UNIVERSIDADE D
COIMBRA

Luís Filipe Dias Neto

Development of Mobility and Satellite Communication Modules in the YAFS Simulator

Dissertation in the context of the Master in Informatics Engineering, specialization in Software Engineering, advised by Professor Dr. David Abreu and Professor Dr. Noé Godinho and presented to the Department of Informatics Engineering of the Faculty of Sciences and Technology of the University of Coimbra.

September 2025

DEPARTAMENTO DE
ENGENHARIA INFORMÁTICA
FACULDADE DE
CIÊNCIAS E TECNOLOGIA
UNIVERSIDADE D
COIMBRA

Luís Filipe Dias Neto

Development of Mobility and Satellite Communication Modules in the YAFS Simulator

Dissertation in the context of the Master in Informatics Engineering, specialization in Software Engineering, advised by Professor Dr. David Abreu and Professor Dr. Noé Godinho and presented to the Department of Informatics Engineering of the Faculty of Sciences and Technology of the University of Coimbra.

September 2025

1 2  9 0

DEPARTAMENTO DE
ENGENHARIA INFORMÁTICA
FACULDADE DE
CIÊNCIAS E TECNOLOGIA
UNIVERSIDADE D
COIMBRA

Luís Filipe Dias Neto

Desenvolvimento de Módulos de Mobilidade e Comunicação por Satélite no Simulador YAFS

Dissertação no âmbito do Mestrado em Engenharia Informática, especialização em Engenharia de Software, orientada pelo Professor Doutor David Abreu e o Professor Doutor Noé Godinho e apresentada ao Departamento de Engenharia Informática da Faculdade de Ciências e Tecnologia da Universidade de Coimbra.

Setembro 2025

Abstract

The rapid proliferation of Internet of Things devices has led to the emergence of the Cloud-to-Thing Continuum, a computing paradigm that integrates cloud, fog, and edge infrastructures. However, ensuring ubiquitous connectivity with the challenges posed by high mobility and limited terrestrial coverage remains critical issues. Satellites, particularly in Low Earth Orbit, offer promising solutions, extending connectivity and improving computing capabilities.

In this context, robust and scalable simulation tools are essential. Network simulation makes it possible to explore and test complex hybrid networks between satellites and terrestrial infrastructures in controlled environments, avoiding the prohibitive costs and logistical difficulties of real-world experimentation. By accurately reproducing the behaviour of such networks, simulations provide a foundation for evaluating performance, optimising designs, and validating new architectures.

To address this need, this dissertation enhances the YAFS simulator by incorporating two new modules: a user mobility module and a satellite communication module. These extensions enable the simulation of dynamic hybrid environments with mobile users and satellite-based connectivity. The main contributions include a comprehensive review of existing simulators, the design and implementation of realistic simulation scenarios, and the validation of the proposed modules.

The developed modules were validated through comparative experiments against iFogSim2 and SatEdgeSim, demonstrating consistency in performance metrics and confirming the accuracy of the proposed models. The enhanced YAFS simulator also successfully reproduced realistic hybrid satellite-terrestrial scenarios. These results confirm the suitability of the new modules for evaluating ubiquitous connectivity solutions and establish YAFS as a robust platform to support future research and the design of next-generation hybrid network architectures.

Keywords

Cloud-to-Thing Continuum, Non-Terrestrial Networks, Internet of Things, Satellites, Network Simulation, Ubiquitous Connectivity.

Resumo

A rápida proliferação de dispositivos da Internet das Coisas levou ao aparecimento do *Cloud-to-Thing Continuum*, um paradigma de computação que integra infraestruturas de nuvem, nevoeiro e periferia. No entanto, garantir a conectividade ubíqua com os desafios impostos pela alta mobilidade e cobertura terrestre limitada continua a ser uma questão crítica. Os satélites, nomeadamente em órbita terrestre baixa, oferecem soluções promissoras, alargando a conectividade e melhorando as capacidades computacionais.

Neste contexto, tornam-se essenciais ferramentas de simulação robustas e escaláveis. A simulação de redes permite explorar e testar redes híbridas complexas entre satélites e infraestruturas terrestres em ambientes controlados, evitando os custos proibitivos e as dificuldades logísticas da experimentação no mundo real. Ao reproduzir com precisão o comportamento destas redes, as simulações fornecem uma base sólida para avaliar o desempenho, otimizar arquiteturas e validar novas soluções.

Para responder a esta necessidade, esta dissertação melhora o simulador YAFS através da incorporação de dois novos módulos: um módulo de mobilidade de utilizadores e um módulo de comunicação por satélite. Estas extensões permitem a simulação de ambientes híbridos dinâmicos com utilizadores móveis e conectividade via satélite. As principais contribuições incluem uma revisão abrangente dos simuladores existentes, a conceção e implementação de cenários de simulação realistas e a validação dos módulos propostos.

Os módulos desenvolvidos foram validados através de experiências comparativas com o iFogSim2 e o SatEdgeSim, demonstrando consistência nas métricas de desempenho e confirmando a precisão dos modelos propostos. O simulador YAFS aprimorado também conseguiu reproduzir cenários híbridos realistas satélite-terrestres. Estes resultados confirmam a adequação dos novos módulos para a avaliação de soluções de conectividade ubíqua e estabelecem o YAFS como uma plataforma robusta para apoiar futuras investigações e o desenho de arquiteturas de rede híbridas de próxima geração.

Palavras-Chave

Cloud-to-Thing Continuum, Redes não terrestres, *Internet of Things*, Satélites, simulação de redes, YAFS, Conectividade ubíqua.

Contents

Acronyms

3D Three-Dimensiona.

5G Fifth Generation.

6G Sixth Generation.

API Application Programming Interface.

C2T Cloud-to-Thing Continuum.

CSMA Carrier Sense Multiple Access.

CSV Comma-Separated Values.

DDF Distributed Data Flow.

DDPG Deep Deterministic Policy Gradient.

DES Discrete Event Simulator.

DST-IoT Direct-to-Satellite IoT.

DVB Digital Video Broadcasting.

FCD (Floating Car Data.

GEO Geostationary Earth Orbit.

GPX GPS Exchange format.

GUI Graphical User Interface.

HSTN Hybrid Satellite-Terrestrial Network.

IoT Internet of Things.

ISL Inter-Satellite Links.

JSON JavaScript Object Notation.

LAN Local Area Network.

LEO Low Earth Orbit.

LoRaWAN Long Range Wide Area Network.

LOS Line of Sight.

LSTM Long Short-Term Memory.

LTE Long Term Evolution.

MEC Mobile Edge Computing.

MEO Medium Earth Orbit.

NCC Network Control Centre.

NEOs Near-Earth objects.

NFV Network Functions Virtualization.

NS-3 Network Simulator 3.

NTN Non-Terrestrial Networks.

OS³ Open Source Satellite Simulator.

QoS Quality of Service.

SatEC Satellite Edge Computing.

SDN Software Defined Networking.

SNS3 Satellite Network Simulator 3.

STK Satellite Tool Kit.

SUMO Simulation of Urban MObility.

TLE Two-Line Element.

UAV Unmanned Aerial Vehicle.

VLEO Very Low Earth Orbit.

VM Virtual Machine.

WAN Wide Area Network.

WLAN Wireless Local Area Network.

YAFS Yet Another Fog Simulator.

List of Figures

List of Tables

Chapter 1

Introduction

STATEMENT ON THE USE OF ARTIFICIAL INTELLIGENCE TOOLS

This document was reviewed and refined with the assistance of Large Language Models, such as ChatGPT, or similar tools, which helped check grammar, correct typos, and enhance clarity. Any Artificial Intelligence-generated text or content is clearly indicated within the document and is used only to a limited extent. The overall content and ideas remain solely the responsibility of the author.

This chapter is structured as follows: In Section 1.1, the motivation behind the development of this work is presented; in Section 1.2, the objectives are defined and the main contributions are highlighted; and finally, in Section 1.3, the structure of the document is described to guide the reader.

1.1 Motivation

Nowadays, the proliferation of Internet of Things (IoT) devices has contributed significantly to an increase in the number of connected devices, characterised by increasing heterogeneity. This scenario introduces new challenges for cloud computing infrastructures, driving the emergence of a new paradigm called Cloud-to-Thing Continuum (C2T) [Bangui et al., 2018]. C2T seeks to integrate distributed computing infrastructures, encompassing cloud, fog and edge computing, with the aim of satisfying the specific needs of various applications. These applications, often characterised by varying sensitivity to latency and high mobility - as in the case of autonomous vehicles - require dynamic adaptation of resources and optimisation in the allocation of workloads [Bendechache et al., 2020].

Despite advances, C2T still faces significant challenges, such as the continuity of connectivity. The high mobility of many devices results in scenarios where terrestrial infrastructure is non-existent, weakened due to disasters, or insufficient, jeopardising connectivity. This failure can have a critical impact on the performance of applications, especially in IoT environments. In these situations, interruption of communication between network nodes can lead to serious operational failures, especially in applications that depend on real-time coordination

[Laroui et al., 2021].

A promising solution to mitigate this problem lies in the use of satellites, which offer global coverage and connectivity even in regions lacking terrestrial infrastructure. This approach is gaining traction, particularly due to the growing focus on satellites in Low Earth Orbit (LEO) - satellites that orbit the Earth at altitudes between roughly 500 and 2,000 km. Compared to satellites in geostationary orbits - which remain fixed relative to a point on Earth at about 36,000 km altitude, satellites in low orbits have lower placement costs and reduced latencies, factors that have encouraged studies to integrate satellite Internet with terrestrial infrastructure [Al-Dulaimy et al., 2024].

For this integration to materialise, it is essential to investigate new network topologies, advanced algorithms for allocating resources, and other fundamental characteristics. However, carrying out tests in real environments is costly and is often hampered by logistical barriers. In this context, simulation plays a crucial role, allowing the replication of real scenarios at a significantly lower cost. This method reduces risks and facilitates the validation of new solutions, reinforcing the importance of simulation tools in this emerging area.

Although there is a wide range of network simulators with different purposes — from analysing lower protocol layers to modelling complex systems — most are unable to simultaneously simulate terrestrial infrastructures and satellite networks in a flexible and integrated manner. Many focus exclusively on terrestrial or satellite environments, and even those that support both are often tailored to particular scenarios, lacking the ability to capture the complexity of C2T systems.

Rather than creating a completely new simulator, a more efficient approach is to adapt an existing tool that already aligns closely with our objectives. In this regard, Yet Another Fog Simulator (YAFS) stands out as a simulator specifically designed for C2T environments, offering a modular and extensible architecture. Its current limitation lies in the absence of native support for satellite simulation and user mobility. Addressing these gaps will enable YAFS to model integrated scenarios involving terrestrial and satellite networks in a comprehensive and effective way.

Based on the discussion presented, the next section will detail the objectives and main contributions of this work.

1.2 Objectives and contributions

The main goal of this dissertation is to help improve the simulator YAFS by implementing two new modules: a mobility module and a satellite module. To achieve this, the following objectives have been defined:

- Identify relevant simulators in the context of C2T and satellite simulation.
- Analyse the limitations and gaps present in existing simulators.

- Determine the requirements and metrics needed for a C2T simulator with support for mobility and satellite integration.
- Define an architecture that makes it possible to simulate C2T using satellites and mobile users.
- Validate the implementation of the proposed modules by comparing results with those obtained from other simulators and from scenarios with predictable outcomes.
- Develop a use case that demonstrates the capabilities of the updated YAFS.

Bearing in mind the objectives described above, this document has produced the following contributions:

- A state-of-the-art survey of C2T simulators and satellite simulators, identifying their capabilities and limitations.
- A couple of modules compatible with the YAFS simulator to be able to simulate satellite communications and user mobility.
- Simulation scenarios that assess connectivity in C2T and explore the use of satellites as potential computing nodes.

In addition, it is expected that the results obtained during this dissertation may lead to the publication of a scientific article contributing to the discussion on ubiquitous connectivity in C2T.

1.3 Dissertation structure

This document is organised into six chapters, as described below:

- Chapter 2 – Presents the key concepts that underpin this work. It begins with a description of C2T, followed by a discussion on the importance of continuous connectivity for the efficient operation of C2T systems and the role of satellites in this context. Subsequently, it provides a brief explanation of how generic network simulators operate. After this, several simulators are compared with our proposed use of YAFS, along with an analysis of libraries that can be employed to simulate satellites. The review includes a critical assessment of existing tools and approaches.
- Chapter 3 - Presents a detailed description of the identified problem, the requirements needed to solve it and the risks associated with implementing the solution.
- Chapter 4 – Describes the developed solution, including an overview of the proposed simulator architecture, details of its implementation, and the mathematical models used to support the implementation.

- Chapter 5 – Describes the tests conducted to evaluate, validate, and demonstrate the usefulness of the new YAFS modules, detailing the conditions under which the tests were performed and the results obtained.
- Chapter 6 - Finally, the last chapter presents the conclusions of this thesis, as well as suggestions for future research and development directions.

Chapter 2

Background and State of the Art

This chapter introduces the fundamental concepts that will form the basis for the subsequent development of this thesis, as well as a review of the work already conducted in the field. The objective is to assess whether Yet Another Fog Simulator (YAFS) is indeed the most suitable simulator to be enhanced, or whether there are other existing simulators with similar capabilities, and to determine if such an update is genuinely needed within the academic community.

The discussion opens with an outline of the Cloud-to-Thing Continuum (C2T), providing the foundations required to understand its scope and associated challenges. It then considers the role of satellite systems in enabling ubiquitous connectivity, discussing both the underlying orbital principles and the potential of hybrid satellite–terrestrial architectures. The focus then shifts to network simulation, where core methodologies, architectural aspects, and inherent limitations are examined. Finally, the chapter reviews the relevant literature, comparing existing simulators, assessing their support for mobility and satellite integration, and evaluating their suitability within the Cloud-to-Things paradigm.

2.1 Cloud-to-Things Continuum: An Overview

The C2T refers to an emerging computing paradigm that encompasses an integrated and distributed infrastructure, from centralised storage and processing in the cloud to devices located at the edge of the network. In this continuous and distributed model, the various layers - Internet of Things (IoT) devices, edge, fog and cloud computing - represented in Figure 2.1 can be coordinated to guarantee real-time transmission of data and provide support for different applications and use cases [Al-Dulaimy et al., 2024].

C2T represents an evolution from the traditional cloud computing model, which centralises data processing and storage. Although the cloud continues to offer scalable storage and processing capacities, it is no longer the sole point of application execution. In the continuum, edge computing operates directly on end devices or on equipment located close to the data source, providing local processing to reduce latency and improve efficiency. In addition, fog computing is

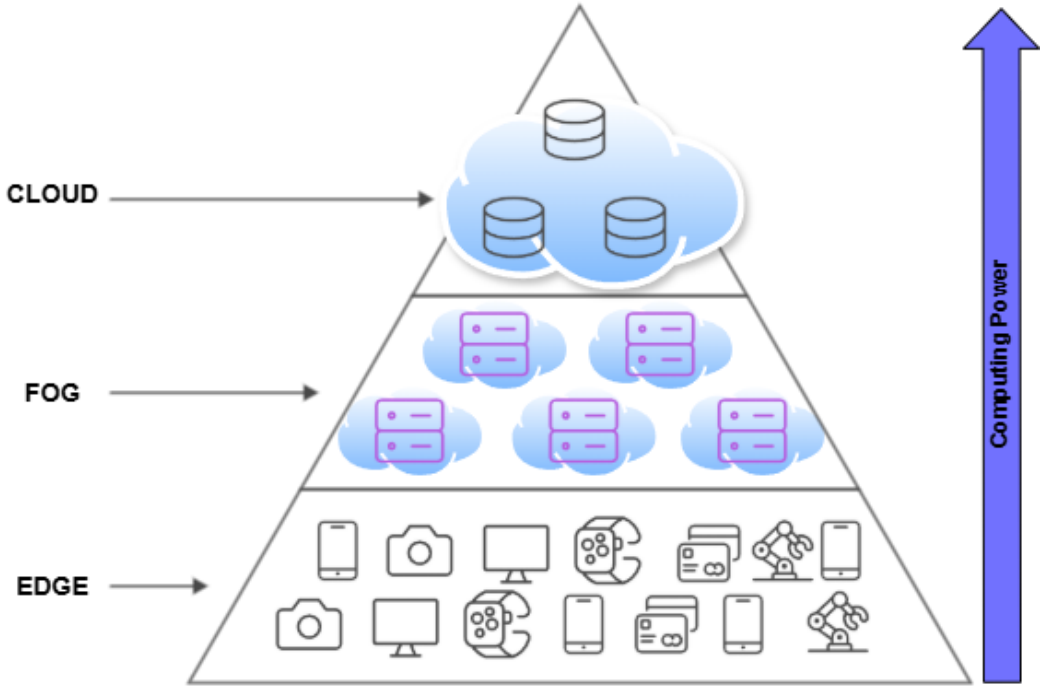


Figure 2.1: The Cloud-to-Things Continuum [Al-Dulaimy et al., 2024]

introduced as an intermediate layer between the cloud and the edge, providing distributed processing capacity that complements the continuum by performing tasks that require closer proximity to the data source [Bendechache et al., 2020].

This computing paradigm aims to respond to the new requirements brought about by the proliferation of IoT devices, which generate large volumes of data and require ever shorter response times. C2T promotes an infrastructure that dynamically adapts the execution of workloads, optimising the allocation of resources between the different layers to meet specific time and location requirements.

To better understand the workings and advantages of C2T, it is essential to explore the different types of computing that comprise it. Each layer C2T plays a specific role in the infrastructure, contributing to an integrated and flexible approach. Depending on the requirements and objectives of each application, one or more of these approaches can be used in combination to achieve the best results. The main types of computing in the continuum are presented below, highlighting their functions and benefits.

2.1.1 Cloud Computing

Cloud computing is a technological model that enables ubiquitous and convenient access to an infrastructure of configurable and shared computing resources. These resources can be provisioned and released quickly and efficiently, allowing users to use only the computing capacity they need at any given time. This flexibility results in the optimisation of the cost and the adaptive management of the

technological infrastructures [NIST, 2011].

In this model, resources are made available in a way that abstracts from their physical location, promoting efficient sharing and optimised use. Scalability is ensured by the capacity for automatic and dynamic provisioning, which allows the infrastructure to be adjusted according to changes in demand. Continuous control and monitoring of resource consumption ensure transparent and effective management, optimising performance and use of services.

Despite its many advantages, cloud computing has some limitations. Firstly, connectivity to remote data centres is an essential precondition, and network failures can significantly compromise performance, especially in time-sensitive applications. Additionally, the available bandwidth may not be sufficient to support the increased volume of data exchanged between the cloud and end devices, resulting in congestion and service degradation. As cloud data centres can be physically distant from end users, response times can be longer, negatively affecting application performance.

In addition, there are security concerns, as resources are shared between multiple users, which can increase the risk of data breaches or unauthorised access. These challenges emphasise the importance of considering complementary solutions, such as edge computing, to mitigate the limitations of cloud computing [Al-Dulaimy et al., 2024].

2.1.2 Fog Computing

Fog computing is a decentralised computing model that aims to move some of the cloud's resources and capabilities to devices located closer to the end users, acting as an intermediary between the centralised cloud infrastructure and the end devices. This approach makes it possible to position processing, storage and connectivity capacities in distributed nodes on the network, such as routers, gateways or other devices with computing capacity and an internet connection.

By bringing computing resources closer to the source of the data, fog computing significantly reduces latency, improves application performance and promotes greater geographical distribution of resources, which contributes to more efficient network utilisation. This architecture is particularly advantageous for time-sensitive applications where speed of response is essential.

However, fog computing has some limitations, since intermediate devices generally have more limited computing and storage resources than centralised data centres in the cloud. Consequently, this approach may not be suitable for all types of workloads, especially those that require large processing or storage capacities. For this reason, fog computing is often used in combination with cloud computing to maximise the advantages of both architectures [Hazra et al., 2023].

2.1.3 Edge Computing

Edge computing is also a decentralised model that aims to bring computing capabilities to the edge of the network, close to the devices that generate the data. Although this model is similar to fog computing, there are important differences between them. Fog computing adopts a hierarchical architecture, where computing resources can be positioned at any point between the cloud and the data source devices (things), acting in multiple layers of intermediary. In contrast, edge computing is limited to computing at the edge of the network, i.e., directly on devices or equipment close to sensors and actuators [Al-Dulaimy et al., 2024].

The advantages of edge computing are similar to those of fog computing, including a significant reduction in latency, improved application performance and efficient use of network resources. This approach is particularly beneficial for applications that require very fast response times, such as analysing data in real time, industrial automation systems and autonomous vehicles. However, due to its proximity to the data source, edge computing can offer an even more immediate response in certain scenarios where local processing is essential for quick decisions.

Furthermore to fog computing and edge computing, other models have emerged, which are more specific implementations adapted to specific objectives, based on the decentralisation and distributed processing principles of these approaches. These emerging models aim further to optimise resource allocation and application performance in specific scenarios.

Table 2.1 summarises the main differences between the types of computing - cloud, fog and edge - highlighting their characteristics, advantages and limitations.

Table 2.1: Comparison of Cloud, Fog, and Edge Computing

Characteristics/Models	**Cloud**	**Fog**	**Edge**
Type of Computing	Centralized	Distributed	Distributed
Resources	Unlimited	Limited	Limited
Energy Consumption	High	Low	Low
Latency	High	Low	Very low
Source Distance	Far	Near	Very near
Typical Applications	General cloud applications	Applications requiring short response times or real-time processing	Applications with critical latency requirements

Connectivity in C2T is an extremely important factor, as a network failure can result in significant implications, such as interruption of services, incomplete computations and degradation in the quality of applications. This is a particularly critical challenge in applications that require real-time response, such as autonomous vehicles or healthcare systems, where response time is essential.

One of the main problems affecting connectivity in the continuum is the high mobility of many IoT devices, such as mobile devices, vehicles and drones. These devices often traverse areas with different levels of stability in network coverage,

which can lead to frequent connection failures. These interruptions negatively affect Quality of Service (QoS) and security, especially in the fog layer, which needs to deal with a large amount of data and mobile devices [Laroui et al., 2021].

Moreover, the exponential increase in the number of devices connected to the network generates a massive volume of data, which can lead to infrastructure overload. This situation creates bottlenecks, resulting in a slower and more unstable connection, which directly affects the system's scalability. These issues highlight the need to develop solutions capable of guaranteeing connection stability and dealing with obstacles that can interfere with communication.

2.2 Ubiquitous Connectivity in C2T: The Role of Satellite Systems

In order to address the challenges of ubiquitous connectivity described above, it is of paramount importance to ensure a stable and reliable connection in C2T, particularly in scenarios characterised by high mobility and large volumes of data, to preserve QoS. This essential requirement has prompted growing interest in the potential of satellite technology as a solution [Al-Dulaimy et al., 2024]. The use of satellites offers the potential to provide coverage in areas where the terrestrial infrastructure may be insufficient or unstable. This represents a promising approach to achieving ubiquitous connectivity, which can be defined as a continuous and uninterrupted connection, regardless of the user's location.

In recent years, advances in satellite technology have reinforced the viability of this alternative. Modern satellites are increasingly equipped with advanced processing capabilities, allowing them to act as fog layer nodes. In addition to relaying data, these capabilities enable the satellites to perform local calculations, thereby reducing latency and optimising the flow of information. Furthermore, investigation of novel orbital paths, particularly low orbits, has yielded substantial advantages in terms of latency, exhibiting a markedly reduced latency compared to the conventional geostationary orbits used for television broadcasting.

These developments make satellites an attractive option to meet C2T connectivity requirements. The utilisation of satellite constellations in Low Earth Orbit (LEO) can facilitate enhanced coverage, accelerate response times and reinforce network resilience. Nevertheless, the adoption of these constellations presents a number of challenges, including the management of a considerable number of satellites, the mitigation of interference and the compensation for effects such as the Doppler effect, which can result in the distortion of the signal due to the high mobility of the satellites.

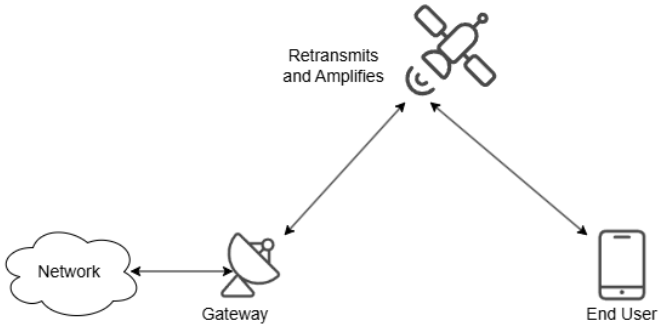
In this context, it is essential to explore the technical and operational characteristics of the different types of satellites and their orbits, analysing how they can be integrated into the C2T ecosystem to ensure stable, high-quality connections.

2.2.1 Fundamentals of Satellite Technology and Orbits

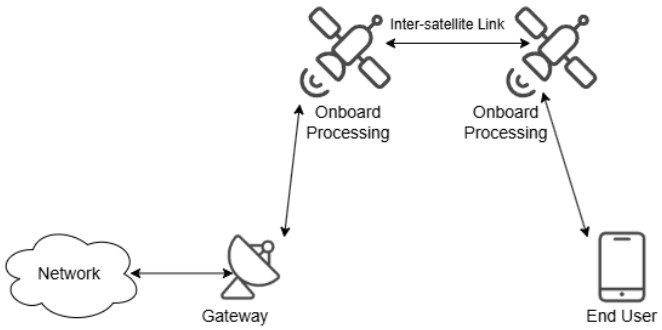
Communication satellites play a crucial role in transmitting and receiving information between gateways and end-users. They can be categorised into two main types: transparent and regenerative, as represented in Figure 2.2.

1. Transparent satellites act as passive repeaters, retransmitting and amplifying the received signal without any additional processing, which simplifies the system but limits its capabilities.

2. Regenerative satellites include onboard processing, which allows operations such as routing, modulation and others to be carried out. These satellites also make it possible to establish Inter-Satellite Links (ISL), which is particularly useful when a satellite does not have direct access to a gateway so that it can forward data to another satellite with available connectivity [Heo et al., 2023].



(a) Transparent approach



(b) Regenerative approach

Figure 2.2: Types of approach to satellite behaviour [Heo et al., 2023]

Satellites can also be categorised based on the altitude of their orbits, a factor that directly influences the coverage and latency of the services provided, which are as follows.

- Geostationary Earth Orbit (GEO), located approximately 36,000 km from Earth, has an orbital speed synchronised with Earth's rotation, meaning that the satellite remains stationary in relation to a fixed point on Earth's surface.

This orbit enables vast territorial coverage, allowing for the entire Earth's surface to be covered with just three satellites. However, the high altitude of the GEO satellites implies significant latency, which can be a limitation for time-sensitive applications.

- Medium Earth Orbit (MEO), located between 5,000 km and 20,000 km altitude, allows for a considerable reduction in latency compared to GEO and requires a moderate number of satellites for global coverage.
- LEO, located less than 5,000 kilometres from the Earth, offers low latency and less signal attenuation. However, for global coverage, a significantly larger number of satellites is needed in LEO, due to its reduced coverage area. On the other hand, MEO and LEO orbits are subject to a high Doppler effect, caused by the high mobility of the satellites, which can distort the signal and require appropriate compensation techniques to ensure communication quality.

In the context of satellite constellations, particularly in LEO orbits, various factors must be taken into account to optimise QoS. The orbital inclination angle, for example, directly influences coverage at different latitudes, with lower inclinations limiting coverage capacity in polar regions. In addition, the number of orbital planes and the number of satellites in each plane have an impact on both the total number of satellites in low orbit and the distance between them, a factor that determines the continuity of the connection and the quality of the signal for users. A higher number of satellites in orbit LEO can improve the signal-to-noise ratio, but implies an increase in the risk of interference and in the complexity of managing link exchange between satellites. Balancing these factors is therefore essential to guarantee an efficient and high-quality connection [Jiang et al., 2023].

Although satellites offer significant advantages in providing coverage, low latency, and resilience in C2T systems, achieving true ubiquitous connectivity requires more than a purely satellite-based solution. In order to leverage the advantages of both architectures, a seamless integration of satellite and terrestrial networks is essential. Such an approach ensures that the system can capitalise on the wide coverage and advanced processing capabilities of satellites, while also benefiting from the high-speed, low-latency connections of terrestrial infrastructure in well-covered areas.

This integration gives rise to Hybrid Satellite-Terrestrial Network (HSTN), an architectural model that combines the unique advantages of both types of networks. These networks enable dynamic allocation of resources, allowing devices to seamlessly switch between satellite and terrestrial links based on factors such as coverage, latency, and data demands. By implementing this type of architecture, it becomes possible to address the diverse connectivity requirements of C2T environments, ensuring stable, efficient, and high-quality communication across all layers.

2.2.2 Hybrid Satellite-Terrestrial Network Architectures

The HSTN architecture consists of integrating satellite networks with conventional terrestrial networks. The terrestrial Internet, which uses base stations and backhaul to connect these stations to the network core, provides broadband to urban and developed areas. Satellites, on the other hand, extend this coverage, allowing connectivity to everyone and everything in rural and difficult-to-reach areas, as illustrated in Figure 2.3. In addition, satellites help reduce the load on the terrestrial network in large cities, as lower-priority data can be transmitted by satellite, leaving the terrestrial network available for critical data that requires low latency and real-time processing [Jiang et al., 2023].

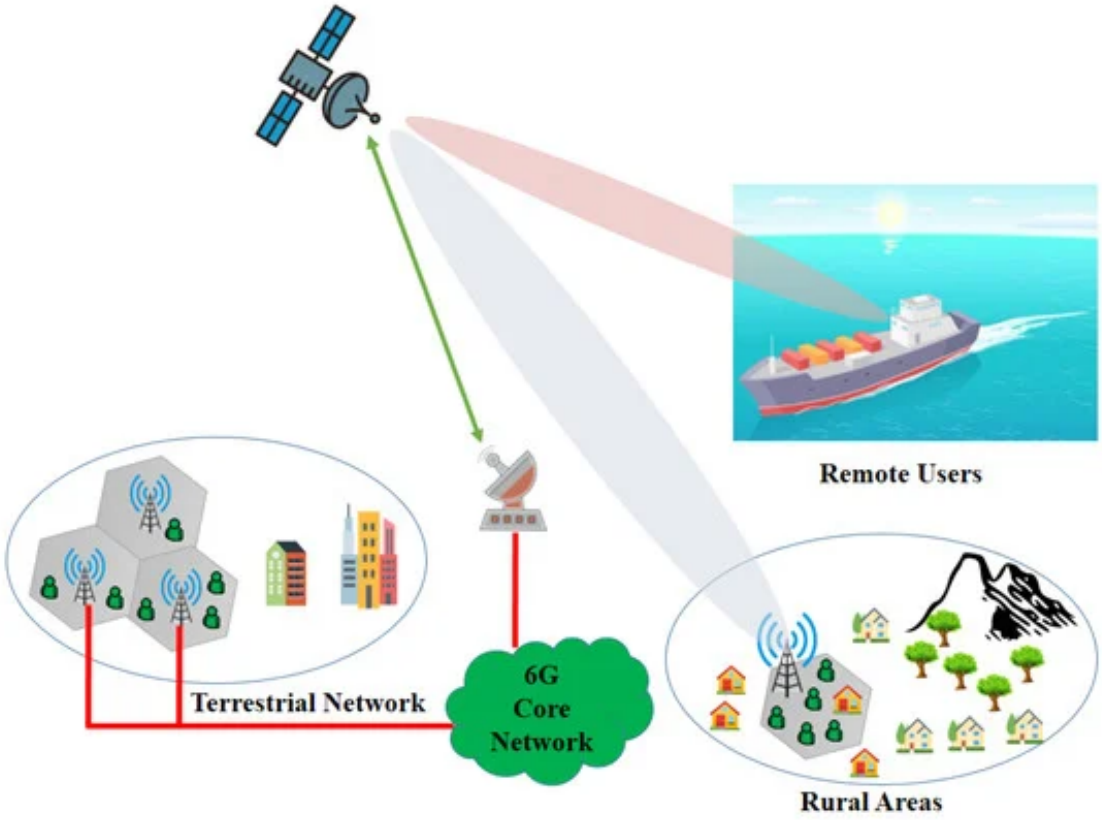


Figure 2.3: Hybrid Satellite-Terrestrial Networks Architecture [Tirmizi et al., 2022]

The main architectures of HSTN are divided mainly into two types: Cooperative and Cognitive [Tirmizi et al., 2022].

1. **Cooperative Architecture**

   In the Cooperative architecture, as the name suggests, the objective is to leverage each type of network to offer more robust and ubiquitous connectivity. This approach is divided into different models:

   - **Coordinated HSTNs**: In this approach, in addition to cooperation between networks, there is a unification of protocols and terminals, allowing for the use of common protocols at the physical layer. Thus,

devices can connect to any of the networks efficiently, automatically selecting the most suitable transmission medium without having to distinguish between satellite and terrestrial networks. This results in more optimised data exchange.

- **HSTN Relay**: In this model, to reduce the attenuation or fading of the satellite signal, relays are used that capture and retransmit the signal. These relays can process, amplify, or simply redirect the signal to the end user. This is particularly advantageous for users in places where direct communication with the satellite is limited, such as inside buildings. It is possible to employ a single relay, a network of relays, or even aerial relays, such as drones, to extend coverage and mitigate shadow zones.

2. **Cognitive architecture**

   Cognitive architectures, meanwhile, are designed to overcome a specific challenge: the limitation of available spectrum. In this approach, secondary users are allowed to share the same spectrum used by a primary network without compromising the QoS of primary users. This sharing improves spectrum efficiency by minimising resource scarcity. There are two main approaches to cognitive networks:

   - **Underlay model**: In this model, secondary users operate with limited transmission power in order to ensure that the interference generated remains within an acceptable level (called interference temperature), preserving the signal quality of primary users.
   - **Overlay model**: In this case, secondary users contribute to the transmission of primary users through cooperative communication techniques, increasing the efficiency of the overall system by helping with data retransmission or signal amplification, benefiting everyone.

   Cognitive HSTN are therefore a promising solution to increase the flexibility and adaptability of hybrid networks. They enable more efficient spectrum use, supporting the growing demand for high-capacity services and high-speed data.

The implementation of HSTN introduces a wide range of technological and operational complexities that must be carefully considered to ensure the success of this new technology. The integration of satellite and terrestrial systems, the implementation of spectrum sharing strategies and the coordination of multiple relay models present significant challenges that require rigorous testing to ensure seamless functionality and optimal performance. However, conducting such evaluations in real-world environments can be prohibitively expensive and logistically challenging, particularly when testing large-scale scenarios or novel configurations.

In this context, network simulation is an invaluable tool. By allowing the creation of virtual environments that accurately replicate the behaviour of real networks, simulations provide an effective and scalable method to validate the feasibility

and performance of new architectures such as HSTN, while avoiding the costs associated with physical testing. Such simulations permit researchers and developers to experiment with different models, identify potential bottlenecks, and optimise system parameters prior to the deployment of these networks in practical scenarios.

2.3 Principles of Network Simulation

Internet simulation entails the creation of a virtual system that is capable of reproducing the operation of a real network, obviating the need to use its physical infrastructure [Jiang et al., 2023]. The system must be capable of simulating the actual behaviour of a network through the use of mathematical formulas or by replicating the values and behaviours observed in operational real-world networks.

They are indispensable for evaluating scenarios that, in the real world, would be either complex or economically infeasible to test. These tools facilitate the evaluation of new protocols, the analysis of existing ones, experimentation with diverse topologies, the testing of innovative configurations, the implementation of novel orchestration techniques, and the validation of emerging technologies and architectures [Sarkar and Halim, 2008].

However, Internet simulators are not capable of replicating all the details of real networks with complete accuracy. Nevertheless, when properly modelled, they can provide results close enough to reality to allow researchers to gain a deeper understanding of their tests [Rampfl, 2013].

The following sections give an overview of the general architecture of a network simulator, the fundamental concepts associated with it and the existing limitations.

2.3.1 Core Concepts and Methodologies

The first fundamental concept is the distinction between a simulator and an emulator. A simulator utilises advanced techniques to model the entire behaviour of the Internet, including the devices that comprise this environment. In contrast, an emulator is limited to simulating the network that connects the end hosts and does not model the hosts themselves. Instead, it uses real operating systems to provide a more authentic representation. Emulators are often used in the context of network preparation and certification, where it is crucial to replicate real conditions [Pan and Jain, 2008].

The second key concept is Discrete Event Simulator (DES), as they are widely used in network simulators due to the discrete nature of networks: messages are transmitted at specific times, rather than continuously. This feature makes DES particularly suitable for modelling communication networks.

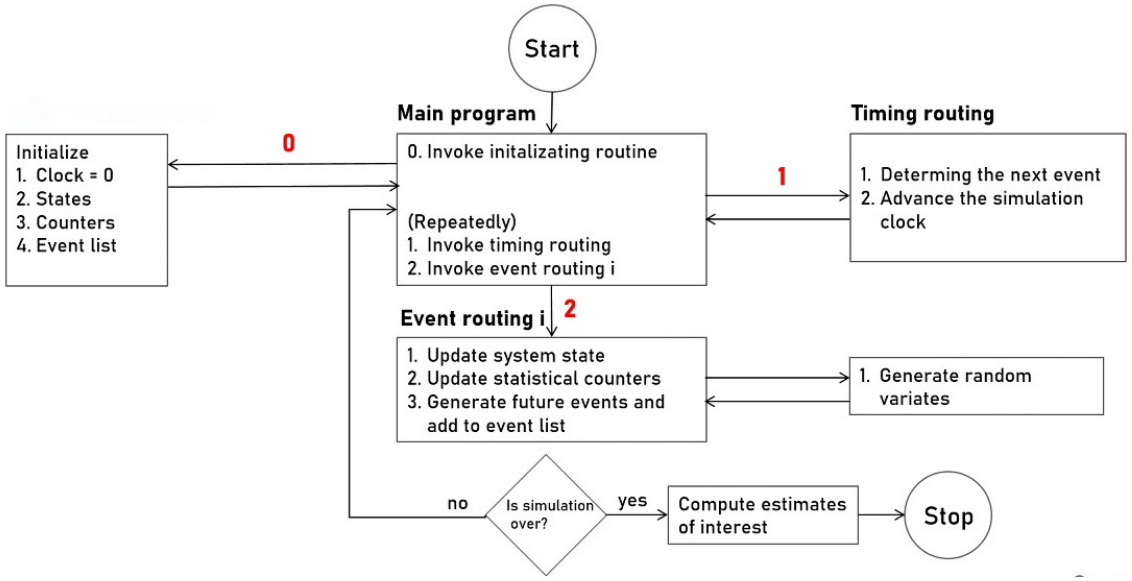


Figure 2.4: Discrete Event Simulator Diagram Flow [Studio, 2022]

The operation of a DES can be described as follows, as illustrated in Figure 2.4:

1. **Initial definition:** The possible states of the system and the simulation clock, which marks the times associated with the events, are configured.
2. **Main cycle:** The simulator runs an iterative cycle with the following steps:
   - Identification of the next event to occur;
   - Updating the simulation clock to the time corresponding to the event;
   - Updating the system state, based on the event processed;
   - Generating new random variables to determine future events.
3. **Termination:** The cycle ends when the simulation time is reached. At this point, the final statistics are computed and the results are presented.

The approach based on discrete events is simple but effective, making it possible to model the dynamic behaviour of communication networks accurately. This capability makes DES an indispensable tool for analysing complex scenarios and developing new technologies.

Other concepts that are important for understanding network simulation are the following.

- **Network topology**

  The topology of a network refers to the arrangement of its nodes and links. Different topologies, such as star, bus and ring, can have a significant influence on network performance. Simulating different topologies allows researchers to assess their impact on network behaviour and optimise their design for specific applications.

- **Latency**

  Latency is the delay between transmitting and receiving data on the network. It is a critical factor in network performance, especially in real-time applications such as videoconferencing and online gaming. High latencies can cause delays and interruptions, jeopardising the user experience.

- **Throughput**

  Throughput quantifies the volume of data successfully transmitted on a network during a specific time interval. It is a metric that reflects the network's ability to transport data. Higher throughput indicates a more efficient network capable of supporting larger volumes of data.

- **Packet Loss**

  Packet loss occurs when data transmitted from the source does not reach its intended destination. This phenomenon can be due to network congestion, transmission errors or hardware failures. Packet loss reduces network performance and may require retransmission of lost data, increasing the system overhead.

- **Bandwidth**

  Bandwidth refers to the maximum data transfer rate on a given network link, usually measured in bits per second. Higher bandwidths allow for faster data transmissions and support applications that require more processing power.

- **Real-time simulation**

  In certain applications, real-time simulation is essential to capture the dynamic behaviour of networks and their response to real-time events. This type of simulation makes it possible to accurately model time-sensitive applications and protocols, such as those used in critical networks or interactive environments.

Bearing in mind the concepts defined above, we can now look at the architecture that makes up a network simulator.

2.3.2 Network Simulator Architecture

The architecture of a network simulator is designed to achieve specific objectives, taking into account requirements such as the type of networks to be simulated, the functionalities to be implemented and the performance parameters to be evaluated. In general, according to [Tymchenko et al., 2021] and [Jiang et al., 2023], the architecture includes several essential components:

- **Input**

In this phase, the simulation scenario is configured, including the definition of the network topology, the node and link parameters and the characteristics of the applications. This is the phase in which the user defines the specifications required for the scenario to be simulated.

- **Simulation Engine**

  The simulation engine is the central core of the simulator, responsible for running the simulations and coordinating the interactions between the different elements of the network. Most network simulators use Discrete Event Simulation to reproduce the behaviour of real networks. This core can be subdivided into the following main categories:

  - **Resource management and allocation:** Implementation of algorithms for efficient management of network resources.
  - **Routing:** Simulation of algorithms responsible for packet routing.
  - **Transmission control:** This component generates the traffic that circulates on the network during the simulation. It can reproduce different types of traffic, such as web browsing, video streaming and file transfer, each with different patterns and characteristics.

- **Network Model**

  This component represents the topology of the network, the different types of nodes, the connections between them and their characteristics, such as latency, bandwidth and processing capacities.

- **Protocol Models**

  Protocol models incorporate the behaviour of various Internet protocols, such as TCP/IP, MAC, among others, allowing for a detailed and faithful representation of the interactions between layers.

- **Results and Analysis**

  Simulators offer tools to collect and analyse simulation results. These results can include metrics such as throughput, latency and packet loss, as well as offering detailed visualisations of network behaviour. Some simulators also integrate Graphical User Interface (GUI), allowing the results to be visually and intuitively represented, which facilitates analysis and interpretation.

To mitigate the complexity inherent in network simulation, certain layers can be abstracted. For example, the physical layer, which is often more demanding in terms of simulation, can be simplified, allowing the focus to be on layers that are more relevant to the area of research. This approach allows for the analysis of higher-level interactions and behaviours, simplifying the simulation process without compromising the objectives of the study.

2.3.3 Limitations

Although network simulation offers numerous advantages, it is also important to emphasise some of its limitations that must be taken into account when using this technology.

- **Scalability limitations**

  Network simulation makes it easy to add and remove components, devices and channels quickly and efficiently. However, a significant increase in the number of elements simulated, coupled with the growing complexity of the network, can result in high memory consumption and considerable time taken to complete the simulation. In extreme cases, the simulator may not even be able to finalise the process, which compromises the usefulness of the approach [Sarkar and Halim, 2008].

- **Accuracy and Validation**

  The accuracy of a simulation depends directly on the precision of the models used to represent the network and its components. Inaccurate models can generate misleading results, leading to wrong conclusions about the network's behaviour. Even with rigorous validation, simulation models can hardly capture all aspects of real networks, which can introduce inaccuracies in the results and limit the applicability of conclusions to real-world scenarios [Sarkar and Halim, 2008].

- **Dynamic scenarios**

  In the real world, networks are highly dynamic, often subject to rapid transformations, such as changes in topology or device failures. If the simulator cannot adequately represent these changes, it may not be able to reproduce certain critical scenarios, limiting the scope of the simulations.

- **Abstraction**

  Although abstracting certain components or layers of the Internet can be an advantage in reducing the complexity of the simulation, it can also introduce significant limitations. By omitting specific details, abstraction can compromise the accuracy of the results, affecting the reliability of the conclusions drawn [Rampfl, 2013].

2.4 Literature Review: Simulating C2T, Mobility, and Satellite Networks

Finally, this section examines other simulators that could have been considered as alternatives to YAFS, in order to assess whether its selection was indeed the most appropriate choice. In addition, it reviews the most commonly used libraries for satellite simulation, which could be integrated into YAFS to extend its capabilities. Lastly, it presents a selection of recent research papers where satellites and

user mobility were combined, highlighting the growing relevance of this trend and reinforcing the need for a simulator capable of accurately replicating such scenarios.

2.4.1 Simulators for Cloud-to-Things Environments

With regard to network simulators lacking native satellite support, the sheer diversity of available tools necessitates a filtering process to focus on those with relevant features and a strong presence within the research community.

The most frequently referenced network simulators include traditional tools such as NS-2 [Issariyakul et al., 2009], NS-3 [Riley and Henderson, 2010], and OMNeT++ [Varga, 2010]. These platforms are widely used for evaluating the performance of low-level network protocols. However, their primary focus on protocol behaviour limits their applicability for modelling higher-level aspects, such as load distribution, end-to-end latency, and computational resource management in distributed C2T environments. As such, these traditional tools are not considered within the scope of this comparative study.

Instead, this analysis concentrates on simulators explicitly designed for, or adaptable to, C2T environments, with the purpose of comparing their capabilities against those of the YAFS. This comparison provides a clearer understanding of the strengths and limitations of YAFS within its current scope, prior to any enhancements proposed in this work. These are the following:

EdgeCloudSim

EdgeCloudSim [Sonmez et al., 2018] is a simulator based on CloudSim [Buyya et al., 2009], with specific features to model Edge computing scenarios. This tool stands out for its modular architecture, which allows users to simulate, in detail, environments involving devices at the edge and in the cloud, supporting critical aspects such as mobility, network management and resource orchestration.

EdgeCloudSim's modular architecture is made up of different modules, shown in Figure 2.5. The core simulation module is responsible for loading and running the scenarios defined in the configuration files, as well as providing a logging mechanism that stores the results of each simulation in Comma-Separated Values (CSV) format, making it easier to analyse them later. The mobility module manages the location of edge devices and clients, taking into account a flexible and customisable mobility model that allows different scenarios and movement patterns to be simulated.

The Load Generator module is responsible for modelling the application's data, allowing specific tasks to be created according to the scenario's configuration, providing a realistic workload that can be adjusted for different simulation requirements. Additionally, the networking module allows you to work with arbitrary topologies, modelling Local Area Network (LAN), Wireless Local Area Network (WLAN) and Wide Area Network (WAN) networks, with options for defining link properties such as delay, capacity and parameter variability during the simulation.

Orchestration is another essential component of EdgeCloudSim's architecture, implemented through a module that allows you to define Virtual Machine (VM) allocation policies and decide where and how to handle client requests, optimising the distribution of tasks between edge and cloud servers. However, one limitation identified is the absence of an energy consumption model, although this can be added through extensions.

In addition, EdgeCloudSim, by using a simple queue-based architecture to calculate network delays, may have limitations when modelling more complex access technologies, which affects the accuracy of the simulation in scenarios with more dynamic networks. The simulator also allows the use of a cost model, but does not support VM migrations or grouping nodes across different layers comprehensively, instead limiting itself to nodes within the same layer. This modular and extensible structure makes EdgeCloudSim a widely used tool for studying edge computing architectures, VM allocation policies and fault tolerance mechanisms.

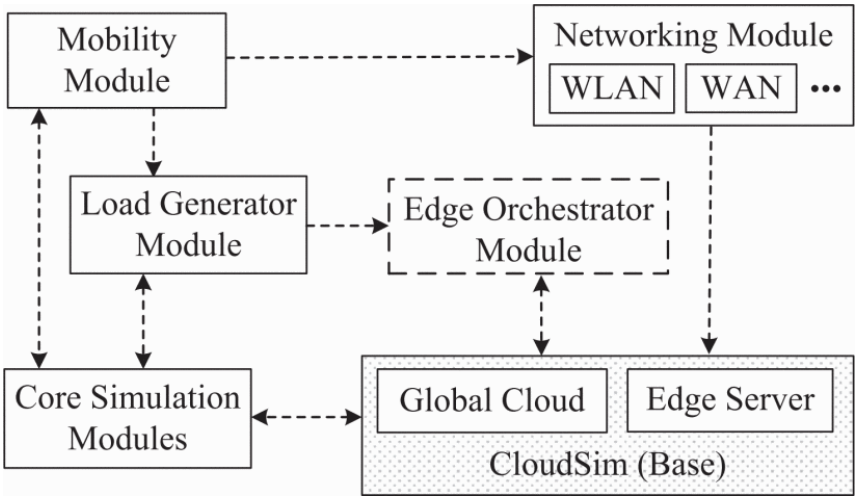


Figure 2.5: EdgeCloudSim Architecture [Sonmez et al., 2018]

FogNetSim++

FogNetSim++ [Qayyum et al., 2018] is a fog computing network simulator developed in C++ as an extension of OMNeT++, one of the classic network simulation platforms. Designed to model complex distributed computing scenarios, FogNetSim++ adopts a modular architecture, shown in Figure 2.6, organised into three main modules: the broker node, the fog nodes and the end devices. The broker acts as a central resource manager, monitoring all fog nodes and coordinating task allocation based on the availability of each node and the scheduling policy, being able to handle queued requests and carrying out scheduling based on a 'first come, first served' approach. This functionality allows the broker to transfer tasks to neighbouring fog nodes via the Handoff manager in the event of node mobility or overload, which is particularly useful in scenarios involving mobile devices.

Fog nodes are responsible for computational processing, responding to requests from end devices. The latter represent the terminal devices that generate the service requests and can move around during communication. With support for different resource scaling algorithms and the ability to adapt to various network

topologies, FogNetSim++ allows you to develop and integrate your own fog node management algorithms, improving performance in distributed computing contexts.

In addition to processing resources, FogNetSim++ includes cost and energy models, making it possible to calculate the costs associated with networks, storage and computing components, as well as the energy consumption of devices. The integration of detailed network models (such as bandwidth, error rate and packet retransmission) and support for various communication protocols make it possible to simulate more realistic network environments adapted to different application scenarios.

For flexible configuration, FogNetSim++ allows network parameters to be defined and customised modules to be incorporated. However, a limitation is the lack of significant updates since its introduction in 2018, which results in challenges in troubleshooting and compatibility with previous versions of OMNeT++ and INET.

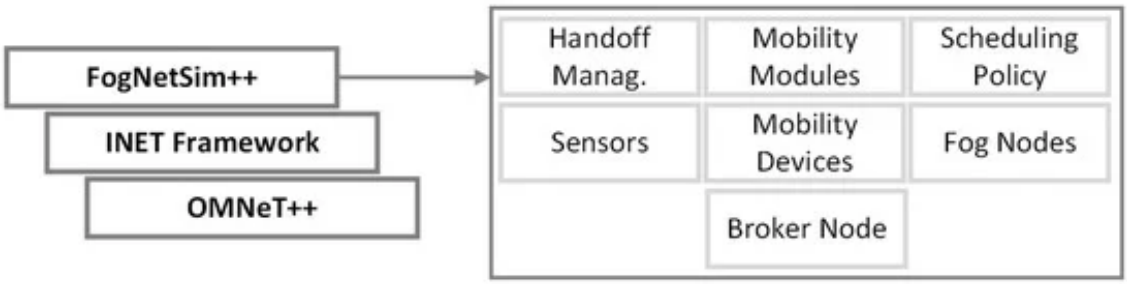


Figure 2.6: FogNetSim++ Architecture [Fahimullah et al., 2023]

iFogSim and iFogSim2

iFogSim [Gupta et al., 2016] is an extension of the CloudSim simulator designed to model and evaluate the impact of resource management techniques in Fog and IoT environments, focussing on critical aspects such as internet congestion, latency, cost and energy consumption.

The iFogSim architecture is structured in layers, as illustrated in Figure 2.7. In the lower layer, IoT devices connect to the physical environment, taking on the role of data sources or response triggers, as required by the simulation. This structure adjusts latency and connectivity parameters according to the scenario being studied.

The Fog devices in iFogSim are arranged in a hierarchical tree topology, where communications occur exclusively between pairs of devices in a parent-child relationship, mimicking the physical arrangements common in Fog computing architectures. This model makes it possible to exploit distributed communication between devices and the cloud, distributing data processing and filtering tasks.

The simulator includes three main services: Infrastructure Monitoring, Resource Management and Data Generation. Infrastructure monitoring tracks the utilisation, availability of devices and the energy monitoring that is especially important, given the critical role of battery life in many IoT devices. In contrast,

resource management (based on allocation and scheduling components) aims to ensure that applications respect QoS requirements and that resource use is efficient. Data generation plays a crucial role in modelling the data produced by sensors, including the way this data is structured, the volume of information transmitted and the frequency with which transmission occurs.

The architecture also allows the creation of application models based on the Distributed Data Flow (DDF) model, where an application is built through modules that process and exchange data. In this model, communication between modules is represented in directed graphs, with edges expressing data dependencies between modules, allowing realistic IoT applications to be recreated.

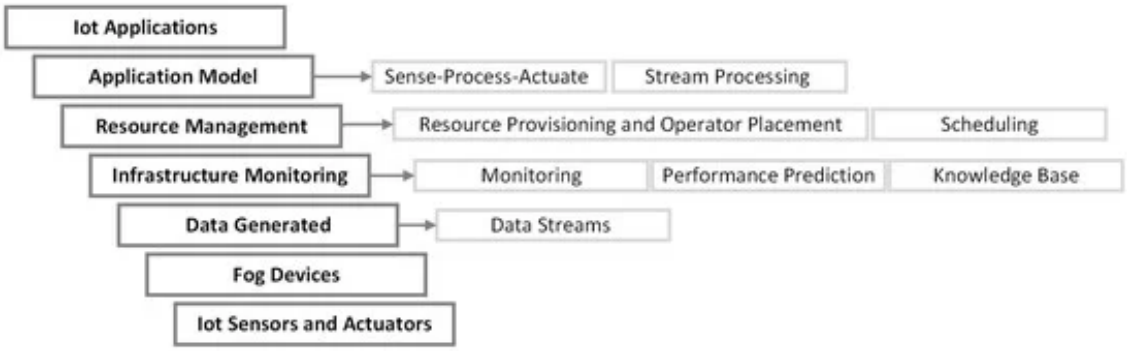


Figure 2.7: iFogSim Architecture [Fahimullah et al., 2023]

The latest version of iFogSim, launched in 2022 and known as iFogSim2 [Mahmud et al., 2021], introduced essential features such as support for the mobility of IoT devices, the creation of groups of Fog devices (clustering) and the management of microservices. These new features are intended to address the tool's previous limitations, making it possible to simulate more complex scenarios, including the migration of application modules and the coordination of modular services, fundamental aspects for a realistic simulation of Fog and IoT environments.

YAFS

YAFS [Lera et al., 2019] is a simulator developed in Python to evaluate the performance of C2T networks and is widely used to analyse resource allocation, scheduling and routing strategies.

Its architecture, pictured in Figure 2.8, is made up of six main modules (core, topology, selection, placement, population and application), allows for the detailed simulation of scenarios in distributed environments and support for inputs in JavaScript Object Notation (JSON) format, facilitating the configuration and reuse of scenarios.

The core module plays the role of integrating and controlling the lifecycle of all the processes, coordinating the execution of the simulation based on data input and output events. This module is developed on top of the SimPy [Monks and Harper, 2023] library, which offers an implementation of DES, allowing simulations in real time or with configurable steps, ideal for environments where the response time must be adjustable.

The topology, in turn, is configured using a JSON, which allows the definition of a flexible and dynamic network structure, including the ability to insert new links and nodes during the simulation run, reflecting real-time changes such as failures or infrastructure upgrades. Integration with the NetworkX [Hagberg et al., 2008] library extends the analytical potential of YAFS, making it possible to calculate advanced metrics such as centrality and clustering coefficient, which are crucial in simulations that seek to mimic complex, interconnected networks.

When modelling applications, YAFS uses a DDF model, similar to the one implemented in iFogSim. Applications are represented as directed graphs, where nodes represent service modules and edges represent the exchange of data between them. This model is especially advantageous for supporting modern paradigms, such as microservices and serverless functions, which require efficient communication between modules.

Messages, or dependencies, can be generated dynamically, with customisable attributes such as the number of instructions and byte size, which affect service time and transmission latency. This structure enables the use of dynamic selection and scheduling policies, thereby increasing the accuracy of simulations involving variable demands, such as selective routing and broadcast operations.

The selection, placement and population modules are fundamental for managing resources throughout the simulation, allowing the allocation of application modules and the distribution of workloads to be managed dynamically in response to changes in the network infrastructure or service demands. The selection module is responsible for choosing the entity that will run a particular application module, while placement determines the initial allocation of each module, and population controls the allocation of load generators, allowing mobility and timing patterns to be defined for the generation of new loads. These modules, combined with the ability to simulate failures and capacity adaptations in the processing nodes, allow for a detailed evaluation of failure management policies and performance optimisation strategies.

Furthermore, the Custom Control module, as its name suggests, allows the inclusion of customised simulation controls that are not originally implemented in the simulator. This functionality makes the simulator adaptable, allowing it to meet the specific requirements of any type of simulation scenario.

Finally, YAFS offers advanced functionalities for monitoring performance metrics such as response time, link latency and resource utilisation, recording the data in CSV files for later analysis.

After this brief introductory analysis of the simulators, it is now important to present a high-level assessment of their main features, based on the information provided in the scientific articles that describe each tool, the available documentation and the simulators' repositories, as well as articles that have previously analysed these simulators [Abreu et al., 2020], [Margariti et al., 2020], [Gill et al., 2021], [Bajaj et al., 2022], [Fahimullah et al., 2023]. Firstly, we present Table 2.2, which summarises the non-technical features of each simulator, followed by Table 2.3, which compares their technical features.

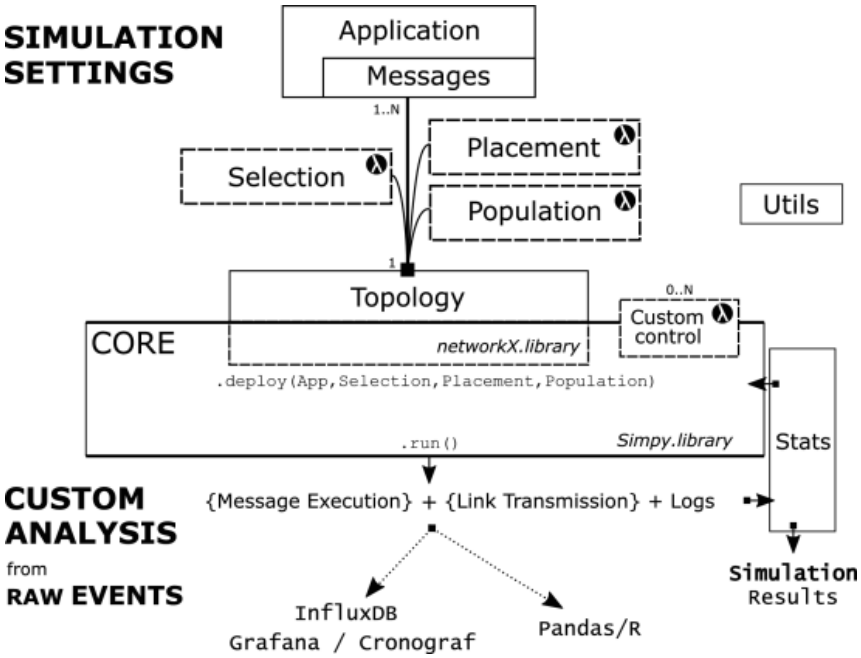


Figure 2.8: YAFS Architecture [Lera et al., 2019]

In Table 2.2, the field "first release" represents the year in which the first version of the simulator was made available, while "last release" refers to the year of the last published update or the last modification to the changelog. The "citations" field reflects the number of times the article associated with the simulator has been cited, based on data from Google Scholar. The "code frequency" is categorised into three levels: high if the source code has been updated in the last two years; moderate, if there have been no recent updates in the last two years; and low, if there have been no significant modifications since the project was created. The response frequency is also classified into three categories: high, when those responsible for the repository respond to issues in less than two weeks; moderate, if the response usually takes more than two weeks; and low, if there is no frequent response to issues. Finally, documentation is rated as high if it includes detailed information such as installation guides, usage examples, Application Programming Interface (API) reference and architecture description; moderate, if this information is only partially available; and low, if documentation is limited and lacks tutorials.

Table 2.2: Non-technical comparison of simulators without satellite support

Characteristics/Simulators	EdgeCloudSim	FogNetSim++	iFogSim	iFogSim2	YAFS
First Release	2018	2018	2016	2022	2018
Last Release	2020	2018	2017	2022	2022
Citations	630	208	1872	194	243
Code Frequency	Moderate	Low	Low	Low	High
Response Frequency	Moderate	Low	Low	High	High
Documentation	High	Low	Moderate	Moderate	High

From this table, it can be seen that the iFogSim2 and YAFS simulators are the most recently updated projects. In the case of iFogSim2, the release date of the first version coincides with the last update, suggesting more recent and active

development. In terms of the number of citations, YAFS, EdgeCloudSim and iFogSim stand out in terms of popularity among researchers, with iFogSim leading the way with 1872 citations, as it is the most established simulator and was the first to be launched. In terms of code frequency, YAFS stands out as the most active project in recent years. The "response frequency" is equally relevant, since a quick response facilitates the development and problem-solving process; here, iFogSim2 and YAFS are the ones that respond most promptly to questions and close issues. Finally, in terms of documentation, YAFS and EdgeCloudSim are the simulators with the most information on use, examples and architecture.

With regard to the technical analysis, "language" refers to the programming language used to develop the tool. "Mobility support" indicates whether the simulator is capable of modelling the movement of devices, which is important for simulating realistic scenarios. "Microservices support" indicates whether the simulator utilises a microservices orchestration model, while "fault injection" represents the capability to introduce faults into the simulated network, which is crucial for testing the robustness of fault-tolerance algorithms. The "cost model" and "energy model" refer to the simulator's ability to calculate energy and monetary costs for a given scenario, reflecting the applicability in real scenarios. Finally, a simulator is considered scalable if it can support an increase in the number of nodes or their capacity without compromising performance.

Table 2.3: Technical comparison of simulators without satellite support

Characteristics/Simulators	**EdgeCloudSim**	**FogNetSim++**	**iFogSim**	**iFogSim2**	**YAFS**
Language	Java	C++	Java	Java	Python
Mobility Support	✓	✓	✗	✓	✓
Microservices Support	✗	✗	✗	✓	✓
Fault Injection	✗	✗	✗	✗	✓
Cost Model	✓	✓	✓	✓	✓
Energy Model	✗	✓	✓	✓	✓
Scalability	✗	✓	✓	✓	✓

This analysis shows that Java is the most common language among simulators. Concerning mobility support, a key feature for applications in fog and cloud environments, all simulators except iFogSim support motion modelling, which is a significant disadvantage of the latter. Support for microservices, a growing feature due to the popularity of modern software architectures, is only present in iFogSim2 and YAFS. As far as fault injection is concerned, only YAFS offers this capability, making it more suitable for complex scenarios and evaluating fault tolerance algorithms. Almost all the simulators provide cost and energy models, a notable exception being EdgeCloudSim, which also stands out negatively for its lack of scalability.

It is therefore possible to emphasise that iFogSim, iFogSim2, and YAFS emerge as the most robust simulators adapted to different research scenarios. Among these, YAFS stands out as the most suitable choice for the purposes of this work due to its combination of modularity, active development, comprehensive documentation, and support for key features such as mobility, microservices, and fault injection. Additionally, the articles analysed indicate that YAFS, along with iFogSim and iFogSim2, reports the largest number of metrics, which significantly

increases its applicability in multiple use cases. This comprehensive set of features makes YAFS particularly advantageous for researchers seeking flexibility and analytical depth in complex simulations, allowing detailed and comparative analysis across different scenarios and configurations.

In addition, the tools mentioned above have been optimised mainly for simulating environments with fog and clouds. However, for satellite simulation, which involves orbital dynamics and the movement of celestial bodies, these simulators are inadequate. To overcome these limitations in YAFS, it is necessary to expand its functionality by incorporating external libraries specialised in orbit simulation.

2.4.2 Modeling Orbital Mechanics and Satellite Mobility

The field of satellite orbit simulation encompasses a vast array of libraries, rendering an exhaustive analysis of all available tools impractical. Consequently, this section will focus on the most widely recognised libraries within the academic community that are open source and implemented in Python. The choice of Python as the main language is motivated by its prevalence in this field, as well as its simplicity and ease of use. Furthermore, YAFS is developed in Python, which facilitates rapid development and integration. The libraries selected for analysis are as follows:

Skyfield

The Skyfield library [Rhodes, 2019] has been developed in pure Python and is designed to accurately calculate the positions of celestial bodies, including stars, planets and satellites, in their orbits around the Earth. The library is compatible with a range of widely used data formats, including Two-Line Element (TLE), CSV and JSON, and its sole dependency is the NumPy package.

The software's principal characteristics are its high degree of precision and the accuracy of its calculations. The results produced by Skyfield are in close agreement with those generated by the United States Naval Observatory and the Astronomical Almanac, with a margin of 0.0005 seconds of arc, thereby ensuring excellence in astronomical calculations. Furthermore, the library is able to forecast the positions of Earth satellites using orbital elements in the standard SGP4 (TLE) format. The library also supports efficient vector calculations based on these elements, which allows for the projection of future trajectories with ease.

Another advantage is the flexibility in data format support, which allows satellite data to be integrated into common formats, thereby facilitating its use in various sources and applications. The fact that Skyfield is written in pure Python eliminates the need for compilation processes, thus ensuring compatibility with various versions of Python, including 2.6, 2.7 and 3.x. Furthermore, the unique dependency on NumPy, a widely used package for mathematical operations and scientific computing, greatly simplifies use. Additionally, the library implements a corrected and updated version of the algorithm described in the Revisiting Spacetrack Report 3 study, which improves the accuracy of orbital predictions.

However, the library has some limitations. The accuracy of the data in TLE format is limited to about 1 kilometre at the time of the epoch, but this accuracy degrades rapidly over time. It is therefore necessary to update the orbital elements frequently or to use historical archives to predict past events. For Near-Earth objects (NEOs), Skyfield does not apply corrections to the travel time of light, as these corrections have no relevant effect in this specific context. This may be less useful for applications that require a more detailed model.

Poliastro

Poliastro [Rodríguez et al., 2022] is a library designed for the analysis and simulation of orbital motion. The library has been developed with the objective of facilitating the manipulation of satellite trajectories and other space objects. It offers an intuitive interface and a robust set of functionalities. Despite its prominence in the field, the project is currently discontinued, as indicated by the "No Maintenance Intended" label on GitHub, which implies the absence of future support and updates.

Poliastro offers a range of analytical and numerical capabilities for the propagation of orbital trajectories, enabling the prediction of the position and velocity of orbiting objects based on mathematical and computational models. Furthermore, the library enables the conversion between disparate orbital representations, including position-velocity vectors and classical orbital elements. A further significant functionality is the ability to transform between coordinate systems, including geocentric and heliocentric references.

Poliastro also incorporates tools for graphical representation of trajectories, thus enabling dynamic visualisations of orbits and movements. Moreover, the library facilitates calculations related to NEOs, including asteroids and comets, thereby supporting investigations of celestial bodies that approach Earth's orbit.

Notwithstanding its capabilities, Poliastro is not without notable limitations. The library does not support the TLE data format, which is widely employed to describe satellite orbits, especially in the context of practical orbital tracking and prediction. Additionally, the project's maintenance status raises concerns regarding its reliability for long-term use, particularly in situations that demand consistent support and compatibility with emerging technologies.

Orbital

Designed for calculations in the field of orbital mechanics and analysis of space trajectories, using Keplerian elements and state vectors. This tool [McLean, 2014] is particularly suitable for modelling orbits, calculating manoeuvres and evaluating orbital properties, offering a comprehensive set of functionalities and support for fundamental operations in astrodynamics.

Its main features include the definition of celestial bodies, with models covering the planets of the Solar System. This allows orbits to be associated with reference bodies such as Earth or Uranus, facilitating accurate and efficient calculations of orbital parameters. The library also allows Keplerian orbits to be modelled, so that orbits can be defined using classical elements. In addition, propagation methods are provided to extrapolate positions along the orbit based on anoma-

lies.

Another notable feature is support for the TLE format, allowing orbits to be created from orbital element data in this format, and using the SGP4 model for position prediction. In addition, the library allows for the calculation and application of impulsive orbital manoeuvres, with subsequent propagation of the resulting orbit, allowing detailed analysis of changes in orbital parameters.

Despite its capabilities, Orbital does have some limitations. The library has been optimised for calculations based on Keplerian orbits, making it less suitable for modelling complex orbital perturbations or for long-term analyses. In addition, as with other tools that use the SGP4 model, the precision of data in the TLE format is limited and significantly degrades over time, depending on the epoch of the data. Finally, the reliance on the Keplerian format can lead to discrepancies with models that take into account non-Keplerian effects, limiting their use in scenarios that require more sophisticated analyses.

AstroPy

AstroPy [Price-Whelan et al., 2022] is a central library for the astronomy and astrophysics community in Python, providing basic tools and functionality for exploring, analysing and processing astronomical data. Developed as a community project, the library is widely used by developers and researchers working with astronomical data, and serves as the basis for a wide range of related packages. Despite its versatility, AstroPy does not specialise in orbital simulations, but provides valuable support for calculations and operations that complement more specific tools in this area.

Its main features include support for data structures and transformations, allowing precise manipulation of astronomical constants, unit systems and celestial coordinates. The library also has advanced features for file input, output and manipulation, with support for standard astronomy formats, making it easy to read and write data efficiently. In the area of scientific computing, AstroPy provides resources for cosmological analysis, statistics, and astronomical data processing, including operations such as convolutions.

Despite its robustness, AstroPy has some limitations. As a general library, it lacks specialisation in orbital simulations and satellite mobility, which may limit its use in certain scenarios. Some of its functionalities also depend on integration with external or related packages, which may require complementary tools. Finally, the wide variety of available features can present a steep learning curve for new users, making initial adoption difficult.

Beyond

Beyond [Schmidt, 2021] is a library dedicated to the study of space flight dynamics, with a focus on developing a simple and intuitive API for space observations. Although not optimised for computational efficiency or performance, the library is suitable for academic applications and for those wishing to better understand the principles of orbital mechanics and flight dynamics.

Its main functionalities include orbital modelling and flight dynamics, integrat-

ing numerical and analytical propagators such as Kepler and SGP4. The library also allows the solution of the Lambert problem and the analysis of relative orbital motion. A listener and event mechanism facilitates the detection of orbital events, such as maxima, nodes, and anomalies, thereby extending its applicability.

Beyond supports input and output operations in widely used formats, such as TLE, and provides integration with external sources of ephemeris and orbital data. It also includes useful functions for advanced calculations such as interpolation, matrix operations and analysis of relative motion in orbital constellations or interplanetary orbits.

Despite its advantages, the library has some limitations. Its dependence on versions of Python 3.6 or higher can make it difficult to use in projects using earlier versions of the language. In addition, Beyond is not optimised for scenarios that require high precision or computational power, and is more suitable for educational purposes and conceptual studies than for practical applications that require extreme rigour.

High-level Evaluation

After presenting the tools, Table 2.4 summarises the main characteristics of the libraries analysed, highlighting the most relevant aspects for a high-level comparison.

The first column identifies the dependencies used by each library. This factor is crucial, since a greater number of dependencies increases the likelihood that updates to them could cause malfunctions in some of the libraries' functionalities.

The code upgrade metric is categorised into three levels: high, when the code has been updated in the last two years; moderate, if it has not been updated in that period; and low, if the code has not been changed since it was created.

The documentation is assessed according to three levels of completeness: high, when it includes detailed information such as installation guides, usage examples, API references and architecture descriptions; moderate, when some of this information is missing; and low, when the documentation is limited and lacks tutorials or practical examples.

Support for the TLE format is an essential criterion, since this format is widely used to enter satellite data. In addition, the ability to predict orbits - both existing and simulated - is of high importance, as it makes it possible to analyse already established trajectories or to create and test new types of orbits.

Finally, it is essential to assess whether the main objective of the library is aligned with the specific requirements of the work in question, ensuring that the tool selected is suitable for the needs of the project.

The comparison table shows that most libraries have minimal dependencies, with the exception of Poliastro, which requires a significantly higher number of packages. As for maintenance and updates, Beyond, Skyfield and Astropy stand out as the most active, while Poliastro has a significant limitation as it is no longer

Table 2.4: Comparison of Python Libraries for Orbital Mechanics and Satellite Motion Simulation

Attribute	Poliastro	Skyfield	Orbital	Astropy	Beyond
Dependencies	Astropy, NumPy, SciPy, Numba, Jplephem, Plotly	NumPy	NumPy	None beyond Python standard	NumPy, sgp4
Code Upgrade	No longer maintained	High	Low	High	Moderate
Documentation	High	High	Moderate	High	Moderate
TLE Support	✗	✓	✓	✓	✓
Predicts Existing Orbits	✗	✓	✓	✗	✓
Predicts Simulated Orbits	✓	✓	✓	✗	✓
Main Focus	Orbit propagation and maneuvers	Prediction of satellite and celestial positions	Modeling of Keplerian orbits	General-purpose astronomy toolkit	Orbital dynamics and event analysis

maintained. The documentation is adequate in all the libraries analysed and most offer TLE support.

For network simulators based on satellite movement, the ability to predict both existing and simulated orbits is essential. In this respect, only Skyfield, Orbital and Beyond fulfil both requirements and stand out as the most complete options. Selecting the most appropriate library will depend on the specific requirements of the simulator. If the focus is on simulating large, scalable networks, Skyfield stands out for its optimised methods for dealing with a large number of simulated objects and its solid support for TLE. For scenarios that require more detail on orbits and orbital simulations, Beyond is the best option, offering advanced tools for orbital modelling and event prediction. For simpler simulators, focused on Keplerian calculations and basic manoeuvres, Orbital is the best choice.

Subsequently, it is imperative to present simulators that offer specific support for satellites. These simulators can utilise the aforementioned libraries to extend their functionalities or possess intrinsic capabilities to address satellite-related operations. Furthermore, it is vital to determine whether these tools are capable of simulating fog and cloud computing environments, enabling a comprehensive evaluation of their potential in complex and distributed scenarios.

2.4.3 Existing Network Simulators with Satellite Support

In the context of simulators with satellite support, as in the cases presented above, a diverse range of options is available for comparison. The selection process was guided by the following criteria:

1. The chosen simulators were deemed the most relevant for scenarios within C2T environments.
2. These simulators originate from well-established and widely recognised tools, with Network Simulator 3 (NS-3) being a notable example.

For the purpose of comparison, the decision was made to adopt YAFS in combina-

tion with the Skyfield library. Skyfield was selected due to its ability to simulate large satellite constellations, including accurate orbital prediction based on TLE and JSON files.

From this point onwards, the analysed simulators and their main characteristics will be presented.

SatEdgeSim

SatEdgeSim [Wei et al., 2020] is a simulator designed to study cloud computing scenarios with satellites. This simulator combines the cloud computing simulation capabilities of CloudSim Plus with the modular architecture of PureEdgeSim [Mechalikh et al., 2019], extending its functionalities to model distributed computing environments based on satellites.

Its architecture, shown in Figure 2.9, includes modules responsible for simulation management, task generation, location management, task orchestration and network communication, enabling detailed modelling of customised scenarios.

The Simulation Manager plays a central role, responsible for overall simulation management and generating results. In addition to processing the results from its dependent modules, it offers additional functionalities, such as calculating the end-to-end delay of tasks and their success or failure rates.

The main purpose of the DataCenters Manager is to create and manage the network nodes representing the satellites in the simulation scenario. This module includes the Tasks Generator, which assigns applications to each device, ensuring that the messages exchanged are sent based on a Poisson distribution. In addition, the Location Manager is responsible for managing satellite positions using the Satellite Tool Kit (STK) satellite mobility library. At the same time, Tasks Orchestration allows different task distribution algorithms to be customised according to the definitions of the scenario being analysed.

The network module is based on PureEdgeSim's network load and bandwidth limitation model, and also incorporates the propagation delay between satellites. This approach significantly increases the realism of the simulation by taking into account the particularities of inter-satellite communications.

Finally, the Scenario Manager is responsible for initialising all the parameters needed to start the simulation.

With its ability to create complex scenarios, accurately model inter-satellite communications and manage multiple network nodes, SatEdgeSim offers a robust platform for analysing task orchestration strategies and evaluating the performance of satellite-based distributed systems.

Satellite Network Simulator 3 (SNS3):

The SNS3 [Puttonen et al., 2014] is an extension of the renowned NS-3 network simulator, specifically adapted to simulate satellite networks based on the Digital Video Broadcasting (DVB) standard. This simulator stands out for its detailed functionalities in the lower layers of the protocol stack, allowing modelling of processes such as modulation, channel coding and data transmission, with sup-

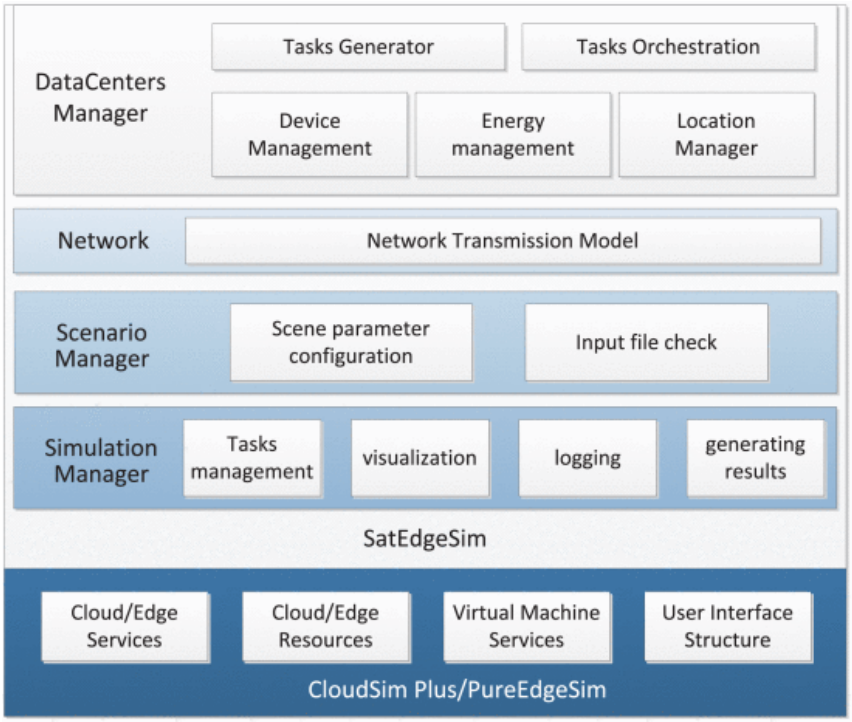


Figure 2.9: SatEdgeSim Architecture [Wei et al., 2020]

port for communications between satellites and ground stations. However, it has limitations when it comes to simulating large satellite constellations, as it does not offer support for satellites in low orbits (LEO), only offering access for a satellite of the type GEO, restricting its applicability to very specific scenarios.

The SNS3 architecture, shown in Figure 2.10 , consists of a structure designed to simulate interactive GEO networks with multiple transparent beams. This structure includes modules and nodes that represent different components of the system, such as user terminals, satellites, gateways, the Network Control Centre (NCC) and ground nodes.

One of the main innovations of SNS3 is the implementation of a new class of network devices, called SatNetDevice, which supports the stacks of the new DVB protocols. In addition, a new implementation of the SatChannel class was developed which allows calculations of received signal power, significantly increasing the realism of simulations in the physical layer.

The simulator also offers the ability to configure hybrid scenarios that integrate terrestrial networks and satellites. In the terrestrial domain, SNS3 supports various access technologies provided by NS-3, such as point-to-point networks, Carrier Sense Multiple Access (CSMA), Wi-Fi and Long Term Evolution (LTE), allowing scenarios to be explored combining terrestrial and satellite-based networks. This integration enables the joint analysis of network performance across different domains.

Another important aspect of the simulator is the modelling of NCC as a module shared between all gateway nodes. This centralised module assumes the existence of a single NCC entity and establishes an ideal communication channel

with the gateways, guaranteeing efficient coordination of network operations.

Although it has limitations related to scalability and support for satellite constellations in low orbits, SNS3 stands out for its extensibility through the integration of other NS-3 modules, making it a versatile and suitable tool for simulating scenarios where it is important to evaluate the lower layers of the network.

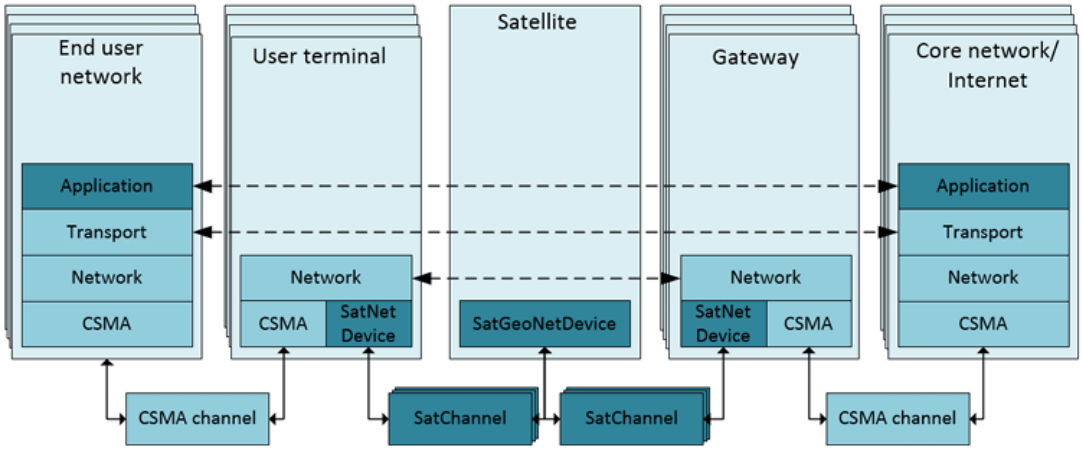


Figure 2.10: Satellite Network Simulator 3 Architecture [Puttonen et al., 2014]

OpenSAND:

OpenSAND [Alenia and CNES, 2006] distinguishes itself from other simulators in this category by its ability to support terrestrial networks and the possibility of configuring and integrating real equipment during simulation. This feature provides a high level of fidelity, making OpenSAND a particularly useful tool for validating applications, protocols and algorithms in realistic scenarios. In addition, the ability to interconnect real applications and hardware significantly expands its field of use.

One of OpenSAND's main features is its ability to emulate satellite communication systems based on the DVB standards, covering everything from the physical layer to the application. This versatility allows for detailed analysis of various aspects of satellite communication, contributing to the evaluation of new algorithms and protocols, as well as the validation of applications under controlled conditions close to reality.

OpenSAND's architecture requires at least three machines for it to work properly, as shown in Figure 2.11. Each machine plays a specific role: one simulates the satellite, another represents the gateway (ground station) and the third simulates the user terminals. This design not only allows for a faithful representation of the system, but also facilitates interconnection with real equipment and applications.

Despite its advantages, the OpenSAND documentation available does not specify the number of satellites that can be simulated simultaneously. This limitation may restrict the simulator's applicability in studies involving extensive satellite constellations, such as mega-constellations of satellites in low orbits (LEO), where scalability is an essential factor.

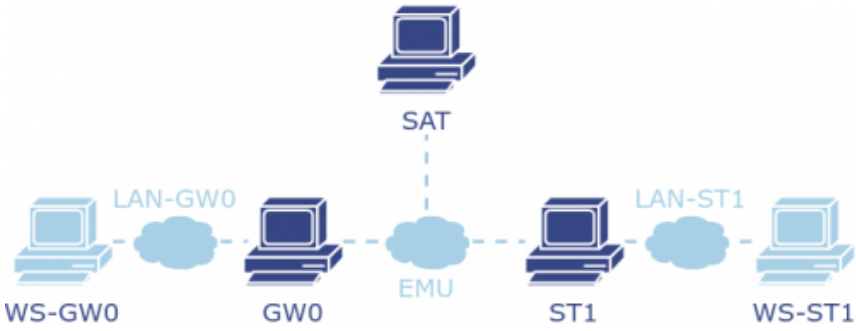


Figure 2.11: OpenSAND Architecture [Alenia and CNES, 2006]

Open Source Satellite Simulator (OS³):

OS³ [Niehoefer et al., 2013] is a simulator developed on the OMNeT++ platform, characterised by its architecture that enables the import of real data from satellites and meteorological conditions. This functionality adds a high level of fidelity to the simulation, allowing more accurate metrics to be calculated and realistic scenarios to be modelled, especially in studies that require rigorous analysis of satellite signals. The detailed architecture of the simulator is illustrated in Figure 2.12.

The main objective of OS³ is to offer a flexible simulation platform for satellite signal evaluation processes. Its architecture has been designed to ensure the separation of essential functionalities into external modules or libraries, minimising inadvertent changes to the system's basic operations. To facilitate the process of configuring simulations, OS³ also includes a GUI developed in Java.

In the graphical interface, users can select satellites and parameters of interest to configure the simulation, as well as receiver types and adverse weather conditions, such as heavy rainfall. The relevant TLE files are automatically downloaded using integrated web service routines, and all necessary files for OMNeT++ are automatically generated. The simulator utilises well-known algorithms, such as SDP4/SGP4, in conjunction with the INETMANET framework to realistically simulate satellite movements. At each step of the simulation, the corresponding communication channel is evaluated, including specific connectivity considerations.

In addition, OS³ provides the user with full access to the satellite positions and corresponding signal strengths, allowing customised features to be integrated into the OMNeT++ code as required. Satellite movement can be visualised in two preconfigured ways in the simulator:

- **Sky-View**, which displays satellite movements in terms of azimuth and elevation, based on the position of the receiver.
- **Map-View**, which uses a global colour map, where each ground station is included through a georeferenced transformation into the OMNeT++ pixel coordinates.

Both visualisation options provide realistic channel properties, enabling detailed

analysis methods such as calculating free-space loss and signal-to-noise ratio at specific positions.

Despite the significant advantages offered by its modular architecture and the integration of real data, OS³ has an important limitation in terms of its scalability: the number of satellites that can be simulated simultaneously is restricted.

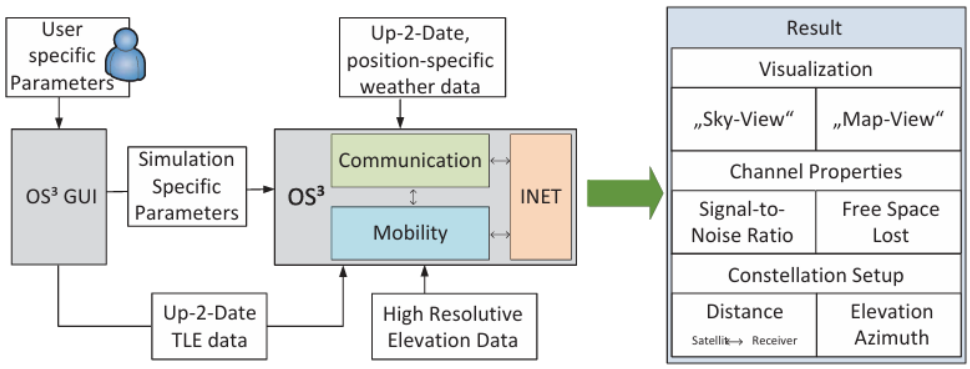


Figure 2.12: Open Source Satellite Simulator Architecture [Niehoefer et al., 2013]

FLoRaSat:

FLoRaSat [Fraire et al., 2022] is a simulation tool based on the OMNeT++ platform, explicitly designed for IoT networks that use direct communications via satellites. This simulator supports an integrated constellation of 16 LEO satellites and 1,500 ground nodes that communicate using the low-power Long Range Wide Area Network (LoRaWAN) protocol. Its architecture, shown in Figure 2.13, allows complex scenarios to be simulated, offering functionalities ranging from orbital dynamics to routing between satellites.

Orbital dynamics in FLoRaSat is modelled with trajectories generated using the Contact Plan Designer tool, an STK plugin. During the simulation, these trajectories are utilised by modules dedicated to satellite mobility, which calculate inter-satellite distances and visualise movements. In this context, devices IoT are considered fixed points on the ground, while satellites are the only mobile elements.

Communication between satellites is made possible by ISL that follow a basic routing model. This model employs a 'right-up' routing method for ascending traffic and a 'left-down' method for descending traffic, ensuring efficient connectivity within the constellation. However, the simulator architecture allows for the easy integration of more advanced routing algorithms, such as Dijkstra or methods based on contact graphs, expanding the possibilities for experimentation.

Another important aspect is the application model, which has been adapted to support delay-tolerant scenarios, allowing devices to buffer data until a satellite is available for transmission. This feature is particularly relevant in Direct-to-Satellite IoT (DST-IoT) networks, where connectivity can be intermittent.

The communication channel is also modelled taking into account factors such as attenuation, interference, and capture effects on the receiver, ensuring a realistic

representation of transmission conditions. FLoRaSat's architecture also includes specialised modules for communication management, such as LoRaNic and LoRaGWNic, which implement the LoRaWAN protocol layers in the ground devices and on board the satellites. The PacketForwarder module is responsible for forwarding the packets to the ground stations, while the ISLPacketForwarder manages communication between the satellites in the constellation.

FLoRaSat also offers an advanced graphical interface that makes it easy to configure scenarios, allowing users to select parameters such as constellations, traffic patterns and atmospheric conditions. The results can be visualised using global maps or azimuthal graphs, providing a clear analysis of the performance of the simulated networks.

Despite its sophistication and versatility, the simulator has some limitations, including support for a fixed number of 16 satellites and 1,500 terrestrial devices. In addition, the lack of mobility in IoT devices represents a significant disadvantage in scenarios that require modelling of mobile devices.

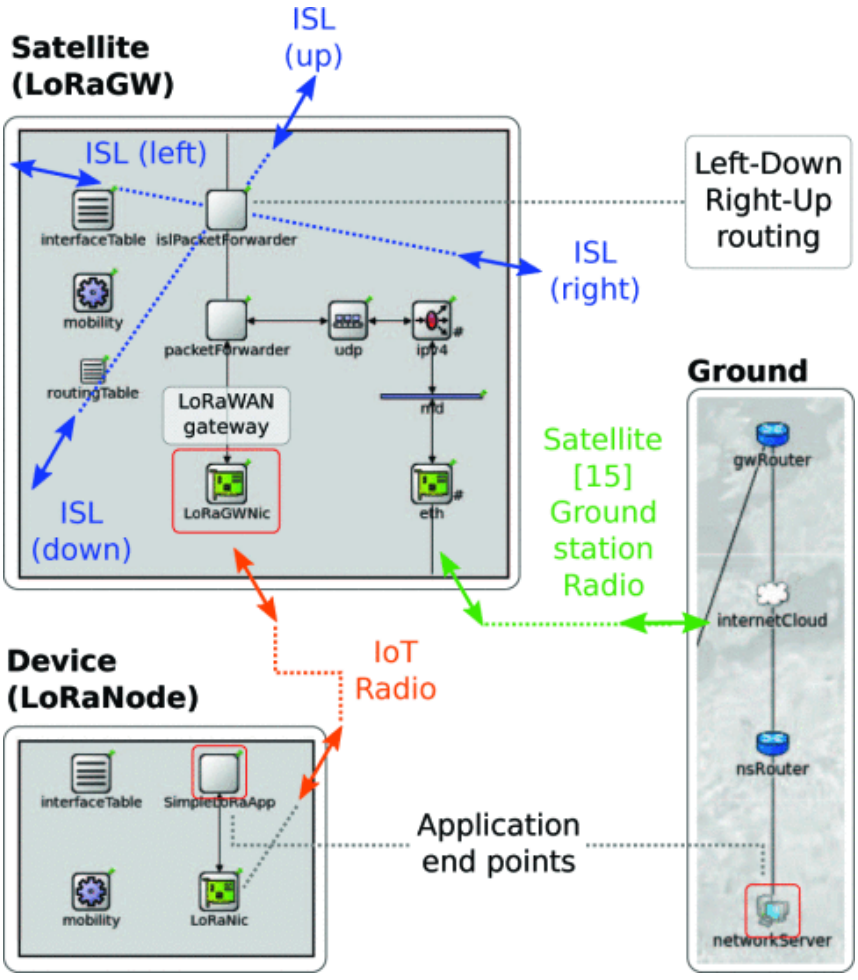


Figure 2.13: FloRaSat Architecture [Fraire et al., 2022]

High-level Evaluation

The Table 2.5 provides a detailed comparative analysis of various simulation tools developed for satellite-enabled C2T, focusing on several essential parameters.

These include support for the application layer, which manages high-level services and user interactions, and the cloud layer, which ensures scalable and efficient large-scale data processing. Furthermore, the fog and edge layer is assessed for its ability to facilitate low-latency processing near data sources, enhancing real-time responsiveness.

A key consideration is mobility support, critical in scenarios involving the dynamic movement of satellites and terrestrial users. The capacity to handle a large number of satellites is another vital aspect, as scalability often determines the effectiveness of simulations in diverse real-world applications. In addition, the ability to simulate satellite orbits is indispensable for accurately representing the movement and positioning of satellites over time.

In particular, these tools also differ in their capacity to handle not only satellite internet scenarios but also terrestrial internet simulations. This dual capability is crucial for evaluating integrated communication systems that involve terrestrial and satellite networks, offering a holistic approach to simulation.

Table 2.5: Comparison of Simulation Tools for satellite-enabled C2T

	SatEdgeSim	**SNS3**	**OpenSAND**	**OS3**	**FLoRaSat**	**YAFS+Skyfield**
Application Layer	✓	✗	✗	✗	✗	✓
Cloud Layer	✓	✗	✗	✗	✗	✓
Fog/Edge Layer	✓	✗	✗	✗	✓	✓
Mobility Support	✓	✗	✗	✓	✓	✓
Satellites Supported	> 1000	< 10	Not Mentioned	< 10	< 100	> 1000
Satellite Orbit Simulation	✓	✗	✗	✓	✓	✓
Terrestrial Support	✗	✗	✓	✗	✓	✓

Among the tools, SatEdgeSim and YAFS+Skyfield emerge as the most comprehensive solutions. These tools stand out because they provide robust support for all layers, including the application, cloud, and fog/edge layers. They also incorporate mobility handling, allowing dynamic simulations of satellite movement and user mobility. In addition, they stand out for their ability to simulate numerous satellites while integrating accurate orbit modelling. However, YAFS+Skyfield stands out more for its support for terrestrial communications.

However, tools such as SNS3 and OpenSAND demonstrate minimal capabilities, without significant support for essential layers and mobility, limiting their applicability in sophisticated scenarios. Meanwhile, OS^3 offers some mobility and orbit simulation capabilities, but lacks layer support, which restricts its versatility. Finally, FLoRaSat offers outstanding capabilities in edge computing and mobility simulation but is limited by its limited ability to handle a large number of satellites.

This comparison underlines the versatility of YAFS+Skyfield for complex, large-scale peripheral computing scenarios with the help of satellites, while pointing out the specific limitations of other tools for researchers and programmers looking for specialised features, making it the ideal choice for development.

Finally, an analysis of scientific articles that address the subtleties of integrating satellites into C2T, with a special emphasis on their use in fog computing environments is presented. A significant number of these studies adopted simulators

as a core instrument for the validation of their hypotheses, thereby underscoring the pivotal role of such instruments in propelling research in this domain.

2.4.4 Studies on Integrating Satellites within the C2T Paradigm

In the paper [Strinati et al., 2020] the authors investigate the integration of terrestrial, aerial and satellite platforms in future mobile networks Sixth Generation (6G) to provide cloud computing services at the edge of the HSTN. The proposed approach aims to embed Mobile Edge Computing (MEC) functionalities in aerial platforms and LEO satellites, thus extending support to devices and network elements in space. They present an architecture that combines communication, computing and storage in an integrated way, using algorithms based on artificial intelligence to dynamically orchestrate resources between terrestrial and non-terrestrial nodes.

The evolution of networks from Fifth Generation (5G) to 6G, as described by [Strinati et al., 2020], will go beyond the current two-dimensional coverage and will enable truly three-dimensional on-demand services. However, challenges such as Three-Dimensiona (3D) interference management, multi-link load balancing, and AI-based common resource orchestration remain promising areas for future research. The preliminary results of the project show that interference management in 3D multiradio systems can make use of innovative techniques, such as the use of additional 3D nodes (drones and satellites) to deal with congestion or extend coverage opportunistically. This work also shows that the integration of Non-Terrestrial Networks (NTN) with terrestrial networks can pave the way for new standards and consolidate the direction for the development of 6G.

Other studies explore Satellite Edge Computing (SatEC) architectures to support IoT in 5G and 6G communication scenarios. These studies demonstrate both the economic and technical feasibility of these solutions. For example, a recent paper [Kim et al., 2021] proposes a SatEC architecture for IoT by formulating a multiobjective optimisation problem that considers latency, computational consumption and transmission power attenuation. This problem was solved using heuristic algorithms, such as multi-objective tabu search, for satellite constellation topologies in LEO and Very Low Earth Orbit (VLEO) orbits. The simulation results show that reducing the altitude of the satellite improves latency and transmission power attenuation, although computational complexity increases. Evaluation of SatEC network slice solutions with different offloading priorities for massive IoT, mission critical IoT, and 6G backhaul showed that LEO constellations offer a better cost-benefit ratio, although VLEO constellations have lower latency. The study also proposed a task ordering algorithm to personalise latency services, highlighting the need for optimised offloading strategies.

In addition, solutions for emergency scenarios such as earthquakes and landslides have been explored.[Zhang et al., 2023] proposed an approach that combines LEO satellite networks with Unmanned Aerial Vehicle (UAV) to provide edge computing services to terrestrial devices. The model considers energy and latency constraints and uses algorithms based on Deep Deterministic Policy Gra-

dient (DDPG) and Long Short-Term Memory (LSTM) for resource allocation and task offloading. The simulation results show that this solution outperforms approaches based on random offloading, providing greater energy efficiency and reduced latency. The use of UAV in conjunction with satellites proved effective in resource allocation in high task volume scenarios or under conditions of limited computing resources.

In the context of satellite-terrestrial collaborative networks, [Jia et al., 2022] proposed a strategy that integrates MEC technology directly into LEO satellites, allowing these satellites to provide computing services to mobile devices on Earth. This approach, which uses a core edge multi-agent cooperation model with Software Defined Networking (SDN) and Network Functions Virtualization (NFV), has proven effective in optimising resource control and improving connectivity in dynamic and heterogeneous networks. The authors also addressed the challenges associated with satellite mobility and the randomness of resource requests and proposed solutions for service migration and efficient resource allocation between satellites.

Furthermore, [Cheng et al., 2022] proposed a dynamic task offloading strategy for SatEC networks to minimise global delay in ground user tasks. The computing capacity of LEO satellites, which is unstable due to power variations, was taken into account when developing an optimised offloading strategy. The approach used Lyapunov optimisation theory, which transforms a stochastic problem into multiple deterministic problems, resulting in efficient optimisation with a significant reduction in global delay and control of energy consumption.

As these studies show, the use of satellites in fog computing systems has increased significantly in recent years. However, despite these advances, there are still no widely accepted or available simulators that can replicate complex scenarios to integrate satellites with fog computing systems, such as those presented in these studies. This suggests that many of the simulators used in the research have been developed in-house by the authors, without reference or standardisation in the field. This highlights an important gap in the field that needs to be addressed to allow the validation and replication of such scenarios in future studies.

Chapter 3

Design and Requirements for YAFS Enhancement

This chapter is organised into three main sections. Section 3.1 presents and describes the problem that motivated the development of this document. Next, Section 3.2 details the functional and non-functional requirements that the system must fulfil in order to address the identified problem. Finally, Section 3.3 identifies the potential risks associated with implementing the system and presents a mitigation plan with strategies to avoid or minimise the occurrence of these risks.

3.1 Problem Statement

As previously outlined in Section 2.1, there are currently several challenges related to maintaining continuous communication throughout C2T systems. With the exponential growth in the number of IoT devices, including those with high mobility, ensuring uninterrupted and reliable connectivity has become increasingly critical.

One of the most promising approaches to achieve this is through the integration of satellites into terrestrial networks, as discussed in Section 2.2. However, testing new satellite-enabled topologies in real-world environments is complex and costly, often prohibitive for research teams. In this context, network simulators emerge as a viable and cost-effective solution, enabling the evaluation of novel architectures and algorithms before deployment. This approach not only reduces financial risk but also allows researchers to experiment with scenarios that would otherwise be impractical to reproduce physically.

A review of existing simulators, presented in the previous chapter, revealed that no single tool currently offers a complete solution for C2T scenarios involving both satellite communications and user mobility. While various platforms provide partial capabilities, none combine these features in a unified and flexible framework. This lack of integrated support forces many researchers to develop custom, ad-hoc simulators, which can be time-consuming, less maintainable, and harder to extend for future work.

In summary, the central problem addressed in this dissertation is the absence of a network simulator that natively supports both satellite communications and realistic user mobility within Cloud-to-Things scenarios. This gap limits researchers' ability to evaluate integrated architectures efficiently and under realistic conditions.

Among the available tools, YAFS stood out as the most promising foundation for extension. Despite its significant limitations in the context of this work—namely, the absence of native support for simulating satellite networks and realistic user mobility—YAFS offers a highly modular architecture, a clear event-driven design, and well-defined interfaces for adding new models. These characteristics make it particularly suitable for upgrades that expand its capabilities without compromising its existing features. Its flexibility and extensibility mean that once these gaps are addressed, YAFS can serve as a powerful and versatile tool for researchers studying satellite-enhanced and mobility-aware C2T systems.

The following section outlines the fundamental requirements for the enhancement of YAFS to bridge these functionality gaps, enabling the simulation of realistic C2T scenarios that integrate both satellite communication and user mobility in a cohesive, research-ready framework.

3.2 Requirements

To address the challenges mentioned, it is essential to first identify the functional requirements of the simulator from the perspective of a researcher, student, or general user, followed by the non-functional requirements.

3.2.1 Functional Requirements

The functional requirements will be presented in the form of user stories, organised in the Table 3.1 according to their priority. In the table, the priority of each user story will be classified into three categories:

- **Must:** These are critical requirements that are essential to ensure that the final product has value and is functional.
- **Should:** Important but not essential requirements that can be postponed if necessary.
- **Could:** Desirable requirements that add value to the product, but whose implementation is not essential.

3.2.2 Non-Functional Requirements

The non-functional requirements that the system must fulfil are as follows:

Table 3.1: Functional Requirements

As a researcher, I want to...	**Priority**
Simulate the orbits of satellites that are already operational in order to analyse their characteristics.	Must
Simulate orbits that do not yet exist to determine whether new configurations offer advantages over the current ones.	Must
Simultaneously simulate satellite and terrestrial networks to evaluate the benefits of hybrid network configurations.	Must
Ensure the simulator implements coverage zones based on the orbital position of satellites and the location of devices on Earth, to assess connectivity and coverage.	Must
Have the simulator implement dynamic handovers, allowing mobile nodes to switch access points as they move between different coverage zones.	Must
Simulate intra-orbital satellite connectivity to study communication, data exchange, and dynamic handovers within the same orbital plane.	Must
Simulate inter-orbital satellite connectivity to study communication and data exchange between satellites in different orbital planes.	Should
Simulate a large number of satellites to replicate real-world scenarios of satellite constellation networks.	Should
Provide an API for configuring satellite-related parameters to customise simulated scenarios according to research requirements.	Should
Support different mobility models to analyse how real mobile devices interact with the satellite and terrestrial networks.	Could
Include a graphical interface to visualise how users connect to various network nodes.	Could
Calculate advanced satellite-related metrics, such as latency, link capacity, and energy efficiency, to enhance performance analysis.	Could

- **Performance:** The simulator must be able to process and simulate a large number of nodes, both terrestrial and satellite, without a significant degradation in performance. Scalability must be an essential feature to support complex and realistic scenarios.
- **Reliability:** Calculations related to orbits, metrics and coverage zones must be carried out on the basis of rigorous and accurate mathematical models, ensuring that the simulator's results reliably reflect the real characteristics of the simulated systems.

3.3 Risk Analysis

Table 3.2 shows the identified risks that could compromise the success of the project, with likelihood and impact assessed according to the matrix 3.1. The values are categorised into three levels:

- Likelihood (1-3):
  1. Unlikely, but possible
  2. Moderate probability; the risk could materialise under certain conditions
  3. Probable or highly probable
- Impact (1-3):
  1. Low impact; minimal disruption or easily mitigated
  2. Moderate impact; would require additional resources to resolve
  3. High impact; could threaten the success or usability of the project

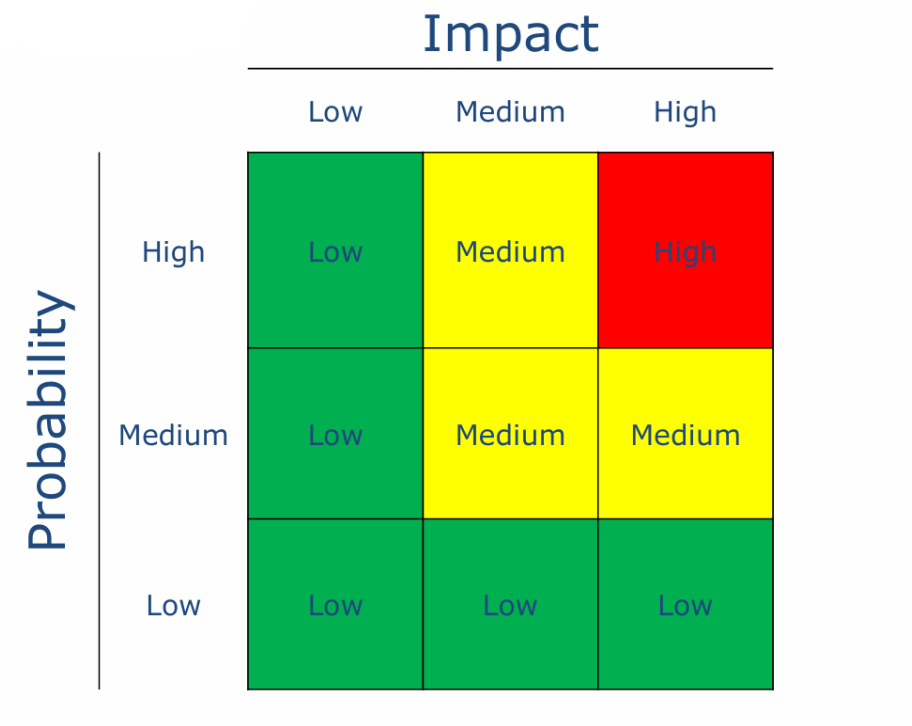


Figure 3.1: Risk Analysis Matrix [Morphy, 2023]

In addition, consequences are identified should each risk materialise, allowing for a more detailed and precise analysis of its potential effects on the project.

It is also crucial to identify strategies to mitigate the risks previously identified, in order to reduce the likelihood of them occurring or minimise the impact if they do materialise.

Table 3.2: Risk Assessment Table

Risk ID	Risk	Likelihood	Impact	Consequence
R1	Dependency on External Libraries and Tools	2	3	The system may stop working properly in the future
R2	Scalability Challenges	1	2	Unable to handle large-scale or realistic scenarios
R3	Simulation Accuracy	2	2	Reduce confidence in the simulator
R4	System Integration Complexity	2	3	Delays and increased development effort
R5	Unrealistic Deadlines	3	2	Inferior quality and incomplete features that could jeopardise the credibility of the project

The first risk, related to dependence on external libraries or third-party tools, raises the possibility that these may become obsolete, no longer be supported, or present incompatibilities with future technologies. Mitigating this risk involves minimising dependence on non-essential external tools and opting for libraries that are widely used, well-maintained and continuously supported by the community or suppliers.

In the case of R2, which refers to performance degradation due to processing a large number of nodes (terrestrial and satellite), mitigation involves carrying out comparative performance assessments during development, as well as implementing efficient algorithms capable of handling large-scale simulations.

R3 addresses the possibility that the mathematical models used to simulate orbits, metrics and coverage zones are not sufficiently accurate. To mitigate this risk, it is essential to use validated mathematical models that have been proven to be able to reproduce the characteristics of real systems reliably.

With regard to risk R4, related to the complexity of integration, mitigation requires integration tests to be carried out from the early stages of the project, allowing potential problems to be identified and resolved early on.

Finally, R5 refers to tight deadlines, which can jeopardise the quality of the project by resulting in inadequate testing, incomplete functionality or quality failures. To address this risk, it is necessary to conduct detailed project planning, utilising tools such as Gantt charts to establish realistic milestones, and regularly evaluate progress to adjust deadlines as needed.

Identifying risks and planning the respective mitigation strategies are fundamental stages of the project, contributing significantly to increasing the likelihood of success and guaranteeing its final quality.

Chapter 4

YAFS Module Development: Architecture and Implementation

This chapter presents the design and implementation of the expanded architecture of the YAFS simulator. It begins with an overview of the enhanced system architecture (Section 4.1), where both the high-level design and the detailed internal components are described. The discussion then moves to the implementation of the developed modules (Section 4.2), focusing first on the user mobility module and subsequently on the satellite mobility module. Finally, the chapter outlines the development methodology adopted throughout this work (Section 4.3), which guided the integration and validation of the new features.

All the implementation is available in the repository YAFS_NTN, which is a fork of the original YAFS project.

4.1 System Architecture of Enhanced YAFS

The updated architecture of the YAFS simulator is based on its original modular structure, introducing new components designed to support the simulation of hybrid satellite-terrestrial environments with dynamic user mobility. These additions directly address the new functional requirements identified in this work, enabling a more realistic and flexible modelling of complex communication scenarios.

This section provides an overview of the updated system architecture, highlighting the newly introduced modules and their integration into the existing YAFS core. In Section 4.1.1, a high-level block diagram is presented to visually represent these architectural changes, followed by a detailed description of the main functionalities and objectives of each new module.

4.1.1 High-Level Architectural Design

To provide a clear understanding of how the new modules integrate into the original YAFS architecture, presented in Section 2.4.1, a block diagram was developed 4.1. In this diagram, the modules shown in grey correspond to the existing components in YAFS, while the coloured modules represent the extensions developed within the scope of this work. The connections between the blocks illustrate the functional relationships established between the different modules.

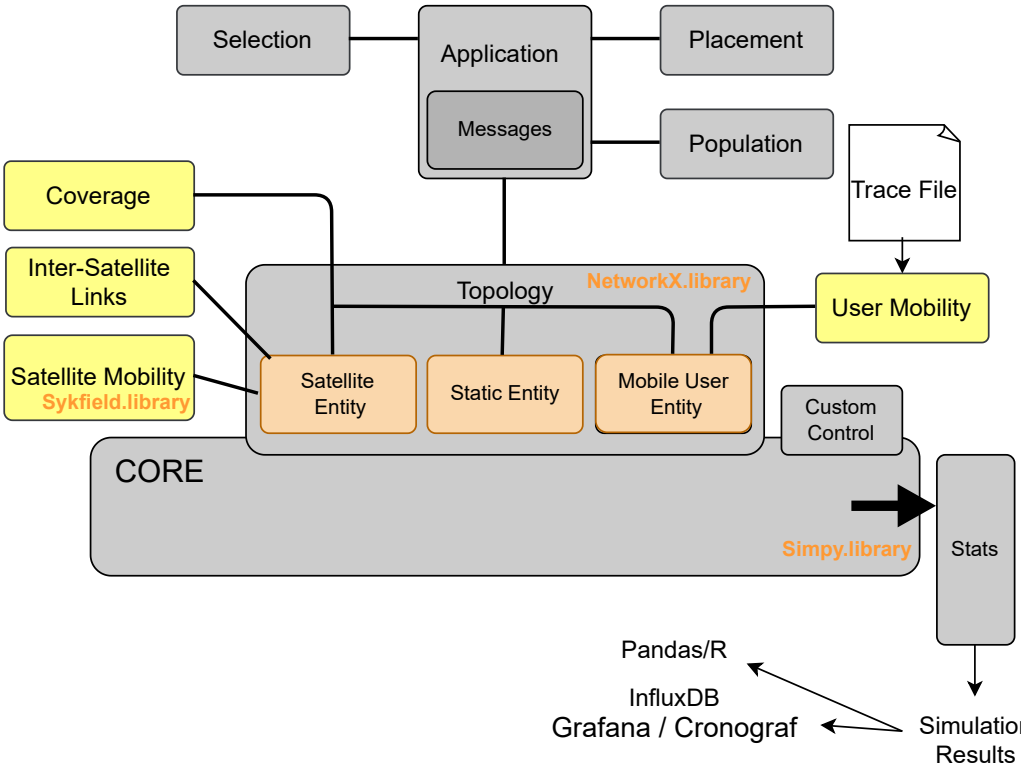


Figure 4.1: High Level Architecture

In this expanded version of the architecture, the topology remains the central element of the core of the system. However, three new types of nodes can now be defined, in addition to those already existing in YAFS:

1. **Satellite Entity:** These represent the satellites in the topology, with their orbits dynamically modelled
2. **Static Entity:** These correspond to the existing nodes in YAFS, which can now include geographical coordinates to simulate, for example, fixed communication towers.
3. **Mobile user Entity:** These represent users on the move with dynamic trajectories in geographic space.

A simulation will continue to contain only one topology, which can be dynamically modified throughout the process by adding or removing nodes and connections. This topology will be able to support several applications, each with its own placement, selection and population policies. The Custom Control module is also present, allowing the inclusion of personalised controls that interact dynamically with the simulation's applications and variables.

The new modules introduced include:

- **User Mobility:** This module will be responsible for managing the movement of mobile users, ensuring that their behaviour is realistically represented in the simulation. To achieve this, trace files containing the detailed coordinates of a user's journey will be used. These routes can be generated using various traffic simulation tools, such as Simulation of Urban MObility (SUMO) [Lopez et al., 2018], which are widely used for urban mobility and transport simulations.

  The process consists of generating, using tools such as SUMO, simulated routes that represent user movements in different contexts. The result of these simulations is often exported in standard formats, such as GPS Exchange format (GPX), which describe the geographical and temporal coordinates of the routes. This output will be integrated into this module, allowing user movements to be simulated realistically and accurately within the simulation environment.

  In this way, this module will therefore be responsible for translating the mobility data generated by external tools, representing the behaviour of mobile users in a detailed and dynamic way. It is important to emphasise that there will only be one module of this type in each simulation, since different types of trajectories can be combined and represented in the same trace file.

- **Satellite Mobility:** This component is responsible for calculating satellite orbits, using integration with the Skyfield library to obtain precise and dynamic trajectories. If it is necessary to work with different types of satellites or different trajectories, several instances of this module can be added to the simulation, with each instance being in charge of managing the specific set of satellites under its responsibility.

  This modular approach guarantees flexibility and scalability in the simulation, allowing it to adapt to more complex scenarios with multiple satellites and different orbital configurations.

- **Coverage:** One of the core modules of this new architecture, which checks the connectivity of end users with satellites and fixed nodes throughout the simulation. If the user is within the coverage area of a node, this module will establish the corresponding connection, creating a new link in the topology. In addition, the module will make it possible to implement different algorithms for choosing the node in scenarios where the user is covered by more than one.

- **Inter-Satellite Links:** This is a key module, especially for simulating LEO satellite constellations, and is responsible for creating and managing the

links between satellites. If there are different types of satellites and orbits, several instances of this module can be created, each instance being in charge of managing the satellites and their respective orbits assigned to it.

This modular approach allows for greater flexibility in the simulation, making it possible to represent different satellite configurations and their interactions within a constellation.

This architecture significantly expands the capabilities of YAFS, offering a flexible and extensible solution for complex simulations involving hybrid networks between satellites and terrestrial infrastructure.

Next, we will take a closer look at each of these new components, detailing their connections to the system and how they interact with the other elements of the architecture.

4.1.2 Detailed Architecture

Starting with the User Mobility module, it receives as input a file with the geographical coordinates corresponding to the users' routes throughout the simulation. After reading the file, a parsing process is performed to convert the information into a format compatible with the system, allowing it to be read sequentially throughout the simulation.

The positions of mobile users are updated by the update position function, which is responsible for dynamically changing the location of mobile nodes in the topology whenever it is invoked. In order for this function to be called repeatedly during the simulation, a mechanism was developed in the simulator core that creates a process within the discrete simulation environment, as is already done for native YAFS modules.

To ensure that these updates occur at the appropriate time, an additional function called next activation was implemented, which calculates and returns to the core the next time instant at which the update function should be executed. In this way, the process created in the core waits until the next activation instant defined by this function and then invokes the update position again.

This entire flow is illustrated in Figure 4.2, where you can see the sequence of interactions between the components involved in the management of user mobility.

Moving on to the Satellite Mobility module and representing its details in Figure 4.3, its operation is conceptually similar to that described above for the user mobility module, but with some significant differences, particularly in terms of the type of input data.

In this case, the module receives a file in JSON format that includes:

- The initial time of the simulation (essential for calculating orbital positions based on TLE);

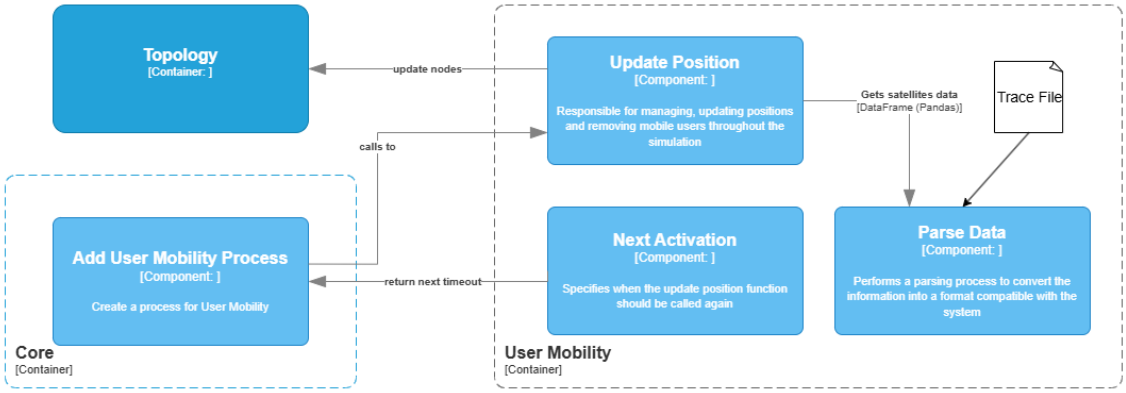


Figure 4.2: Details of the user mobility module

- The total duration of the simulation, which should coincide with the actual duration of the simulated scenario (additional aspects on this point will be addressed in the section dedicated to implementation details);
- The TLE describing the orbits of the satellites in a given constellation, or, alternatively, links to external sources, such as CelesTrak, from which the module can obtain the orbital data.

After reading and processing the input file, the module invokes the Skyfield library API, which is responsible for calculating the position of each satellite throughout the duration of the simulation. These calculations, which involve computationally intensive operations, are performed before the actual start of the simulation to avoid compromising the performance of the discrete execution environment. Thus, the spatial coordinates of the satellites are determined in advance for each relevant time instant, allowing the module to only access these already processed data during the simulation, efficiently updating the position of each satellite in the topology without introducing delays or overhead in execution time.

As in the User Mobility module, the simulator core will register this component as a discrete process, using a dedicated function created for this purpose. This function will repeatedly invoke the update satellite position operation, which is responsible for updating the location of active satellites.

The frequency of these updates is controlled by the next activation function, which determines the next moment when the system should reactivate the process. Thus, the cyclical behaviour of this module is managed in the same way as the other simulator processes, ensuring consistency and synchronisation between the various components of the system.

Consequently, the Coverage module, as mentioned above, is responsible for ensuring the connection of mobile users to the network, either through a satellite or a fixed node. This connection is verified and updated periodically, based on a temporal distribution algorithm, which determines the next moment in time when the update function should be invoked.

As in the previous modules, the simulator core creates a dedicated process for

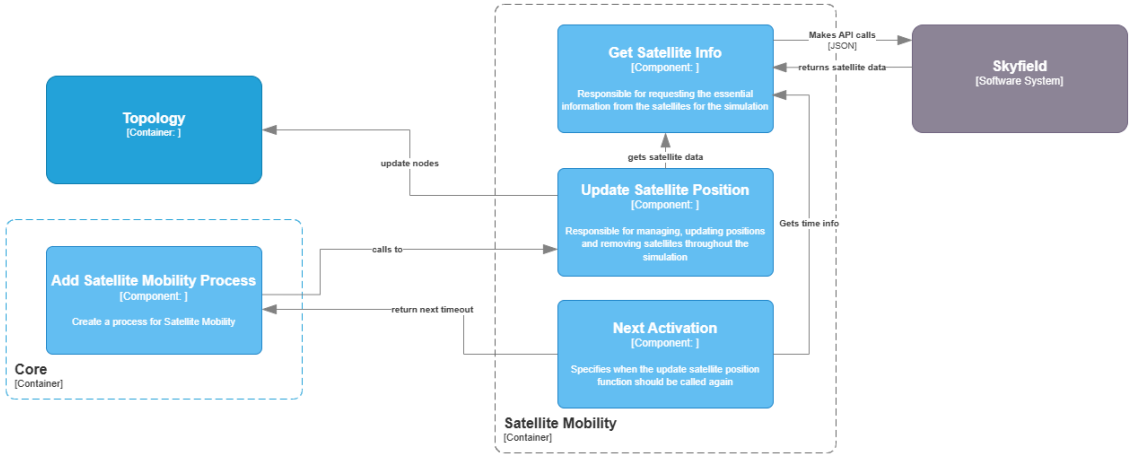


Figure 4.3: Details of the satellite mobility module

this module, which cyclically calls the update function responsible for checking and updating the links between users and available access nodes.

The decision on which network node the user should connect to is delegated to the designated connectivity policy function. This function can be customised, allowing different selection criteria to be implemented according to the scenario to be simulated. For example, one can opt for a policy that connects the user to the node closest in terms of geographical distance, or to another with better connection quality, among other possibilities.

This modular approach represented in Figure 4.4 preserves the YAFS philosophy, promoting flexibility and adaptability, since different connectivity policies can be easily integrated and tested according to study requirements.

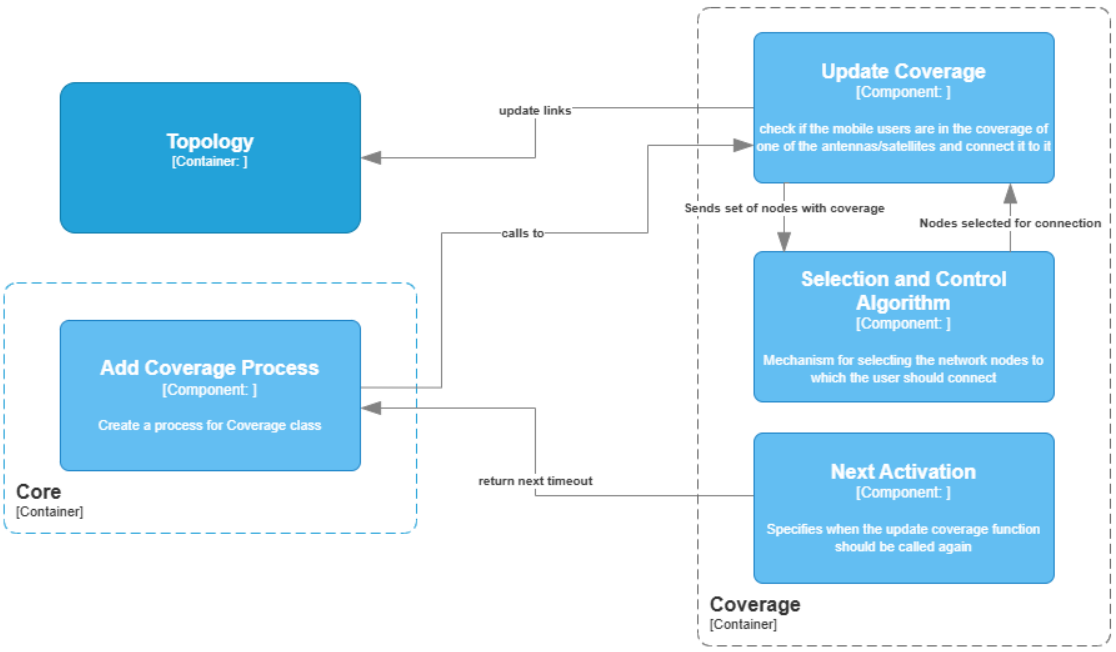


Figure 4.4: Details of the coverage module

Finally, the ISL Manager module, represented in Figure 4.5 is responsible for periodically updating the links between satellites. These links are defined based on the Link Strategy function, which can be customised with different link algorithms, allowing you to configure how the satellites in a constellation communicate with each other according to specific criteria.

To support this functionality, the module uses two auxiliary functions. The ISL distance function calculates the distance between satellite pairs, allowing the propagation time of the signals to be estimated. The Line of Sight (LOS) limit function assesses whether, from a physical point of view, the link between two satellites is feasible, taking into account visibility and other obstruction factors.

As with the other modules, there is a dedicated function at the core of YAFS that registers this module as a simulator process. This process is executed periodically, according to the instants defined by the get next activation function, thus ensuring that the intersatellite connections are updated efficiently and synchronised throughout the simulation.

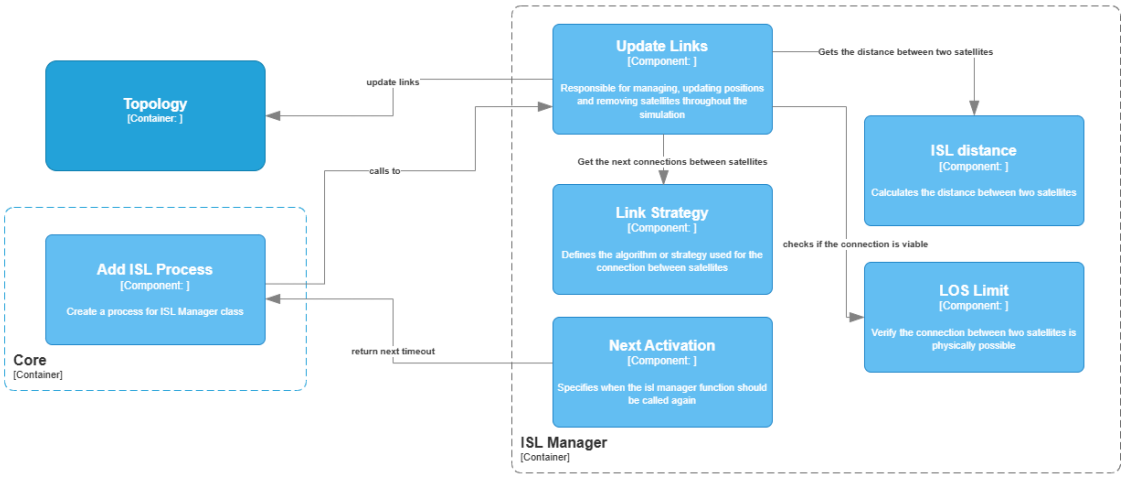


Figure 4.5: Details of the ISL Manager module

4.2 Implementation Details of Developed Modules

Following the architectural description of each module, this subsection details the implementation of their respective functionalities and the rationale behind key design decisions, including the selection of algorithms, data structures, and latency calculations.

4.2.1 User Mobility Module Implementation

The first logical step in the development of this module was to define the supported formats for mobility trace files. Two formats were selected: the (Floating Car Data (FCD) output format from SUMO and the GPX format. The choice of

the SUMO format is justified by its ease of use when generating mobility scenarios, its ability to be exported as a CSV file, and the straightforward integration of such files for processing within YAFS. The GPX format was included due to its widespread adoption in representing trajectories of moving objects on the Earth's surface, such as vehicles or users carrying mobile devices.

Based on these input files, the module performs data extraction and transformation into a Pandas DataFrame, structured as shown in Table 4.1. This structure includes the simulation timestep, the user identifier, and the corresponding latitude and longitude at that specific moment.

To model the association between mobile users and terrestrial nodes, a coverage-based system was implemented. Each node is defined by a circular coverage area, and a connection is established based on the geographical distance between a user and the node. This distance is computed using the Haversine formula, which determines the great-circle distance between two points on a sphere using their latitudes and longitudes. The Haversine distance d is calculated as follows [U.S. Census Bureau, 2004]:

$$d = 2R \cdot \arcsin\left(\sqrt{a}\right) \tag{4.1}$$

$$\text{where} \quad a = \sin^2\left(\frac{\Delta\phi}{2}\right) + \cos(\phi_1) \cdot \cos(\phi_2) \cdot \sin^2\left(\frac{\Delta\lambda}{2}\right) \tag{4.2}$$

Here, R represents the Earth's radius, $\Delta\phi$ and $\Delta\lambda$ denote the differences in latitude and longitude (in radians), and ϕ_1, ϕ_2 are the respective latitudes.

A user is considered to be connected to a node if they are located within its coverage radius. By default, the user connects to the nearest available node within range. However, the system architecture was designed to be modular and extensible, thereby allowing the implementation of alternative connectivity models beyond circular coverage, allowing more sophisticated or domain-specific mobility interactions to be explored within the simulation environment.

Table 4.1: Structure of the processed mobility DataFrame

timestep	user_id	pos_x	pos_y
0	1	100.0	200.0
1	1	105.0	205.0
2	1	110.0	210.0

4.2.2 Satellite Mobility Module Implementation

This module is responsible for retrieving orbital data using the Skyfield library. One of its core features lies in the treatment of time. While Skyfield performs computations based on real time, the YAFS simulator operates using a discrete simulation time. Therefore, to ensure temporal synchronisation between both systems, it is essential to align these two distinct time scales.

To achieve this, the module requires as input the temporal unit adopted by the simulation. This allows real-time values to be converted into the corresponding simulation time steps. For instance, if the simulation runs for 10 time units and each unit corresponds to one second, then Skyfield will compute orbital data for an equivalent real-time span of 10 seconds, ensuring temporal consistency across both systems.

Given that this module may potentially manage thousands of satellites, computational efficiency becomes a critical concern. In many cases, the simulation scenario is geographically localised, making it unnecessary to track all satellites at all times. To address this, two distinct strategies were implemented for retrieving satellite information, as described in the Mobility Module. In the first, all satellites are loaded and tracked throughout the entire simulation. In the second approach, a dynamic method is employed, whereby only satellites that are both visible and proximate to the simulation area are retrieved at each time step. This significantly reduces computational overhead by excluding satellites irrelevant to the current context.

The output of both approaches is a Pandas DataFrame with the following structure:

Table 4.2: Structure of the satellite mobility DataFrame

timestep	satellite_id	pos_x	pos_y	pos_z	azimuth
0	1	10000.0	20000.0	30000.0	45.0
1	1	10100.0	20100.0	30100.0	46.0
2	1	10200.0	20200.0	30200.0	47.0

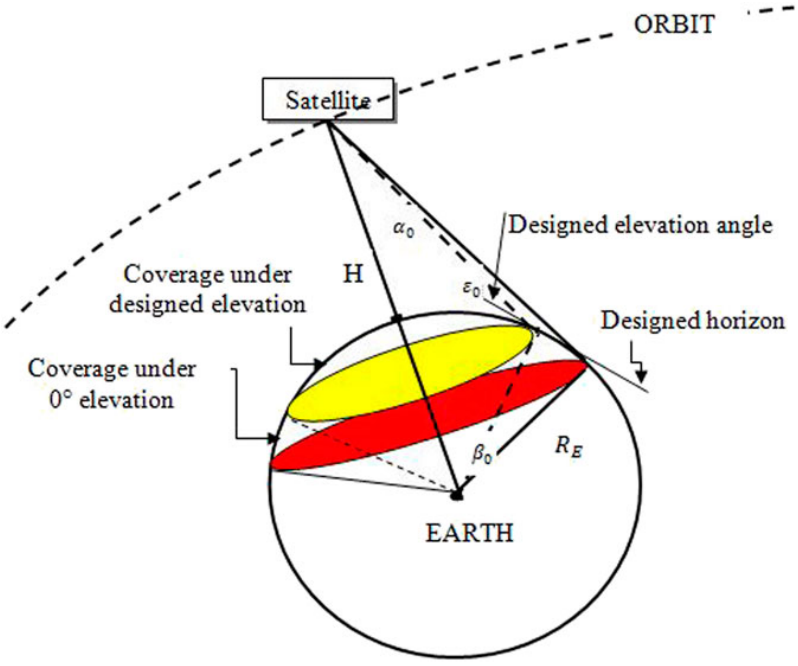


Figure 4.6: Coverage Area Model [Cakaj, 2021]

Another critical component is the satellite coverage model. For this, an algo-

rithm adapted from [Cakaj, 2021] was employed, allowing the calculation of each satellite's coverage area based on geometric considerations. As illustrated in Figure 4.6, the central angle of visibility α_0 is determined using the following expression:

$$\sin(\alpha_0) = \left(\frac{R_e}{H}\right) \cdot \cos(\varepsilon_0) \tag{4.3}$$

where:

- R_e is the Earth's radius,
- H is the altitude of the satellite,
- ε_0 is the minimum elevation angle required for communication.

Based on α_0, the angle β_0 is then given by:

$$\beta_0 = 90^\circ - \varepsilon_0 - \alpha_0 \tag{4.4}$$

The ground coverage radius can subsequently be estimated by:

$$\text{Coverage} = R_e \cdot \beta_0 \tag{4.5}$$

To assess whether a user falls within a satellite's coverage area, the sub-satellite point — the point on the Earth's surface directly beneath the satellite — is computed. The distance between this subpoint and the user's position is then calculated. If this distance is less than the coverage radius, the user is considered to be within communication range.

Beyond coverage, another key factor in establishing satellite-user communication is the propagation delay, i.e., the time required for data to travel from the user to the satellite. This is computed using Skyfield's `altaz` method, which returns the spatial distance between two positions. The propagation delay is then obtained by dividing this distance by the speed of light in vacuum, yielding a result in seconds. This value can be converted to any other temporal unit as defined in the simulation configuration.

An additional crucial element is the modelling of ISL. In this work, each satellite is designed to establish four connections: two with neighbouring satellites in the same orbital plane and two with satellites in adjacent planes, as shown in Figure 4.7.

The physical feasibility of an ISL depends on the existence of a clear LOS — i.e., no obstruction by the Earth. To validate this, two main calculations are required.

The first is the Euclidean distance between two satellites p and q, given by:

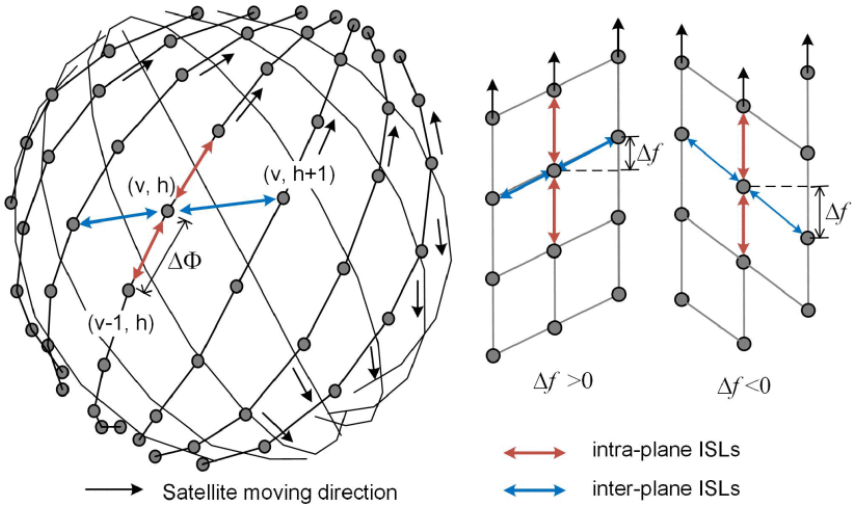


Figure 4.7: ISL Model [Chen et al., 2021]

$$\|uv\| = \Big[(h_p + R_E)^2 + (h_q + R_E)^2 - 2(h_p + R_E)(h_q + R_E) \cdot \big(\cos\theta_p \cos\theta_q + \cos(\phi_p - \phi_q) \sin\theta_p \sin\theta_q\big)\Big]^{1/2} \quad (4.6)$$

The second defines a geometric limit, representing the maximum distance at which a direct link is possible without obstruction by Earth's curvature:

$$l^*(p,q) = h_p(h_p + 2R_E) + h_q(h_q + 2R_E) \quad (4.7)$$

where:

- h_p, h_q are the altitudes of satellites p and q, respectively,
- R_E is the Earth's radius,
- θ, ϕ represent latitude and longitude.

The LoS condition is then evaluated by comparing both values:

$$\|uv\| > \sqrt{l^*(p,q)} \quad (4.8)$$

If this condition is satisfied, the Earth is obstructing the line of sight between the two satellites, making communication unfeasible. Conversely, if:

$$\|uv\| \leq \sqrt{l^*(p,q)} \quad (4.9)$$

the connection is deemed physically viable and may be established.

4.3 Development Methodology

An agile methodology will be used to implement this mobility and satellite module. The choice of this approach is due to the nature of the project, which does not have fixed requirements from the outset, allowing new functionalities to be added throughout development.

As can be seen in Figure 4.8, the Agile methodology is divided into six phases, which are completed in each sprint. In the specific case of this project, each sprint will last one week. At the end of each week, a meeting will be held to present the work carried out, reflect on the results achieved, plan the tasks for the following week and adjust the strategy where necessary.

The advantages of adopting this approach include the continuous updating of the project and regular feedback, which enables improvements to be made throughout the process and not just at the end. In addition, constant communication is reinforced by the meetings held at the end of each sprint. Finally, the methodology offers flexibility, allowing new functionalities to be introduced at later stages of the project. For these reasons, this methodology is ideally suited to the nature of this project.

Figure 4.8: Agile Methodology [Laoyan, 2024]

Chapter 5

Evaluation and Results

5.1 Experimental Design and Setup

In order to evaluate the integration of the newly developed modules into the pre-existing YAFS infrastructure, and to validate both their correct functioning and their utility in addressing the emerging challenges of ubiquitous connectivity, it was first necessary to define the experimental environment in which the tests were conducted. Accordingly, Section 5.1.1 describes the experimental design, including the simulation environment, the definition of test scenarios, and the procedures for data collection. Section 5.1.3 then presents the validation of the developed modules through a set of comparative experiments against established simulators. Finally, Section 5.3 analyses performance and illustrates the capabilities of the enhanced YAFS through two representative use-case scenarios involving hybrid networks and satellite-enabled fog computing.

5.1.1 Simulation Environment

The experiments were carried out on a personal computer with the following hardware and software specifications: an *Intel(R) Core(TM) i5-9300HF* CPU running at 2.40 GHz with 8 logical cores, 16 GB of RAM, two SSD storage devices (a 250 GB NVMe drive and a 1 TB SATA drive), and an *NVIDIA GeForce GTX 1050* GPU. The operating system was *Windows 11*. Given that the simulator was implemented in Python, the same language was used in version `3.11.4`, alongside the YAFS framework in its latest available release (`0.3`). The following Python libraries were employed to support the simulation workflow:

- `numpy` (`1.26.4`) — for numerical computations and matrix operations;
- `networkx` (`3.2.1`) — for representing network topologies and performing graph-based operations;
- `matplotlib` (`3.8.4`) — for generating plots and visualising results;
- `pandas` (`2.2.0`) — for data storage, manipulation, and analysis;

- `skyfield` (`1.49`) — for computing and generating satellite trajectories;
- `Sphinx` (`7.2.6`) — for enhancing the simulator's documentation;
- `virtualenv` (`20.26.2`) — for executing all experiments within an isolated virtual environment, ensuring reproducibility and preventing conflicts with system-level dependencies.

5.1.2 Test Scenarios Definition

To evaluate the proposed modules, four distinct tests were designed. The first two tests focus on assessing each module in isolation. For this purpose, experiments previously conducted in other simulators were replicated in order to determine whether similar behavioural trends could be observed when using the newly integrated YAFS modules.

The third test consists of an integration assessment, in which both modules operate simultaneously. Since there are no prior studies that combine user mobility scenarios with satellite-based communication models, the developed integration test follows a relatively straightforward design. This makes it possible to predict the expected outcomes with reasonable accuracy, thereby facilitating the verification of the correct interaction and functional compatibility between the modules.

Finally, the fourth test is a performance benchmark aimed at evaluating the simulator's behaviour under a demanding use case, representative of scenarios in which this updated version of YAFS could serve as a valuable tool for future research investigations.

Audio Translation Service Test - iFogSim2

The first test was designed with the primary objective of validating the YAFS mobility module and its associated coverage mechanism. To this end, a case study previously implemented in the iFogSim2 simulator [Mahmud et al., 2021] — an updated version of iFogSim whose main innovation is the incorporation of user mobility — was selected. This choice is appropriate as it enables a direct comparison between the behaviour of the YAFS mobility module and the results obtained in a simulator already validated for such purposes.

The selected scenario implements an audio translation service, an application that requires intensive pre-processing due to variations in voice quality depending on the speaker's volume and the surrounding noise. For devices such as smartphones, performing this processing locally in real time, or offloading it directly to the cloud, is often unfeasible due to latency and resource constraints. In this context, fog computing emerges as a promising solution, providing computational capacity close to the user and thus ensuring faster response times.

The application simulating this service is modelled as a Directed Acyclic Graph (DAG), illustrated in Figure 5.1. The *Client* module, running on the smartphone, is responsible for collecting sensor data and forwarding it for processing. The *Processing* module is ideally deployed on a fog gateway located as close as possible to the end user to minimise latency and enable real-time interaction. This

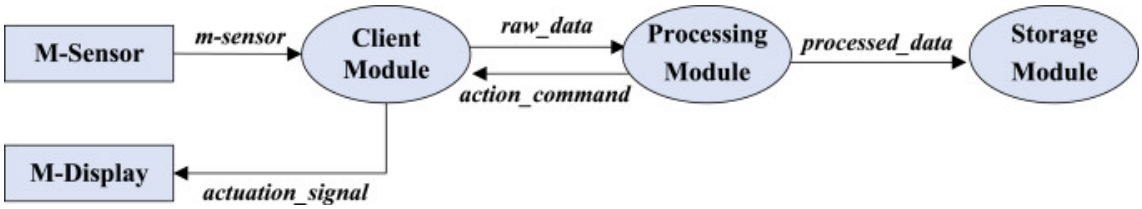


Figure 5.1: Audio Translation Service Application [Mahmud et al., 2021]

Figure 5.2: Network Topology for ATS application [Mahmud et al., 2021]

module performs audio data analysis operations, including noise filtering, intensity classification, and speech segmentation, before returning the results to the *Client* module for presentation to the user. Finally, the *Storage* module retains both the raw data collected from smartphones and the processed results, with the purpose of periodically updating the artificial intelligence model executed in the *Processing* module. Due to the large volume of stored data and the persistence requirements, this module is hosted in the cloud. The attributes of the modules are those shown in Table 5.1.

Table 5.1: Module Attributes for ATS Application

Module	RAM (GB)	Input (MB)	Output (MB)	CPU Instructions
Client	0.10	2.0	2.5	500
Processing	4.00	2.5	1.5	2500
Storage	4.00	1.0	1.0	1000

The network topology replicated that of the original iFogSim2 study, illustrated in Figure 5.2 and also adapted from the same source. It consists of 118 fog gate-

ways, represented in red, distributed across 12 distinct zones within Melbourne's Central Business District (CBD). Each zone has a proxy server, represented in blue, acting as an intermediary between fog nodes and the cloud data centre. The infrastructure specifications, as well as the characteristics of the application modules, were adapted from the original study's tables and are summarised in Table 5.2 and Table 5.3.

Table 5.2: General Configuration of Resources for ATS Test

Simulation Duration: 500 seconds				
Parameter	Cloud	Proxy	Fog	Smartphone
Number of Devices	10	12	118	50
Processing Speed (IPT)	44800	4000	3000	500
RAM (GB)	16	16	8	1

Table 5.3: Communication Parameters Between Resources for ATS Test

Connection	Cloud ↔ Proxy	Proxy ↔ Fog	Fog ↔ Smartphone
Bandwidth (Mbps)	100	50	100
Latency (PR in sec)	0.1	0.004	0.004

The test compared three distinct module migration policies in environments with user mobility:

- **Cloud-Centric Migration:** Whenever a module migration is required, the decision is delegated to the cloud, which determines the most appropriate destination node for re-deploying the service.
- **Non-Hierarchical Migration:** The migration decision is made locally by the node currently hosting the module, selecting the nearest neighbouring node to the end user without consulting any higher-level entity in the hierarchy.
- **Intra/Inter-Cluster Migration:** If the user remains within the same geographic cluster, the non-hierarchical migration mechanism is applied. If the user moves to a different cluster, the decision is escalated to a higher level, namely the proxy server or the cloud.

Two key performance metrics were evaluated in this study:

1. **Average Migration Time** — the average time required for a module to migrate from a source node to a destination node under each migration policy.
2. **Average Network Usage** — the average size of messages exchanged during the migration process, including the transfer of the module itself if it is not already present at the destination node.

The aim of this test is to verify whether the YAFS mobility module can reproduce behavioural trends similar to those observed in iFogSim2. The primary focus is

not on obtaining identical numerical results but on maintaining the same relative performance patterns among the migration policies. If such trends are preserved, it can be concluded that the YAFS mobility module is correctly implemented and functionally reliable.

Satellite edge computing Test - SatEdgeSim

The second test follows a similar objective to the previous one, but in this case aims to validate the satellite communication module. For this purpose, a benchmark scenario from the SatEdgeSim simulator was adopted as the reference case.

The virtual environment consists of multiple satellite constellations organised into three distinct layers. At the top, the *cloud computing centre* layer comprises 18 satellites in high-altitude orbits, providing the greatest computational resources. Below this is the *edge data centre* layer, composed of 24 satellites with slightly fewer resources than the cloud tier. Finally, the *mist layer* consists of 1,000 satellites distributed across six different low-altitude orbits. Nodes in the mist layer are responsible for generating computational tasks, which are then allocated to an execution node according to a given deployment strategy.

The evaluation focuses on comparing different deployment strategies. Task allocation occurs in two phases:

1. **Preliminary selection:** All satellites capable of receiving a given task are identified. For a satellite to be eligible, it must have a communication link with the source node and be within the maximum transmission distance.
2. **Final selection:** From the pre-selected satellites, one is chosen according to the strategy in use.

In the *weighted greedy* strategy, four evaluation indicators are considered: transmission distance, energy consumption, number of concurrent tasks, and computational capacity. Since three of these indicators are optimised when their values are smaller, and one — computational capacity — is optimised when its value is larger, the latter is transformed into a "smaller-is-better" indicator by considering the estimated processing time required for the task. These indicators are then normalised to account for differences in units and magnitudes, preventing bias in the decision-making process. Following normalisation, weighting factors are applied: 0.30 for transmission distance, 0.15 for energy consumption, 0.25 for the number of concurrent tasks, and 0.30 for computational capacity. The satellite with the lowest overall score is selected as the execution node.

In addition to the *weighted greedy* method, three alternative strategies were analysed:

- **Round Robin:** Tasks are cyclically assigned to the satellites in the eligible list, prioritising the node with the lowest number of ongoing tasks.
- **Trade-Off:** The same four evaluation indicators are used as in the *weighted greedy* approach, but without normalisation or weighting. Instead, a direct multiplicative combination of their values determines the selected node.

- **Traditional Polling:** Satellites in the eligible list are selected sequentially in a fixed rotation order.

Three different applications were used to test these strategies. Although they share the same Directed Acyclic Graph (DAG) structure, they differ in computational requirements and maximum allowable completion times, as detailed in Table 5.4. The DAG model includes the following modules:

- *user_request* — represents the task generated by the mist node;
- *task_data* — processes the task, deployed on the node selected by the deployment strategy;

Table 5.4: Module Attributes for the Three Applications

AUGMENTED REALITY				
Module	RAM (MB)	Input (MB)	Output (MB)	CPU Instructions (MI)
User_request	25	12	0.8	0
Task_data	25	12	0.8	60000
E_HEALTH				
User_request	13	80	80	0
Task_data	13	80	80	200000
HEAVY_COMP_APP				
User_request	9	0.4	0.4	0
Task_data	9	0.4	0.4	15000

The parameters of the simulated network are as follows: the simulation lasts for 10 minutes and the connections between the satellites have a bandwidth of 1000 Mbps.

The primary performance metric evaluated in this test is the **end-to-end delay**, defined as the time elapsed from the moment a task is sent until the corresponding result reaches the originating node. This metric is of critical importance in satellite-based computation scenarios, as it directly reflects the timeliness and efficiency of the system under varying deployment strategies.

Satellite as a bridge test

The third test aims to validate the integration between the user mobility and the satellite mobility modules. For this purpose, the simulation environment is set in an area with limited terrestrial communication infrastructure — specifically, the Peneda–Gerês National Park.

The validation procedure is structured into three distinct phases:

1. Exclusive use of the terrestrial network;
2. Exclusive use of the satellite network;
3. Combined use of terrestrial and satellite networks.

This phased approach enables a comparative assessment of each network's individual performance, as well as the effectiveness of their integration in supporting user mobility under varying levels of connectivity and infrastructure availability.

The chosen application for this test is an **Emergency Response Application**, illustrated in Figure 5.3. It automatically detects emergencies such as hiking accidents or road incidents by collecting data from device sensors (accelerometer, microphone). When a critical event is suspected, the data is sent to the cloud, located in an urban area, where an AI model processes the information to determine if emergency services should be triggered. Processed data is stored in a database for continuous retraining and refinement of the AI model. If an emergency is confirmed, the cloud sends a notification containing essential incident information, such as the user's coordinates, the route taken, and any additional user-provided data.

The attributes of the modules that comprise the emergency application are listed in Table 5.5. The messages exchanged between components are designed to accommodate the transmission of high-quality images and videos, alongside other sensor data, to meet the requirements of emergency communication. The cloud layer is expected to handle the highest CPU load, primarily due to AI model execution and sensor data processing.

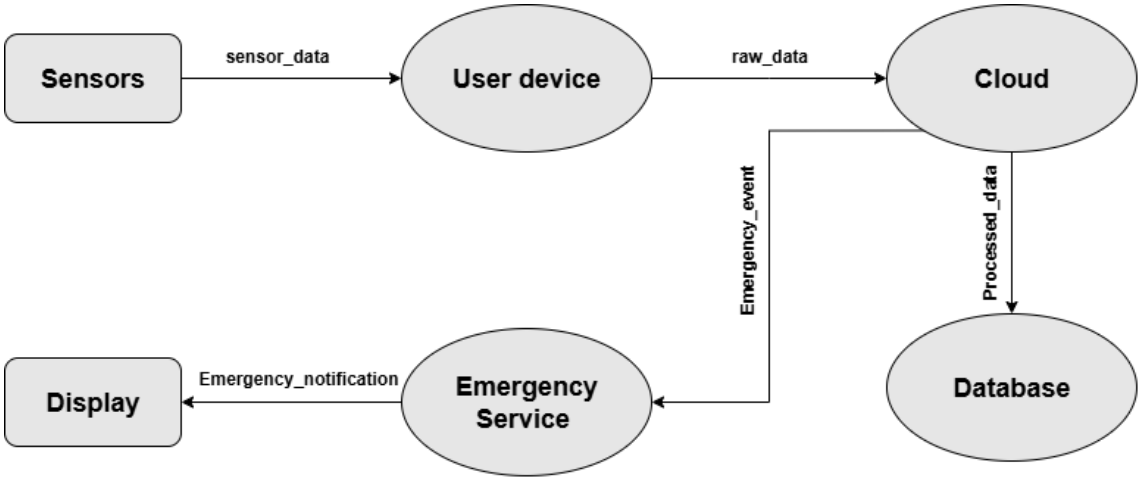


Figure 5.3: Emergency Response Application

Table 5.5: Attributes for the emergency application modules

Module	RAM (MB)	Input (MB)	Output (MB)	CPU Instructions (MI)
User Device	50	25.0	25.0	100
Cloud	2000	25.0	10.0	5000
Database	200	30.0	1.0	200
Emergency Service	100	10.0	10.0	100

The network topology consists of an edge layer comprising smartphones and the emergency service that receives alerts. Devices connect either to terrestrial communication towers via the mobile network or directly to satellites. Three terrestrial communication towers are connected to two proxy nodes, which in turn

maintain a direct link to the cloud and host the database. Satellites can connect to both smartphones and the cloud.

The general configuration of resources is detailed in Table 5.6, while the communication parameters for each link type are presented in Table 5.7. Whenever the propagation rate is marked as "calculated in the simulator," it is computed dynamically using the distance between nodes and the speed of light.

Table 5.6: General configuration of resources

Simulation Duration: 3600 seconds					
Parameter	**Cloud**	**Proxy**	**Radio Tower**	**Satellite**	**Smartphone**
Number of Devices	1	2	25	648	1000
Processing Speed (IPT)	100000	5000	2500	1500	500
RAM (GB)	16000	16	8	4	1

Table 5.7: Communication parameters between resources

Connection	**Bandwidth (Mbps)**	**Propagation Rate (s)**
Cloud ↔ Proxy	1000	0.09
Proxy ↔ Radio Tower	1000	0.004
Satellite ↔ Satellite	1000	calculated in the simulator
Radio Tower ↔ Radio Tower	1000	0.001
Radio Tower ↔ Smartphone	300	0.002
Satellite ↔ Smartphone	200	calculated in the simulator
Smartphone ↔ Sensors	10	0.001
Smartphone ↔ Display	10	0.001

The simulation runs for one hour to provide a comprehensive evaluation of mobility impacts—particularly those related to satellite movement—on communication and application performance.

The locations of terrestrial communication towers were sourced from a community-based platform providing infrastructure coordinates. A 5 km coverage range was assigned to each tower to ensure a stable connection whenever terrestrial coverage exists, while intentionally leaving certain areas without mobile network coverage, as can be seen in the Figure 5.4. The satellite constellation employed is OneWeb, selected for its moderate number of satellites—making it more suitable for ISL (Inter-Satellite Link) modelling than denser constellations such as Starlink. This constellation operates under a Keplerian orbital model, already implemented in the ISL connection module.

During the simulation, each user device attempts to connect to the best available network—terrestrial or satellite—based on signal strength and availability.

The test performance is assessed using the following metrics:

- End-to-end delay;
- Task failure rate;

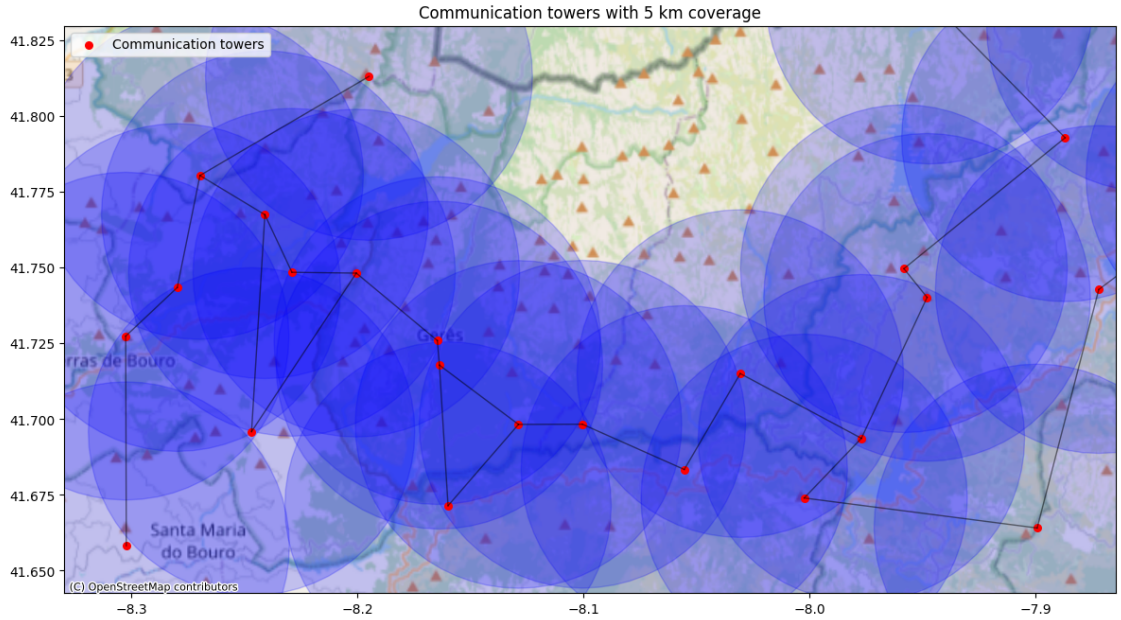


Figure 5.4: Terrestrial coverage in Peneda-Gerês Park

- Task success rate;
- Proportion of failed tasks due to: excessive delay, user mobility, or satellite mobility;
- Percentage of time users maintained network coverage.

Given that this test aims to validate the integrated functioning of both modules, the expected results are:

- Lowest end-to-end delay in the terrestrial-only scenario;
- Highest coverage and improved task success rate in the combined terrestrial–satellite scenario;
- Higher task failure rate in terrestrial-only use due to coverage gaps in remote areas;
- Increased latency-related failures in satellite-only use;
- Balanced performance and greater resilience to mobility-induced disruptions in the combined approach.

Performance Test

The final experiment does not primarily aim to validate the newly developed YAFS modules, but rather to demonstrate their applicability in relevant scenarios within the context of C2T, with a particular focus on architectures that integrate terrestrial and satellite networks. The case study employed remains identical to

that of the previous experiment, centred on the region of Peneda-Gerês National Park, with no changes made to the network topology, communication parameters, or deployed applications.

However, in contrast to the previous test, this experiment focuses on analysing the performance of *fog computing* when combining terrestrial nodes with satellite-based processing capabilities (*satellite fog computing*), as opposed to a scenario where satellites are used solely as communication relays without active involvement in task processing. The central objective of this test is therefore to evaluate whether the inclusion of *fog computing* in satellites results in improved overall system performance when compared to a traditional architecture based exclusively on terrestrial edge nodes.

To support this analysis, a node selection algorithm was employed, adapted from the *weight greedy* algorithm previously used in Test 2. The only changes introduced in this adaptation involve the removal of the energy consumption parameter, previously considered in the selection process, as this test focuses primarily on the *end-to-end delay* metric rather than on optimising energy usage. Accordingly, the selection of the execution node — whether in terrestrial or satellite *fog* environments — is based on three criteria: (i) the distance between the user and the candidate node, (ii) the number of tasks currently being processed on the node, and (iii) the computational capacity available at the time of allocation. The weights assigned to each criterion were, respectively, 5:3:2.

The task submission process begins with the user sending the request to the nearest edge device — which may be either a terrestrial communication tower or a satellite with reception capability. Upon receiving the task, the device applies the selection algorithm to determine the most suitable node for execution, based on the aforementioned criteria.

In order to assess the performance of the two described scenarios, the following evaluation metrics were defined:

- **End-to-End Delay:** A fundamental metric for C2T systems, it measures the total time elapsed from task submission to total response reception. Analysing this delay is critical to determining whether the system meets its time-sensitive requirements.
- **Throughput (Tasks completed per unit of time):** Indicates the number of tasks processed throughout the simulation, providing a clear view of the system's responsiveness and processing capacity under each architectural model.
- **Average Task Processed by location:** This metric provides insight into the algorithm's tendency when selecting the node on which each task will be executed.

5.1.3 Test Procedures and Data Collection

The simulations were executed on a personal computer, as previously described, within a virtual environment to ensure greater control over the libraries, their versions, and all associated dependencies. To reduce statistical errors, each test scenario was executed 30 times. The results generated follow the default output format of the YAFS simulator, consisting of two separate files: one recording the processing of messages at the nodes, and the other registering the traversal of messages across the network links, along with their respective latencies, arrival times, and processing durations.

Using these two records, many of the metrics were directly derived from the simulator outputs. In the case of end-to-end delay, the calculation was performed by identifying that, within the same task, all messages share the same identifier; thus, the metric was computed as the reception time of the last message minus the sending time of the first message. To determine task losses caused by mobility, the analysis was based on discontinuities in the sequence of task identifiers — for example, if the sequence progresses from ID 2 to ID 4, the missing ID 3 indicates that the corresponding task failed.

5.2 Validation of Developed Modules

After presenting the test scenarios, their objectives, and the data collection methodology, this section focuses on the analysis of the results obtained from the validation experiments conducted to assess both the functionality of the implemented modules and their integration within the overall system.

5.2.1 Audio Translation Service Test – iFogSim2 Comparison

The primary objective of this initial test was to validate the operation of the mobility modules. The evaluation was conducted by comparing the results obtained in the YAFS simulator with those from the iFogSim2 environment. The key aspect here is not a direct comparison of absolute values, but rather the observation of whether both simulators exhibit similar behavioural trends. Nevertheless, obtaining consistent and reasonably close numerical values is also relevant.

Starting with the analysis of the first chart (Figure 5.5), we observe that the trends in YAFS align with those reported for iFogSim2: the cloud-centric approach results in the highest volume of data migration, followed by the inter-cluster strategy, and finally the non-hierarchical approach. In terms of absolute values, all results remain relatively close to those in the original study.

Moving to the *average time to migrate modules* metric (Figure 5.6), the trend in the YAFS test is again consistent with the original results. However, the absolute values obtained with YAFS are slightly higher than those observed in iFogSim2.

This difference may be explained by the fact that the original study did not dis-

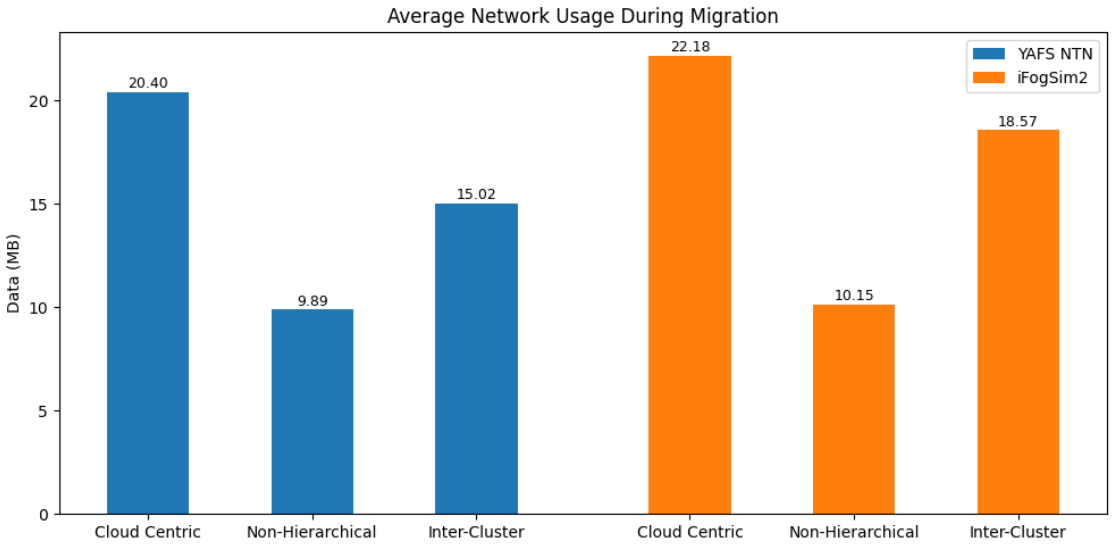


Figure 5.5: Average Data Usage

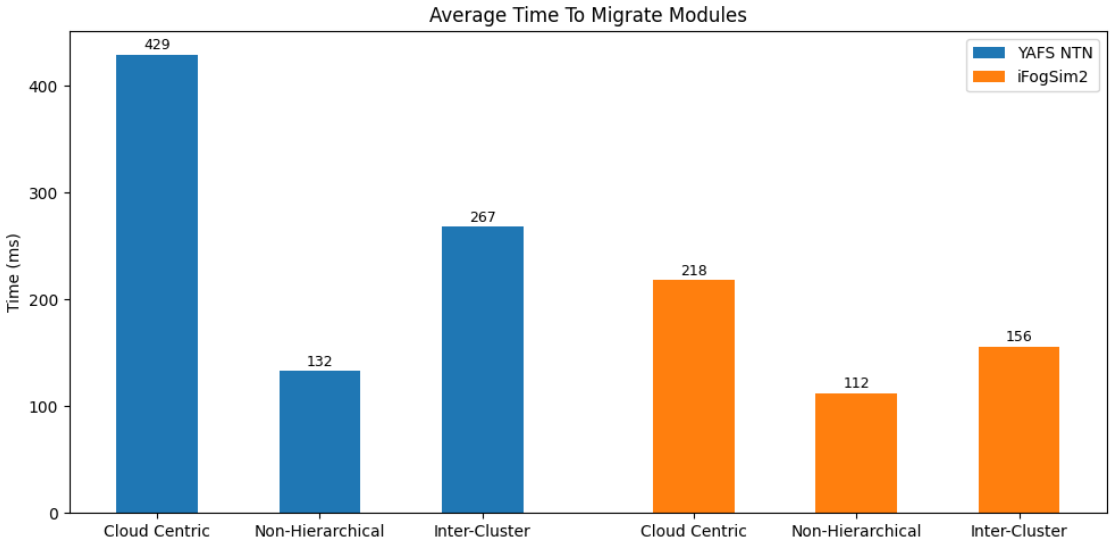


Figure 5.6: Average Time to Migrate

close the specific algorithm used to generate user mobility patterns. In the YAFS implementation, mobility was modelled using SUMO, which may have resulted in higher mobility levels overall. Consequently, the selection of a suitable node for migration may require more time in the YAFS simulations, leading to the observed increase in migration delays.

From these results, it can be concluded that the mobility modules are functioning correctly within the simulator and producing trends consistent with the established baseline from iFogSim2.

5.2.2 Satellite Edge Computing Test – SatEdgeSim Comparison

As in the first experiment, the objective of this test was to compare behavioural trends and verify whether the values obtained are reasonably close to those reported in the original SatEdgeSim study.

Analysing Figure 5.7, which presents the end-to-end delay results obtained with YAFS, and comparing them to those in the original experiment (Figure 5.8), it is evident that the overall trend is preserved. In both cases, the *weight greedy* strategy consistently produces the lowest latency values, followed by the *trade-off*, *round robin*, and, finally, *tradi polling* approaches. The absolute latency values obtained in YAFS also remain close to those in the reference experiment.

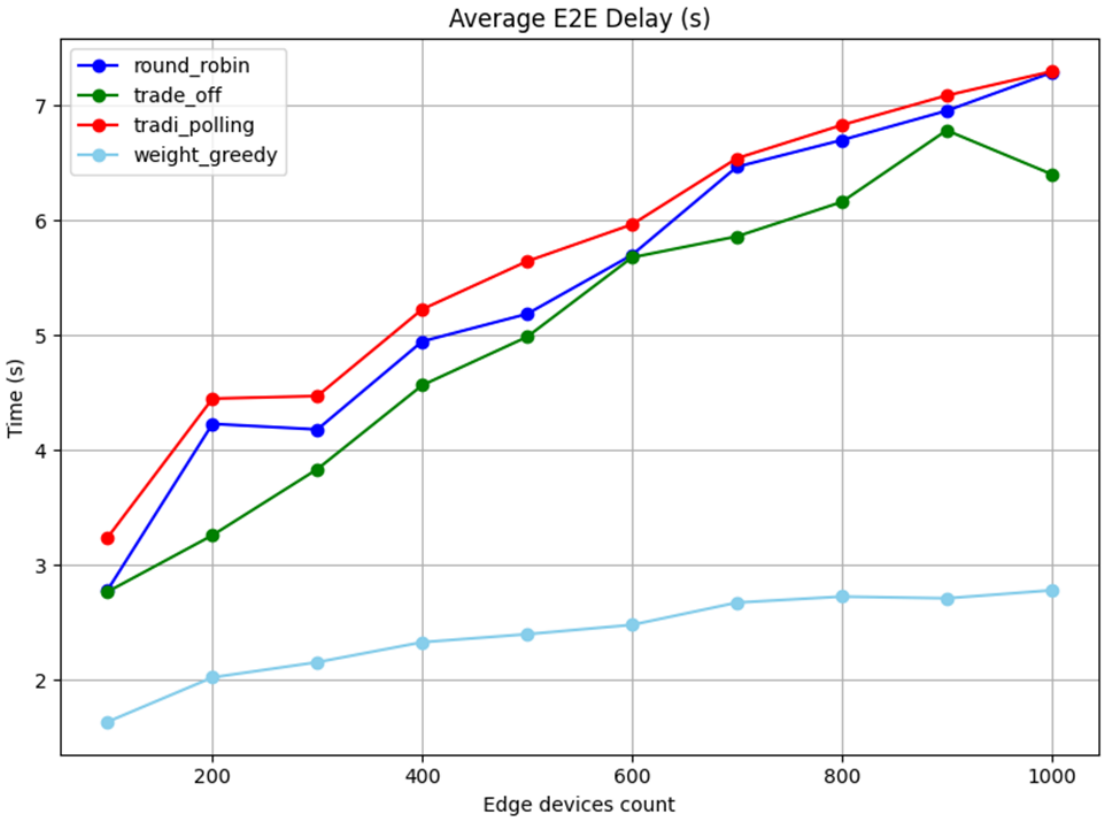


Figure 5.7: End-to-end Delay in YAFS

Minor discrepancies between the two sets of results may be attributed to the absence of specific reference values in the original study for satellite energy consumption in both idle and maximum operational states. These parameters can influence the selection of execution nodes and, consequently, the final latency measurements.

Despite these small differences, the results obtained in YAFS are in strong agreement with the original SatEdgeSim findings. This confirms that the satellite communication module operates correctly and integrates seamlessly with the rest of the system.

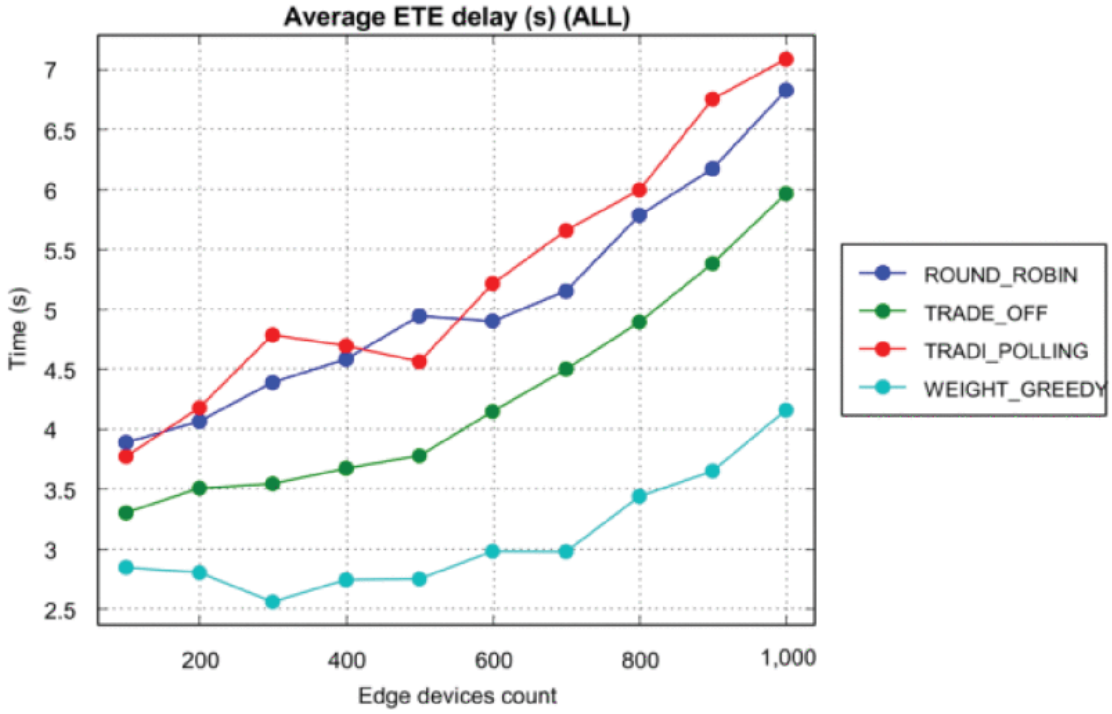


Figure 5.8: End-to-end Delay in SatEdgeSim

5.2.3 Satellite as a Bridge Test

The third and final validation experiment differs from the previous ones in that it is not a comparative reproduction of an existing study. Instead, its purpose is to verify whether, in a scenario where the expected outcome is known in advance, the results produced by YAFS align with those expectations.

Starting with the end-to-end delay shown in Figure 5.9, the results indicate that the use of satellites alone incurs the highest latency, followed by the terrestrial-only network. The hybrid network configuration — combining both satellite and terrestrial links — delivered the lowest latency overall. It was anticipated that the hybrid configuration would yield slightly higher latency than the terrestrial-only setup, given that messages would occasionally traverse satellite links. However, in this particular scenario, where significant terrestrial coverage gaps were present, the terrestrial-only network suffered additional delays as messages waited for connectivity, substantially increasing the overall latency. The hybrid configuration avoided this issue entirely, as the combined networks ensured full coverage throughout the simulation.

In this experiment, the application's maximum acceptable end-to-end delay was set to **30 seconds**. Any task exceeding this threshold was considered a latency failure.

Turning to Figure 5.10, which represents the percentage of successfully executed tasks, the satellite-only configuration again performed the worst, followed by the terrestrial-only network. As expected, the hybrid configuration achieved the highest performance, delivering a **100% task success rate**. This was not only due

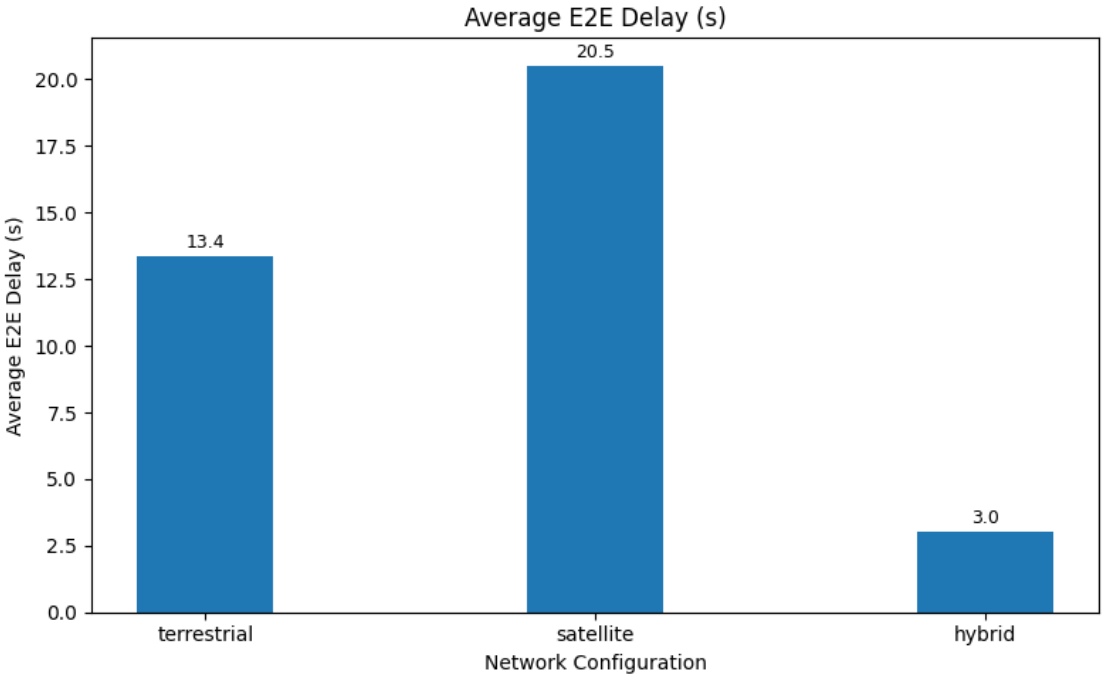


Figure 5.9: ETE Delay for Satellite as a Bridge Test

to its relatively low average end-to-end delay of approximately three seconds, but also because its complete coverage eliminated packet loss due to connectivity issues. Although the satellite-only network also provided full coverage, its much higher latency caused frequent violations of the 30-second limit, resulting in task failures.

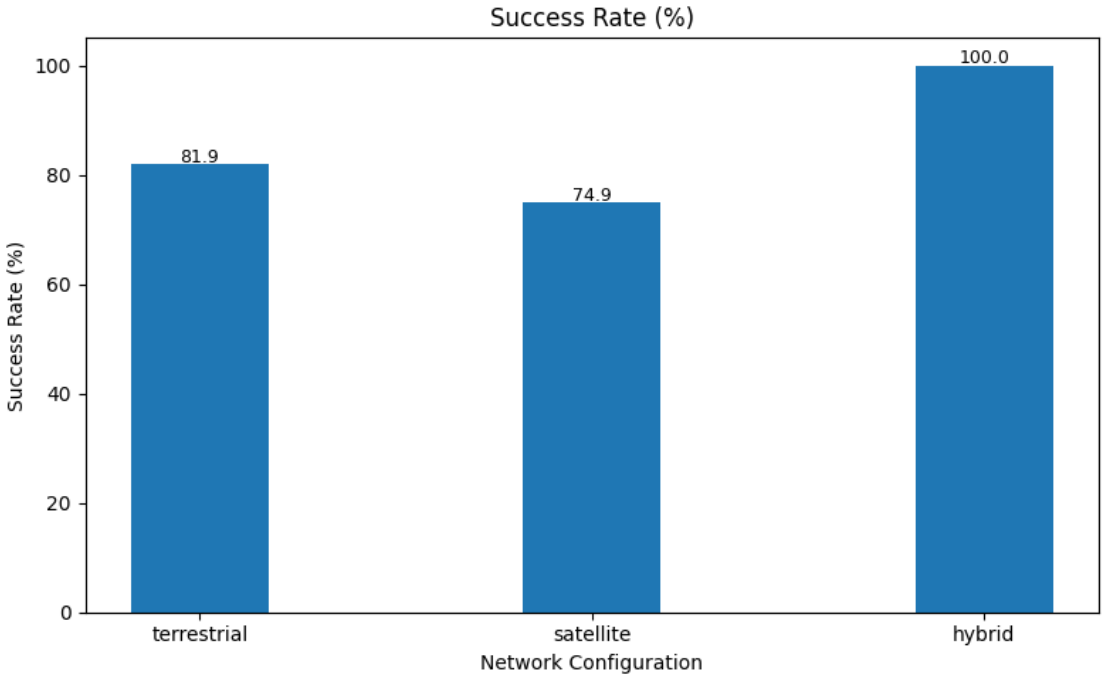


Figure 5.10: Successfully Executed Tasks for Satellite as a Bridge Test

Finally, Figure 5.11 presents the proportion of task failures, categorised by cause: latency or mobility. Consistent with the previous observations, both the satellite-only and hybrid configurations experienced no failures due to mobility, with all satellite-only failures being latency-related (above the 30-second threshold). Conversely, the terrestrial-only network was the only configuration to suffer mobility-related losses, caused by intermittent coverage gaps.

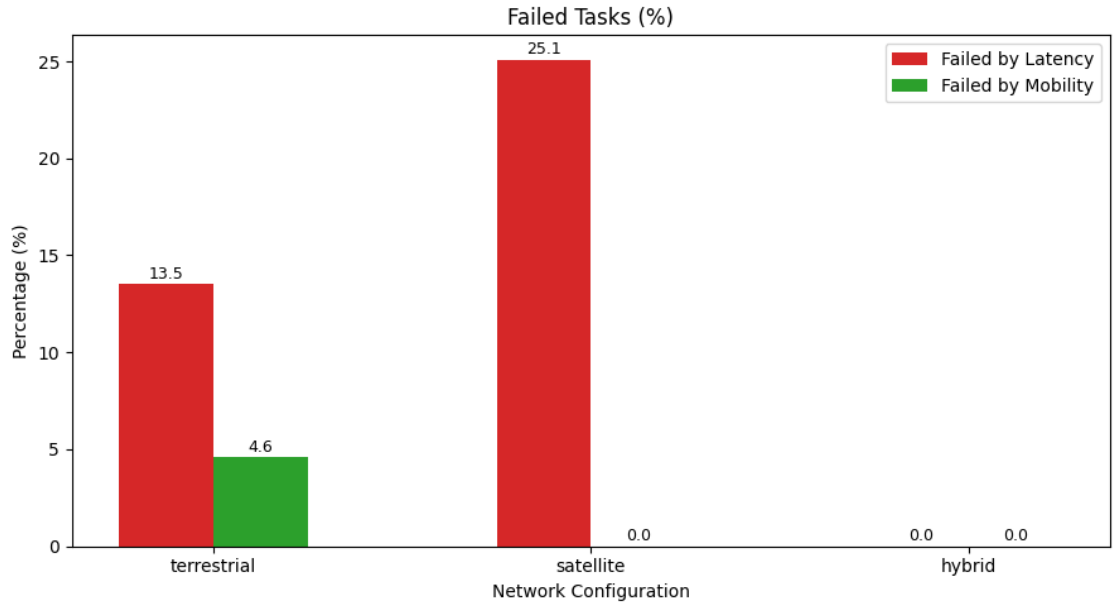


Figure 5.11: Task Failures for Satellite as a Bridge Test

Overall, these results confirm the initial hypothesis: the hybrid configuration, leveraging both satellite and terrestrial networks, provides the most reliable and efficient solution, combining complete coverage with acceptable latency for time-sensitive applications.

5.3 Performance Analysis and Use-Case Scenarios

Following the analysis of the previous experiments, it is equally important to present the use-case scenarios that demonstrate the practical utility of the updated YAFS. In particular, the third experiment described earlier serves as a real-world example, illustrating the impact of hybrid networks in areas with limited terrestrial infrastructure.

5.3.1 Use-Case 1 — Hybrid Networks for Connectivity in Remote Areas

The primary objective of this scenario was to evaluate how the combination of satellite and terrestrial links could improve performance in regions with poor terrestrial network coverage. The results, as discussed in Section 5.2.3, confirmed

that hybrid configurations consistently reduced latency and achieved 100% task completion within the application's 30-second deadline. This highlights the capability of the enhanced YAFS to model complex coverage scenarios and to evaluate trade-offs between latency, connectivity, and reliability.

5.3.2 Use-Case 2 — Performance Test: Satellite and Terrestrial Fog Computing

The second use-case scenario, referred to as the *Performance Test*, aimed to determine whether combining fog computing in satellites with fog computing in terrestrial nodes could deliver significantly better performance than using fog computing solely on terrestrial nodes.

As shown in Figure 5.12, the use of fog computing markedly reduced the overall end-to-end delay compared to the non-fog baseline. However, the difference between deploying fog computing only on terrestrial nodes and deploying it on both terrestrial and satellite nodes was relatively small. Figure 5.13 further reveals that, in all fog-enabled cases, the majority of tasks were processed at the fog layer, which contributed to lower delays. Meanwhile, Figure 5.14 indicates that there was no significant variation in the number of tasks completed per unit of time across the different configurations.

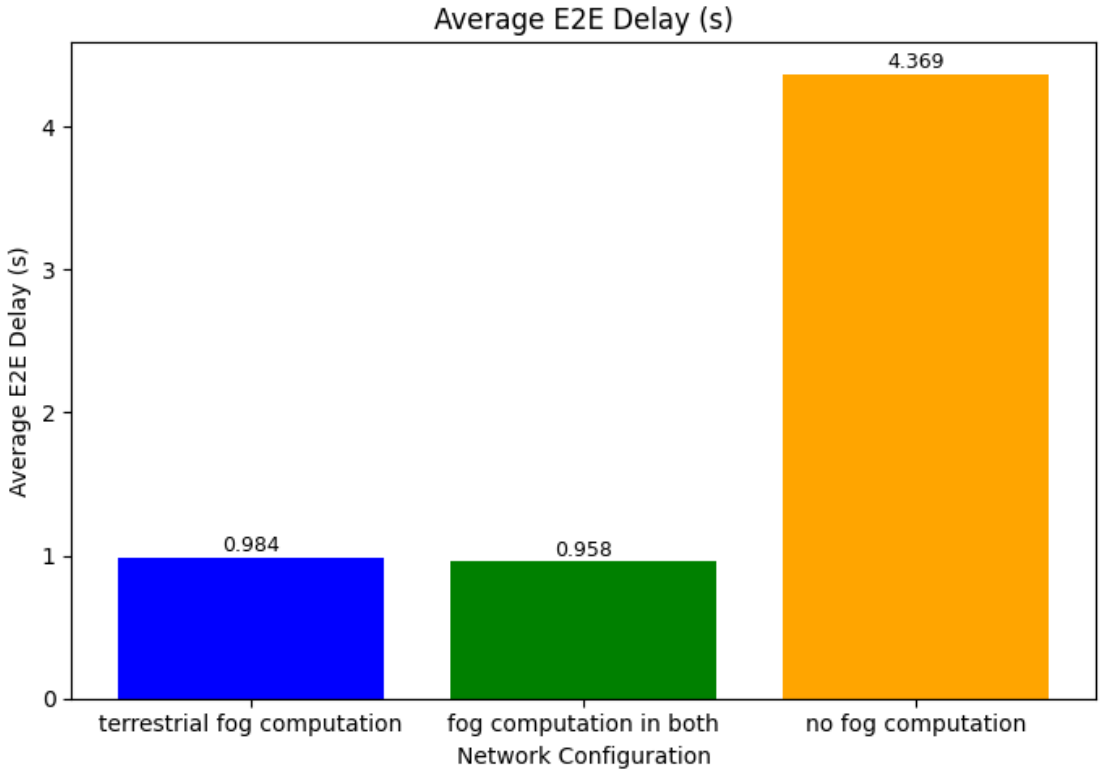


Figure 5.12: ETE Delay - Performance Test

The limited performance gap between the two fog configurations may be linked to the node selection algorithm used for task allocation. This underlines a key

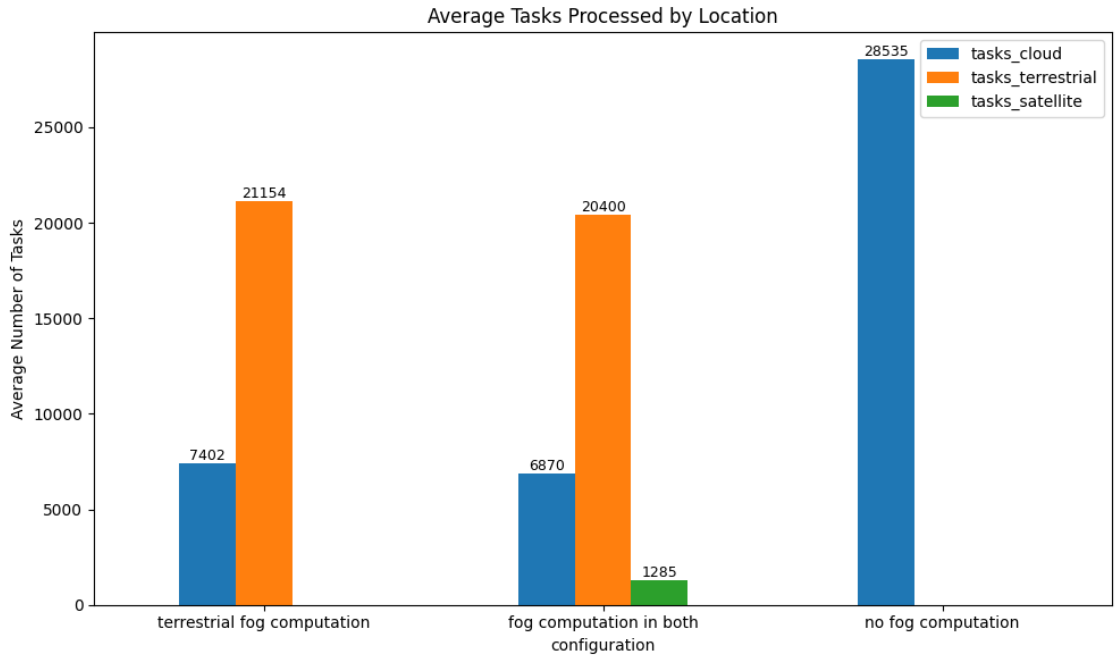


Figure 5.13: Average Tasks Processed by location - Performance Test

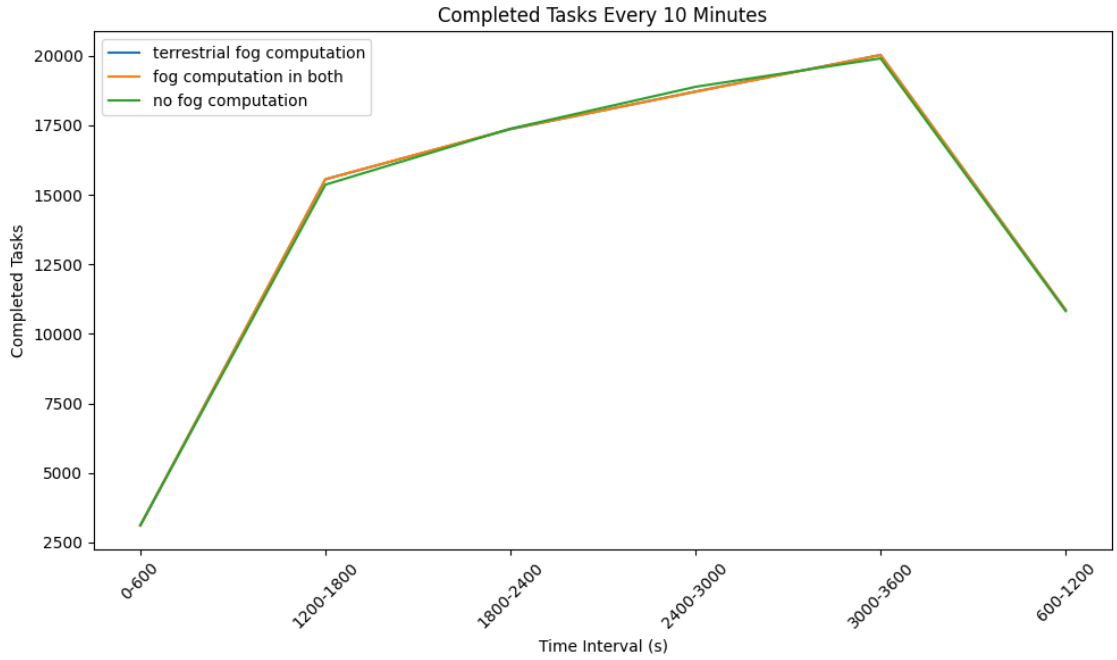


Figure 5.14: Tasks completed every 10 minutes

advantage of the enhanced YAFS: it enables researchers to explore alternative task allocation strategies to determine whether improved algorithms could fully leverage the potential benefits of satellite-based fog computing.

In summary, these use cases demonstrate the versatility of the enhanced YAFS for studying emerging trends in satellite-terrestrial computing and provide a practical tool for future research in C2T and space-based edge computing.

Chapter 6

Conclusion and Future Work

6.1 Summary of Achievements and Contributions

This dissertation addressed the challenge of ensuring connectivity within the C2T, particularly in scenarios characterised by high device mobility and limited terrestrial infrastructure, factors that can compromise service continuity. In response to this challenge, the work focused on enhancing YAFS through the implementation of two modules: one dedicated to user mobility and another to satellite communication. The primary objective was to enable the simulation of hybrid satellite-terrestrial networks and dynamic user scenarios, which would be logistically and economically unfeasible to evaluate in real-world conditions.

Throughout this dissertation, the limitations and gaps of existing simulators were identified and analysed, leading to the definition of the functional and non-functional requirements necessary to equip YAFS to support use cases representative of the identified problem. Following the implementation, the developed modules were rigorously validated through comparison with tests conducted using other simulators, as well as against scenarios with predictable outcomes. Additionally, a practical use case was developed to demonstrate the utility and applicability of these new modules.

The results indicate that the implemented modules effectively achieve the proposed objectives, confirming the reliability and robustness of the enhanced YAFS simulator for modelling hybrid C2T networks and scenarios involving high user mobility.

6.2 Limitations of the Study

Despite the implementations and advances presented in this work, several aspects were not considered, and certain simplifications imposed limitations on the final solution. The first limitation concerns the modelling of links, both between satellites and between satellites and end-users. In the current implementation, connections are assumed to be ideal, without accounting for noise, signal degra-

dation, or environmental factors such as adverse weather conditions.

Similarly, coverage modelling for both communication towers and satellites has been simplified, being represented as a circular area with a fixed radius rather than employing more sophisticated propagation and coverage models. Additionally, only two types of trace files were considered for simulating user mobility, which restricts the flexibility of movement patterns.

These limitations highlight potential avenues for future work, where more detailed link modelling, advanced coverage representations, and broader support for mobility and orbital parameters could further enhance the simulator's realism and utility

6.3 Future Research Directions

Considering the limitations identified in this study, several avenues for future research and enhancement of the simulator can be proposed. One key direction is the incorporation of more realistic link models, which would account for factors such as signal degradation, noise, interference, and environmental conditions (e.g., weather effects). This improvement would provide a more accurate representation of communication performance in both satellite and terrestrial links.

Another potential enhancement involves the development of advanced coverage models for both satellites and terrestrial communication infrastructure. Moving beyond simple circular coverage areas to models that incorporate terrain, antenna patterns, and propagation phenomena would significantly increase the fidelity of the simulations.

Furthermore, the simulation UAV, such as drones, represents a promising research direction. Integrating UAV into the simulator would enable the exploration of hybrid networks involving terrestrial, aerial, and satellite nodes, providing valuable insights into mobility management, resource allocation, and dynamic network orchestration in complex 3D environments.

Additional opportunities include expanding support for a wider variety of trace file formats to better model user mobility, as well as implementing satellite modules capable of simulating diverse orbital configurations beyond existing TLE-based constellations. These improvements would enhance the flexibility and applicability of the simulator, enabling researchers to investigate a broader range of C2T scenarios with high fidelity.

Another important direction for future work is the formal contribution of the developed modules to the official YAFS repository. A pull request will be prepared with the aim of merging the mobility and satellite communication extensions into the main branch of YAFS, thereby ensuring their availability to the wider research community. This step will not only enhance the simulator's core functionality but also foster collaboration and continuity in its development by enabling other researchers to build upon and refine these contributions.

References

David Perez Abreu, Karima Velasquez, Marilia Curado, and Edmundo Monteiro. A comparative analysis of simulators for the cloud to fog continuum. *Simulation Modelling Practice and Theory*, 101, 5 2020. ISSN 1569190X. doi: 10.1016/J.SIMPAT.2019.102029.

Auday Al-Dulaimy, Matthijs Jansen, Bjarne Johansson, Animesh Trivedi, Alexandru Iosup, Mohammad Ashjaei, Antonino Galletta, Dragi Kimovski, Radu Prodan, Konstantinos Tserpes, et al. The computing continuum: From iot to the cloud. *Internet of Things*, 27:101272, 2024. URL `https://www.sciencedirect.com/science/article/pii/S2542660524002130`.

Thales Alenia and Space CNES, Viveris Technologie. Opensand. https://www.opensand.org/, 2006. Accessed at:09/01, 2025.

Karan Bajaj, Bhisham Sharma, and Raman Singh. Comparative analysis of simulators for iot applications in fog/cloud computing. *International Conference on Sustainable Computing and Data Communication Systems, ICSCDS 2022 - Proceedings*, pages 983–988, 2022. doi: 10.1109/ICSCDS53736.2022.9760897.

Hind Bangui, Said Rakrak, Said Raghay, and Barbora Buhnova. Moving to the edge-cloud-of-things: recent advances and future research directions. *Electronics*, 7(11):309, 2018.

Malika Bendechache, Sergej Svorobej, Patricia Takako Endo, and Theo Lynn. future internet simulating resource management across the cloud-to-thing continuum: A survey and future directions. *Future Internet*, 12(6):95, 2020. doi: 10.3390/fi12060095. URL `www.mdpi.com/journal/futureinternet`.

R Buyya, R Ranjan ... international conference on ..., and undefined 2009. Modeling and simulation of scalable cloud computing environments and the cloudsim toolkit: Challenges and opportunities. *ieeexplore.ieee.org*, pages 1–11, 2009. URL `https://ieeexplore.ieee.org/abstract/document/5192685/?casa_token=tKHURgjaAaQAAAAA:ZYR0ol6YGUa5HeMuRh4-ERyFfr4_rq3VEImJDmNqWlbyHXocqh0iy2emamxpRjGMHDNsIHyAfA`.

Shkelzen Cakaj. The parameters comparison of the "starlink" leo satellites constellation for different orbital shells. *Frontiers in Communications and Networks*, 2:643095, May 2021.

Quan Chen, Giovanni Giambene, Lei Yang, Chengguang Fan, and Xiaoqian Chen. Analysis of inter-satellite link paths for leo mega-constellation networks. *IEEE Transactions on Vehicular Technology*, 70(3):2743–2755, 2021.

Lei Cheng, Gang Feng, Yao Sun, Mengjie Liu, and Shuang Qin. Dynamic computation offloading in satellite edge computing. In *ICC 2022-IEEE International Conference on Communications*, pages 4721–4726. IEEE, 2022.

Muhammad Fahimullah, Guillaume Philippe, Shohreh Ahvar, and Maria Trocan. Simulation tools for fog computing: A comparative analysis. *Sensors 2023, Vol. 23, Page 3492*, 23:3492, 3 2023. ISSN 1424-8220. doi: 10.3390/S23073492. URL https://www.mdpi.com/1424-8220/23/7/3492/htmhttps://www.mdpi.com/1424-8220/23/7/3492.

Juan A Fraire, Pablo Madoery, Mehdi Ait Mesbah, Oana Iova, and Fabrice Valois. Simulating lora-based direct-to-satellite iot networks with florasat. In *2022 IEEE 23rd International Symposium on a World of Wireless, Mobile and Multimedia Networks (WoWMoM)*, pages 464–470. IEEE, 2022.

M Gill, D Singh Computer Science Review, and undefined 2021. A comprehensive study of simulation frameworks and research directions in fog computing. *Elsevier*, 2021. URL https://www.sciencedirect.com/science/article/pii/S1574013721000319?casa_token=3t4rk9hz3f4AAAAA:LO54cW0WogwyKBm_SBpwJA_YHkiKHHGlicrY_u5ewK-DB7gmvPOQmarjxFppZMgJGs6NmWqXqg.

Harshit Gupta, Amir Vahid Dastjerdi, Soumya K. Ghosh, and Rajkumar Buyya. ifogsim: A toolkit for modeling and simulation of resource management techniques in internet of things, edge and fog computing environments. *Software - Practice and Experience*, 47:1275–1296, 6 2016. ISSN 1097024X. doi: 10.1002/spe.2509. URL https://arxiv.org/abs/1606.02007v1.

Aric Hagberg, Pieter J Swart, and Daniel A Schult. Exploring network structure, dynamics, and function using networkx. Technical report, Los Alamos National Laboratory (LANL), Los Alamos, NM (United States), 2008.

A Hazra, P Rana, M Adhikari, and T Amgoth. Fog computing for next-generation internet of things: fundamental, state-of-the-art and research challenges. *Computer Science Review*, 48:100549, 2023. doi: 10.1016/j.cosrev.2023.100549. URL https://www.sciencedirect.com/science/article/pii/S1574013723000163.

Jehyun Heo, Seungwoo Sung, Hyunwoo Lee, Incheol Hwang, and Daesik Hong. Mimo satellite communication systems: A survey from the phy layer perspective. *IEEE Communications Surveys and Tutorials*, 25:1543–1570, 2023. ISSN 1553877X. doi: 10.1109/COMST.2023.3294873.

T Issariyakul, E Hossain, T Issariyakul, and E Hossain. *Introduction to network simulator 2 (NS2)*. Springer, 2009. ISBN 978-1-4614-1406-3.

Min Jia, Liang Zhang, Jian Wu, Shiyao Meng, and Qing Guo. Collaborative satellite-terrestrial edge computing network for everyone-centric customized services. *IEEE Network*, 37(5):197–205, 2022.

Weiwei Jiang, Yafeng Zhan, Xiaolong Xiao, and Guanglin Sha. Network simulators for satellite-terrestrial integrated networks: A survey. *IEEE Access*, 2023. doi: 10.1109/ACCESS.2023.3313229.

Taeyeoun Kim, Jeongho Kwak, and Jihwan P Choi. Satellite edge computing architecture and network slice scheduling for iot support. *IEEE Internet of Things journal*, 9(16):14938–14951, 2021.

Sarah Laoyan. What is agile methodology? (a beginner's guide) [2024] • asana. https://asana.com/pt/resources/agile-methodology, 2024. Accessed at:09/01, 2025.

Mohammed Laroui, Boubakr Nour, Hassine Moungla, Moussa A. Cherif, Hossam Afifi, and Mohsen Guizani. Edge and fog computing for iot: A survey on current research activities & future directions. *Computer Communications*, 180: 210–231, 12 2021. ISSN 0140-3664. doi: 10.1016/J.COMCOM.2021.09.003.

Isaac Lera, Carlos Guerrero, and Carlos Juiz. Yafs: A simulator for iot scenarios in fog computing. *IEEE Access*, 7:91745–91758, 2019. ISSN 21693536. doi: 10.1109/ACCESS.2019.2927895.

Pablo Alvarez Lopez, Michael Behrisch, Laura Bieker-Walz, Jakob Erdmann, Yun-Pang Flötteröd, Robert Hilbrich, Leonhard Lücken, Johannes Rummel, Peter Wagner, and Evamarie Wießner. Microscopic traffic simulation using sumo. In *The 21st IEEE International Conference on Intelligent Transportation Systems*. IEEE, 2018. URL `https://elib.dlr.de/124092/`.

Redowan Mahmud, Samodha Pallewatta, Mohammad Goudarzi, and Rajkumar Buyya. Ifogsim2: An extended ifogsim simulator for mobility, clustering, and microservice management in edge and fog computing environments. *Journal of Systems and Software*, 190, 9 2021. ISSN 01641212. doi: 10.1016/j.jss.2022.111351. URL `https://arxiv.org/abs/2109.05636v2`.

Spiridoula V. Margariti, Vassilios V. Dimakopoulos, and Georgios Tsoumanis. Modeling and simulation tools for fog computing-a comprehensive survey from a cost perspective. *Future Internet*, 12, 5 2020. ISSN 19995903. doi: 10.3390/FI12050089.

Frazer McLean. Razerm/orbital: High level orbital mechanics package. https://github.com/RazerM/orbital, 2014. Accessed at:09/01, 2025.

Charafeddine Mechalikh, Hajer Taktak, and Faouzi Moussa. Pureedgesim: A simulation toolkit for performance evaluation of cloud, fog, and pure edge computing environments. *2019 International Conference on High Performance Computing and Simulation, HPCS 2019*, pages 700–707, 7 2019. doi: 10.1109/HPCS48598.2019.9188059.

T Monks and A Harper. Improving the usability of open health service delivery simulation models using python and web apps. *NIHR Open Research*, 3: 48, 2023. doi: 10.3310/nihropenres.13467.2. URL `https://doi.org/10.3310/nihropenres.13467.2`. Version 2; Peer Review: 3 Approved.

Tamzin Morphy. Risk assessment matrix 3 by 3 example with free download. https://www.stakeholdermap.com/risk/risk-assessment-matrix-simple-3x3.html, 2023. Accessed at:09/01, 2025.

Brian Niehoefer, Sebastian Šubik, and Christian Wietfeld. The cni open source satellite simulator based on omnet++. In *Proceedings of the 6th International ICST Conference on Simulation Tools and Techniques*, pages 314–321, 2013. ISBN 2013.251580.

NIST. The nist definition of cloud computing. *National institute of science and technology, special publication*, 800(2011):145, 2011. URL `https://cloudinfosec.wordpress.com/wp-content/uploads/2013/05/the-nist-definition-of-cloud-computing.pdf`.

Jianli Pan and Raj Jain. A survey of network simulation tools: Current status and future developments. *Email: jp10@ cse. wustl. edu*, 2(4):45, 2008.

Price-Whelan, Adrian M. Astropy Collaboration, Pey Lian Lim, Nicholas Earl, Nathaniel Starkman, Larry Bradley, and Shupe et al. The Astropy Project: Sustaining and Growing a Community-oriented Open-source Project and the Latest Major Release (v5.0) of the Core Package. *apj*, 935(2):167, August 2022. doi: 10.3847/1538-4357/ac7c74.

Jani Puttonen, Sami Rantanen, Frans Laakso, Janne Kurjenniemi, Kari Aho, and Guray Acar. Satellite model for network simulator 3. *SIMUTools 2014 - 7th International Conference on Simulation Tools and Techniques*, pages 86–91, 2014. doi: 10.4108/ICST.SIMUTOOLS.2014.254631.

Tariq Qayyum, Asad Waqar Malik, Muazzam A Khan Khattak, Osman Khalid, and Samee U Khan. Fognetsim++: A toolkit for modeling and simulation of distributed fog environment. *IEEE Access*, 6:63570–63583, 2018.

Sebastian Rampfl. Network simulation and its limitations. In *Proceeding zum seminar future internet (FI), Innovative Internet Technologien und Mobilkommunikation (IITM) und autonomous communication networks (ACN)*, volume 57. Citeseer, 2013.

Brandon Rhodes. Skyfield — documentation. https://rhodesmill.org/skyfield/, 2019. Accessed at:09/01, 2025.

George F Riley and Thomas R Henderson. The ns-3 network simulator. In *Modeling and tools for network simulation*, pages 15–34. Springer, 2010.

Juan Luis Cano Rodríguez et al. poliastro/poliastro: poliastro 0.17. 0 (scipy us'22 edition), 2022. Accessed at:09/01, 2025.

Nurul I Sarkar and Syafnidar A Halim. Simulation of computer networks: simulators, methodologies and recommendations. In *5th International Conference on Information Technology and Application (ICITA'08), Cairns, Australia*, pages 420–425, 2008.

Fabian P. Schmidt. Beyond modules — beyond 0.7.5 documentation. https://beyond.readthedocs.io/en/stable/api/index.html, 2021. Accessed at:09/01, 2025.

Cagatay Sonmez, Atay Ozgovde, and Cem Ersoy. Edgecloudsim: An environment for performance evaluation of edge computing systems. *Transactions on Emerging Telecommunications Technologies*, 29:39–44, 11 2018. ISSN 21613915. doi: 10.1002/ETT.3493.

Emilio Calvanese Strinati, Sergio Barbarossa, Taesang Choi, Antonio Pietrabissa, Alessandro Giuseppi, Emanuele De Santis, Josep Vidal, Zdenek Becvar, Thomas Haustein, Nicolas Cassiau, Francesca Costanzo, Junhyeong Kim, and Ilgyu Kim. 6g in the sky: On-demand intelligence at the edge of 3d networks. *ETRI Journal*, 42:643–657, 10 2020. ISSN 22337326. doi: 10.4218/ETRIJ.2020-0205.

Software Solutions Studio. A gentle introduction to discrete-event simulation | software solutions studio. Available online at: https://softwaresim.com/blog/a-gentle-introduction-to-discrete-event-simulation, 2022. Accessed on 8th January 2025.

Syed Bilal Raza Tirmizi, Yunfei Chen, Subhash Lakshminarayana, Wei Feng, and Aziz A. Khuwaja. Hybrid satellite–terrestrial networks toward 6g: Key technologies and open issues. *Sensors 2022, Vol. 22, Page 8544*, 22:8544, 11 2022. ISSN 1424-8220. doi: 10.3390/S22218544. URL `https://www.mdpi.com/1424-8220/22/21/8544/htmhttps://www.mdpi.com/1424-8220/22/21/8544`.

Oleksandr Tymchenko, Bohdana Havrysh, Mariya Nazarkevych, Oleksandr O Tymchenko, and Orest Khamula. Architecture of the simulator of the personal local wireless networks: Examples of implementation. In *IntelITSIS*, pages 317–329, 2021.

U.S. Census Bureau. How do i calculate the distance between two latitude/longitude points using the geometry functions? `https://web.archive.org/web/20041108132234/http://www.census.gov/cgi-bin/geo/gisfaq?Q5.1`, 2004. Accessed via the Internet Archive on August 16, 2025.

Andras Varga. *OMNeT++*, pages 35–59. Springer Berlin Heidelberg, 2010. ISBN 978-3-642-12331-3. doi: 10.1007/978-3-642-12331-3_3. URL `https://doi.org/10.1007/978-3-642-12331-3_3`.

Junyong Wei, Suzhi Cao, Siyan Pan, Jiarong Han, Lei Yan, and Lei Zhang. Satedgesim: A toolkit for modeling and simulation of performance evaluation in satellite edge computing environments. *2020 12th International Conference on Communication Software and Networks, ICCSN 2020*, pages 307–313, 6 2020. doi: 10.1109/ICCSN49894.2020.9139057.

Hongxia Zhang, Shiyu Xi, Hongzhao Jiang, Qi Shen, Bodong Shang, and Jian Wang. Resource allocation and offloading strategy for uav-assisted leo satellite edge computing. *Drones*, 7(6):383, 2023.