placement (ESP) problem.

To address these challenges, we formalized both problems as Mixed-Integer Linear Programming (MILP) models that capture operational and temporal constraints under space-domain conditions. Because of their combinatorial nature, both problems are NP-hard, making exact methods impractical at constellation scale. We therefore designed heuristic and learning-based strategies that approximate data-driven optimal decisions in real time while adapting to dynamic topologies and variable workloads. Additionally, we developed the MeteorNet emulation framework as a reproducible software environment for testing and validating edge resource management policies under realistic orbital and network dynamics.

6.1 Edge Server Activation

We implemented autonomous controllers that independently decide when to activate in-orbit edge servers using only locally available information. Four controller paradigms were evaluated: two novel contributions, namely Fuzzy Logic (FL) and Reinforcement Learning (RL), and two baselines, Access and Bernoulli. Both FL and RL were designed to minimize activation time while maintaining a low task failure rate, explicitly trading energy usage for MEC task reliability.

Results show that FL and RL consistently reduce server activation time and task failures relative to the baselines, yielding improvements of roughly 20–40% depending on

the load regime. FL achieves these gains with low computational complexity during both training and operation while offering interpretable rules that make it suitable for resource-constrained spacecraft. RL delivers comparable performance while being easier to adapt to new input features or mission objectives, thanks to its model-agnostic learning pipeline. Together, these controllers demonstrate that intelligent ESA can deliver significant efficiency gains without compromising quality of service (QoS), advancing the state of the art in distributed on-orbit edge resource management.

6.2 Edge Server Placement

This work demonstrates the feasibility and performance potential of deploying edge servers in LEO constellations and positions the ESP problem at the core of latency optimization under a fixed budget K. We proposed **Farthest Point Sampling (FPS)** as a contribution for topology-aware space placement and compared it against space baselines Along-Orbit (AO) and Cross-Plane (CP), as well as against a terrestrial cloud scenario and hybrid configurations.

Across emulated constellations, FPS reduced mean round-trip time (RTT) by 15–20% relative to AO and CP while also improving hop-count balance. At $K = 10$, space placement with FPS outperformed the terrestrial cloud baseline, confirming that well-informed space deployments can surpass gateway-centric designs even as network density grows. Hybrid deployments did not always produce the lowest average RTT, but they consistently reduced variance, providing more uniform service across geographically dispersed users.

6.3 Contributions and Findings

The results obtained through emulation substantiate three main contributions. **(1) Edge Server Activation (ESA).** FL and RL controllers, our proposed methods, outperform Access and Bernoulli baselines across varying task rates, cutting activation time and failure rates by about 20–40% while preserving high utilization. FL is lightweight and interpretable, reducing engineering effort on constrained platforms, whereas RL provides easier reconfiguration for new inputs and objectives through retraining. This highlights a practical trade-off between explainability and adaptability that can guide autonomous on-orbit control designs. **(2) Edge Server Placement (ESP).** FPS, our proposed placement

strategy, consistently improves mean RTT and balances network load compared with AO and CP, with 15–20% gains observed across budgets and scenarios, and with space FPS surpassing terrestrial clouds for $K = 10$. This confirms that topology-aware space placement is essential to unlock Space Cloud latency advantages under realistic dynamics. **(3) Real-Time Constellation Emulation (MeteorNet).** The MeteorNet platform integrates orbital propagation, SDN-based network emulation, and containerized execution into a single, reproducible environment, enabling rapid prototyping of ESA and ESP strategies with production-grade networking stacks. This open and modular integration allows researchers to evaluate edge resource orchestration policies end-to-end without proprietary simulators while providing the telemetry required for data-driven learning and validation.

Collectively, these contributions establish a foundation for autonomous, scalable, and energy-efficient space-edge computing systems, demonstrating that intelligent activation, strategic placement, and real-time emulation can jointly advance the design and operation of next-generation space networks.

6.4 Future Work

Future research can extend this work along several directions that deepen integration between networking, computing, and control in space. First, extending MeteorNet to support inter-constellation coordination and cross-layer optimization would enable the study of multi-orbit architectures and global task-routing strategies. Second, integrating energy-aware models and radiation-induced degradation effects would improve the realism of compute and communication behavior over mission lifetimes. Third, exploring collaborative learning among satellites—via federated or swarm reinforcement learning—could enable robust adaptation to traffic variability and partial visibility without central coordination. Fourth, enriching ESA and ESP objectives with multi-tenant QoS, fairness, and energy budgets would bring the optimization targets closer to operational reality. Finally, hardware-in-the-loop experiments that couple on-orbit hardware prototypes with MeteorNet would smooth integration with operational deployments and accelerate the maturation of self-learning computing systems in space.

Appendix

APPENDIX A

Fundamentals of Fuzzy Control

This summary is based in Chapter 8 (*An Introduction to Fuzzy Control*) of *Fuzzy Logic, An Introductory Course for Engineering Students* book [27].

A Overview of Fuzzy Control Systems

Fuzzy logic control provides a robust framework for decision-making in systems characterized by uncertainty, imprecision, or nonlinear behavior. Unlike classical control strategies that rely on precise mathematical models, fuzzy control uses linguistic rules and approximate reasoning to model expert knowledge and derive control actions. A fuzzy controller typically consists of three stages: *fuzzification*, *inference*, and *defuzzification*. This annex summarizes their theoretical foundations and mathematical formulations.

B Fuzzification

Let x be a crisp input variable, and X the universe of discourse. Fuzzification maps x to one or more fuzzy sets A_i characterized by a membership function $\mu_{A_i}(x)$:

$$\mu_{A_i} : X \to [0, 1]. \tag{A.1}$$

A membership function quantifies the degree to which x belongs to the fuzzy set A_i. Common shapes include the triangular, trapezoidal, and Gaussian functions:

Triangular:

$$\mu_A(x) = \begin{cases} 0, & x \le a, \\ \dfrac{x-a}{b-a}, & a < x \le b, \\ \dfrac{c-x}{c-b}, & b < x < c, \\ 0, & x \ge c, \end{cases} \tag{A.2}$$

Trapezoidal:

$$\mu_A(x) = \begin{cases} 0, & x \le a, \\ \dfrac{x-a}{b-a}, & a < x \le b, \\ 1, & b < x \le c, \\ \dfrac{d-x}{d-c}, & c < x < d, \\ 0, & x \ge d, \end{cases} \tag{A.3}$$

Gaussian:

$$\mu_A(x) = \exp\left[-\frac{(x-c)^2}{2\sigma^2}\right], \tag{A.4}$$

where a, b, c, d define shape parameters, and (c, σ) denote mean and standard deviation.

C Rule Base and Inference Mechanism

Fuzzy rules describe control knowledge in linguistic form. A typical rule has the structure:

$$R_k : \text{IF } x_1 \text{ is } A_1^k \text{ AND } x_2 \text{ is } A_2^k \text{ THEN } y \text{ is } B^k, \tag{A.5}$$

where A_i^k and B^k are fuzzy sets for inputs and outputs, respectively.

Given a set of M rules, the inference mechanism determines the firing strength ω_k of each rule:

$$\omega_k = \min\left[\mu_{A_1^k}(x_1), \mu_{A_2^k}(x_2), \ldots, \mu_{A_n^k}(x_n)\right]. \tag{A.6}$$

For a rule with multiple antecedents connected by "OR," the firing strength is given by the maximum operator:

$$\omega_k = \max\left[\mu_{A_1^k}(x_1), \mu_{A_2^k}(x_2), \ldots, \mu_{A_n^k}(x_n)\right]. \tag{A.7}$$

The implication step computes the fuzzy output set for each rule:

$$\mu'_{B^k}(y) = \omega_k \wedge \mu_{B^k}(y), \tag{A.8}$$

where $\wedge$ denotes the fuzzy intersection (usually implemented as min).

The aggregation of all rule outputs yields a combined output fuzzy set:

$$\mu_B(y) = \bigvee_{k=1}^{M} \mu'_{B^k}(y), \tag{A.9}$$

where $\bigvee$ represents the fuzzy union (commonly max).

D Defuzzification

Defuzzification converts the aggregated fuzzy output $\mu_B(y)$ into a crisp control signal y^*. The most widely used method is the *centroid of area (CoA)*:

$$y^* = \frac{\int_Y y \cdot \mu_B(y)\, dy}{\int_Y \mu_B(y)\, dy}. \tag{A.10}$$

E Summary

Fuzzy control systems provide a bridge between human reasoning and automatic control. By transforming qualitative knowledge into quantitative decision-making, they are well-suited for dynamic, uncertain, and nonlinear environments such as satellite networks. Their interpretability and low computational complexity make them particularly appealing for onboard resource orchestration where model-based controllers are infeasible.

APPENDIX B

Actor–Critic Algorithms

This summary on Actor-Critic algorithms is based in *Natural Actor-Critic Algorithms* [48].

A Introduction

Actor–critic reinforcement-learning algorithms provide an online approximation to policy iteration in Markov decision processes (MDPs). They maintain two coupled structures: (i) a *critic*, which estimates the state-value function through temporal-difference (TD) learning, and (ii) an *actor*, which updates the policy parameters in the direction of an estimated policy-gradient. This two-component architecture enables simultaneous policy evaluation and improvement under function approximation, making it effective for large or continuous state spaces. Unlike pure value-based methods, actor–critic algorithms guarantee stability when compatible features are used.

B The Policy-Gradient Framework

We consider a discrete-time MDP with finite state and action spaces S and A. At time t, the system is in state $s_t \in S$, selects $a_t \in A$ according to a stochastic policy π, receives reward r_{t+1}, and transitions to s_{t+1} with probability $P(s_t, a_t, s_{t+1})$. The long-run average reward under policy π is

$$J(\pi) = \sum_{s \in S} d_\pi(s) \sum_{a \in A} \pi(s,a) R(s,a),$$

where $d_\pi(s)$ is the stationary distribution of the induced Markov chain and $R(s,a) = \mathbb{E}[r_{t+1} | s_t = s, a_t = a]$ is the single-stage expected reward when action a is taken in state s.

For parameterized randomized policies $\pi_\theta(s,a)$, the gradient of the average reward is

$$\nabla J(\pi) = \sum_{s \in S} d_\pi(s) \sum_{a \in A} \nabla \pi(s,a) Q_\pi(s,a), \tag{B.1}$$

where $Q_\pi(s,a)$ is the expected differential reward associated with a state-action pair (s,a), defined by

$$Q_\pi(s,a) = \sum_{t=1}^{\infty} \mathbb{E}[r_{t+1} - J(\pi) | s_0 = s, a_0 = a, \pi]. \tag{B.2}$$

Likewise, the expected differential reward associated with a state s when actions are selected according to policy π, is defined by

$$V_\pi(s) = \sum_{a \in A} \pi(s,a) Q_\pi(s,a). \tag{B.3}$$

The Poisson equation under policy pi is given by

$$J(\pi) + V_\pi(s) = \sum_{a \in A} \pi(s,a) \Big[R(s,a) + \sum_{s' \in S} P(s,a,s') V_\pi(s') \Big]. \tag{B.4}$$

C Two-Timescale Stochastic Approximation Algorithms

Coupled stochastic recursions with step sizes of different magnitudes underpin actor–critic convergence. Let X_t, Y_t evolve as

$$X_{t+1} = X_t + \alpha_t (f(X_t, Y_t) + N_{t+1}^{(1)}), \tag{B.5}$$

$$Y_{t+1} = Y_t + \beta_t (g(X_t, Y_t) + N_{t+1}^{(2)}), \tag{B.6}$$

where $\{N_t^{(1)}\}, \{N_t^{(2)}\}$ are martingale-difference sequences, f, g are Lipschitz continuous functions, and the step-sizes satisfy

$$\sum_t \alpha_t = \sum_t \beta_t = \infty, \quad \sum_t (\alpha_t^2 + \beta_t^2) < \infty, \quad \beta_t = o(\alpha_t). \tag{B.7}$$

Hence X_t (critic) evolves on the *faster* timescale, and Y_t (actor) on the slower one. Under assumptions (B1)–(B3) the iterates satisfy

$$(X_t, Y_t) \to (\mu(Y^\star), Y^\star) \quad \text{a.s.}$$

where $\mu(\cdot)$ and $Y^\star$ are globally asymptotically stable equilibria of the associated ODEs.

D Policy Gradient with Function Approximation

We now extend the policy gradient framework to the case in which the action–value function Q^π for a fixed policy π is approximated by a learned function approximator. If the approximation is sufficiently accurate, it can replace Q^π in the policy gradient equations while still pointing approximately in the direction of the true gradient. Sutton et al. [48] showed that if the approximation $\hat{Q}^\pi_w$ with parameter $w \in \mathbb{R}^{d_1}$ is *compatible* and minimizes the mean–squared error, then such substitution is valid.

$$E^\pi(w) = \sum_{s\in S} d^\pi(s) \sum_{a\in A} \pi(s,a) \big[Q^\pi(s,a) - \hat{Q}^\pi_w(s,a)\big]^2, \tag{B.8}$$

where compatibility means

$$\nabla_w \hat{Q}^\pi_w(s,a) = \nabla \log \pi(s,a).$$

We employ a linear function approximation of the form

$$\hat{Q}^\pi_w(s,a) = w^\top \psi_{sa},$$

where the features ψ_{sa} are defined as the compatible features $\psi_{sa} = \nabla \log \pi(s,a)$.

The Fisher information matrix, previously defined in Equation (6), can then be expressed as

$$G(\theta) = \mathbb{E}_{s\sim d^\pi, a\sim\pi}[\psi_{sa}\psi_{sa}^\top] = \sum_{s\in S} d^\pi(s) \sum_{a\in A} \pi(s,a)\, \psi_{sa}\psi_{sa}^\top. \tag{B.9}$$

To further reduce estimator variance, we consider a baseline $b(s)$ and redefine the mean–squared error as

$$E^\pi(w) = \sum_{s\in S} d^\pi(s) \sum_{a\in A} \pi(s,a) \big[Q^\pi(s,a) - w^\top \psi_{sa} - b(s)\big]^2. \tag{B.10}$$

Let $w^\star = \arg\min_w E^\pi(w)$ denote the optimal parameter.

Lemma D.1. *The optimal weight parameter $w^\star$ for any given policy π satisfies*

$$w^\star = G(\theta)^{-1} \mathbb{E}_{s\sim d^\pi, a\sim\pi}[Q^\pi(s,a)\, \psi_{sa}]. \tag{B.11}$$

Proof provide by [48])

Lemma D.2. *For any given policy π, the minimum–variance baseline $b^\star(s)$ in the action–value estimator corresponds to the state–value function:*

$$b^\star(s) = V^\pi(s) = \sum_{a \in A} \pi(s,a) Q^\pi(s,a). \tag{B.12}$$

Proof provide by [48])

From these lemmas it follows that $w^{\star\top}\psi_{sa}$ is a least–squares optimal parametric representation of the advantage function $A^\pi(s,a) = Q^\pi(s,a) - V^\pi(s)$. Thus, the critic in the actor–critic structure can use $w^\star$ to approximate $A^\pi(s,a)$ efficiently and with minimum variance.

The temporal–difference (TD) error, used for online learning, is given by

$$\delta_t = r_{t+1} - \hat{J}_{t+1} + \hat{V}_{s_{t+1}} - \hat{V}_{s_t}, \tag{B.13}$$

where $\hat{V}_{s_t}$ and $\hat{J}_{t+1}$ are unbiased estimates of the differential value and the average reward, respectively:

$$\mathbb{E}[\hat{V}_{s_t} | s_t, \pi] = V^\pi(s_t), \quad \mathbb{E}[\hat{J}_{t+1} | s_t, \pi] = J(\pi).$$

The TD error δ_t is an unbiased estimate of the advantage function $A^\pi(s_t, a_t)$, and therefore it can be used in place of A^π in actor updates to estimate the policy gradient direction incrementally.

E Actor–Critic Algorithms

Algorithm 1—*Regular-Gradient Actor–Critic*—uses estimates of the ordinary gradient rather than the natural gradient. The algorithm maintains parameter vectors v_t (critic), θ_t (actor), and a scalar estimate $\bar{r}_t$ of the average reward. Let the temporal-difference error be

$$\delta_t = r_{t+1} - \bar{r}_t + v_t^\top f_{s_{t+1}} - v_t^\top f_{s_t}, \tag{B.14}$$

where f_s denotes the feature vector of state s. Then, with decreasing step-sizes $\{\alpha_t\}$ and $\{\beta_t\}$ satisfying the two-timescale conditions above, the updates are

$$\bar{r}_{t+1} = \bar{r}_t + \alpha_t(r_{t+1} - \bar{r}_t), \tag{B.15}$$

$$\delta_t = r_{t+1} - \bar{r}_t + v_t^\top (f_{s_{t+1}} - f_{s_t}), \tag{B.16}$$

$$v_{t+1} = v_t + \alpha_t \delta_t f_{s_t}, \tag{B.17}$$

$$\theta_{t+1} = \Gamma(\theta_t + \beta_t \, \delta_t \, \psi_{s_t a_t}), \tag{B.18}$$

where $\Gamma(\cdot)$ is a projection operator ensuring boundedness of θ_t.

The TD-error δ_t is an unbiased estimate of the advantage function $A_\pi(s_t, a_t) = Q_\pi(s_t, a_t) - V_\pi(s_t)$; thus, update (B.18) ascends the expected average-reward gradient.

Assumptions (A1)–(A3) guarantee irreducibility of the Markov chain, differentiability of $\pi_\theta(s, a)$, and linear independence of basis functions. Under these and standard stochastic-approximation conditions,

$$\theta_t \to \Theta^\star \subseteq \{\theta : \nabla J(\theta) = 0\} \quad \text{almost surely.}$$

All proofs and convergence arguments are given in [48].

Appendix C

The SGP4 Model

SGP4 summary and equations are recovered from *SPACETRACK REPORT NO. 3* [49]

A NORAD Two-Element Sets (TLEs) and SGP4

NORAD Two-Line Element (TLE) sets encode a satellite's mean Keplerian elements and related perturbation terms at a given epoch. Each TLE contains the mean motion, eccentricity, inclination, right ascension of the ascending node, argument of perigee, and mean anomaly, along with secular drag terms. The SGP4 model takes a TLE at epoch and propagates the satellite's state forward or backward in time, accounting for Earth's oblateness (J_2, J_3, J_4), atmospheric drag, and short/long-period gravitational effects to produce osculating position and velocity in an Earth-centered inertial frame. The workflow is: read TLE $\rightarrow$ recover corrected mean quantities $\rightarrow$ form constants $\rightarrow$ apply secular effects $\rightarrow$ add long/short-period terms $\rightarrow$ convert to position/velocity.

B The SGP4 Model

The following equations reproduce the SGP4 formulation. All constants and symbols not defined below are defined in Section C. The original mean motion n_o'' and semimajor axis a_o'' are first recovered from input elements and equations

$$
\begin{aligned}
a_1 &= \left(\frac{k_e}{n_o}\right)^{2/3}, \qquad \delta_1 = \frac{3}{2}\frac{k_2}{a_1^2}\frac{(3\cos^2 i_o - 1)}{(1-e_o^2)^{3/2}}, \\
a_o &= a_1\left(1 - \tfrac{1}{3}\delta_1 - \delta_1^2 - \tfrac{134}{81}\delta_1^3\right), \\
\delta_o &= \frac{3}{2}\frac{k_2}{a_o^2}\frac{(3\cos^2 i_o - 1)}{(1-e_o^2)^{3/2}}, \\
n_o'' &= \frac{n_o}{1+\delta_o}, \qquad a_o'' = \frac{a_o}{1-\delta_o}.
\end{aligned}
$$

If perigee < 156 km:

$$s^* = \begin{cases} a_o''(1-e_o) - s + a_E, & \text{if perigee } \in [98, 156) \text{ km}, \\ 20/XKMPER + a_E, & \text{if perigee } < 98 \text{ km}, \end{cases}$$

and replace $(q_o - s)^4$ by

$$(q_o - s^*)^4 = \left(\sqrt[4]{(q_o - s)^4} + s - s^*\right)^4$$

Define (with appropriate s and $(q_o - s)^4$):

$$\theta = \cos i_o, \quad \xi = \frac{1}{a_o'' - s}, \quad \beta_o = \sqrt{1 - e_o^2}, \quad \eta = a_o'' e_o \xi,$$

$$C_2 = (q_o - s)^4 \xi^4 n_o'' (1-\eta^2)^{-7/2} \left[a_o'' \left(1 + \tfrac{3}{2}\eta^2 + 4e_o\eta + e_o\eta^3\right) + \tfrac{3}{2}\frac{k_2\xi}{(1-\eta^2)}\left(-\tfrac{1}{2} + \tfrac{3}{2}\theta^2\right)(8 + 24\eta^2 + 3\eta^4) \right]$$

$$C_1 = B^* C_2$$

$$C_3 = \frac{(q_o - s)^4 \xi^5 A_{3,0} n_o'' a_E \sin i_o}{k_2 e_o}$$

$$C_4 = 2n_o''(q_o - s)^4 \xi^4 a_o'' \beta_o^2 (1-\eta^2)^{-7/2} \left(\left[2\eta(1 + e_o\eta) + \tfrac{1}{2}e_o + \tfrac{1}{2}\eta^3 \right] - 2\frac{k_2\xi}{a_o''(1-\eta^2)} \left[3(1-3\theta^2)\left(1 + \tfrac{3}{2}\eta^2 - 2e_o\eta - \tfrac{1}{2}e_o\eta^3\right) + \tfrac{3}{4}(1-\theta^2)(2\eta^2 - e_o\eta - e_o\eta^3)\cos 2\omega_o \right] \right)$$

$$C_5 = 2(q_o - s)^4 \xi^4 a_o'' \beta_o^2 (1-\eta^2)^{-7/2} \left(1 + \tfrac{11}{4}\eta(\eta + e_o) + e_o\eta^3\right)$$

$$D_2 = 4a_o''\xi C_1^2$$

$$D_3 = \frac{4}{3}a_o''\xi^2(17a_o'' + s)C_1^3$$

$$D_4 = \frac{2}{3}a_o''\xi^3(221a_o'' + 31s)C_1^4$$

The secular effects of atmosheric drag and gravitation are included through the equations

$$
\begin{aligned}
M_{DF} &= M_o + \left[1 + \frac{3k_2(-1+3\theta^2)}{2a_o''^2\beta_o^3} + \frac{3k_2(13-78\theta^2+137\theta^4)}{16a_o''^4\beta_o^7}\right] n_o''(t-t_o)\\
\omega_{DF} &= \omega_o + \left[\frac{-3k_2(1-5\theta^2)}{2a_o''^2\beta_o^4} + \frac{3k_2(7-114\theta^2+395\theta^4)}{16a_o''^4\beta_o^8} + \frac{5k_4(3-36\theta^2+49\theta^4)}{4a_o''^4\beta_o^8}\right] n_o''(t-t_o)\\
\Omega_{DF} &= \Omega_o + \left[\frac{-3k_2\theta}{a_o''^2\beta_o^4} + \frac{3k_2(4\theta-19\theta^3)}{2a_o''^4\beta_o^8} + \frac{5k_4\theta(3-7\theta^2)}{2a_o''^4\beta_o^8}\right] n_o''(t-t_o)\\
\delta\omega &= B^* C_3(\cos\omega_o)(t-t_o),\\
\delta M &= -\frac{2}{3}(q_o-s)^4 B^* \xi^4 \frac{a_E}{e_o\eta}\left[(1+\eta\cos M_{DF})^3 - (1+\eta\cos M_o)^3\right]\\
M_p &= M_{DF} + \delta\omega + \delta M,\\
\omega &= \omega_{DF} - \delta\omega - \delta M\\
\Omega &= \Omega_{DF} - \frac{21}{2}\frac{n_o''k_2\theta}{a_o''^2\beta_o^2}C_1(t-t_o)^2,\\
e &= e_o - B^*C_4(t-t_o) - B^*C_5(\sin M_p - \sin M_o)\\
a &= a_o''\left[1 - C_1(t-t_o) - D_2(t-t_o)^2 - D_3(t-t_o)^3 - D_4(t-t_o)^4\right]^2\\
I_L &= M_p + \omega + \Omega + n_o''\Big[\tfrac{3}{2}C_1(t-t_o)^2 + (D_2+2C_1^2)(t-t_o)^3\\
&\quad + \tfrac{1}{4}(3D_3+12C_1D_2+10C_1^3)(t-t_o)^4\\
&\quad + \tfrac{1}{5}(3D_4+12C_1D_3+6D_2^2+30C_1^2D_2+15C_1^4)(t-t_o)^5\Big]\\
\beta &= \sqrt{1-e^2}, \qquad n = \frac{k_e}{a^{3/2}}
\end{aligned}
$$

When epoch perigee height < 220 km, truncate a and I_L after the C_1 term and drop the terms involving $C_5,\ \delta\omega,\ \delta M$

Add the long-period periodic terms

$$
\begin{aligned}
a_{xN} &= e\cos\omega,\\
I_{LL} &= \frac{A_{3,0}\sin i_o}{8k_2a\beta^2}(e\cos\omega)\left(3+\frac{5\theta}{1+\theta}\right),\\
a_{yNL} &= \frac{A_{3,0}\sin i_o}{4k_2a\beta^2}, \qquad I_{LT} = I_L + I_{LL}, \qquad a_{yN} = e\sin\omega + a_{yNL}.
\end{aligned}
$$

Solve Kepler's equations (for $E + \omega$) by defining

$$U = I_{LT} - \Omega$$

and using the iteration equation

$$(E + \omega)_{i+1} = (E + \omega)_i + \Delta(E + \omega)_i$$

with

$$\Delta(E + \omega)_i = \frac{U - a_{yN}\cos(E + \omega)_i + a_{xN}\sin(E + \omega)_i - (E + \omega)_i}{-a_{yN}\sin(E + \omega)_i - a_{xN}\cos(E + \omega)_i + 1}.$$

and

$$(E + \omega)_1 = U$$

The following equations are used to calculate preliminary quantities needed for short-period periodics

$$\begin{aligned}
e\cos E &= a_{xN}\cos(E + \omega) + a_{yN}\sin(E + \omega),\\
e\sin E &= a_{xN}\sin(E + \omega) - a_{yN}\cos(E + \omega),\\
e_L &= \sqrt{a_{xN}^2 + a_{yN}^2}, \qquad p_L = a(1 - e_L^2),\\
r &= a(1 - e\cos E),\\
\dot{r} &= k_e\sqrt{\frac{a}{r}}\, e\sin E, \qquad r\dot{f} = k_e\sqrt{\frac{p_L}{r}},
\end{aligned}$$

and

$$\begin{aligned}
\cos u &= \frac{a}{r}\left[\cos(E + \omega) - a_{xN} + \frac{a_{yN}(e\sin E)}{1 + \sqrt{1 - e_L^2}}\right],\\
\sin u &= \frac{a}{r}\left[\sin(E + \omega) - a_{yN} - \frac{a_{xN}(e\sin E)}{1 + \sqrt{1 - e_L^2}}\right],\\
u &= \tan^{-1}\left(\frac{\sin u}{\cos u}\right).
\end{aligned}$$

$$\Delta r = \frac{k_2}{2p_L}(1-\theta^2)\cos 2u, \qquad \Delta u = -\frac{k_2}{4p_L^2}(7\theta^2-1)\sin 2u,$$
$$\Delta\Omega = \frac{3k_2\theta}{2p_L^2}\sin 2u, \qquad \Delta i = \frac{3k_2\theta}{2p_L^2}\sin i_o \cos 2u,$$
$$\Delta\dot{r} = -k_2 n\frac{1}{p_L}(1-\theta^2)\sin 2u,$$
$$\Delta(r\dot{f}) = k_2 n\frac{1}{p_L}\left[(1-\theta^2)\cos 2u - \tfrac{3}{2}(1-3\theta^2)\right].$$

The short -period periodics are added to give the osculating quantities

$$r_k = r\left[1 - \frac{3}{2}\frac{k_2}{p_L^2}\sqrt{1-e_L^2}\,(3\theta^2-1)\right] + \Delta r, \qquad u_k = u + \Delta u,$$
$$\Omega_k = \Omega + \Delta\Omega, \qquad i_k = i_o + \Delta i, \qquad \dot{r}_k = \dot{r} + \Delta\dot{r}, \qquad (r\dot{f})_k = r\dot{f} + \Delta(r\dot{f}).$$

Then unit orientation vectors are calculated by

$$\mathbf{U} = \mathbf{M}\sin u_k + \mathbf{N}\cos u_k \tag{C.1}$$
$$\mathbf{V} = \mathbf{M}\cos u_k - \mathbf{N}\sin u_k \tag{C.2}$$

where

$$\mathbf{M} = \begin{bmatrix} -\sin\Omega_k \cos i_k \\ \cos\Omega_k \cos i_k \\ \sin i_k \end{bmatrix} \tag{C.3}$$
$$\mathbf{N} = \begin{bmatrix} \cos\Omega_k \\ \sin\Omega_k \\ 0 \end{bmatrix} \tag{C.4}$$

Then position and velocity are given by

$$\mathbf{r} = r_k\,\mathbf{U} \tag{C.5}$$
$$\dot{\mathbf{r}} = \dot{r}_k\,\mathbf{U} + (r\dot{f})_k\,\mathbf{V} \tag{C.6}$$

C Physical-Mathematical Constants and Symbols

Table C.1: Constants used by SGP/SGP4.

Definition	**Value**
$\frac{1}{2}J_2 a_E^2$	5.413080×10^{-4}
$-\frac{3}{8}J_4 a_E^4$	$0.62098875 \times 10^{-6}$
10^{-6}	1.0×10^{-6}
$(q_o - s)^4$ (er)^4	$1.88027916 \times 10^{-9}$
s (er)	1.01222928
$\frac{2}{3}$	0.66666667
J_3	-0.253881×10^{-5}
$k_e \left(\frac{\text{er}}{\text{min}}\right)^{3/2}$	$0.743669161 \times 10^{-1}$
XKMPER kilometers per Earth radii	6378.135
XMNPDA time units per day	1440.0
AE distance units per Earth radii	1.0
DE2RA radians per degree	$0.174532925 \times 10^{-1}$
π	3.14159265
$\frac{\pi}{2}$	1.57079633
2π	6.2831853
$\frac{3\pi}{2}$	4.71238898

Table C.2: SGP4 Symbols used.

Symbol	Description
n_o	SGP-type "mean" mean motion at epoch
e_o	"Mean" eccentricity at epoch
i_o	"Mean" inclination at epoch
M_o	"Mean" mean anomaly at epoch
ω_o	"Mean" argument of perigee at epoch
Ω_o	"Mean" longitude of ascending node at epoch
$\dot{n}_o$	Time rate of change of "mean" mean motion at epoch
$\ddot{n}_o$	Second time rate of change of "mean" mean motion at epoch
B^*	SGP4-type drag coefficient
$k_e = \sqrt{GM}$	(G: Newton's constant, M: Earth mass)
a_E	Earth equatorial radius
J_2, J_3, J_4	Earth's second/third/fourth zonal harmonics
$(t - t_o)$	Time since epoch
$k_2 = \frac{1}{2}J_2 a_E^2, \quad k_4 = -\frac{3}{8}J_4 a_E^4$	Auxiliary constants
$A_{3,0} = -J_3 a_E^3$	Auxiliary constant
q_o, s	SGP4/SGP8 density function parameters
$B = \frac{1}{2}C_D \frac{A}{m}$	Ballistic coefficient (SGP8)

Publication record

1. International Journal Papers

[1] **Rojas Milla, C.**, Patrone, F., Fraire, J. A., & Marchese, M. (2025). *From Emerging LEO Satellite Constellations to the Space Cloud: Emulation Platforms and Orchestration Methods*. Computer Networks (Elsevier).
DOI: 10.1016/j.comnet.2025.111970

[2] **Rojas, C.**, Badini, N., Patrone, F., Fraire, J. A., & Marchese, M. (2025). *Quantifying Edge Server Placement in LEO Constellations: A Comparative Emulation of Space and Terrestrial Cloud*. International Journal of Satellite Communications and Networking (Wiley).
DOI: 10.22541/au.175672474.48833925/v1

2. International Conference Papers

[3] **Rojas, C.**, Fraire, J. A., Patrone, F., & Marchese, M. (2025). *On the Latency Trade-off Between Space and Terrestrial Clouds in Non-Terrestrial Networks*. ASMS/SPSC 2025, Sitges, Spain.
DOI: 10.1109/ASMS/SPSC64465.2025.10946050

[4] **Rojas, C.**, Fraire, J. A., Patrone, F., & Marchese, M. (2024). *Fuzzy Logic-Based Orchestration of Multi-Access Edge Computing in LEO Satellite Constellations*. IEEE ICC Workshops 2024, Denver, USA.
DOI: 10.1109/ICCWorkshops59551.2024.10615305

[5] **Rojas, C.**, Fraire, J. A., Patrone, F., & Marchese, M. (2024). *Advanced Constellation Emulation and Synthetic Datasets Generation for Non-Terrestrial Networks*. IEEE MeditCom 2024, Madrid, Spain.
DOI: 10.1109/MeditCom61057.2024.10621248

[6] **Rojas, C.**, Fraire, J. A., Patrone, F., Gotta, A., & Marchese, M. (2023). *Continuous Time Emulation for Software-Defined Non-Terrestrial Edge Computing Networks*. European Wireless 2023, Rome, Italy.
https://ieeexplore.ieee.org/document/10461445

Participation in Research Projects

MISSION: Machine Learning and Innovative Satellite System Orchestration, Marie Skłodowska-Curie, Horizon 2020 (Grant No. 101008233).

The project focuses on advancing Non-Terrestrial Networks (NTN) through novel approaches in edge computing, satellite communications, and real-time emulation frameworks.

Activity: Contributed to the design, implementation, and evaluation of emulation platforms for LEO satellite constellations. Participated in the development of latency trade-off analyses between space and terrestrial clouds and orchestration strategies for multi-access edge computing.

RESTART: Telecommunications of the Future, Italian National Recovery and Resilience Plan (NRRP), NextGenerationEU, Partnership PE00000001.

National partnership aimed at strengthening Italy's role in future telecommunications, focusing on cloud-native, satellite-enabled, and AI-driven network infrastructures.

Activity: Contributed to the work package on Space Cloud research within RESTART. Designed and implemented experiments on hybrid cloud architectures and server placement strategies. Actively involved in dissemination of research results through international conferences and journals.

Space Edge Computing for 6G NTN, French National Research Agency (ANR-22-CE25-0014-01).

Research on integrating edge computing and machine learning for next-generation satellite networks, with a focus on reinforcement learning and fuzzy-logic orchestration methods.

Activity: Co-developed models for server activation policies in LEO constellations. Contributed to the evaluation of KPIs, reinforcement learning models, and fuzzy logic controllers for edge orchestration in NTN.

Attendance to conferences and workshops

Attended Conference in Sitges, Spain, 15–18 Sept. 2025, presenting: C. Rojas, J. A. Fraire, F. Patrone, M. Marchese, *On the Latency Trade-off Between Space and Terrestrial Clouds in Non-Terrestrial Networks*, 12th Advanced Satellite Multimedia Systems Conference and 18th Signal Processing for Space Communications Workshop (ASMS/SPSC).

Attended Conference in Denver, CO, USA, 9–13 June 2024, presenting: C. Rojas, J. A. Fraire, F. Patrone, M. Marchese, *Fuzzy Logic-Based Orchestration of Multi-Access Edge Computing in LEO Satellite Constellations*, IEEE International Conference on Communications Workshops (ICC Workshops).

Attended Conference in Madrid, Spain, 9–12 Sept. 2024, presenting: C. Rojas, J. A. Fraire, F. Patrone, M. Marchese, *Advanced Constellation Emulation and Synthetic Datasets Generation for Non-Terrestrial Networks*, IEEE International Mediterranean Conference on Communications and Networking (MeditCom).

Attended Workshop in Rome, Italy, 3 Oct. 2023, presenting: F. Patrone, C. Rojas, *Continuous Time Emulation for Software-Defined Non-Terrestrial Edge Computing Networks*, European Wireless 2023; 28th European Wireless Conference.

Awards and scientific recognitions

Best Paper Recognition – 12th Advanced Satellite Multimedia Systems Conference and 18th Signal Processing for Space Communications Workshop (ASMS/SPSC 2025), Sitges, Spain. The paper *On the Latency Trade-off Between Space and Terrestrial Clouds in Non-Terrestrial Networks* was selected as one of the top four best papers of the conference and invited for extension in a special issue of the International Journal of Satellite Communications and Networking (Wiley).

Bibliography

[1] R. Radhakrishnan, W. W. Edmonson, F. Afghah, R. M. Rodriguez-Osorio, F. Pinto, and S. C. Burleigh, "Survey of Inter-Satellite Communication for Small Satellite Systems: Physical Layer to Network Layer View," *IEEE Communications Surveys & Tutorials*, vol. 18, no. 4, pp. 2442–2473, 2016. [Online]. Available: http://ieeexplore.ieee.org/document/7466793/

[2] S. Lee, A. Hutputanasin, A. Toorian, W. Lan, R. Munakata, J. Carnahan, D. Pignatelli, and A. Mehrparvar, "Cubesat design specification rev. 13," 2014.

[3] I. d. Portillo, S. Eiskowitz, E. F. Crawley, and B. G. Cameron, "Connecting the other half: Exploring options for the 50% of the population unconnected to the internet," *Telecommunications Policy*, vol. 45, no. 3, p. 102092, 2021. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0308596120301828

[4] P. Cassara, A. Gotta, M. Marchese, and F. Patrone, "Orbital Edge Offloading on Mega-LEO Satellite Constellations for Equal Access to Computing," *IEEE Communications Magazine*, vol. 60, no. 4, pp. 32–36, Apr. 2022. [Online]. Available: https://ieeexplore.ieee.org/document/9755271/

[5] K. Liolis, A. Geurtz, R. Sperber, D. Schulz, S. Watts, G. Poziopoulou, B. Evans, N. Wang, O. Vidal, B. Tiomela Jou, M. Fitch, S. Diaz Sendra, P. Sayyad Khodashenas, and N. Chuberre, "Use cases and scenarios of 5G integrated satellite-terrestrial networks for enhanced mobile broadband: The SaT5G approach," *International Journal of Satellite Communications and Networking*, vol. 37, no. 2, pp. 91–112, 2019. [Online]. Available: https://onlinelibrary.wiley.com/doi/abs/10.1002/sat.1245

[6] 3GPP, "TR 38.811: Study on new radio (nr) to support non-terrestrial networks," 3GPP Technical Report, 2020. [Online]. Available: https://www.3gpp.org/dynareport/38811.htm

[7] European Union, "Regulation (eu) 2023/588 establishing the union secure connectivity programme for 2023–2027," Official Journal of the European Union, 2023, secure Connectivity Programme (IRIS^2).

[8] European Commission, "Eu awards iris^2 concession contract for secure connectivity," European Commission press release, 2024.

[9] European Space Agency, "Eu secure connectivity programme (iris^2) overview," ESA programme page, 2023.

[10] ITU-R, "Recommendation itu-r m.2160: Framework and overall objectives of the future development of imt for 2030 and beyond," International Telecommunication Union Radiocommunication Sector, 2023.

[11] ——, "Report itu-r m.2516: Future technology trends of terrestrial imt systems towards 2030 and beyond," International Telecommunication Union Radiocommunication Sector, 2022.

[12] C. Lin and et al., "Integration of satellite communication and mec for 6g iot applications," *IEEE Wireless Communications*, vol. 29, no. 6, pp. 12–18, 2022.

[13] 6G-IA, "Research priorities on non-terrestrial networks (ntn)," 6G Smart Networks and Services Industry Association report, 2024.

[14] EAGER Consortium, "Architectures, services, and technologies towards 6g non-terrestrial networks," Project white paper, 2023.

[15] Hexa-X Consortium, "Hexa-x deliverable d1.2: Expanded 6g vision, use cases and societal values," Project deliverable, 2021.

[16] F. Spinelli and V. Mancuso, "Toward enabled industrial verticals in 5g: A survey on mec-based approaches to provisioning and flexibility," *IEEE Communications Surveys & Tutorials*, vol. 23, no. 1, pp. 596–630, 2021.

[17] J. Pfandzelter and D. Bermbach, "Qos-aware resource placement strategies in leo satellite constellations," *IEEE Transactions on Network and Service Management*, vol. 19, no. 1, pp. 25–38, 2022.

[18] S. Pizzi, A. Tropeano, G. Araniti, and A. Molinaro, "Enhancing orbital edge computing through isl-aided federation of satellite constellations," *IEEE Open Journal of the Communications Society*, pp. 1–1, 2025.

[19] M. Giordani, M. Polese, and M. Zorzi, "Toward 6g networks: Use cases and technologies," *IEEE Communications Magazine*, vol. 61, no. 3, pp. 55–61, 2023.

[20] W. Saad, M. Bennis, and M. Chen, "A vision of 6g wireless systems: Applications, trends, technologies, and open research problems," *IEEE Network*, vol. 38, no. 1, pp. 134–142, 2024.

[21] O-RAN Alliance, "O-ran: Towards an open and smart ran," White Paper, 2023.

[22] H. Yang and X. Li, "Ai-driven resource management for non-terrestrial networks," *IEEE Communications Surveys & Tutorials*, 2024.

[23] R. Zhang and Y. Liu, "Ai-native mec architectures for 6g non-terrestrial networks," *IEEE Transactions on Network and Service Management*, 2025.

[24] C. Sonmez, A. Ozgovde, and C. Ersoy, "Fuzzy Workload Orchestration for Edge Computing," *IEEE Transactions on Network and Service Management*, vol. 16, no. 2, pp. 769–782, Jun. 2019. [Online]. Available: https://ieeexplore.ieee.org/document/8651335/

[25] Y. Lyu, Z. Liu, R. Fan, C. Zhan, H. Hu, and J. An, "Optimal Computation Offloading in Collaborative LEO-IoT Enabled MEC: A Multiagent Deep Reinforcement Learning Approach," *IEEE Transactions on Green Communications and Networking*, vol. 7, no. 2, pp. 996–1011, Jun. 2023. [Online]. Available: https://ieeexplore.ieee.org/document/9809971/

[26] W. Jiang, Y. Zhan, X. Xiao, and G. Sha, "Network simulators for satellite-terrestrial integrated networks: A survey," *IEEE Access*, vol. 11, pp. 98 269–98 292, 2023.

[27] E. Trillas and L. Eciolaza, *Fuzzy Logic: An Introductory Course for Engineering Students*. Springer Publishing Company, 2015.

[28] S. Jenei, "Continuity in Zadeh's compositional rule of inference," *Fuzzy Sets and Systems*, vol. 104, no. 2, pp. 333–339, 1999. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S016501149700198X

[29] J. Mendel, "Fuzzy logic systems for engineering: a tutorial," *Proceedings of the IEEE*, vol. 83, no. 3, pp. 345–377, 1995.

[30] C. Rojas, J. A. Fraire, F. Patrone, and M. Marchese, "Advanced constellation emulation and synthetic datasets generation for non-terrestrial networks." Madrid, Spain: IEEE International Mediterranean Conference on Communications and Networking (MeditCom), 2024.

[31] B. Chun, D. Culler, T. Roscoe, A. Bavier, L. Peterson, M. Wawrzoniak, and M. Bowman, "Planetlab: an overlay testbed for broad-coverage services," *SIGCOMM Comput. Commun. Rev.*, vol. 33, no. 3, p. 3–12, Jul. 2003. [Online]. Available: https://doi.org/10.1145/956993.956995

[32] B. White, J. Lepreau, L. Stoller, R. Ricci, S. Guruprasad, M. Newbold, M. Hibler, C. Barb, and A. Joglekar, "An integrated experimental environment for distributed systems and networks," *SIGOPS Oper. Syst. Rev.*, vol. 36, no. SI, p. 255–270, Dec. 2003. [Online]. Available: https://doi.org/10.1145/844128.844152

[33] Ansys, "Systems tool kit," Html, 2025. [Online]. Available: https://www.ansys.com/products/missions/ansys-stk

[34] S. P. Hughes, "General mission analysis tool (gmat)," NASA, Tech. Rep., 2016.

[35] S. Kassing, D. Bhattacherjee, A. B. Águas, J. E. Saethre, and A. Singla, "Exploring the "internet from space" with hypatia," ser. IMC '20, Proceedings of the ACM Internet Measurement Conference. New York, NY, USA: Association for Computing Machinery, 2020, p. 214–229. [Online]. Available: https://doi.org/10.1145/3419394.3423635

[36] Z. Lai, H. Li, and J. Li, "Starperf: Characterizing network performance for emerging mega-constellations." 2020 IEEE 28th International Conference on Network Protocols (ICNP), 2020, pp. 1–11.

[37] M. Team, "Mininet: An instant virtual network on your laptop," Git, 2021. [Online]. Available: https://github.com/mininet/mininet

[38] Satellite Communication and heterogeneous Networking Laboratory (SCNL), "Meteornet," Git, University of Genoa, Italy, 2024. [Online]. Available: https://gitlab.com/camilo.rojas/meteornet

[39] D. Vallado, P. Crawford, R. Hujsak, and T. Kelso, "Revisiting spacetrack report #3," vol. 3. Collection of Technical Papers - AIAA/AAS Astrodynamics Specialist Conference, 08 2006.

[40] K. Benzekki, A. El Fergougui, and A. Elbelrhiti Elalaoui, "Software-defined networking (SDN): a survey," *Security and Communication Networks*,

vol. 9, no. 18, pp. 5803–5833, 2016. [Online]. Available: https://onlinelibrary.wiley.com/doi/abs/10.1002/sec.1737

[41] P. Berde, M. Gerola, J. Hart, Y. Higuchi, M. Kobayashi, T. Koide, B. Lantz, B. O'Connor, P. Radoslavov, W. Snow *et al.*, "Onos: towards an open, distributed sdn os." Proceedings of the third workshop on Hot topics in software defined networking, 2014, pp. 1–6.

[42] S. Singh and N. Singh, "Containers & Docker: Emerging roles & future of Cloud technology." 2016 2nd International Conference on Applied and Theoretical Computing and Communication Technology (iCATccT), 2016, pp. 804–807.

[43] MongoDB, "Mongodb," Html, 2025. [Online]. Available: https://www.mongodb.com

[44] L. Yan and N. McKeown, "Learning Networking by Reproducing Research Results," *ACM SIGCOMM Computer Communication Review*, vol. 47, no. 2, 2017.

[45] C. Rojas, J. A. Fraire, F. Patrone, and M. Marchese, "Fuzzy logic-based orchestration of multi-access edge computing in leo satellite constellations." Denver, CO, USA: IEEE International Conference on Communications (ICC), 2024.

[46] ——, "On the latency trade-off between space and terrestrial clouds in non-terrestrial networks." Barcelona, Spain: 12th Advanced Satellite Multimedia Systems Conference and 18th Signal Processing for Space Communications Workshop (ASMS/SPSC), 2025.

[47] D. Eddy, M. Ho, and M. J. Kochenderfer, "Optimal ground station selection for low-earth orbiting satellites," *arXiv preprint arXiv:2410.16282*, 2024.

[48] S. Bhatnagar, R. S. Sutton, M. Ghavamzadeh, and M. Lee, "Natural actor–critic algorithms," *Automatica*, vol. 45, no. 11, pp. 2471–2482, 2009. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0005109809003549

[49] F. R. Hoots, "Spacetrack report no. 3, models for propagation of norad element sets," *http://www. itc. nl/-bakker/orbit. html*, 1980.

Design, Emulation, and Control of Edge Computing Systems for the Space Cloud in LEO Satellite Networks

Camilo José Rojas Milla