UNIVERSIDADE FEDERAL DO PARÁ
INSTITUTO DE TECNOLOGIA
PROGRAMA DE PÓS-GRADUAÇÃO EM ENGENHARIA ELÉTRICA

Redesigning CAVIAR: A Framework for Next-Generation 6G/B6G Simulations

João Pedro Barbosa Albuquerque

DM 34/2025

UFPA / ITEC / PPGEE
Campus Universitário do Guamá
Belém-Pará-Brasil
2025

UNIVERSIDADE FEDERAL DO PARÁ
INSTITUTO DE TECNOLOGIA
PROGRAMA DE PÓS-GRADUAÇÃO EM ENGENHARIA ELÉTRICA

João Pedro Barbosa Albuquerque

Redesigning CAVIAR: A Framework for Next-Generation 6G/B6G Simulations

A dissertation submitted to the examination committee in the graduate department of Electrical Engineering at the Federal University of Pará in partial fulfillment of the requirements for the degree of Master of Science in Electrical Engineering with emphasis in Telecommunications.

Orientador: Prof. Dr. Aldebaro Klautau

UFPA / ITEC / PPGEE
Campus Universitário do Guamá
Belém-Pará-Brasil
2025

Dados Internacionais de Catalogação na Publicação (CIP) de acordo com ISBD
Sistema de Bibliotecas da Universidade Federal do Pará
Gerada automaticamente pelo módulo Ficat, mediante os dados fornecidos pelo(a) autor(a)

B238r Barbosa Albuquerque, João Pedro.
Redesigning CAVIAR: A Framework for Next Generation 6G/B6G Simulations / João Pedro Barbosa Albuquerque. — 2025.
81 f. : il. color.

Orientador(a): Prof. Dr. Aldebaro Klautau
Dissertação (Mestrado) - Universidade Federal do Pará, Instituto de Tecnologia, Programa de Pós-Graduação em Engenharia Elétrica, Belém, 2025.

1. Co-simulation. 2. Digital twin. 3. 3GPP. 4. 5G. 5. 6G. I. Título.

CDD 621.3822

UNIVERSIDADE FEDERAL DO PARÁ
INSTITUTO DE TECNOLOGIA
PROGRAMA DE PÓS-GRADUAÇÃO EM ENGENHARIA ELÉTRICA

Redesigning CAVIAR: A Framework for Next-Generation 6G/B6G Simulations

AUTHOR: **JOÃO PEDRO BARBOSA ALBUQUERQUE**

DISSERTAÇÃO DE MESTRADO SUBMETIDA À BANCA EXAMINADORA APROVADA PELO COLEGIADO DO PROGRAMA DE PÓS-GRADUAÇÃO EM ENGENHARIA ELÉTRICA, SENDO JULGADA ADEQUADA PARA A OBTENÇÃO DO GRAU DE MESTRE EM ENGENHARIA ELÉTRICA NA ÁREA DE TELECOMUNICAÇÕES.

APROVADA EM: 27/10/2025

BANCA EXAMINADORA:

Prof. Dr. Aldebaro Klautau
(Orientador – PPGEE/ITEC/UFPA)

Prof. Dr. Glauco Gonçalves
(Avaliador Interno - PPGEE/ITEC/UFPA)

Prof. Dr. Bruno Souza Lyra Castro
(Avaliador Externo ao Programa – CAMPUS CASTANHAL/UFPA)

VISTO:

Prof. Dr. Diego Lisboa Cardoso
(Coordenador do PPGEE/ITEC/UFPA)

Dedico este trabalho a todos que me ajudaram a chegar até aqui, com apoio, paciência e incentivo

Acknowledgements

The completion of this work was only possible thanks to the support, patience, and encouragement of many people, to whom I am deeply grateful.

First and foremost, I thank my family, the source of unwavering support and comfort during the most challenging moments of this demanding endeavor. Their presence—often discreet yet constant—was a source of strength in times of difficulty and consolation during moments of burden.

I am also profoundly grateful to my professors and advisor, Aldebaro, whose dedication and patience guided me throughout this journey.

I am also thankful for my colleagues in the workspace, whose camaraderie made even the most challenging moments more bearable and enjoyable. Special thanks go to the CAVIAR team, João Borges, and Felipe Bastos, whose support has been pivotal to this research and discovery journey.

My heartfelt gratitude extends to my girlfriend and future wife, Byanka Fernandes, who has endured the many ups and downs of university life alongside me. Her steadfast support has been invaluable, and I am forever grateful for her unwavering presence.

To LASSE (Research and Development Center for Telecommunications, Automation, and Electronics), my heartfelt gratitude for the opportunity to conduct this work in such a fertile environment for both technical and personal growth. My experience in the laboratory provided not only knowledge but also invaluable experiences that I will carry with me for life.

"What?"

–Richard Nixon

Resumo

A evolução das redes móveis rumo à sexta geração (6G) e à consolidação do Open Radio Access Network (Open RAN) demanda ferramentas de simulação flexíveis, modulares e capazes de integrar comunicação, mobilidade e inteligência artificial em cenários ultra-realistas. Este trabalho apresenta a mais recente versão do framework Communication Networks, Artificial Intelligence and Computer Vision with 3D Computer-generated Imagery (CAVIAR), uma ferramenta de co-simulação alinhada às diretrizes do 3rd Generation Partnership Project (3GPP), capaz de integrar módulos independentes para simulação tridimensional (3D), mobilidade, comunicações e inteligência artificial, orquestrados de forma assíncrona para execução em tempo real. O CAVIAR incorpora modelagem de canais por traçado de raios (ray tracing) utilizando Sionna-Ray Tracing (Sionna-RT) e Wireless InSite, simulação de redes de quinta geração (5G) com Network Simulator 3 (ns-3) e módulo 5G-Lena, técnicas de aceleração de cálculos de propagação e monitoramento interativo por meio de InfluxDB e Grafana. Para validar o ferramental, foram implementados três casos de uso representativos alinhados aos estudos descritos nas especificações do 3GPP Release 19: (i) monitoramento de animais em áreas rurais com alocação dinâmica de Physical Resource Blocks (PRBs) via Open RAN, (ii) busca e resgate urbano com detecção de pessoas usando You Only Look Once (YOLO) sobre enlace de Non-Terrestrial Network (NTN), e (iii) coordenação de robôs em ambientes internos com Integrated Sensing and Communication (ISAC) multimodal combinando Light Detection and Ranging (LiDAR) e imagem Red, Green, Blue (RGB). Os resultados demonstram que o CAVIAR atinge Indicadores-Chave de Desempenho (Key Performance Indicators – KPIs) com alta precisão e baixo uso de recursos computacionais (Unidade Central de Processamento – CPU, Unidade de Processamento Gráfico – GPU e Memória de Acesso Aleatório – RAM), mesmo em cenários complexos, possibilitando a execução de aplicações de rede sensíveis à latência e com alta confiabilidade, reforçando sua aplicabilidade para pesquisas avançadas em 5G-Advanced e 6G.

Palavras-chave: Redes móveis, Co-simulação, Gêmeo digital, Open RAN, ISAC, 5G, 6G, 3GPP

Abstract

The evolution of mobile networks toward the sixth generation (6G) and the consolidation of the Open Radio Access Network (Open RAN) require flexible, modular simulation tools capable of integrating communications, mobility, and artificial intelligence in ultra-realistic scenarios. This work presents the latest version of the Communication Networks, Artificial Intelligence and Computer Vision with 3D Computer-generated Imagery (CAVIAR) framework, a co-simulation framework aligned with the 3rd Generation Partnership Project (3GPP) guidelines, capable of integrating independent modules for three-dimensional (3D) simulation, mobility, communications, and artificial intelligence, orchestrated asynchronously for real-time execution. CAVIAR incorporates channel modeling via ray tracing using Sionna-Ray Tracing (Sionna-RT) and Wireless InSite, fifth-generation (5G) network simulation with Network Simulator 3 (ns-3) and the 5G-Lena module, propagation calculation acceleration techniques, and interactive monitoring through InfluxDB and Grafana. To validate the framework, three representative use cases aligned with studies described in the 3GPP Release 19 specifications were implemented: (i) rural animal monitoring with dynamic Physical Resource Block (PRB) allocation via Open RAN, (ii) urban search and rescue with You Only Look Once (YOLO)-based human detection over Non-Terrestrial Network (NTN) links, and (iii) indoor robot coordination with multimodal Integrated Sensing and Communication (ISAC) combining Light Detection and Ranging (LiDAR) and Red, Green, Blue (RGB) imaging. The results show that CAVIAR achieves Key Performance Indicators (KPIs) with high accuracy and low computational resource usage even in complex scenarios, enabling latency-sensitive and high-reliability network applications, thus reinforcing its applicability for advanced research in 5G-Advanced and 6G.

Key-words: Mobile networks, Co-simulation, Digital twin, Open RAN, ISAC, 5G, 6G, 3GPP

List of figures

List of tables

List of acronyms

3GPP 3rd Generation Partnership Project.

A2S Aerial-to-Space.

AI Artificial Intelligence.

API Application Programming Interface.

BER Bit Error Rate.

CAVIAR Communication Networks, Artificial Intelligence and Computer Vision with 3D Computer-generated Imagery.

CBR Constant Bit Rate.

CPU Central Processing Unit.

CQI Channel Quality Indicator.

DASH Dynamic Addaptative Streaming over HTTP.

DT Digital Twin.

EIRP Equivalent Isotropic Radiated Power.

FTP File Transfer Protocol.

GPU Graphics Processing Unit.

GSCM Geometry-Based Stochastic Channel Models.

HARQ Hybrid Automatic Repeat Request.

IoT Internet of Things.

IoU Intersection of Union.

IP Internet Protocol.

ISAC Integrated Sensing and Communication.

JSON JavaScript Object Notation.

KPIs Key Performance Indicators.

KPM Key Performance Metrics.

LEO Low Earth Orbit.

LiDAR Light Detection and Ranging.

LLM Large Lenguage Models.

LTE Long Term Evolution.

MAC Medium Access Control.

mAP mean Average Precision.

MIMO multiple-input multiple-output.

MPCs Multi-Path Components.

NATS Neural Autonomic Transport System.

NORI New Open RAN Interface.

NR New Radio.

NSA Non-Standalone.

NTN Non-Terrestrial Network.

OFDMA Orthogonal Frequency Division Multiple Access.

OS Operation System.

PID Proportional-Integral-Derivative.

PRB Physical Resource Block.

QoE Quality of Experience.

QoS Quality-of-Service.

R&D Research and Discovery.

RAM Random-Access Memory.

RAN Radio Access Network.

RC RAN Control.

RGB Red, Green, Blue.

RIC RAN Intelligent Controller.

RL Reinforcement Learning.

RR Round Robin.

RSS Received Signal Strength.

RSSI Received Signal Strength Indicator.

RT Ray Tracing.

RTF Real-Time Factor.

RTMP Real-Time Message Protocol.

RTSP Real-Time Streaming Protocol.

RTT Round Trip Time.

S2G Space-to-Ground.

S&R Search and Rescue.

SDN Software-Defined Networking.

SINR Signal-to-Interference-Plus-Noise Ratio.

SNR Signal-to-Noise Ratio.

SSIM Structural Similarity Index.

TCP Transmission Control Protocol.

TR Technical Report.

UAV Unmanned Aerial Vehicle.

UE User Equipment.

UE4 Unreal Engine 4.

UPA Uniform Planar Array.

V2X Vehicle-to-Everything.

VR/AR Virtual and Augmented Reality.

WCDMA Wideband Code Division Multiple Access.

WI Wireless InSite.

xApp eXtended Application.

XR Extended Reality.

YOLO You Only Look Once.

Table of Contents

1 INTRODUCTION

The increasing adoption of autonomous systems across various sectors, including transportation, public safety, manufacturing, and logistics, is transforming how connectivity and intelligence are integrated into physical environments. Drones, self-driving vehicles, and industrial robots are now routinely deployed to perform tasks remotely, often requiring real-time control, situational awareness, and coordination across distributed systems. These applications place stringent demands on the underlying communication infrastructure, including low latency, high-reliability data exchange, and seamless integration with Artificial Intelligence (AI) modules and sensor-rich environments (DAO et al., 2024).

Beyond robotics and automation, new classes of applications are emerging that rely heavily on advanced wireless capabilities at both the physical and application layers. Integrated Sensing and Communication (ISAC), Extended Reality (XR), and immersive technologies such as Virtual and Augmented Reality (VR/AR) are becoming integral to sectors such as healthcare, education, industrial training, and entertainment. These services often involve tight coupling between data processing, user interaction, and wireless transmission. Moreover, open concepts such as the Open Radio Access Network (RAN) have gained relevance in this new technological era. A key component is the RAN Intelligent Controller (RIC), a Software-Defined Networking (SDN)-based remote intelligent controller that monitors RAN metrics in real time and enables control of base station functionalities through service models, namely Key Performance Metrics (KPM) and RAN Control (RC). These control capabilities are realized by execution applications, or eXtended Application (xApp)s, which play a central role in the RIC ecosystem. In front of this, these new technologies are central to ongoing discussions in standardization bodies such as 3rd Generation Partnership Project (3GPP) (GIULIANO, 2024), which aim to support these use cases with stringent Quality-of-Service (QoS) requirements and new radio designs.

Although these services appear straightforward and reasonable to reproduce in real life, the telecommunications industry needs a way of reproducing these new features in a more real-compliant manner, which leads to the usage of testbeds and simulators (PEDERSEN et al., 2024). However, simulating such systems accurately—while accounting for wireless communication dynamics, real-time decision-making, and physical mobility—remains a major challenge. Traditional simulation approaches struggle to capture the interactions between network behavior, real network traffic, and dynamic 3D environments with sufficient fidelity and timing accuracy.

In this context, the concept of Digital Twin (DT) in 5G/6G environment has been born. Aiming to create high-fidelity virtual replicas of real-world systems for continuous monitoring, prediction, and control (ALKHATEEB; JIANG; CHARAN, 2023). While promising, building

DTs for complex wireless-enabled cyber-physical systems requires accurate modeling of radio propagation, traffic behavior, and physical interactions in a synchronized manner. This is particularly difficult in site-specific environments where Ray Tracing (RT) methods are used for channel modeling. Tools such as Sionna (HOYDIS et al., 2023) and Wireless InSite[1] provide powerful RT-based simulation capabilities, but they are computationally expensive, often taking seconds to minutes per snapshot to generate channels, which undermines real-time compliance for applications that require real information or low-latency control loops.

In parallel, still in DT implementation context, mobility simulators like AirSim (SHAH et al., 2017) (for drone simulation), CARLA (DOSOVITSKIY et al., 2017) (for autonomous driving), and recent satellite mobility and communication platform (KEMPTON; RIEDL, 2021) have become essential for creating realistic movement patterns in 3D environments. However, these frameworks are often developed in a non-compliant manner or lack standardized mechanisms for data exchange, making integration with communication simulators and emulators nontrivial. This interoperability challenge becomes even more pronounced when combining them with 5G network emulators that support advanced features (e.g., position-based beamforming (MIAO et al., 2019)). Commonly used platforms like 5G-Lena[2] and the ns-3 mmWave module[3] are valuable for evaluating system-level behaviors. Still, they are not inherently designed for real-time co-simulation or seamless integration with external mobility and application frameworks. They typically lack Application Programming Interface (API)s for time-synchronized data exchange with other simulators, limiting their ability to reproduce interactive, closed-loop scenarios.

These limitations underscore a critical gap: the lack of unified frameworks capable of combining site-specific or stochastic radio modeling, realistic mobility, and real-time communication emulation under a single synchronized environment. This gap motivates the development of novel hybrid co-simulation platforms capable of tightly integrating ray tracing-based or stochastic propagation, dynamic mobility engines, and low-latency network emulation with bidirectional data flows.

1.1 Related Work

To address the limitations of traditional network simulators, recent efforts (RAVIGLIONE et al., 2024; PEGURRI et al., 2025; PEGURRI et al., 2024) have proposed modular and hybrid simulation platforms capable of integrating heterogeneous tools across multiple domains (e.g., event-based and continuous simulation). These approaches typically build on the ns-3 (RILEY; HENDERSON, 2010) core for event scheduling and timing orchestration, while incorporating advanced components such as site-specific channel modeling via Sionna-RT and mobility frameworks like SUMO (BEHRISCH et al., 2011). However, because their primary focus is

[1] https://www.remcom.com/wireless-insite-propagation-software
[2] https://5g-lena.cttc.es/
[3] https://apps.nsnam.org/app/mmwave/

on achieving high-fidelity physical simulation, they lack full integration with ultra-realistic 3D environments, which is essential for enabling functionalities such as computer vision and real-time multimedia traffic generation (e.g., video streaming).

Other efforts (ZHAO et al., 2025) rely on real-world measurements to create accurate physical models for realistic testing and simulation, also integrating mobility simulators. While these approaches provide high fidelity in their specific domains, they lack generality when applied to diverse scenarios (e.g., rural, indoor, or large-scale heterogeneous environments). Similarly, work such as (JIA et al., 2021) addresses cyber-physical integration but still falls short in incorporating ultra-realistic 3D features, which are crucial for AI-driven use cases like object or people recognition. Finally, mission planning platforms such as (WALL, 2024) provide state-of-the-art capabilities for physical modeling and payload analysis, but their closed-source and commercial nature limits their adaptability and integration with broader research ecosystems.

In summary, although these platforms represent meaningful progress toward hybrid and high-fidelity simulation, they still lack the combination of open accessibility, ultra-realistic 3D environments, and flexible real-time data interaction required for next-generation use cases. These limitations motivate the development of a new framework capable of bridging these gaps: Communication Networks, Artificial Intelligence and Computer Vision with 3D Computer-generated Imagery (CAVIAR), developed in (KLAUTAU et al., 2021; BORGES et al., 2024), presents a comprehensive DT that enables the integration of mobility, 3D, communications, and AI modules to address challenges regarding dataset creation and DT context-based applications. However, both implementations lack (i) network-level type-of evaluations, since they don't rely on a network-level emulator, (ii) optimized orchestration for faster and seamless execution, and (iii) usage of ultra-realistic 3D for better computer vision evaluations.

To fill these gaps, this work presents the updated version of the legacy CAVIAR framework (BORGES et al., 2024): a flexible, open-source, hybrid co-simulator designed for real-time, end-to-end experimentation with 5G/6G and beyond networks. Previous implementation lacks photorealism, Key Performance Indicators (KPIs) evaluation and monitoring tool, and this current redesign extends CAVIAR to support seamless orchestration among modules, ultra-realistic 3D environments, enhanced by the usage of Unreal Engine 4 (SANDERS, 2016) main visual features (e.g., humans, cars, buildings), and network-level evaluations, now using the well-known ns-3 (RILEY; HENDERSON, 2010), addressing the interoperability and timing limitations observed in existing approaches. Current CAVIAR's architecture supports asynchronous execution, is hardware-compliant, and adopts a distributed, service-oriented design, enabling seamless integration of real-time traffic, interactive control, and visualization via a monitoring tool.

This work is validated through three representative use cases—rural animal monitoring, indoor human-robot coordination, and urban Search and Rescue (S&R) after a disaster—evaluating both resource utilization and network-level KPIs performance.

The main contributions of this work are summarized as follows:

- Design and implementation of a customizable, open-source[4] hybrid co-simulation framework that integrates mobility simulators, ray tracing tools, and a full-stack 5G/6G environment for ultra-realistic experimentation.
- Development of a Neural Autonomic Transport System (NATS)[5]-based orchestrator supporting asynchronous scheduling and sample-rate–aware coordination across heterogeneous modules.
- Integration of a real-time monitoring interface for time-aligned visualization and live interaction with the simulation loop.
- Experimental validation in three application scenarios, analyzing computer resources utilization and network-level KPIs metrics.

1.2 Scientific Outcomes

During the development of this dissertation, side research efforts were carried out focusing on the implementation of new channel models in the ns-3 and 5G-Lena environment, as well as the study of latency-sensitive functionalities in cloud-based network functions under the 5G standard topology. These efforts resulted in the following publications at national and international scientific venues, which form the basis and framework for this dissertation:

- **Flexible Channel Model Configuration for Scalable 5G-LENA Simulations** – Accepted at *The International Conference on ns-3 (ICNS3)*, 2025. This work extends 5G-Lena with additional channel models, enabling flexible large-scale 5G simulations and benchmarking their accuracy–cost trade-offs against the default 3GPP 38.901 model.
- **Assessing Power Allocation Efficiency for RAN in Cloud-based Systems for 5G Networks** – Accepted at the *XLIII Simpósio Brasileiro de Telecomunicações e Processamento de Sinais (SBrT)*, 2025. This work evaluates the impact of deploying a latency-sensitive power allocation function at the RAN, MEC, and cloud levels in 5G networks, analyzing channel capacity and RTT using ns-3 with the 5G-Lena module.
- **Enabling NS-3 Simulations Integrated with Latest Versions of Open RAN Near-RT RICs** – Accepted at the *XLIII Simpósio Brasileiro de Telecomunicações e Processamento de Sinais (SBrT)*, 2025. This work presents NORI, a modular interface integrating ns-3 with the Open RAN Near-RT RIC via the E2 interface, enabling realistic 5G simulations with metric collection and performance evaluation in disaggregated RAN environments.

[4] https://github.com/lasseufpa/caviar
[5] https://nats.io/

1.2.1 Additional Contributions

The author also contributed to the following interdisciplinary research:

- **Network Digital Twin for Route Optimization in 5G/B5G Transport Slicing with *What-If* Analysis** – Presented at the *IEEE International Conference on Communications (ICC)*, 2025. This work proposed a DT-based platform using GNNs for route optimization in 5G/B5G transport slicing.

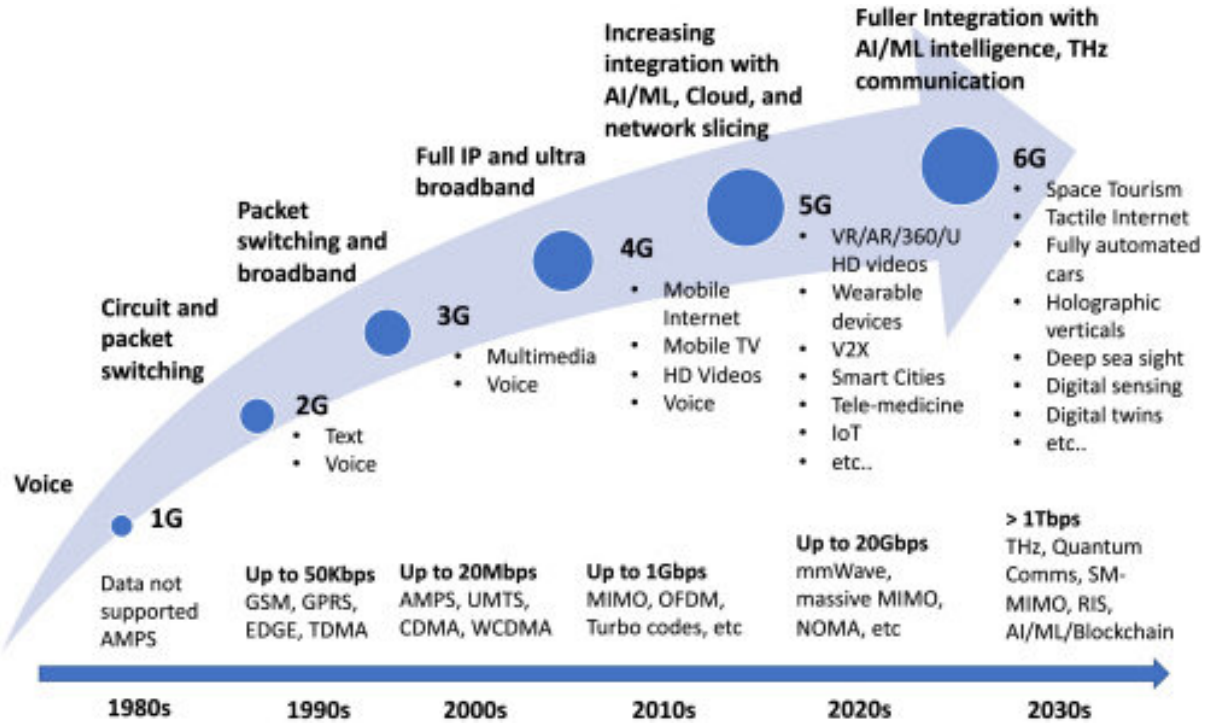
1.3 Dissertation Structure

The remainder of this dissertation is organized as follows: Chapter 2 presents a comprehensive overview of the evolution of simulation techniques in mobile networks, tracing their development from early analytical approaches to modern co-simulation approaches adopted in the 3GPP standardization process. Chapter 3 introduces the proposed CAVIAR's framework update, describing its architecture, synchronization model, and modules. Chapter 4 details the experimental evaluation through representative use cases. Chapter 5 presents KPIs and use case-specific results obtained via the monitoring system, along with the analysis of computational resource utilizations. Finally, Chapter 6 summarizes the main contributions of this work, discusses its limitations, and suggests possible directions for future research.

2 EVOLUTION OF MOBILE NETWORK SIMULATION FRAMEWORKS

The evolution of mobile networks has been characterized by continuous technological advancements, transitioning from basic voice-centric systems to highly sophisticated multi-service platforms. As illustrated in Fig. 1, the early generations (1G–3G) primarily emphasized voice and multimedia transmission, with limited focus on higher-layer protocol interactions and complex service requirements. In contrast, the 4G and 5G eras introduced a wide range of standardized features and applications, enabling diverse use cases that define modern communication systems. Specifically, in the context of 5G, novel paradigms such as multiple-input multiple-output (MIMO) and Vehicle-to-Everything (V2X) communications have gained significant relevance.

Figure 1 – Evolution of mobile networks and associated features across generations.



Source: (RANAWEERA et al., 2022)

Advancements beyond 5G (5G-Advanced) toward 6G envision even more dynamic and heterogeneous use cases, with the concept of DT becoming a cornerstone for next-generation networks. Digital twins are expected to provide high-fidelity representations of physical environments and enable context-aware applications. Furthermore, the integration of AI into mobile networks has emerged as a transformative paradigm, supporting functionalities ranging from advanced sensing (ZHANG et al., 2022) to optimized air-interface operations, such as super

massive MIMO beamforming (HOYDIS et al., 2020), and resource allocation in Open-RAN scenarios.

Within this evolutionary landscape, simulation tools have played a pivotal role in enabling research, standardization, and prototyping. Initially designed to assess physical-layer developments, simulators have progressively incorporated complex system-level capabilities aligned with 3GPP specifications. This progression has led to the emergence of modern co-simulation frameworks, capable of integrating communication, sensing, mobility, and intelligent decision-making into unified virtual environments. The subsequent sections provide an overview of this evolution, tracing the development of mobile network simulators from their early inception to the co-simulation tools shaping 5G-Advanced and 6G research.

2.1 Early Simulation in Mobile Networks

At the initial stages of mobile network research, simulation efforts emerged as analytical and highly simplified tools designed primarily to evaluate link-level or system-level results. These early models were intended mainly for network planning and physical-layer assessment rather than providing a comprehensive, end-to-end system evaluation. In other words, simulators at this stage focused on characterizing fundamental physical behaviors and their influence on higher-layer protocols, such as Medium Access Control (MAC) or application-layer mechanisms, without incorporating packet-level information typical of more advanced network-level simulations.

During this period, *Monte Carlo* techniques (MOONEY, 1997) were commonly employed to estimate key performance metrics, such as error rates and throughput distributions under varying channel conditions. Within this context, link-level and early system-level simulators (e.g., (MANN, 2003; WIESER; PŠENÁK, 2005)) became popular for analyzing modulation schemes, coding strategies, and diversity techniques, offering valuable insights into Bit Error Rate (BER) performance for technologies such as Wideband Code Division Multiple Access (WCDMA).

Beyond the trends in this simulation context, several distinct methods for simulation and reproduction have emerged. During the development of early simulators, various techniques were introduced to make simulations more feasible and standardized. One prominent approach is event-driven simulation, which discretizes the simulation process into a series of time-ordered events. This technique ensures scalability, synchronization, and efficiency, and remains widely used today in network simulators such as ns-3 and OMNeT++ (VARGA, 2010).

Additionally, methods for evaluating diversity in telecommunications have also evolved. Two common evaluation strategies are drop-based and site-specific evaluations. The former typically involves random sampling and is not tied to any particular location, offering a generalized assessment. In contrast, site-specific evaluations are more deterministic and tailored to particular

environments, producing results that reflect the unique characteristics of a given site.

Despite their relevance, these early simulation frameworks exhibited significant limitations. They lacked support for dynamic mobility, realistic network traffic integration, and multi-layer evaluation, restricting their applicability for investigating the increasingly complex behaviors of next-generation wireless systems. Moreover, no widely adopted standard simulation platforms were available during this period, leaving simulation efforts largely constrained to niche research applications.

2.2 Simulation in the 3GPP Standardization Context

With the emergence of the 3GPP as the leading standardization body for mobile networks, simulation methodologies within its ecosystem have undergone substantial evolution, which initially relied on drop-based statistical modeling at the link level. These approaches progressively incorporated system-level abstractions to enable large-scale performance evaluations, particularly during the development of 3GPP Releases 8–10 for 4G/Long Term Evolution (LTE). In this context, platforms such as ns-2, ns-3, and OMNeT++ (augmented with modules like SimuLTE (VIRDIS; STEA; NARDINI, 2014)) provided researchers with standardized environments for performance analysis and protocol evaluation.

For LTE-Advanced, additional complexities such as multi-element antenna configurations (e.g., Uniform Planar Array (UPA)) drove the introduction of spatial channel modeling (LIU et al., 2018), as formalized in 3GPP Technical Report (TR) 36.873. These advancements allowed simulators to capture spatial correlation and beamforming effects more accurately, reflecting the increasing sophistication of mobile network architectures.

More recent developments, under the 5G context, emphasize spatially consistent site-specific modeling, leveraging 3D maps and ray tracing to achieve highly realistic propagation environments, also using tools such as (ZUBOW et al., 2024), an ns-3 Sionna site-specific module. In parallel, a new generation of AI-native simulation tools has emerged (e.g., GenAi (ZOU et al., 2025)), offering dynamic model creation and automation, but raising questions about generalization and potential overfitting to specific environments.

Under the 6G investigations, the promise is to have next-generation models for some ultra-realistic 3D environments evaluations with AI-native features for air interface use cases, ISAC integration, and employments with XR. Besides, Open-RAN, which starts being investigated in 5G and is more consolidated in 6G, with some use cases such as resource allocation using xApps (QAZZAZ et al., 2024). Some tools, like GenOnet (REZAZADEH et al., 2024), for Large Lenguage Models (LLM)-based simulations or co-simulation tools will definitely be special for 6G use cases.

Table 1 summarizes this evolution, highlighting the transition from early statistical ap-

proaches to advanced site-specific and AI-assisted methodologies. While generative AI and other data-driven techniques are expected to play a significant role in future simulation ecosystems, in this work, ray tracing remains a cornerstone for site-specific analysis due to its physical interpretability and standardization readiness. In the context of this work, it is focused on ray tracing-based methodologies, acknowledging the growing relevance of generative and AI-based approaches.

Table 1 – Evolution of simulation techniques in the 3GPP ecosystem

Era / Release	Simulation Focus	Key Tools and Models	Notable Features
4G / LTE (Rel. 8–10)	Link/system-level hybrid modeling	ns-3,2, SimuLTE, Drop-based	Link adaptation, Hybrid Automatic Repeat Request (HARQ)/Channel Quality Indicator (CQI) modeling, scalable multi-cell simulations
LTE-Advanced (Rel. 11–13)	3D spatially consistent MIMO modeling	3GPP TR 36.873, open-source 3D Geometry-Based Stochastic Channel Models (GSCM) implementations	Elevation-aware channels, improved beamforming support, spatial consistency
5G / NR (Rel. 15–17)	Site-specific and multi-numerology simulations	3GPP TR 38.901, ns-3 NR module, Simu5G	Ray tracing-ready channels, Non-Terrestrial Network (NTN), V2X sidelink, slicing investigations
6G (Rel. 18+)	AI-native, and ultra-realistic 3D-based environments	GenOnet, and co-simulation tools, Open-RAN	AI air-interface modeling, ISAC integration, realistic 3D scenario for XR, resource allocation

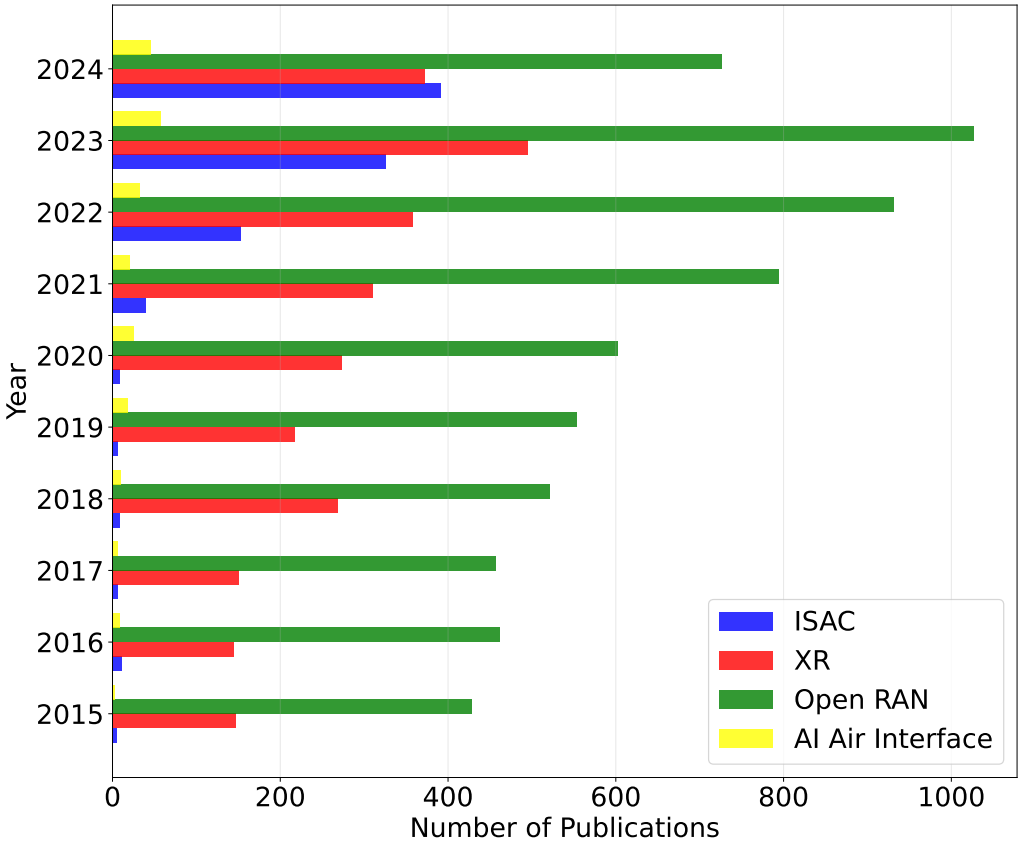
This trajectory illustrates several key trends: (*i*) the evolution from purely simple link/system-level simulation models toward ultra-realistic network-level simulation models, site-specific approaches; (*ii*) increasing use of 3D spatial consistency and ray tracing for enhanced realism; and (*iii*) the integration of AI within the physical-aligned simulation paradigm.

Nevertheless, despite the significant progress in simulation methodologies, current 3GPP-aligned tools still face limitations when addressing problems that require the joint modeling of ultra-realistic 3D environments and AI-native grown use cases. This is particularly evident in emerging areas such as ISAC, discussed in (PENG et al., 2025a), and AI applications, where the interplay between sensing, communication, and adaptive learning remains only partially captured by existing frameworks.

2.3 Emergence of Co-Simulation Approaches in the Context of 6G

As previously discussed, despite the significant advancements in 3GPP-aligned simulators, many emerging features currently under investigation—such as ISAC and Open RAN—are not fully supported by conventional tools like ns-3 and OMNeT++. For instance, as illustrated in Fig. 2, the number of academic publications in areas such as ISAC has increased by more than 50% between 2015 and 2023, reflecting the growing relevance of these technologies in the research landscape. In this context, traditional simulation frameworks have proven insufficient for accurately reproducing such specialized use cases, particularly those with strong industrial applicability, thereby motivating the exploration of alternative approaches.

Figure 2 – Trends in 5G–6G research for selected use cases and areas of investigation.



Source: The author (2025)

As discussed in (MANALASTAS et al., 2024), existing simulators often struggle to replicate advanced use cases, primarily due to their limited applicability in industry-oriented scenarios and their inability to generate practical, context-rich results. In contrast, DT-enabled co-simulation approaches are emerging as a promising alternative, offering the potential to combine high-fidelity modeling with real-world relevance for both research and industrial deployment. Table 2 presents an overview of known platforms developed for evaluating telecommunication

parameters. Within the scope of this work, DTs are considered as co-simulators. The table highlights that only DT-based approaches successfully achieve high accuracy in both practical relevance and industrial utility. In contrast, standalone simulators fall short in these aspects, with a high impact in Research and Discovery (R&D) and medium results in practicality and industrial-based applications. Even though not discussed within the scope of this work, testbeds and analytical models are not pivotal for industry evaluations, since they do not provide high-accuracy results.

Table 2 – Comparison of traditional software for multiple telecommunications-aligned evaluations, based on Fig. 1 of (MANALASTAS et al., 2024).

Software	Analytical Models	Testbeds	Simulators	Digital Twins
Practicality of results	Low	High	Medium	High
Utility to R&D	Medium	Medium	High	Low
Utility to industry	Low	Medium	Medium	High
Risk to live networks	Low	Medium	Low	Medium
Required Resources	Low	High	Medium	Medium

Thereby, co-simulation frameworks enable industry-based evaluations and practicality of results by integrating multiple specialized simulators, each contributing unique capabilities to reproduce complex, multi-domain scenarios. For example, tools such as SUMO can be used for vehicular mobility modeling, while Sionna provides site-specific channel modeling based on ray tracing, together supporting the development of full-fidelity DTs without incurring excessive computational overhead. These characteristics make co-simulation an attractive approach for addressing the limitations of traditional tools, paving the way for more accurate and industry-relevant 6G simulations.

The latest updates to the CAVIAR framework aim to incorporate these co-simulation principles, enabling the integration of diverse modules for communication, mobility, and environment modeling. These updates, detailed in the following chapter, position CAVIAR as a powerful framework for exploring next-generation use cases in 5G-Advanced and 6G/B6G contexts.

3 CAVIAR: A FRAMEWORK FOR 6G HYBRID SIMULATIONS

In the context of the growing demand for co-simulation frameworks that integrate mobility, communications, and AI techniques, this work presents the most up-to-date version of the CAVIAR framework. The initial version of CAVIAR was introduced in (KLAUTAU et al., 2021), focusing on providing a tool for generating datasets that combined mobility and communication features, typically leveraging SUMO and Wireless InSite, with an emphasis on MIMO channel generation. A subsequent update (BORGES et al., 2024) introduced new features and a codebase restructuring to enhance usability and modularity.

Over the past three years, with the advancement of DT concepts as discussed in Chapter 2, CAVIAR has evolved beyond its original purpose. While its early iterations primarily supported dataset creation, the framework has been enhanced to enable real-time representation of physical entities—commonly referred to as physical twins in the DT paradigm—within a digital environment. In other words, CAVIAR was now capable of functioning as a fully integrated, in-loop virtual twin, bridging the gap between simulation and real-time evaluations.

Despite these advancements, prior versions of CAVIAR still exhibited limitations. In particular, the lack of robust synchronization mechanisms adversely affected the sample rate, introducing timing inconsistencies across the co-simulation steps. Additionally, its reliance on 3D assets generated using the basic OpenStreetMap (VARGAS-MUNOZ et al., 2020) tool restricted its potential for computer vision–oriented applications. Furthermore, its evaluation of KPIs remained simplistic, relying on basic thresholds rather than comprehensive network-level performance metrics.

In the following sections, an overview of the previous CAVIAR implementation is provided (Section 3.1). Additionally, the updates and enhancements introduced in this new version of CAVIAR are described. Section 3.2 details the (re)implementation of the 3D, mobility, communication, and AI modules. Section 3.3 presents the redesigned core architecture, highlighting its improved synchronization mechanisms and modular structure. Section 3.4 discusses additional features integrated into the framework to extend its applicability for complex, real-time co-simulation scenarios.

3.1 Previous CAVIAR

Before presenting the updates and showing what the current CAVIAR implementation is capable of, it is essential to review the performance of the previous version, with emphasis on (BORGES et al., 2024). As illustrated in Fig. 3, the last CAVIAR integrated three distinct

modules: (i) communications, (ii) mobility and 3D, and (iii) AI. These modules relied on an orchestrator for message exchange, implemented using NATS. NATS provides the infrastructure for data exchange in the form of segmented messages and operates over Transmission Control Protocol (TCP)/Internet Protocol (IP) following a publish/subscribe paradigm via text-based protocol.

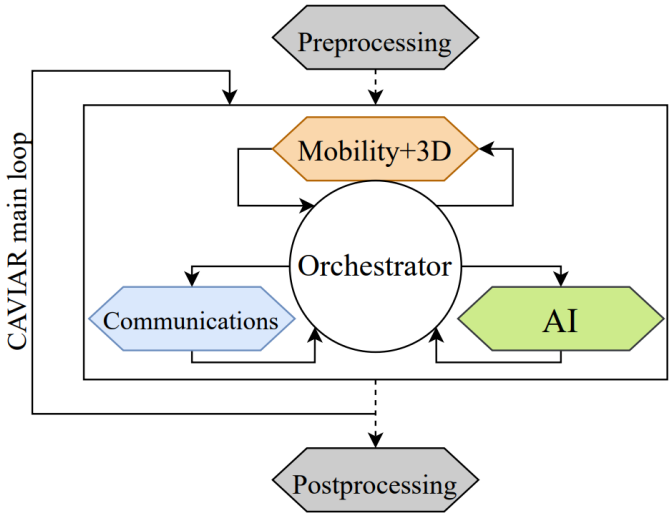


Figure 3 – Overview of the previous CAVIAR full integration.

Source: (BORGES et al., 2024)

Regarding the supported modules, the mobility and 3D models were merged for simplicity. However, these models were simplistic, as no photorealistic assets or enhancements were incorporated (e.g., realistic human representations), which limited the fidelity of the DT in this context. In the communications module, Sionna RT was employed for site-specific channel calculation, while the AI module integrated a decision tree and You Only Look Once (YOLO). Nonetheless, the communications module lacked mechanisms to accelerate ray tracing efficiency for faster digital twin execution and did not provide built-in support for advanced ray tracers to generate channel datasets. Furthermore, realistic data evaluation at the network level was not implemented, reducing the reliability of KPIs assessments.

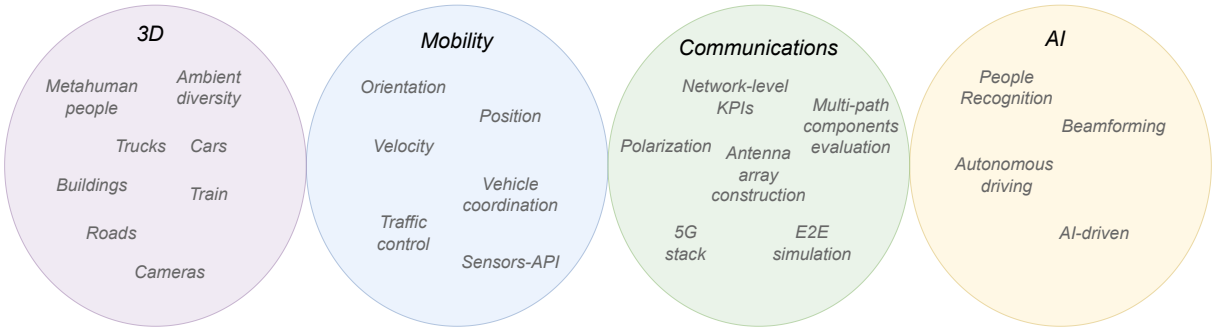
In summary, the previous CAVIAR lacked built-in support for ultra-realistic scenes, efficient and high-fidelity RT execution for digital twin applications, and realistic data for KPIs-oriented network-level evaluation. The next sections present the updates and new implementations that address these limitations.

3.2 CAVIAR Modules: Division and Features

In this updated version of CAVIAR, the framework is organized into distinct modules, as illustrated in Fig. 4. Unlike the previous version, the 3D simulation component is no longer tightly coupled to the mobility module. Instead, it has been decoupled and redesigned as an

independent module with its own set of functionalities. This change represents a key enhancement, as the co-simulation tool no longer depends exclusively on a mobility model that embeds 3D representations. Consequently, each module can now evolve independently, improving flexibility and enabling the simulation of more complex, heterogeneous scenarios.

Figure 4 – Module division within the CAVIAR framework.



Source: Author

To summarize, as depicted in Fig. 4, the main modules of CAVIAR and their features are condensed as follows:

- **3D Module:** Provides realistic representations of people, vehicles, buildings, and road networks to support diverse simulation scenarios, including V2X and cellular use cases. It also includes camera integration for immersive visualization and enables the creation of complex environments with varying levels of ambient detail.
- **Mobility Module:** Handles dynamic features such as position, orientation, and velocity for all simulated entities, supporting multi-user and multi-vehicle scenarios. It also incorporates vehicle coordination, traffic management, and sensor APIs for multisensor data simulation.
- **Communications Module:** Manages wireless network simulation, including support for 5G/6G features, MIMO channel modeling with site-specific or stochastic calculation, and configurable KPIs evaluation for end-to-end performance analysis.
- **AI Module:** Enables integration of AI-driven functionalities, such as adaptive control, online learning, and data-driven decision-making for enhanced simulation intelligence.

The following sections describe in detail each of these modules, discussing their features, updates, and role within the updated CAVIAR framework.

3.2.1 3D: Unreal Engine 4

In the context of CAVIAR, Unreal Engine 4 (UE4)[1] serves as the sole 3D simulation platform currently supported. UE4 is a versatile, high-fidelity engine widely adopted in both the gaming and simulation industries for creating immersive and interactive virtual environments. Its advanced rendering pipeline and physics engine enable the modeling of complex objects and interactions with remarkable realism. By integrating UE4 into CAVIAR, the framework can deploy rich, physically accurate environments that support multiple simulation use cases, including vehicle scenarios, pedestrian mobility studies, and diverse urban or rural layouts. This modular integration allows users to build and manipulate environments in a seamless way, independent of the underlying mobility models.

Beyond static 3D visualization, UE4 enables dynamic interaction with simulated entities, supporting multi-user simulations, vehicle traffic coordination, and pedestrian behavior modeling. These features are critical for end-to-end scenarios that require a combination of communications, mobility, and context-aware applications, such as testing computer vision algorithms or evaluating network performance in realistic visual settings. Fig. 5 provides a visual example of the 3D rendered features in UE4 as part of the CAVIAR framework, illustrating the range of assets and environmental diversity achievable with this module, such as: (i) metahumans for human virtual representation, (ii) common assets (e.g., cars, trucks, etc.), (iii) ambient diversity for multiple scenarios evaluation, (iv) cameras for multi-modal information, and (v) buildings and roads for higher quality urban representation.

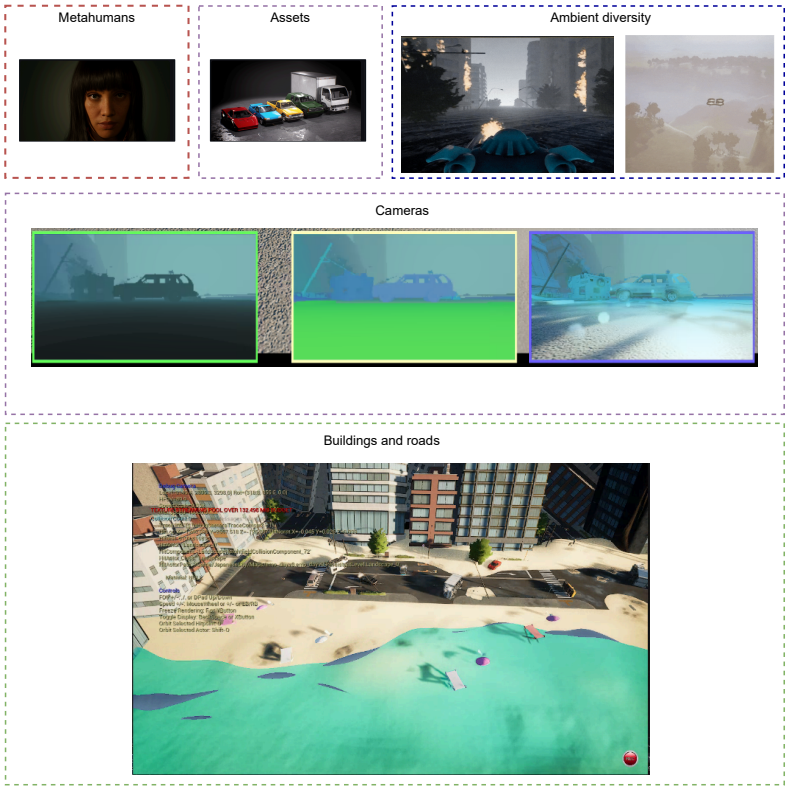
3.2.2 Mobility: AirSim

For mobility simulation, CAVIAR integrates Microsoft AirSim (SHAH et al., 2017) as its primary platform for vehicular and aerial mobility features. AirSim is an open-source simulator that provides high-fidelity environments for autonomous vehicles, offering a server-based communication interface that supports cars, drones, and other robotic agents. Its control mechanisms are secured using Proportional-Integral-Derivative (PID)-based controllers, enabling precise and stable movement of vehicles and drones within the simulated space. This makes AirSim particularly suitable for scenarios involving complex vehicle dynamics, multi-agent navigation, and aerial robotics experiments.

The integration between the 3D environment (rendered in UE4) and AirSim within CAVIAR is achieved via a lightweight broker interface. This broker establishes communication through a set of API calls using `Python`, enabling seamless interaction between modules. For each interaction, simple functions (e.g., `getLidarData()`, `simSetVehiclePose()`) are invoked, facilitating real-time data exchange and control of mobility entities. This modular design allows users to easily retrieve sensor data, update vehicle states, or adjust trajectories

[1] https://www.unrealengine.com/

Figure 5 – Features of a rendered 3D in Unreal Engine 4 within the CAVIAR framework.



Source: The author (2025)

without disrupting the co-simulation loop. Table 3 summarizes the key features supported by AirSim within the CAVIAR framework.

Although AirSim provides a wide range of features for mobility simulation, some limitations remain in its current integration with CAVIAR. Notably, it does not include built-in traffic management capabilities for large-scale vehicular coordination, which constrains its use in complex multi-vehicle traffic scenarios. Additionally, the current implementation is limited to Unreal Engine 4, as Unreal Engine 5 integration is not yet supported. Addressing these gaps may require support platforms, such as CARLA, which offers advanced traffic management and broader support for urban driving simulations, but is not yet integrated into CAVIAR.

3.2.3 Communications: Wireless InSite, Sionna-RT, and ns-3

The communications modules in the updated CAVIAR framework integrate three tools: Wireless InSite for high-fidelity site-specific channel modeling, Sionna-RT for real-time ray tracing channel simulation, and ns-3 for 5G network-level simulation. This combination allows CAVIAR to support both physical-layer and network-layer analysis, offering flexibility for end-to-end evaluation of complex communication scenarios. In particular, ns-3 introduces support for standardized 5G components (e.g., numerologies, scheduling policies) and configurable stochastic channel models, while Wireless InSite and Sionna-RT extend CAVIAR's capabilities

Table 3 – Supported features of AirSim within CAVIAR.

Feature	**Support**
Car simulation	✓
Drone simulation	✓
LIDAR data collection	✓
Camera sensor (RGB/Depth)	✓
GPS sensor	✓
PID-based control	✓
Unreal Engine 4 interface integration	✓
Unreal Engine 5 interface integration	✗
Multi-agent simulation	✓
Custom vehicle models	✓
Real-time API access	✓
Built-in traffic management	✗

for physically accurate, site-specific propagation modeling.

3.2.3.1 Sionna-RT

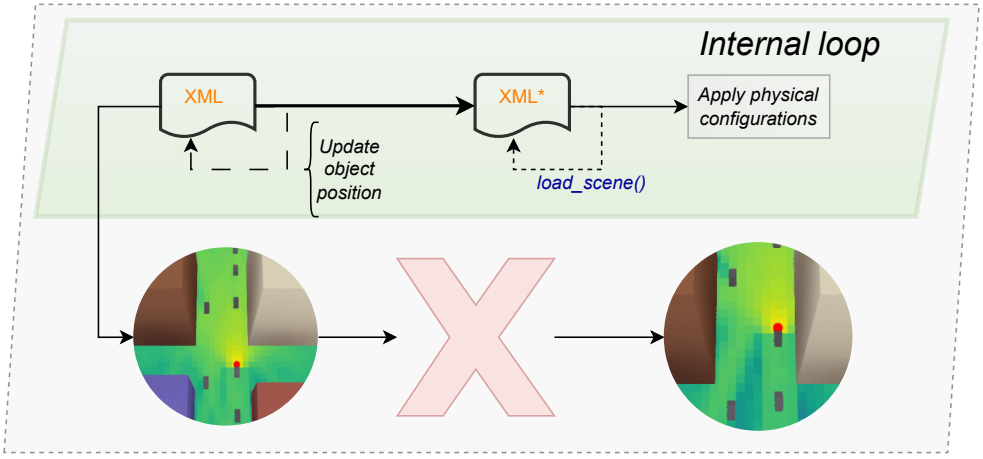
For fast site-specific channel modeling, CAVIAR integrates Sionna-RT, an open-source ray tracing library designed for accurate, differentiable wireless channel simulation. Built on top of the TensorFlow (DEVELOPERS, 2022) framework, Sionna-RT provides a flexible environment for modeling multipath propagation, antenna configurations, and material interactions in complex 3D scenarios. Its integration enables CAVIAR to support high-fidelity physical-layer evaluations, including MIMO channel characterization, Doppler-aware mobility, and polarization effects, making it a powerful tool for digital twin and advanced wireless research.

In previous CAVIAR implementations, the integration with Sionna-RT was functional but computationally inefficient for in-loop simulations. Specifically, updating the position of mobile nodes within the ray tracing environment required rewriting and reloading the entire scene definition file (Mitsuba XML format) for each new position. As illustrated in Fig. 6a, this approach involved modifying the receiver coordinates (update objects positions) within the scenario file, reloading the entire scene (i.e., all the assets that compose the simulated scenario) using the `load_scene()` function, and reapplying all relevant configurations (e.g., antenna array parameters, polarization, material properties). While technically feasible, this procedure introduced substantial computational overhead, often resulting in channel calculations taking several seconds or even minutes per snapshot, even for relatively simple environments.

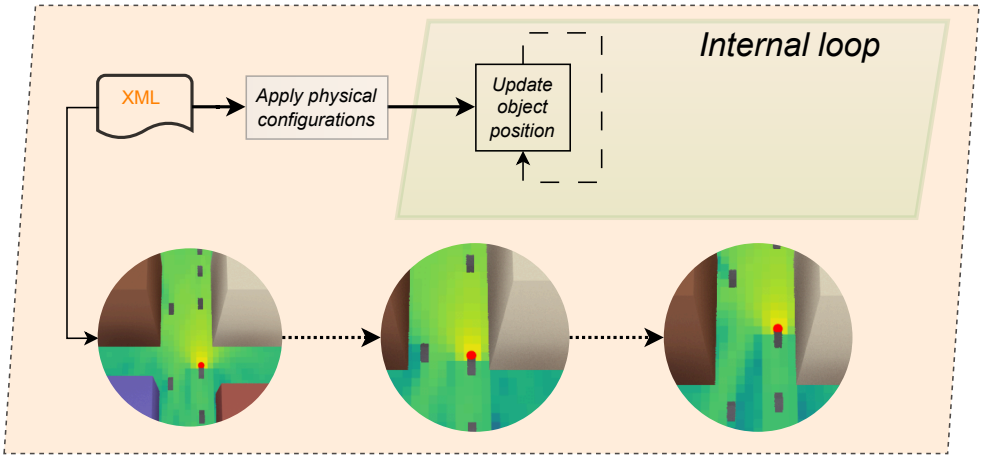
The updated version of CAVIAR mitigates this bottleneck by leveraging the new functionalities introduced in the latest Sionna-RT release (AOUDIA et al., 2025). Rather than reloading the entire scene, the updated integration loads the environment once at initialization and subse-

Figure 6 – Workflow for mobility update in Sionna-RT.

(a) Workflow for mobility update in Sionna-RT within CAVIAR's previous implementation.



(b) Workflow for mobility update in Sionna-RT within CAVIAR's newest implementation.



Source: The author (2025)

quently manipulates receiver attributes (e.g., position, orientation, velocity) directly through the Sionna API in the internal loop, as despite in Fig. 6b. This significantly reduces the computational overhead for per-step updates and enables smoother mobility modeling, in real-time or near-real-time simulation loops. In comparison, as a general overview of Fig. 6, the previous implementation introduced significant computational overhead, forcing the simulation to skip certain intermediate position updates in order to align with higher-granularity position data. In contrast, the updated implementation minimizes this overhead, thereby reducing the need for such skips and allowing smoother, more continuous position tracking. This improvement aligns with the modular life-cycle management discussed in Section 3.3, ensuring that the communication module can operate efficiently within the broader co-simulation architecture.

Despite these advancements, some limitations persist. Sionna-RT's computational cost still scales with scene complexity, which can pose challenges for large, multi-node environments or when high temporal resolution is required. This will be discussed in Section 3.4, where this issue is bypassed, using some RT acceleration techniques.

3.2.3.2 Wireless Insite

In the previous version of the CAVIAR framework, integration with the Wireless InSite (WI) ray tracing simulator was not supported. This new implementation introduces a complete workflow to enable the use of WI as a site-specific channel simulator within the CAVIAR environment. The integration process follows a structured logic composed of two main stages: simulation configuration and run-time execution, which will be discussed in Section 3.3.1. Despite its advantages in providing high-fidelity ray tracing datasets, this integration is not suitable for real-time interaction due to the computational overhead of WI. Still, it is particularly valuable for applications involving dataset generation, channel model validation, and deterministic channel simulation.

The WI-based simulation is configured entirely through the graphical interface of WI, available exclusively for the Windows operating system. In the scenario configuration step, the 3D meshes that compose the environment are imported into the software, and several configurations must be applied to define the simulation behavior. These include assigning appropriate physical properties to each imported object, configuring the desired transmission parameters, defining the placement, radiation pattern, and polarization of both transmitter and receiver antennas, and setting the propagation model along with the output types to be generated. As an output of this configuration, two key files are produced: `scene.xml`, which contains the general scene configuration parameters, and `nodes.txrx`, which stores the initial positions of transmitter and receiver nodes. After the creation of these files, CAVIAR can use them to perform site-specific calculations in its framework environment.

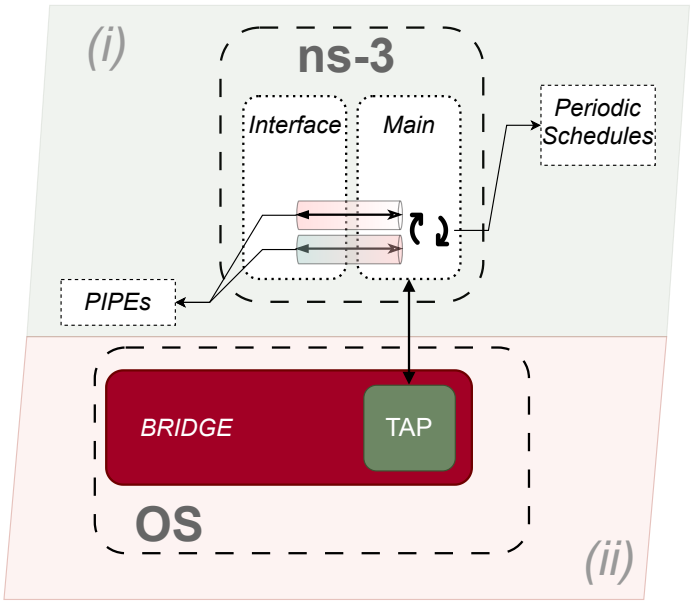
3.2.3.3 ns-3

Unlike the previously discussed communication modules, ns-3 plays a distinct role as an event-based network simulator. As previously introduced, ns-3 provides a flexible environment for simulating network behavior across multiple nodes, incorporating standardized mobility models and protocol stacks. However, by itself, ns-3 does not provide a fully implemented 5G architecture for end-to-end network-level simulations. To address this gap, CAVIAR integrates the 5G-Lena module, which extends ns-3 with a 5G Non-Standalone (NSA) core implementation, enabling consistent and realistic modeling of 5G network behavior. The implementation of ns-3 within CAVIAR is divided into two main components: (i) the interface between CAVIAR's co-simulation core and the ns-3 simulation itself, and (ii) implementation of a ns-3 base code for real-traffic integration.

As illustrated in Fig. 7, part *(i)*, the interface employs a Linux-based pipe file as a low-level, raw I/O queue. This mechanism enables two-way communication between the ns-3-caviar interface and the ns-3 main code, allowing the event-driven ns-3 simulation to remain synchronized with external co-simulation events. Specifically, ns-3 reads data from the pipe in a blocking, synchronized manner, waiting for new inputs before proceeding. These inputs are

scheduled using ns-3's native `Simulator::Schedule()` functionality, enabling customized scheduling of external updates (e.g., refreshing a physical channel state every 1 ms). While pipes behave like files, they operate as lightweight communication channels, minimizing I/O overhead compared to conventional file operations.

Figure 7 – ns-3 module implementation within the CAVIAR environment.



Source: The author (2025)

The second component of this integration leverages ns-3's `TapBridge` module to enable the use of real traffic within the simulated network. As illustrated in Fig. 7, part *(ii)*, this module creates a TAP (network TUN/TAP) interface, relying on a bridge network, in promiscuous mode, acting as a sink for external application traffic, inside the user's Operation System (OS). This allows CAVIAR to incorporate real-world network applications, such as File Transfer Protocol (FTP), video streaming protocols (e.g., Real-Time Streaming Protocol (RTSP), Real-Time Message Protocol (RTMP), Dynamic Addaptative Streaming over HTTP (DASH)), and others, directly into the ns-3 environment. With these features, ns-3 effectively functions as a network emulator, handling routing and protocol behaviors internally while supporting live traffic exchange. Importantly, integrating real applications imposes strict constraints on ns-3's scheduler, which must remain aligned with wall-clock time for accurate emulation. While these real-time scheduling mechanisms for ns-3 emulation are outside the scope of this dissertation, further details can be found in (OTTENS et al., 2025).

3.2.4 AI: YOLO, Decision Tree

Similar to the mobility and communications modules, the AI module presented in this section is a reimplementation, with specific enhancements, of the one developed in the previous version of CAVIAR. The first integrated technique is YOLO (DIWAN; ANIRUDH;

TEMBHURNE, 2023), a state-of-the-art deep learning model for real-time object detection that combines high accuracy with low inference latency. In the CAVIAR context, YOLO enables the identification of people and assets, supporting in-loop interactions with other modules. As illustrated in Fig. 8, YOLO interacts with the virtual environment by processing semantic data, here understood as image-based information, to correctly label the assets, such as people, trucks, cars, etc. This capability is crucial for monitoring specific items, as investigated in use cases, described in Chapter 4.

Figure 8 – A YOLO example of object detection under an ultra-realistic CAVIAR scenario.

Source: The author (2025)

The second approach is the Decision Tree model, initially developed in the last CAVIAR version and preserved in the current release. While no additional enhancements were introduced, its implementation now supports online deployment, enabling dynamic inference during runtime and seamless integration with other modules.

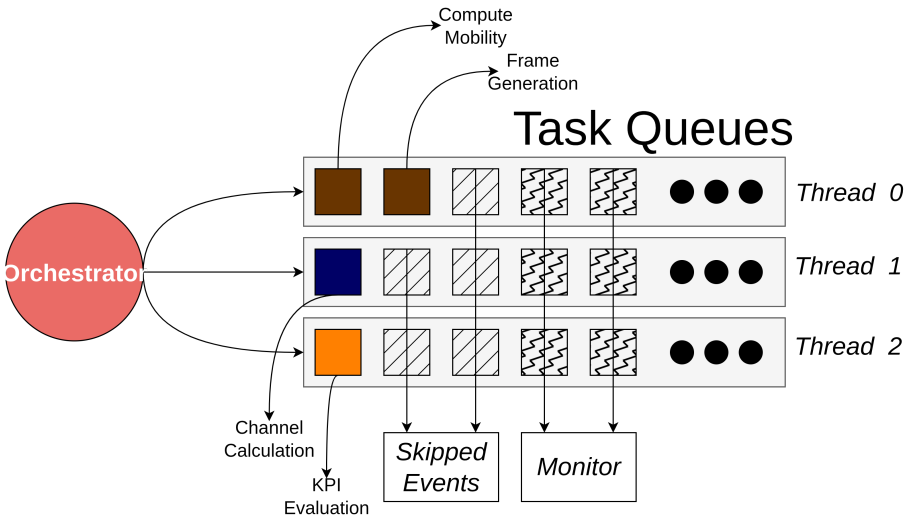
3.3 CAVIAR's Core Architecture and Orchestration

The implementation of individual modules represents only one part of the co-simulation framework. To achieve a fully integrated and real-time capable environment, CAVIAR requires an orchestration layer responsible for coordinating the execution of these heterogeneous components and managing their data exchange. This orchestration is performed by the CAVIAR core, which acts as a mediator between modules, ensuring interoperability, synchronization, and efficient information flow. In other words, the core enables the independent modules (3D, mobility, communications, and AI) to operate cohesively within a unified co-simulation loop.

The CAVIAR core architecture relies on two fundamental technologies: the NATS messaging system and Python's `asyncio` library, detailed in (FOWLER, 2022), combined with multiple threads. NATS is a high-performance, lightweight messaging platform designed for distributed systems. Within CAVIAR, it serves as the backbone for inter-module communication, providing a publish-subscribe model that allows modules to exchange data in a decoupled and scalable manner. This design ensures that modules can be dynamically added, removed, or replaced without disrupting the overall system.

To handle the concurrency demands of co-simulation, the core leverages Python's `asyncio` library for asynchronous event-driven execution. This approach enables the orchestrator to schedule and execute tasks concurrently, in case some module executes multiple tasks (e.g., the AirSim module used to compute mobility and generate frames via 3D camera). By avoiding blocking operations and combining them with threads, this library improves responsiveness. It allows CAVIAR to maintain real-time performance, even when dealing with computationally intensive tasks like ray tracing or live traffic processing in 3D environments. Illustrating this behavior visually, Fig. 9, the orchestrator manages the `asyncio` events within each thread-queue (i.e., task queues) of the co-simulation environment, where each event represents a task to be executed by a module (e.g., compute mobility, frame generation, etc.). Control events, such as monitoring parameters, are also scheduled in task queues. In case of an asynchronous execution, the events may also be skipped, which is synchronized by the orchestrator.

Figure 9 – Events representation in CAVIAR's task queues for multiple threads. Brown boxes represent events of the first thread, blue of the second thread, and orange of the third thread. Special events, such as *skipping* and *monitor* are represented by hatch and zig-zag line markers, respectively.

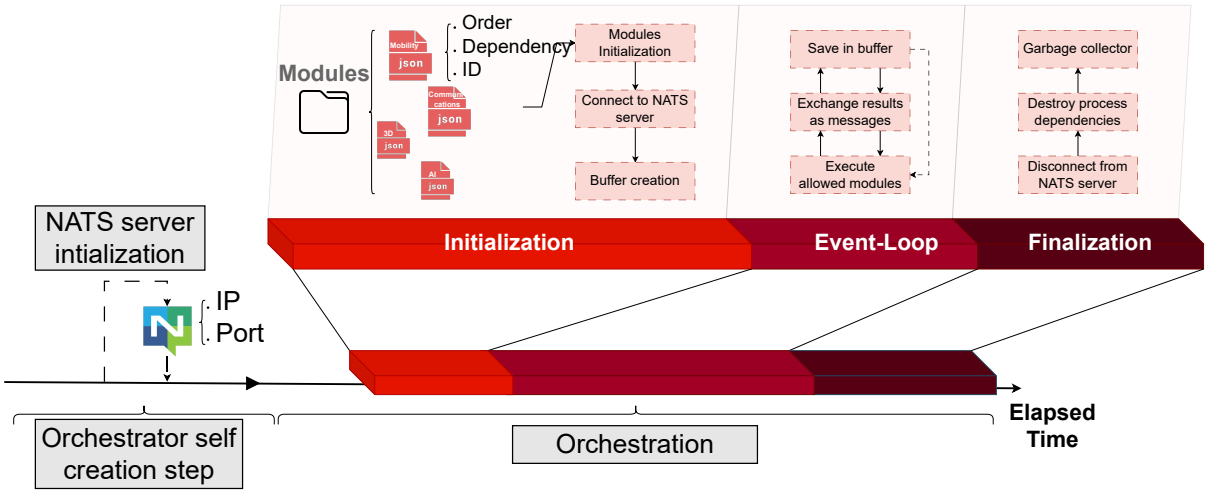


Source: The author (2025)

3.3.1 Life-cycle Management

The life-cycle management process in CAVIAR begins with the self-initialization of the orchestrator environment. As illustrated in Fig. 10, this process starts with launching a NATS server at a specified IP address and port. Acting as a message broker, NATS establishes a publish-subscribe architecture, on top of TCP, that underpins all inter-module communication, ensuring that data flows between modules remain decoupled and fault-tolerant.

Figure 10 – CAVIAR's orchestrator overview over the elapsed time.



Source: The author (2025)

Once the NATS server is running, the orchestrator proceeds with the orchestration step, where it initializes the available modules. In the initialization process, the orchestrator automatically detects modules located in the CAVIAR *Modules* directory, where each module is associated with a configuration file written in JavaScript Object Notation (JSON) format. These configuration files define essential initialization parameters, including: (i) *ID*, which classifies the module as belonging to one of the categories (mobility, communications, 3D, or AI); (ii) *Order*, which determines the initialization sequence in cases where a module depends on others being initialized first; and (iii) *Dependency*, which specifies the data requirements for the module to operate. For example, the Sionna-RT communication module depends on receiver position data from the mobility module to perform site-specific channel calculations. The module can also be independent of any other module (e.g., the 3D module is not reliant on any other module). After reading these configurations, the orchestrator dynamically connects each module to a dedicated NATS subject following the standardized naming convention `"kernel.<ID>.<module_name>"`, ensuring consistent routing of messages across the system.

Since different modules can have different sample rates, the orchestrator allocates an asynchronous ring buffer for each module that prevents information overlaps, due to large data orders from one producer module to the other that is handling such data. These buffers are

governed by Python's `asyncio` event loop, enabling non-blocking read and write operations. This design prevents simulation stalls, as modules are not required to wait for buffer updates to proceed. Besides, if a buffer is empty, the module simply skips its action for that cycle. In contrast, if the buffer is full, the oldest entry is dropped in favor of the most recent data, guaranteeing that modules always operate on up-to-date information. Besides, the buffer is simple and can be customized.

After completing the initialization phase, the orchestrator transitions to the event loop stage, which governs the continuous execution of the co-simulation. Conceptually, this event loop can be viewed as a persistent `while(true)` cycle that runs until a termination trigger occurs (e.g., simulation completion, critical error, or user interruption). Within this loop, the orchestrator performs three primary functions, which are summarized here and described in detail in Section 3.3.2.

The orchestrator monitors the availability of modules within the co-simulation environment and determines their execution eligibility. A module is allowed to execute if: (i) new messages are available in its buffer, enabling it to consume fresh data, or (ii) it has no dependencies, allowing it to run independently of other components. The orchestrator enforces each module's configured execution rate. This rate control may involve skipping specific execution steps for modules with higher sample rates, since it's not capable of executing faster due to Central Processing Unit (CPU) constraints. I.e., the orchestrator accounts for the global sample rate limitation imposed by the host CPU. When a module requests an execution frequency that exceeds the processing capacity of the orchestrator, the module cannot operate at its intended timing and is instead aligned to the orchestrator's available execution rate. This behavior prevents undefined timing errors but may result in event skipping or timing drifts.

Summarizing, modules with slower or medium rates can typically execute each step as scheduled, though medium-rate modules may approach the system's processing limits—leaving them susceptible to event skipping under conditions such as thermal throttling or CPU load fluctuations. In contrast, modules with faster sample rates inevitably experience skipped events, as the orchestrator cannot sustain their requested execution frequency.

After executing the eligible modules while respecting the orchestrator's processing constraints, the system proceeds to handle inter-module communication. At this stage, the orchestrator exchanges messages through the previously initialized NATS server, ensuring that updated data are published to the appropriate subjects and stored in each module's dedicated circular buffer for subsequent processing.

Finally, upon simulation termination, the orchestrator enters the finalization phase. In this stage, the orchestrator performs a controlled shutdown of all active components to prevent the occurrence of zombie processes, which may negatively impact the OS, as discussed in (MUSA, 2024). The first step involves disconnecting all modules from the NATS server in the reverse order of their initialization, ensuring that dependent modules are terminated only after their

dependencies are safely released. For example, the last initialized module is the first to be disconnected, minimizing the risk of unexpected behaviors.

Following this disconnection process, the orchestrator cleans up all process dependencies created within each module's runtime environment. For instance, in the AirSim module, separate subprocesses are spawned for drone movement and video streaming handling, which are explicitly terminated during this step. Finally, a garbage collection routine is executed to remove any residual objects and free allocated resources. Once all modules have been finalized, the orchestrator itself is automatically destroyed, leveraging Python's standard behavior for object lifecycle management.

3.3.2 Event Loop Coordination

As summarized in the previous section, the event loop constitutes the core execution cycle of CAVIAR, constrained by the orchestrator's sample rate. Within this stage, messages are exchanged between modules and stored in their respective circular buffers. However, beyond simple message passing, this stage also governs the coordination of module execution across the co-simulation environment. Although not explicitly depicted in Fig. 10, the event loop is responsible for managing when and how each module executes within the overall system timeline.

The coordination strategy implemented in this stage can be categorized into two distinct execution modes: *asynchronous* and *synchronous* execution, which will be discussed in the next sections.

3.3.2.1 Asynchronous Execution

Asynchronous co-simulation in the context of CAVIAR refers to a non-blocking execution model in which modules operate without directly halting the execution of others. In this design, no module can block the event loop due to the unavailability of data or other dependencies; instead, modules execute whenever their required inputs are available. This approach allows multiple modules to perform computations within the same event cycle, ensuring a seamless and responsive co-simulation. Such non-blocking behavior is particularly critical for use cases requiring real-time interaction, as it tries to minimize latency caused by inter-module dependencies.

The workflow for asynchronous execution is illustrated in Fig. 11. As shown, the initialization process occurs outside the event loop, where the orchestrator sequentially initializes each module according to the configuration parameters described in Section 3.3.1. Once the event loop begins, only modules without dependencies (e.g., `Module_1`) are eligible to execute. After completing its first execution, `Module_1` produces an output message, which the orchestrator immediately forwards to its dependent module (`Module_2`), storing it in the corresponding

buffer. In the next execution cycle, both `Module_1` and `Module_2` run in parallel, producing outputs that are forwarded to their respective destinations (`Module_2` and `Module_3`). By the third execution cycle, all three modules (`Module_1`, `Module_2`, and `Module_3`) operate in parallel, and this iterative process continues until the simulation concludes.

While this asynchronous model ensures fluid integration and supports real-time applications, it also introduces a propagation delay in data availability across dependent modules. For instance, in a co-simulation involving vehicle mobility, channel modeling, and end-to-end network simulation, the results of the final module (e.g., network-level KPIs computation) will only become available after the preceding modules (mobility and channel) have completed their respective cycles. This means that the first meaningful KPIs outputs may only be obtained after several event loop iterations, introducing a slight delay in the overall information pipeline.

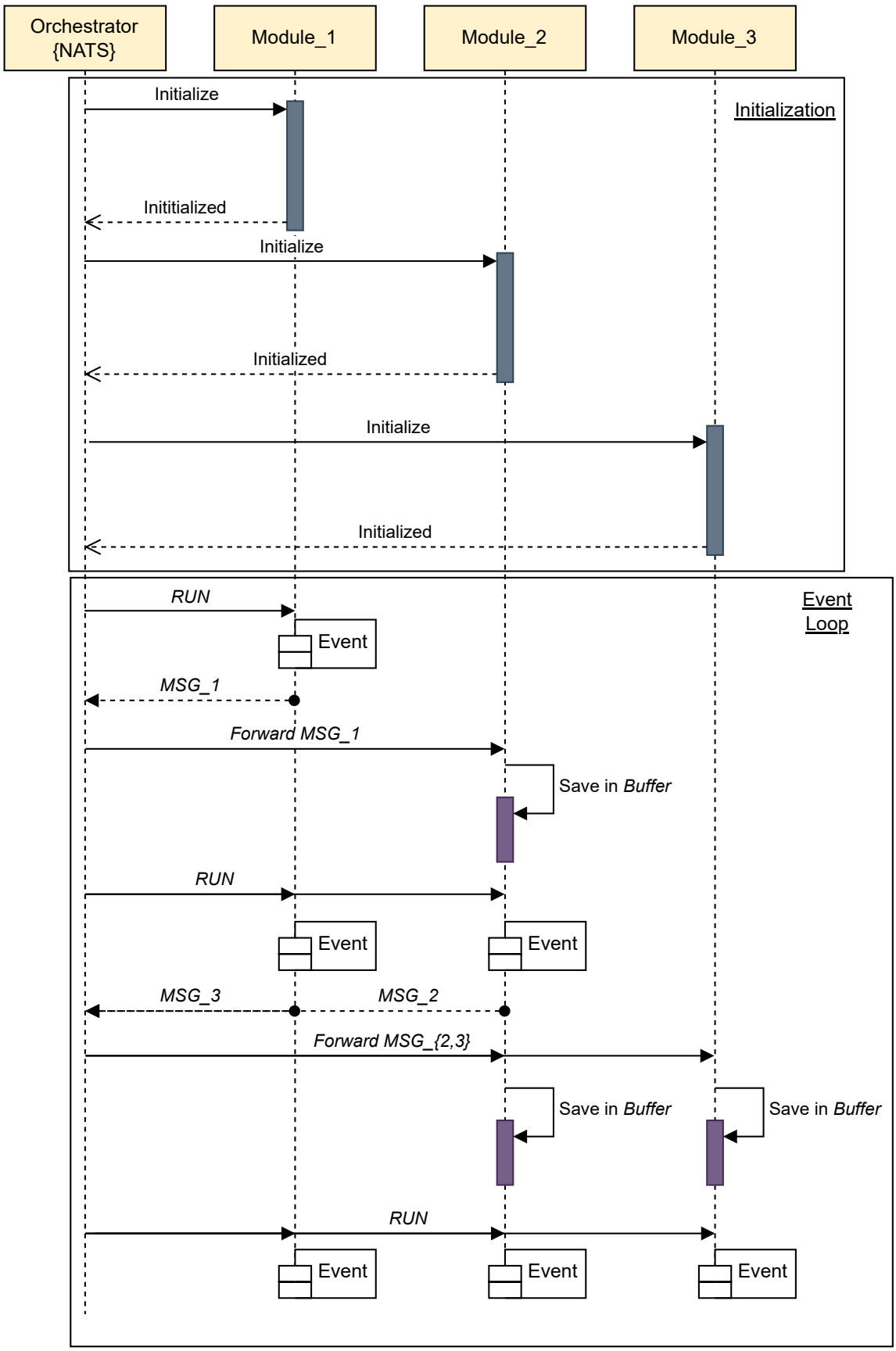
3.3.2.2 Synchronous Execution

In contrast to the asynchronous approach described in Section 3.3.2.1, synchronous execution operates in a blocking mode. In this scheme, the orchestrator enforces strict sequential execution, meaning that the processing of one module can block the execution of others, including those without explicit dependencies. This approach guarantees that each module executes exactly once per event cycle, producing outputs that are temporally aligned. As a result, the outputs of all modules within a cycle—such as those from the first, second, and third modules (for the case of three modules)—are synchronized to the same simulation step, ensuring strong temporal consistency across the co-simulation.

This behavior is illustrated in Fig. 12. As in the asynchronous model, the initialization process remains unchanged; however, the event loop now adheres to a strictly ordered execution scheme. During each cycle, `Module_1` executes first, generating its output, which is then forwarded to `Module_2`. Crucially, while `Module_2` executes, `Module_1` pauses, preventing it from starting a new iteration. This process continues sequentially until the final module (in this case, `Module_3`) completes its execution. Only after the last module has finished does the event loop restart, allowing `Module_1` to begin its next cycle. This strict ordering propagates through all modules, ensuring that their outputs remain fully synchronized.

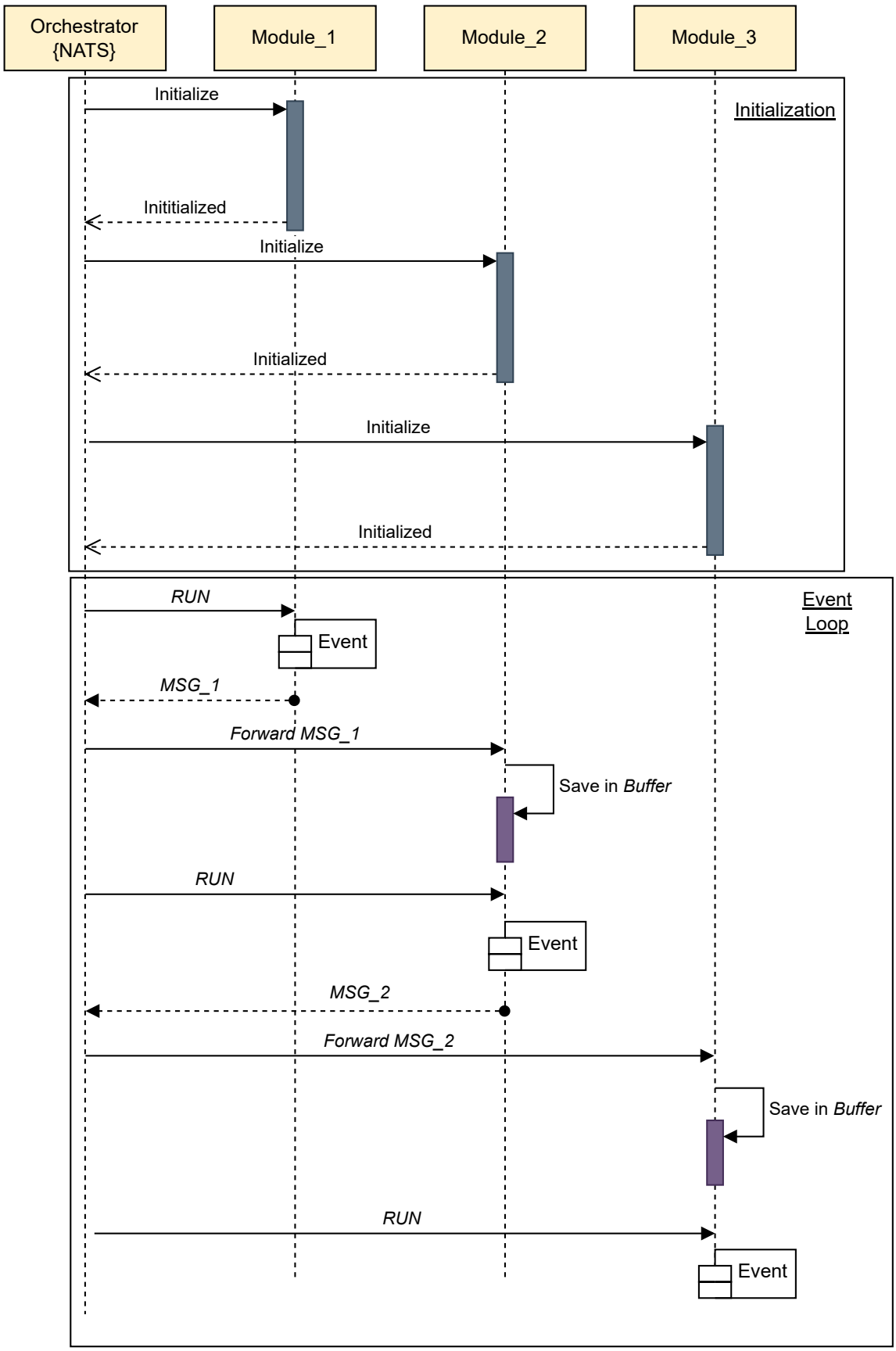
The primary benefit of this approach is the production of highly consistent and temporally aligned outputs. Because data are generated in a strictly ordered manner, the resulting time series are more accurate, making synchronous execution particularly suitable for scenarios requiring high-precision data collection—such as generating datasets for training or evaluating time-sensitive models. This advantage will be further discussed in Section 3.4. However, this strict synchronization introduces significant computational overhead. Modules are forced to pause and resume frequently, resulting in increased latency and reduced responsiveness. Consequently, while synchronous execution improves temporal alignment, it is impractical for real-time applications that require low-latency interaction.

Figure 11 – CAVIAR's execution scheduling over an asynchronous type of co-simulation.



Source: The author (2025)

Figure 12 – CAVIAR's execution scheduling over a synchronous type of co-simulation.



Source: The author (2025)

3.4 Side Features

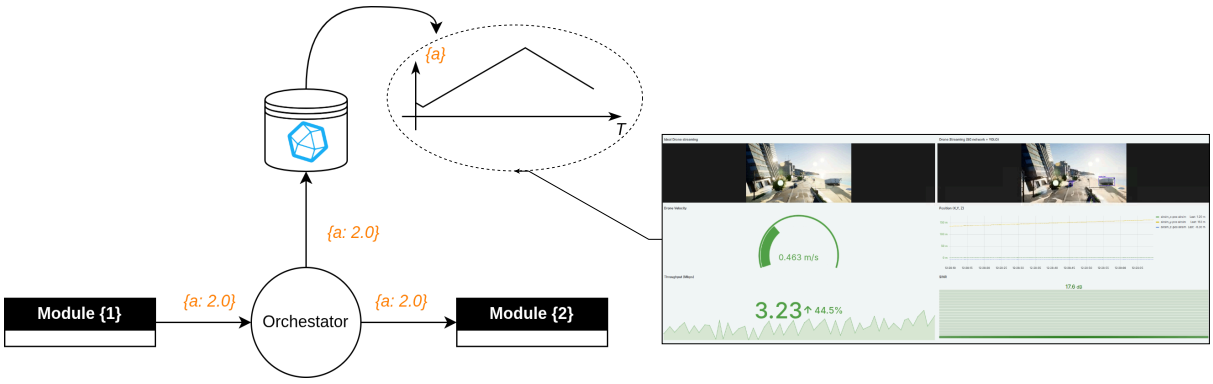
Beyond the newest CAVIAR core implementation, in this version of CAVIAR, some side features were implemented. In this section, the features of monitoring in Section 3.4.1 and the RT acceleration techniques in Section 3.4.2 will be demonstrated.

3.4.1 Monitor Tool

The monitoring tool developed for CAVIAR provides a live-interaction utility for saving and visualizing key data and metrics generated by each module during co-simulation. This tool enables users to monitor the system's performance in real time and supports the permanent storage of simulation data for later analysis. The monitoring system integrates two open-source platforms: InfluxDB[2] and Grafana[3]. InfluxDB serves as a time-series database designed for high-performance storage and retrieval of timestamped data, enabling efficient logging of module outputs and system metrics. Grafana, on the other hand, provides an interactive and highly customizable dashboard interface that allows users to visualize data streams in real time, configure personalized views, and create alerts or performance indicators tailored to specific use cases.

Both InfluxDB and Grafana are deployed as Docker services within a dedicated network, ensuring modularity, isolation, and streamlined communication between the components. As illustrated in Fig. 13, the CAVIAR core architecture captures relevant information from each module by duplicating its output data and forwarding it to InfluxDB. Grafana then queries this database to retrieve and render the data dynamically on the user dashboard, enabling continuous monitoring of system behavior throughout the simulation.

Figure 13 – Monitor tool overview.



Source: The author (2025)

[2] https://www.influxdata.com/
[3] https://grafana.com/

3.4.2 RT Augmentation and Scenario Simplification

As introduced in Section 3.2.3.1, ray tracing (RT) channel calculations in CAVIAR involve computationally intensive operations, often resulting in high execution times that limit their applicability for real-time co-simulations. To address this challenge, CAVIAR incorporates two complementary techniques: *RT augmentation* and *scenario simplification*, as proposed in (MODESTO et al., 2025). While RT augmentation operates dynamically during the event loop to accelerate channel updates, scenario simplification is applied during the initialization phase, focusing on reducing the geometric complexity of the simulated environment. In the current implementation, scenario simplification is exclusively applied to the Sionna module.

The scenario simplification process occurs during the initialization of the communication module and is performed by reducing the number of faces in each mesh that composes the simulated scene. This mesh decimation process decreases the computational burden of the ray tracing solver by simplifying the underlying geometry while preserving the essential propagation characteristics of the environment. Importantly, CAVIAR delegates control over this simplification to the user, who specifies a reduction rate (e.g., a percentage of face count reduction), enabling a trade-off between accuracy and computational efficiency based on the specific requirements of the experiment.

As illustrated in Fig. 14, RT augmentation, on the other hand, is performed during the event loop and leverages interpolation between two previously computed site-specific Multi-Path Components (MPCs) (which is the variable **H** in the representative figure). Using buffered information from two distinct channel computations, the method generates additional intermediate channels that align with the update rate of the end-to-end simulation. The number of interpolated channels, K, is determined by:

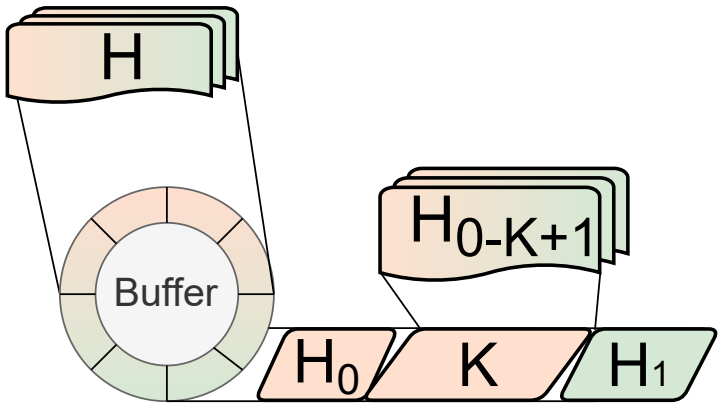
$$K = \left\lceil \frac{\Delta T}{t} \right\rceil, \tag{3.1}$$

where t represents the desired channel update interval in the end-to-end simulation, and ΔT denotes the time difference between two consecutive site-specific channel calculations. This approach significantly reduces the need for frequent full ray tracing computations, improving performance without compromising the temporal granularity required by real-time applications.

3.5 CAVIAR Overview

In summary, the latest updates to the CAVIAR framework primarily focus on enhancing its core infrastructure and refining the implementation of its constituent modules. This redesigned version incorporates new capabilities for user interaction through an integrated monitoring system, as well as RT acceleration techniques to mitigate the computational burden of site-specific channel calculations. These improvements aim to strengthen the framework's real-time performance and extend its applicability to more demanding use cases.

Figure 14 – Augmentation process under the buffered MPCs information.



Source: The author (2025)

Significant advancements have been made in the communications module, including the integration of full 5G end-to-end network simulation using ns-3 and 5G-Lena, as well as support for Wireless InSite to enable more accurate site-specific channel modeling. While these may appear as incremental enhancements, they substantially broaden the scope of CAVIAR by enabling advanced co-simulation scenarios and supporting the evaluation of new technologies under realistic conditions. For instance, the updated framework now provides the necessary infrastructure for experimenting with high-priority use cases such as ISAC and NTN, facilitating their testing in real-time with its deployment within a network-level simulation environment, summarized in Table 4. Specifically, as described in (PENG et al., 2025b), CAVIAR supports multimodal ISAC, with emphasis on Relay-based multimodal ISAC and Interaction-based Multimodal ISAC. At the same time, features introduced in the previous version—such as mobility enhancements, MIMO support, and AI-driven functionalities—have been preserved and incorporated into the updated architecture, ensuring continuity while improving system robustness

Table 4 – Comparison of CAVIAR features support between versions.

Native Features	**Previous CAVIAR**		**This CAVIAR**	
	Support	Built-in	Support	Built-in
End-to-end network-level simulations	✗	✗	✓	✓
Real-time use cases (e.g., video streaming)	✗	✗	✓	✓
Site-specific datasets creation	✓	✓	✓	✓
Flexible mobility entities (cars, drones, etc.)	✓	✓	✓	✓
KPIs evaluations	✓	✗	✓	✓
User interaction via monitor tool	✗	✗	✓	✓
Time series datasets	✓	✗	✓	✓
Multi-sensor support	✓	✗	✓	✓

A comparative summary of the features supported in the previous and updated CAVIAR versions, aligned with 5G-to-6G advancements as defined in Release 19 (LIN, 2024; LIN, 2025), is presented in Table 5. Since ns-3 is a fully integrated and modular code base, it allows merging different modules under the same code line. This capability enables the integration of NTN features within the CAVIAR use cases, including those related to Internet of Things (IoT), as ns-3 facilitates seamless interoperability between these modules. Beyond this, ISAC (in our context, multimodal), which was not supported in the previous version, is now incorporated. With the inclusion of ultra-realistic 3D environments, LiDAR, and semantic outputs, these features can now be well-aligned with the 3D environment representation of a potential DT.

Table 5 – Comparison of Release 19 feature support between CAVIAR versions.

Feature (Rel. 19)	**Previous CAVIAR**		**This CAVIAR**	
	Support	Built-in	Support	Built-in
RP-234007: NR MIMO	✓	✓	✓	✓
RP-234036: NR mobility enhancements	✓	✓	✓	✓
RP-234080: XR	✓	✗	✓	✓
*RP-234078: NTN for NR	✗	✗	✓	✓
**RP-234077: NTN for IoT	✗	✗	✓	✗
RP-234039: AI for NR air interface	✓	✓	✓	✓
RP-234054: Enhancements for AI for NG-RAN	✓	✓	✓	✓
RP-234055: AI for mobility in NR	✓	✓	✓	✓
***RP-234069: ISAC for NR	✓	✗	✓	✓

**Note:* Support for NTN, using (SANDRI et al., 2023), for New Radio (NR) has been newly incorporated.
***Note:* Support for IoT, using (REYNDERS; WANG; POLLIN, 2018).
****Note:* New CAVIAR supports ISAC with built-in features.

In the next Chapter, some use cases that are aligned with the most up-to-date 3GPP concepts, using the new CAVIAR features and support, will be presented.

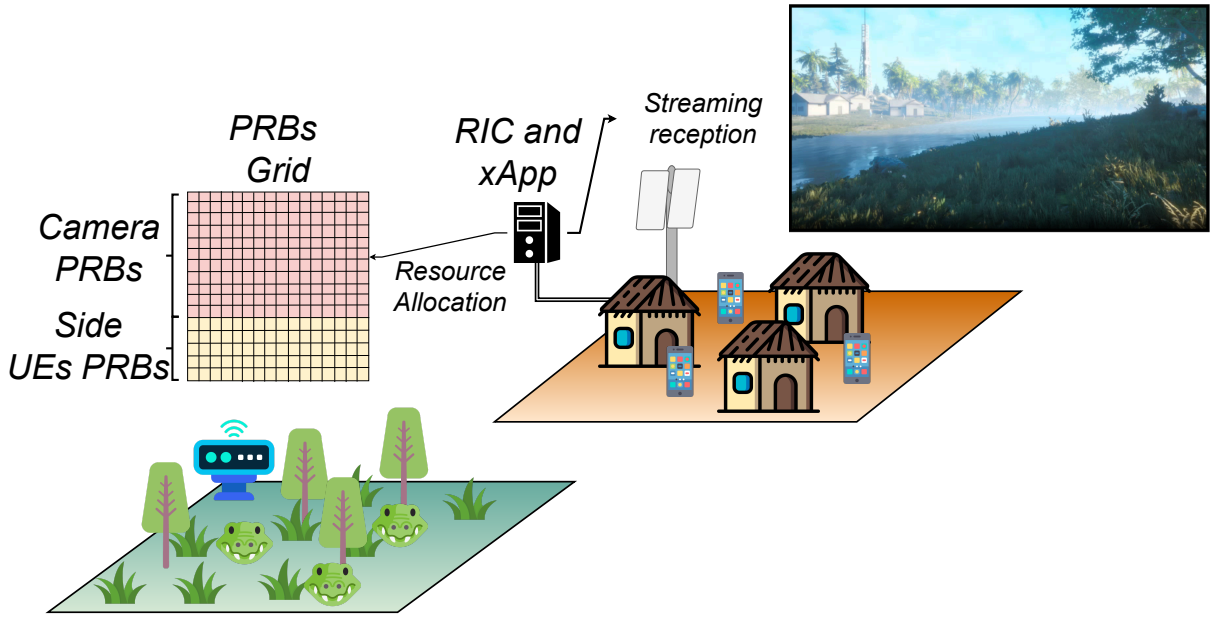
4 USE CASES

In this Chapter, the three representative use cases will be shown: (i) rural, (ii) urban, (iii) indoor. All the explanation of these use cases can be found at Github[1] for further implementations.

4.1 Rural - Animal Monitoring

The rural animal monitoring use case is designed to demonstrate how network resources can be allocated dynamically in a network environment, focusing on the Physical Resource Block (PRB)s available within the RAN. This use case involves four User Equipment (UE)s, one of which transmits live video of wildlife (e.g., crocodiles) and the others act as side UEs, consuming network resources and generating interference (intercell interference). As illustrated in Fig. 15, all communications pass through the RAN and reach the core network, where the video-streaming, which will serve as monitoring visualization for the user, server performs post-processing, and the RIC manages the PRB allocation at the RAN MAC layer.

Figure 15 – Animal monitoring use case overview.



Source: The author (2025)

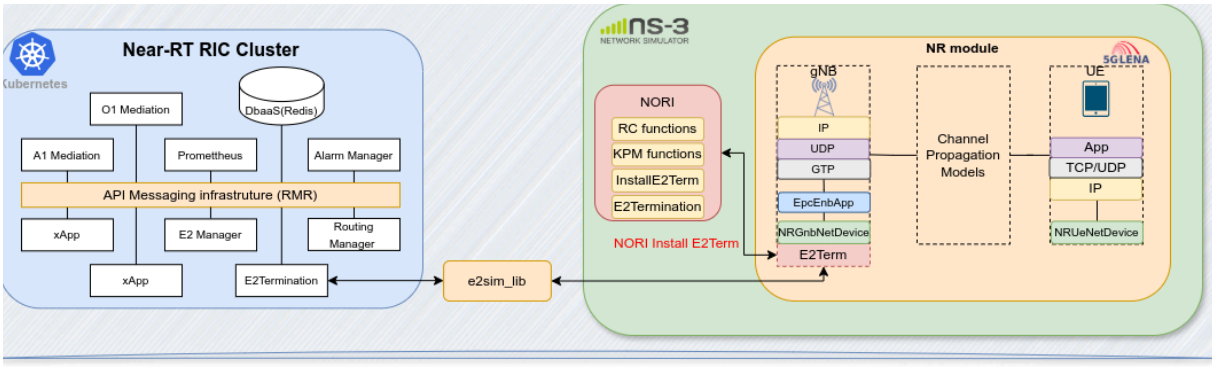
To integrate this use case, the RIC and the latest version of Open RAN module, the New Open RAN Interface (NORI) module[2], is employed, which implements the E2 termination. The E2 interface facilitates communication between the RAN and the RIC, allowing for the flow of

[1] https://github.com/lasseufpa/caviar
[2] https://github.com/lasseufpa/nori

information necessary for real-time decision-making. The NORI module enables the application of two primary services from the RIC: (i) the KPM service, and (ii) the RC service. The first service provides monitoring of key RAN metrics for training AI models and other analytical purposes, while the second allows the RIC to make informed decisions and take actions within the RAN, using special indication messages that are transmitted via the E2Sim interface[3].

The communication flow and interaction between the RIC cluster, including entities such as the xApp and the ns-3 simulation environment, are shown in Fig. 16. This diagram illustrates how the NORI module supports RC and KPM functionalities within the CAVIAR framework, enabling a seamless integration of the simulation environment with real-time control and monitoring. It is important to note that all these technologies are not a new module, but work inside the ns-3 module as a side tool.

Figure 16 – General architecture of the integration between the ns-3 environment and NORI's module.



Source: The author (2025)

The deployment of the xApp occurs within the RIC cluster, as previously described. The xApp is responsible for managing the physical resources available in the RAN. The algorithm used by the xApp to manage PRB allocation is shown in Algorithm 1. This implementation is based on a simplified version of (NAHUM et al., 2023), where a Reinforcement Learning (RL) Orthogonal Frequency Division Multiple Access (OFDMA)-based approach is used, excluding buffer delays and only considering throughput, with a minimum required rate of resources to the camera user.

In practice, the xApp operates as follows: when the co-simulation event loop begins, it establishes a connection with the RAN and retrieves KPM metrics for each slice. In this context, slices represent groups of devices that share common 5G-level requirements. For this use case, the camera stream operates on an exclusive slice, while the other three UEs share the remaining network resources. Using the KPM metrics as input, the xApp performs an online RL interaction to calculate the maximum, minimum, and dedicated number of PRBs for each slice. These

[3] https://github.com/wineslab/o-ran-e2sim

allocations are updated iteratively, allowing the xApp to refine its decision-making based on real-time observations.

Once the allocation decisions are made, the xApp sends an indication message with the updated PRB rates to the RAN through the RIC using the E2 termination interface. Upon receiving this message, the RAN either updates its allocation or maintains the previous configuration if no changes are required. All these operations are performed within the ns-3 simulation environment, ensuring consistency with the underlying network emulation.

Algorithm 1: xApp connects to RAN and performs PRB scheduling during the event-loop

Input: KPM metrics from UEs for each slice
Output: PRB allocation rate (dedicated, maximum, and minimum) for each slice
Connect to the RIC;
while *True* **do**
  Obtain KPM metrics from UEs for each slice;
  Define $S = [s_1, s_2, \ldots, s_n]$;
  // s_i stores the dedicated, maximum, and minimum PRB rates for slice i
  Calculate the PRB allocation rate for each slice, assuming throughput metric;
  for *each slice* $s \in S$ **do**
    Calculate action vector $s\{dedicated, maximum, minimum\}$;
    Update values in S;
  end
  Send updated PRB allocations to the RAN, via RIC;
  if *RIC response indicates different rates from the previous indication* **then**
    Adjust the PRB rates according to the RIC response;
  end
end

To provide a comparison with standard non-Open RAN configurations, the proposed RL-based scheduling approach was evaluated against a simple OFDMA Round Robin (RR) allocation scheme. In the latter case, PRBs are distributed equally among users, relying only on buffer status and throughput information from each UE. As previously discussed, the RL approach enforces a minimum allocation of 70% of PRBs to the camera UE. This constraint is necessary to ensure that the video quality requirements of the camera stream are met. For the side UEs, the maximum throughput is configured as a Constant Bit Rate (CBR) of 40 kbps, while the camera UE can achieve up to 10 Mbps with an average of 3 Mbps. The quality of the video stream is therefore highly dependent on the number of PRBs allocated to the camera UE, which directly impacts the user's Quality of Experience (QoE). To capture this effect, a QoE evaluation based on the Structural Similarity Index (SSIM) metric was integrated into the analysis. The SSIM is computed frame by frame, comparing each transmitted frame with its corresponding reference frame (before transmission within the 5G network). The closer the resulting value is to 1, the more similar the transmitted frame is to the original; conversely, values approaching 0

indicate high distortion or loss of visual fidelity.

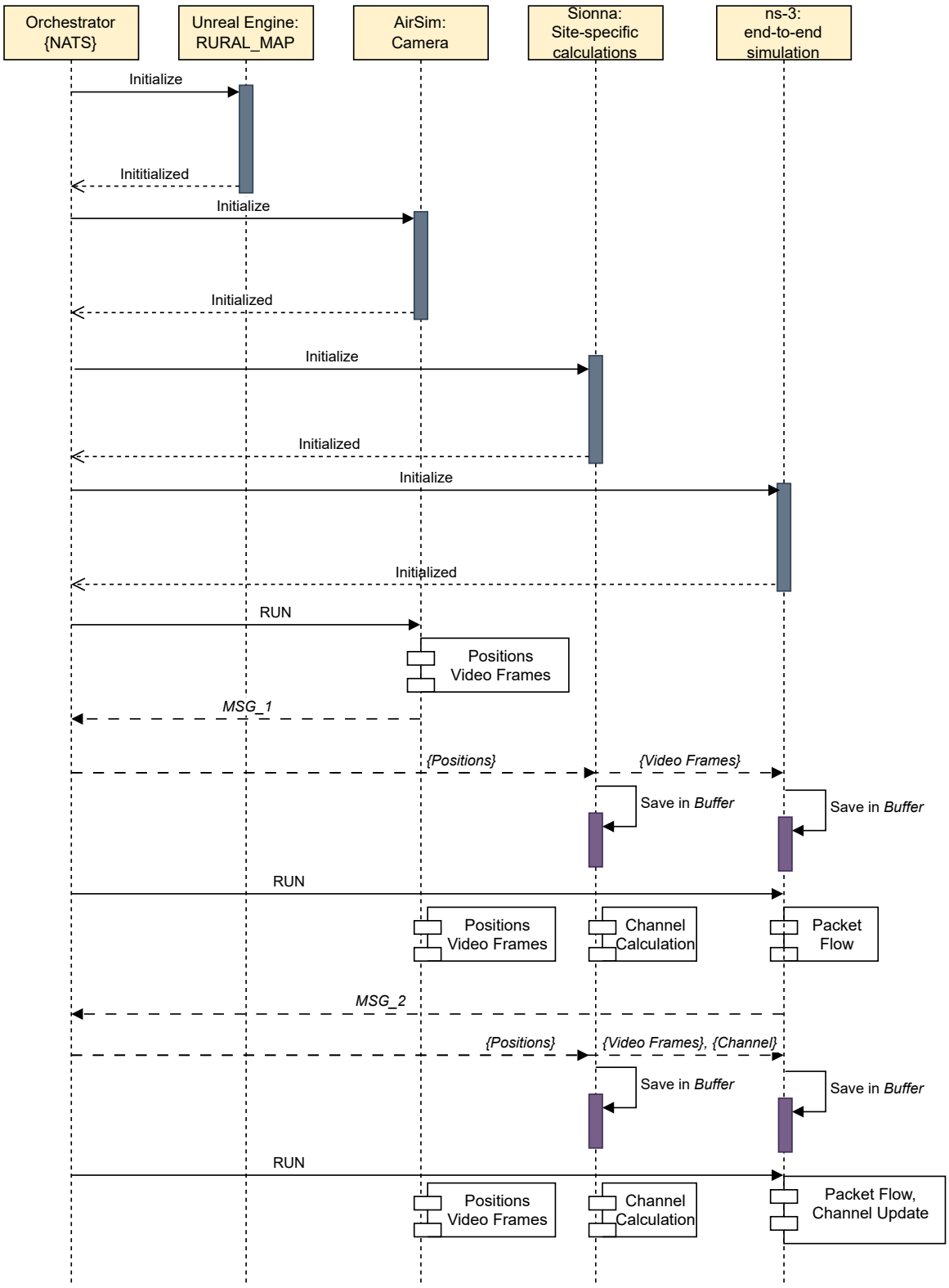
As with the asynchronous execution strategy discussed in Section 3.3.2.1, the interaction between modules in this use case follows an event-driven, non-blocking paradigm. Fig. 17 provides a detailed overview of how the modules interact throughout the event loop. This use case combines multiple components: ns-3 (enhanced with 5G-Lena and the NORI module), Sionna for site-specific channel modeling, and AirSim for computer vision and camera API data acquisition.

The co-simulation process begins with the initialization of all modules. In the first event loop iteration, modules that do not depend on external inputs—namely, the 3D map and AirSim—are executed. During this phase, AirSim captures the positions of the four mobile devices (one camera-equipped and three additional UEs) and retrieves image frames from the camera, transmitting these as messages to the orchestrator. The orchestrator then forwards this data: position information is sent to Sionna for site-specific channel calculation. At the same time, video frames are encapsulated into packets in ns-3 for transmission over the emulated network. These outputs are stored in their respective buffers for use in subsequent iterations.

In the second iteration, all modules execute concurrently using the updated information, enabling Sionna to calculate site-specific channels and ns-3 to process packet flows based on the previous video frames. The orchestrator once again routes these outputs to their corresponding modules, ensuring the correct flow of information across the system. While not explicitly depicted in Fig. 17 for simplicity, several auxiliary tools facilitate these operations. For example, in AirSim, video frames are processed using the FFMPEG[4] codec to generate an RTSP stream, while channel updates in ns-3 adhere to a 1 ms interval, achieved through the augmentation techniques described in Section 3.4.2.

[4] https://ffmpeg.org/

Figure 17 – Message flow diagram between modules during the rural animal monitoring use case.

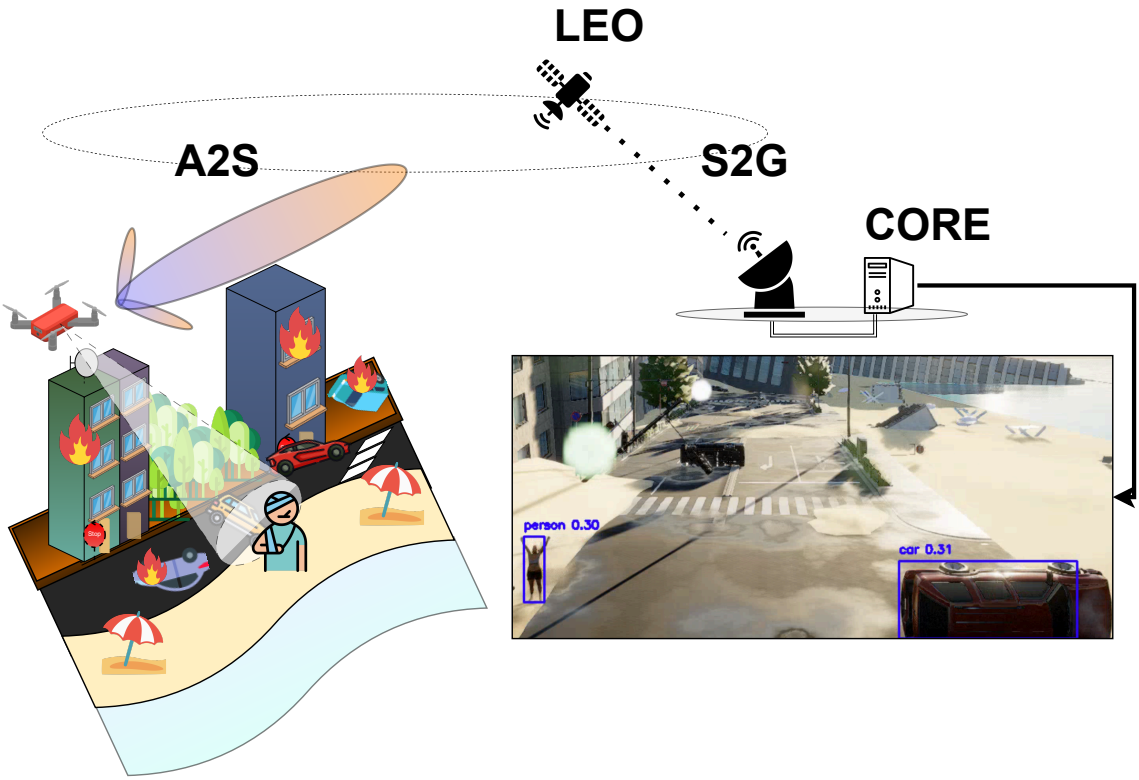


Source: The author (2025)

4.2 Urban - Search and Rescue

This use case models an urban S&R scenario in the aftermath of a disaster, wherein an autonomous drone performs a pre-programmed path-planning mission to identify potentially trapped individuals. The drone traverses a defined set of waypoints, scanning the environment using onboard Red, Green, Blue (RGB) cameras to detect human figures, represented here by Unreal Engine metahuman actors. Unlike the previous use case (Section 4.1), which involves continuous user-based monitoring, this S&R scenario performs fully automated target detection using a YOLO-based AI module. The drone captures live video streams, which are then transmitted over a simulated end-to-end communication system to a remote server, where the YOLO model performs real-time human detection. Fig. 18 presents a high-level architectural overview of the S&R use case.

Figure 18 – Overview of the urban Search and Rescue use case.



Source: The author (2025)

The communication simulation in this use case focuses on a Unmanned Aerial Vehicle (UAV)-to-satellite link, modeled stochastically based on the TR 38.811 NTN-Urban channel model, modeled in (3GPP, 2017). A Low Earth Orbit (LEO) satellite is considered in the simulation, at two different altitudes (600 km and 1200 km); however, its mobility is not explicitly modeled using a dedicated satellite mobility module. The communication path thus represents a minimal version of the system presented in (MARCHESE; MOHEDDINE; PATRONE, 2020), excluding the IoT ground segment and focusing solely on Aerial-to-Space (A2S) and Space-to-

Ground (S2G) interactions, where the S2G link is abstracted without in-air communication, but assuming a perfect link. As with previous scenarios, the video transmission is achieved using FFMPEG-based RTSP streaming. The stream is captured in real time and analyzed remotely for human detection.

This implementation relies on four core modules within the CAVIAR framework. The 3D module is responsible for managing the virtual environment and rendering the drone's surroundings using RGB cameras. The AirSim module handles the UAV flight dynamics and its interaction with the virtual world, including camera control. The ns-3 module enables full end-to-end network emulation, incorporating the NTN stochastic channel model to simulate A2S communication, including the S2G link abstraction until it reaches the server, which is also the core of the 5G NSA network. Finally, the AI module applies YOLO-based object detection to identify humans in the transmitted video stream.

Table 6 – Simulation parameters for UL/DL in LEO-600 and LEO-1200 in the S and Ka bands, obeying the specification Tables in TR 38.821.

Co-simulation parameters	**LEO-600 (km)**		**LEO-1200 (km)**	
	S	Ka	S	Ka
Carrier frequency (GHz) [Table 6.1.3.2-1]	2	30	2	30
System bandwidth (MHz) [Table 6.1.3.2-1]	30	400	30	400
Satellite EIRP density (dBW/MHz) [Table 6.1.1.1-1]	34	4	40	10
Satellite maximum antenna gain (dBi) [Table 6.1.1.1-3]	30	30	38.5	38.5
*Beamforming periodicity (s) [Table 4.2-2]	1			
UE (Drone) features				
Noise figure (dB) [Table 6.1.1.1-3]	1.2		1.2	
Tx transmit power (dBm) [Table 6.1.1.1-3]	33		33	
Tx antenna gain (dBi) [Table 6.1.1.1-3]	43.2		43.2	

**Note:* According to the standards, *"Each satellite has the capability to steer beams towards fixed points on earth using beamforming techniques. This is applicable for a period of time corresponding to the visibility time of the satellite."* In this work, an arbitrary value for beamforming periodicity is assumed.

In general, the co-simulation follows the parameter sets defined in alignment with the TR 38.821 specifications, as summarized in Table 6. Two LEO cases are considered: LEO-600 and LEO-1200, corresponding to orbital altitudes of 600 km and 1200 km, respectively. Both are simulated for the S-band and Ka-band. The S-band, typically ranging from 2 GHz to 4 GHz (picked 2GHz, according to the TR 38. 821), is widely employed in applications such as weather radar and certain communication satellites. In contrast, the Ka-band, operating in the 26-40 GHz range (picked 30 GHz), has become increasingly favored for high-throughput satellite communications. The primary distinction between these bands in the context of the co-simulation parameters lies in the Equivalent Isotropic Radiated Power (EIRP) density, which represents the total power radiated by a transmitter as if emitted from a hypothetical isotropic

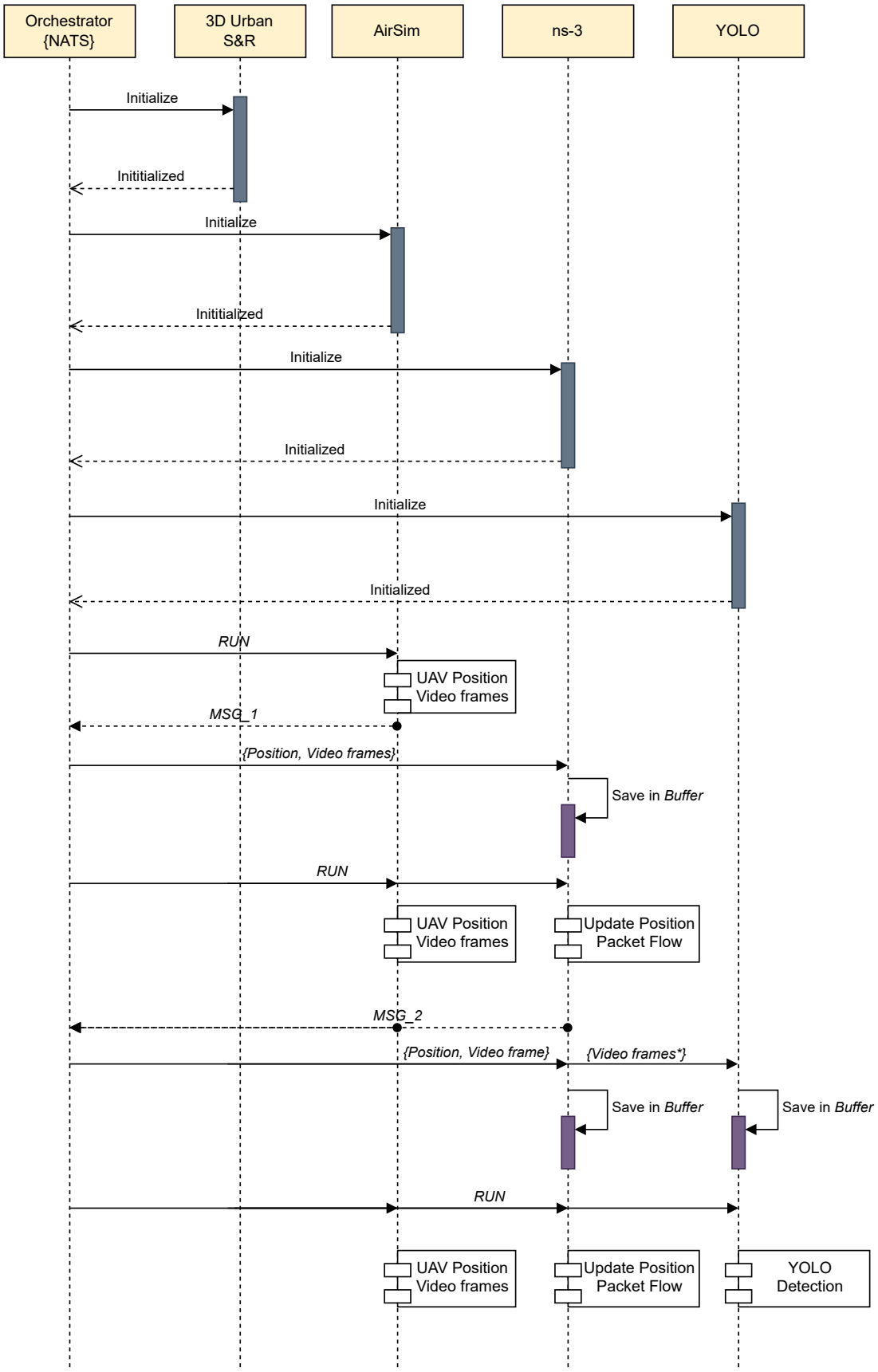
antenna, normalized by the whole bandwidth. This value is used to compute the satellite transmit power according to:

$$\text{TxPower}_{\text{dBm}} = (EIRP + 30) + 10 \cdot \log_{10}\left(\frac{BW}{10^6}\right), \tag{4.1}$$

where the $EIRP$ is the EIRP density in dBW/MHz of the satellite and BW is it's bandwidth in Hz. Moreover, the maximum antenna gain of the satellite also changes with the satellite's altitude.

The communication between these modules adheres to the CAVIAR asynchronous messaging framework described earlier in Section 3.3.2.1. Fig. 19 illustrates the flow of messages and processing steps among the modules, from drone camera acquisition to final detection inference.

Figure 19 – The asynchronous message exchanging in urban-NTN use case.

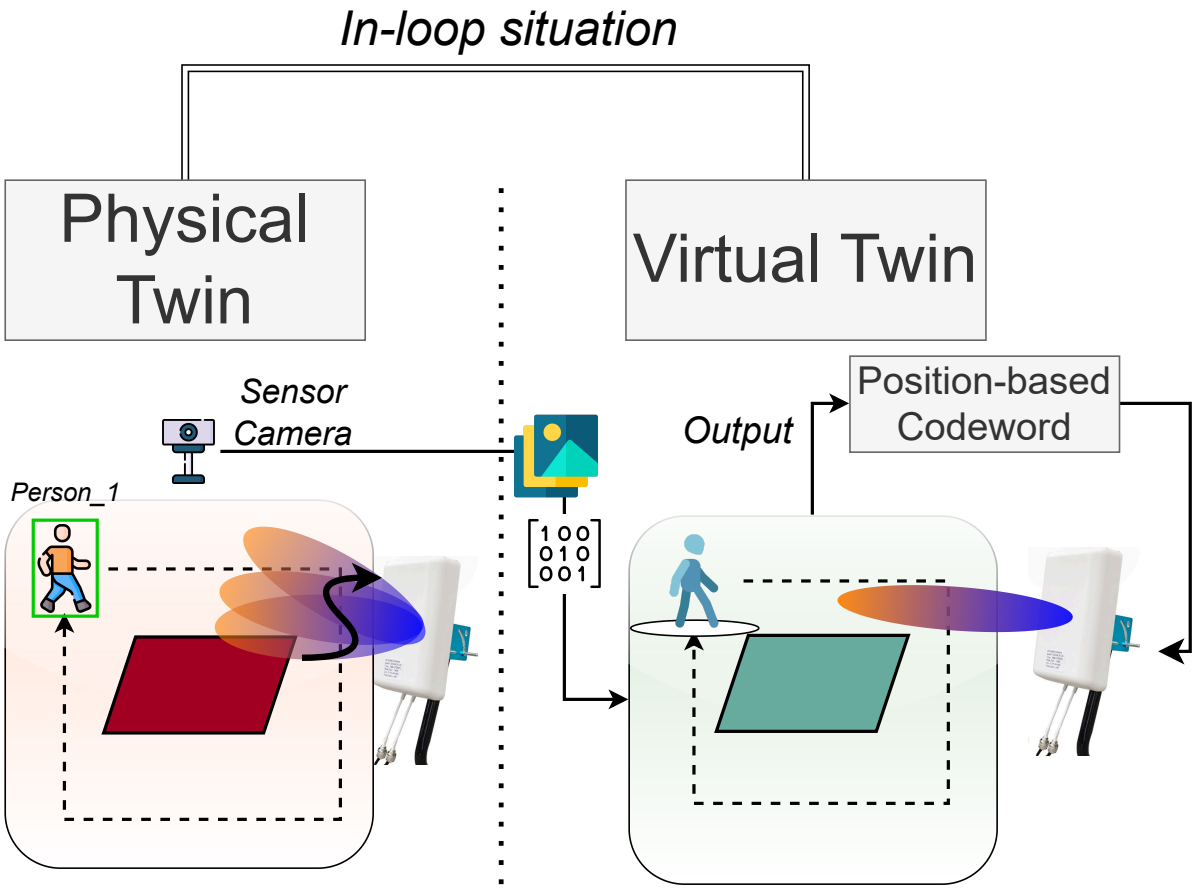


Source: The author (2025)

4.3 Indoor - Robot Coordination with Multi-Modal ISAC

This use case implements a simplified version of a multi-modal ISAC system, inspired by the architecture described in (PENG et al., 2025b). The implementation represents a full in-loop DT configuration (though not fully closed-loop), where the physical and virtual components of the twin maintain continuous communication using the CAVIAR framework. As illustrated in Fig. 20, a single edge node is deployed to collect and process multi-sensor data—specifically Light Detection and Ranging (LiDAR) and RGB imagery—providing critical features for channel estimation and analysis. On the physical side, a Mikrotik multi-antenna array with 32 elements is used to transmit the signal, operating according to a pre-programmed codebook, with 64 codewords as resolution, as described in (SONG et al., 2023). In this use case, position-based beamforming is investigated as an alternative to the standard beam sweeping approach, using LiDAR and RGB information to estimate the receiver's position and compute the appropriate steering vector for beamforming evaluation, assuming one antenna element at the receiver.

Figure 20 – Overview of the indoor robot coordination use case with multi-modal ISAC integration.



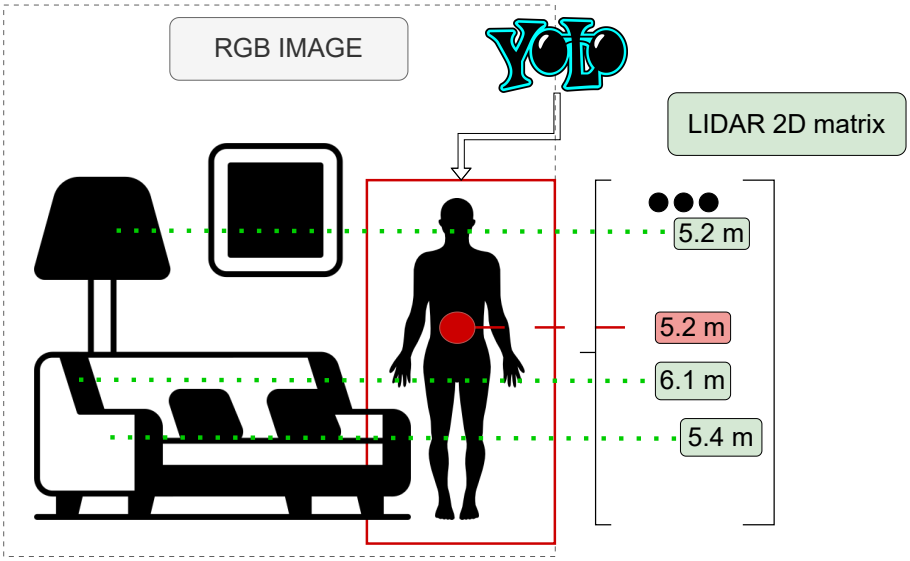
Source: The author (2025)

The exchange of features between the physical and virtual twins is achieved using an

Intel® RealSense™ sensor, which delivers both LiDAR depth-per-pixel data and high-resolution RGB imagery at a high sampling rate. This data stream serves as the primary input for detecting and tracking a person within the monitored indoor environment. First, the LiDAR data is processed to estimate the distance between the person and the sensor. Next, a YOLO-based object detection algorithm is applied to the RGB images, identifying the person's location in the x–y plane. By combining these modalities, the system accurately estimates the person's 3D position (x, y, z coordinates) relative to the sensor.

In the virtual environment, the detected person is represented by an animated robotic agent designed to mimic average human proportions and motion patterns (e.g., walking animations). This virtual representation ensures a realistic mapping between the physical and digital entities, enabling position-aware channel modeling in the simulated twin. By updating the robot's position in real time, the framework enables dynamic estimation of position-specific communication channels.

Figure 21 – Fusion of YOLO and LiDAR to provide a person's estimated position in the virtual twin.



Source: The author (2025)

Fig. 21 illustrates the process of merging these technologies into a cohesive workflow. The process begins with the camera sensor capturing a LiDAR depth map and an RGB image. The depth map is represented as a two-dimensional matrix, where each cell corresponds to a pixel in the scene, with values indicating the measured distance between the camera and the objects in the corresponding field of view. Concurrently, a YOLO-based object detection algorithm is applied to the RGB image to locate the target person and determine the bounding box surrounding them. The center pixel of this bounding box is identified as the primary reference point for the person's position. Using the pixel coordinates from the RGB image and the corresponding depth value from the LiDAR matrix, the system performs a transformation to estimate the 3D coordinates (x, y, z) of the person relative to the camera frame. These coordinates are then mapped into the

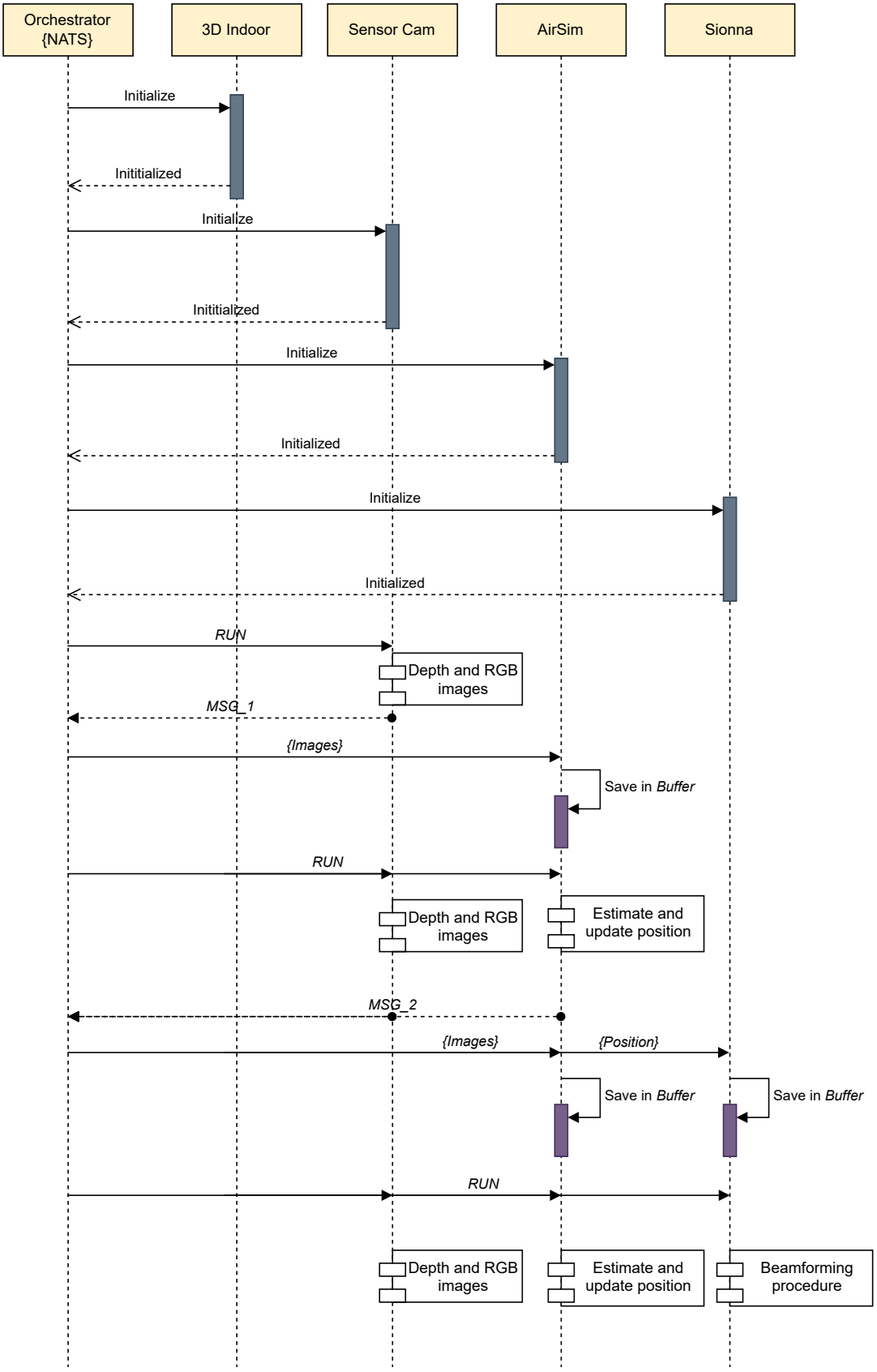
virtual environment, where the robot's pose is updated to match the physical entity's estimated position.

Once the pose of the virtual robot is updated, the new position is used to compute a site-specific channel using Sionna, where the receiver is set to 1 isotropic antenna element. It is important to note that this approach estimates position and channel behavior within the virtual twin, and small discrepancies between the physical and digital environments (e.g., slight variations in room dimensions) may occur. Nevertheless, this approximation provides sufficient accuracy for realistic evaluation of communication performance, as further demonstrated in Section 5.2.3. Although humans are used in this use case, the framework is equally applicable to other devices, such as industrial robots and extended reality (XR) equipment, which were not investigated here.

Moreover, the site-specific channel calculation enables the evaluation of two beamforming methods: (i) beam sweeping and (ii) position-based beamforming, as mentioned previously. In the beam sweeping approach, the transmitter iterates through the entire codebook, testing each codeword to identify the one that maximizes the effective channel. In contrast, the position-based beamforming approach uses the estimated position to calculate azimuth and zenith angles, which are then used to determine the corresponding codeword directly. These methods differ primarily in time efficiency and received power, investigated in Section 5.2.3.

Fig. 22 summarizes the asynchronous communication among the CAVIAR modules during this process, from sensor acquisition through pose updating, channel modeling, and network simulation. An important note is that, in this use case, the AI component based on YOLO is not implemented as a standalone module but rather as a feature embedded within the mobility module. This reflects its function as a perception tool that does not require a dedicated life-cycle management process, as discussed in Section 3.3.1.

Figure 22 – The asynchronous message exchanging in indoor-ISAC use case.



Source: The author (2025)

5 RESULTS

5.1 Real-Time and Performance Metrics

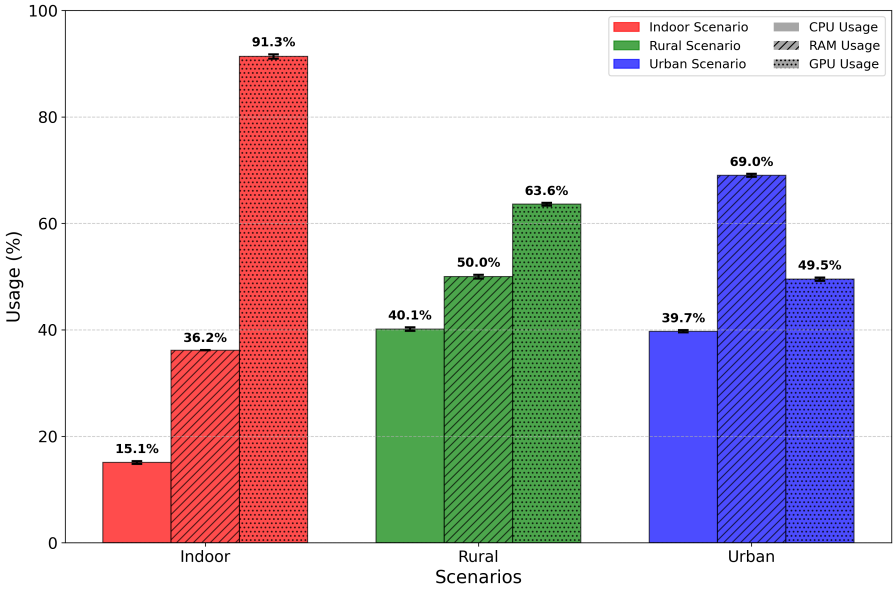
Initially, a performance evaluation was conducted, along with an analysis of computational resources and Real-Time Factor (RTF), performed on the machine described in Table 7.

Table 7 – Simulation hardware specifications.

CPU	13th Gen Intel® Core™ i7-13700HX × 24
RAM	32 GiB
GPU	NVIDIA GeForce RTX 4060

The usage of computational resources can be categorized into three main components: (i) the Graphics Processing Unit (GPU), which is primarily utilized for ray tracing and rendering in the context of this work; (ii) the Random-Access Memory (RAM), which is dynamically allocated for both rendering tasks and the execution of module-specific and core functions; and (iii) the CPU, which is continuously used for executing functions across the entire framework. Figure 23 illustrates the utilization of these resources for each explored use case.

Figure 23 – Average CPU, GPU, and RAM usage among all use cases.



Source: The author (2025)

The Indoor use case shows the highest GPU usage (averaging nearly 92%), primarily due to its reliance on both rendering and ray tracing. Although the indoor environment is spatially limited, it features a high level of detail (i.e., a greater number of objects in the scene), which

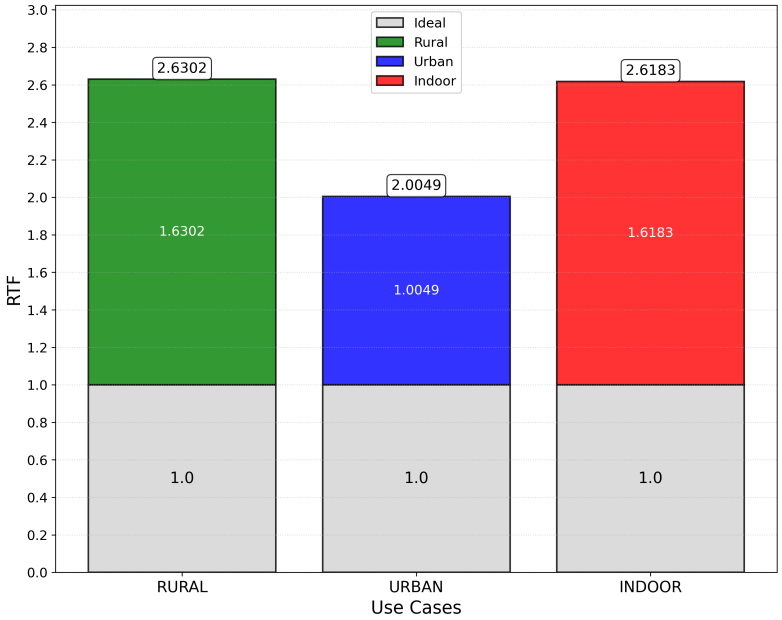
increases the complexity of both rendering and ray tracing-based channel calculations, resulting in a greater number of interactions. Conversely, CPU usage for this use case is relatively low (15.1% on average), given that function executions are minimal, which also explains the reduced RAM usage (36.2%).

Similarly, the Rural use case exhibits moderate GPU usage (63.6%), as it also employs rendering and ray tracing, though with fewer interactions and less scene complexity, resulting in a more optimized scenario. CPU and RAM usage increase in this case, averaging 40.1% and 50%, respectively, due to a higher number of functions being executed—stemming from the inclusion of additional processing entities (RIC + xApp) and the use of CPU-based simulators for network evaluation.

In the Urban use case, GPU usage drops significantly (reaching 49.5%) because this scenario involves only 3D rendering, with no ray tracing-based channel modeling. Instead, a stochastic channel model is used. As a result, similar to the Rural case, there is considerable CPU usage (39.7%) and even higher RAM allocation (69%), primarily due to the network simulator but also owing to the computationally intensive nature of the stochastic channel model, particularly on the CPU.

In addition to average computational resource usage, DT-based approaches must also be assessed in terms of real-time compliance, that is, how closely each iteration of the event loop reproduces real-world timing. This is captured by the RTF, defined as the ratio between simulated time and real elapsed time, where the ideal value is one. Fig. 24 shows the RTF across all evaluated use cases, each simulated for 240 seconds. Results indicate similar RTF values across scenarios, since the same core modules are executed. The rural and indoor cases achieve an RTF of about 2.6, meaning that simulating a given interval requires roughly twice the real-world duration. In contrast, the urban NTN case achieves a 23.9% lower RTF, highlighting ray tracing as the dominant source of computational overhead.

Figure 24 – The overall RTF for all use cases.



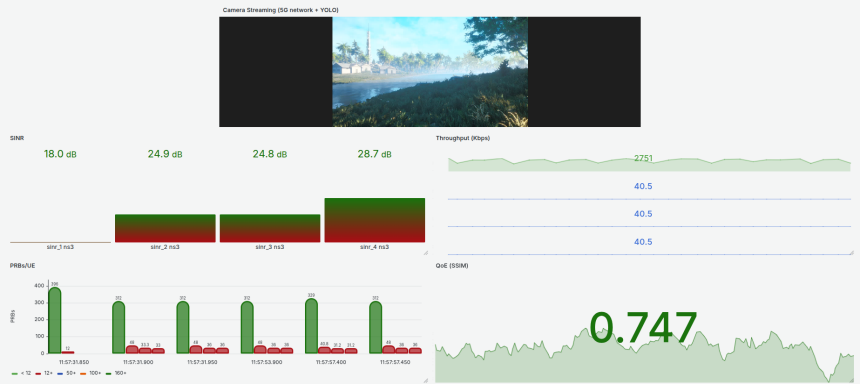
Source: The author (2025)

5.2 Specific Use Cases Results

5.2.1 Rural Results

The first observable results for the rural use case are illustrated in Fig. 25. The dashboard shows the video streaming from the rural scenario's camera when using the RL-based scheduler. Several KPIs metrics are also displayed, including throughput per UE, Signal-to-Interference-Plus-Noise Ratio (SINR) values, and, more specifically to this scenario, the number of PRBs allocated to each user. As discussed in Section 4.1, the allocation of PRBs directly impacts user throughput. Consequently, under the two scheduler configurations—RR and RL—the PRB allocation evolves differently over time.

Figure 25 – Dashboard for the rural use case containing the main results: throughput, number of allocated PRBs for each UE, SINR, SSIM for QoE, and the RTSP streaming.

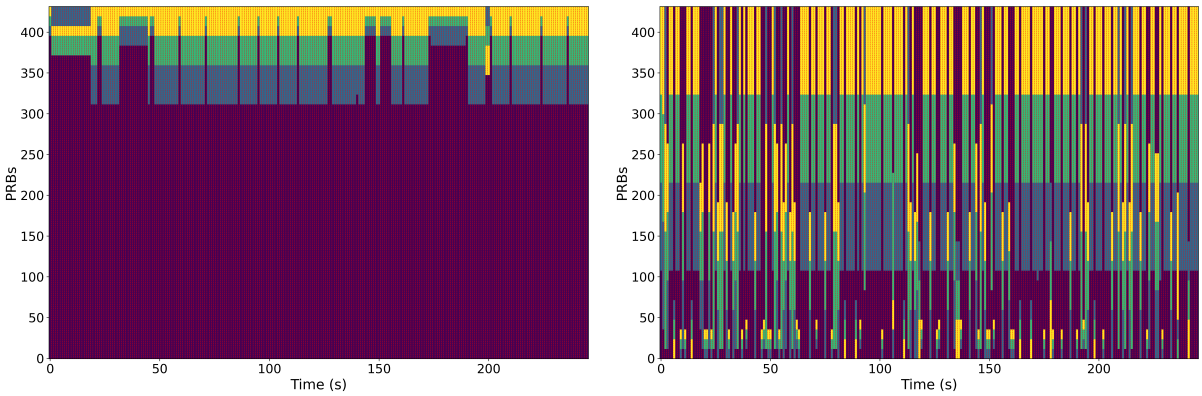


Source: The author (2025)

As shown in Fig. 26, the use of distinct scheduling methods under the OFDMA access scheme results in different PRB allocations during the 240-second co-simulation. In the RL-based scheduler, Fig. 26a, with a fixed minimum allocation of 70% of PRBs, nearly the entire time–frequency grid is assigned to the camera UE (purple blocks). Conversely, in the baseline RR scheduler, Fig. 26b, the allocation is more evenly distributed across users, despite the side UEs requiring only 40 kbps as a minimum throughput (see Section 4.1).

Figure 26 – Resource allocation during the co-simulation using RL and RR schedulers. Purple PRBs are allocated to the camera UE, while yellow, green, and blue blocks represent side UEs.

(a) PRB allocation over time using the RL approach. (b) PRB allocation over time using the RR approach.



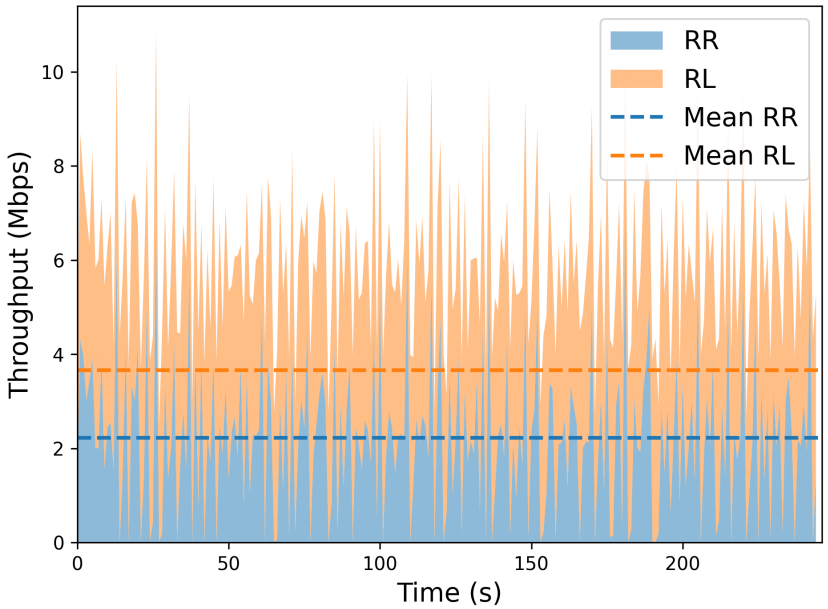
Source: The author (2025)

This inefficient use of PRBs under the RR scheduler negatively impacts the throughput of the video stream, leading to degraded QoE. Fig. 27 presents the average and instantaneous throughput for both approaches. With the RL scheduler, throughput never falls below 1 Mbps and maintains an average of 3.8 Mbps, which is well-suited for RTSP-based applications. In contrast, under the RR scheduler, the camera UE throughput frequently drops to zero and peaks only at 4 Mbps, resulting in an average of 2.1 Mbps, below the 3 Mbps requirement specified in Section 4.1. Since both scheduling methods ensured that the side UEs achieved their required throughput, the corresponding results are omitted from the plots.

The QoE metric, evaluated using SSIM, is summarized in Table 8. Results show that even with the RR scheduler, the average SSIM remains in the acceptable range (0.7, or 70% similarity between reference frames and transmitted frames). However, the minimum values indicate episodes of visual degradation, producing pixel distortion or buffer issues with FFMPEG. With the RL scheduler, the average SSIM improves to 0.75, while the minimum values never fall below 0.7, thus avoiding perceptible quality degradation and ensuring a smoother, higher-quality video streaming experience.

For visualization, Fig. 28 shows a comparison between two frames, both as a result of the frame passing within the 5G network, assuming the RR (Fig. 28a) and the RL (Fig. 28b)

Figure 27 – Throughput evolution and average values under RL and RR schedulers.



Source: The author (2025)

Table 8 – SSIM values for QoE assessment under RL and RR schedulers.

Method	Maximum	Minimum	Average
RL	0.7790	0.7124	0.7513
RR	0.7705	0.6388	0.7260

schedulers. In the RR case, some corrupted blocks, caused by misuse of the RAN resources, produce visual artifacts, since the decoder is not able to recreate the video. Unlike the former, the second presents non-perceptible corrupted blocks, which lead to a higher quality frame (without distortion).

Figure 28 – Comparison of a frame passed within the 5G network, for both RR and RL-based schedulers.

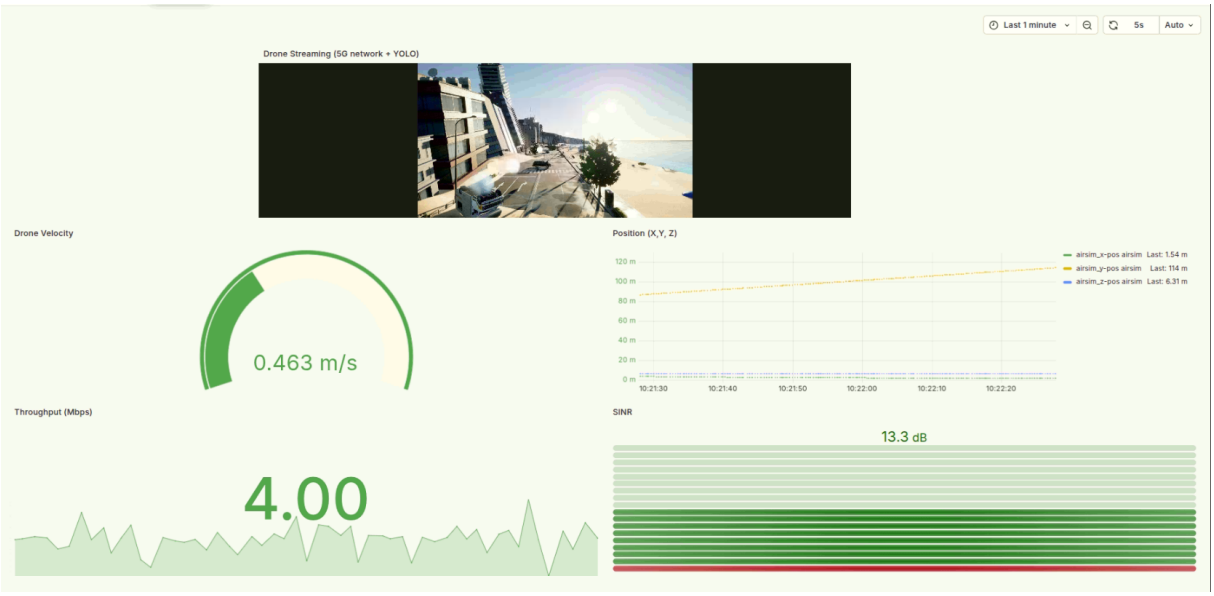
(a) RR-based. (b) RL-based.

Source: The author (2025)

5.2.2 Urban Results

In the NTN use case, the first observable results are presented in Fig. 29. The dashboard provides a consolidated view of key information for the end-to-end simulation. The RTSP video stream, originating from the drone, is transmitted through the satellite via the A2S link and reaches the ground node through the abstracted S2G link. At the ground node, the incoming bitstream is processed for object detection using YOLO, and the video is reconstructed via FFMPEG. In parallel, telemetry from the drone—such as position and velocity—is recorded. KPIs, including application-level throughput and SINR, are also monitored. Each SINR time-series plot represents the average value computed over one-second intervals.

Figure 29 – Urban dashboard with the features: RTSP live-streaming, drone velocity, drone position, measured end-to-end throughput, measured SINR assuming the NTN channel.



Source: The author (2025)

From the SINR time series, it is possible to derive additional statistical measures such as channel capacity and spectral efficiency. Channel capacity is computed according to the Shannon theorem, assuming a Gaussian channel model:

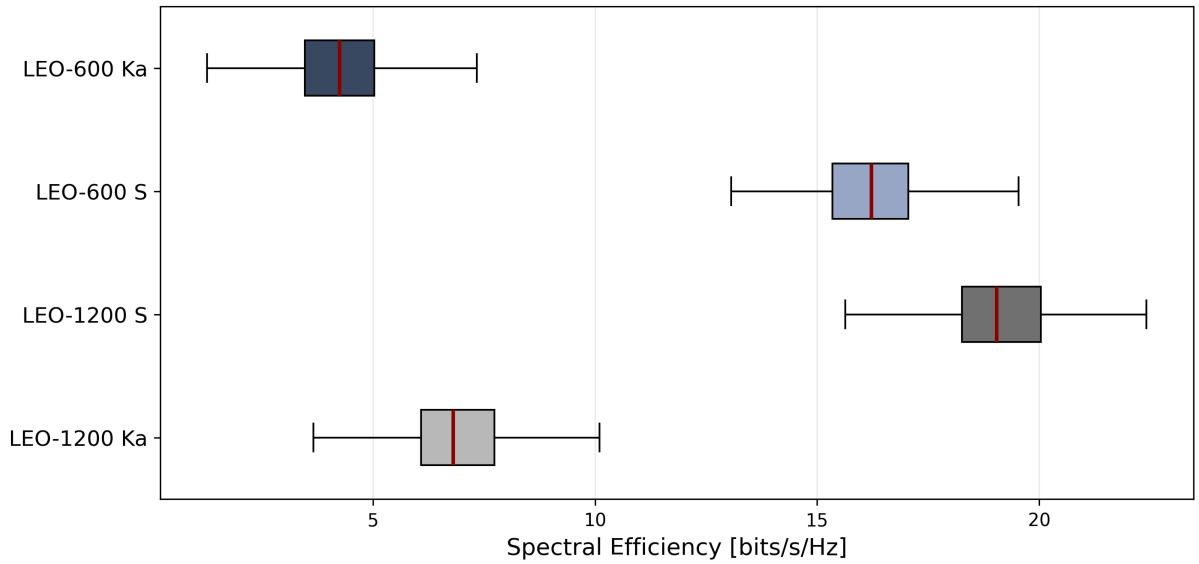
$$C = BW \cdot \log_2(1 + SNR), \tag{5.1}$$

where BW is the channel bandwidth in Hz and SNR is the estimated or measured Signal-to-Noise Ratio (SNR). Spectral efficiency is given by the same expression, normalized by bandwidth.

Fig. 30 presents the statistical distribution of spectral efficiency for each configuration described in Table 6. As observed, the LEO-600 Ka scenario yields the lowest spectral efficiency due to its lower EIRP density and higher carrier frequency (30 GHz), which increases both large-scale and small-scale fading effects. Conversely, the LEO-1200 S-band configuration

achieves the highest values, benefiting from a lower carrier frequency (2 GHz), which reduces propagation losses, combined with a higher EIRP.

Figure 30 – Spectral efficiency statistics for the LEO-600,1200 with S or Ka-band.



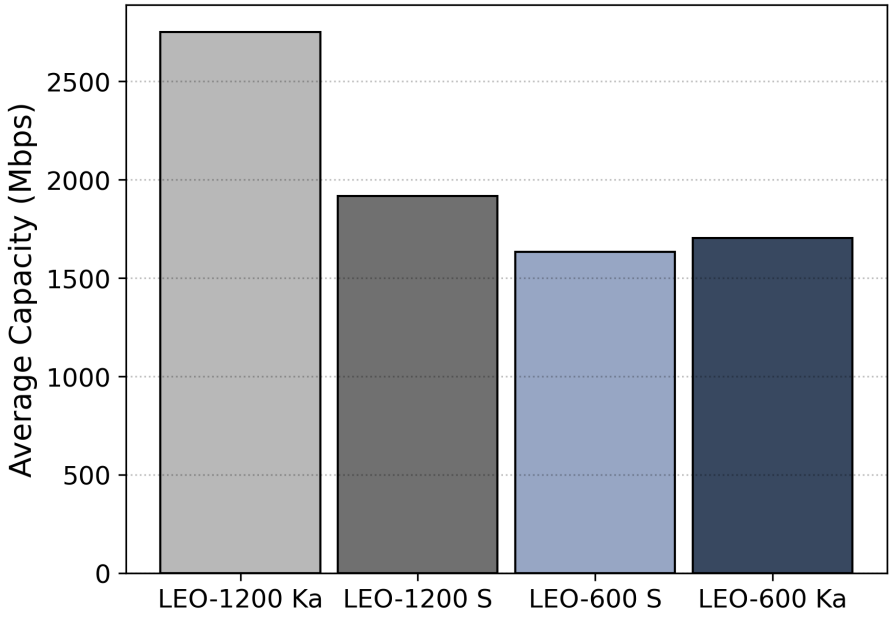
Source: The author (2025)

On the other hand, as shown in Eq. 5.1, channel capacity increases with the available bandwidth. This relationship is evident in Fig. 31, where the LEO-1200 Ka configuration achieves the highest average capacity (2.7 Gbps). This result stems from its wider bandwidth (400 MHz), enabled by operation at a higher carrier frequency, which provides access to a larger spectrum allocation. In contrast, the LEO-1200 S configuration attains higher spectral efficiency but is limited to a 30 MHz bandwidth, resulting in a lower channel capacity (1.85 Gbps). The LEO-600 S configuration presents the lowest capacity, as it combines the smallest bandwidth with a lower EIRP density. Meanwhile, the LEO-600 Ka configuration outperforms the LEO-600 S due to its larger bandwidth, despite exhibiting lower spectral efficiency than its S-band counterpart.

When analyzing the KPIs results in Table 9, throughput exhibits low variability across the four use cases. This behavior is explained by the nature of the RTSP video streaming application, which requires a minimum capacity threshold and fluctuates up to a maximum of approximately 6.5 Mbps — a value reached in all scenarios. In contrast, the Round Trip Time (RTT) values show significant variation, particularly in their minimum and maximum extremes. The highest RTT is observed in the LEO-600 Ka configuration, while the lowest occurs in the LEO-1200 Ka case, which also benefits from the highest channel capacity. In terms of average RTT, the best performance is achieved by the LEO-600 S configuration, closely followed by the LEO-1200 S scenario.

Another important metric to evaluate in the computer vision context—given that YOLO is employed in this use case—is the mean Average Precision (mAP). The mAP is a standard

Figure 31 – The average channel capacity for the four use cases.



Source: The author (2025)

Table 9 – KPIs measured for uplink in LEO-600 and LEO-1200 scenarios, for S-band and Ka-band.

Scenario	Throughput (Mbps)			RTT (ms)		
	Minimum	Maximum	Average	Minimum	Maximum	Average
LEO-1200 Ka	0.000127	6.35883	3.85067	13.20	387.0	28.28
LEO-1200 S	0.000336	6.32352	3.89329	8.17	457.0	18.30
LEO-600 S	0.000396	6.05758	3.87990	7.20	452.0	16.42
LEO-600 Ka	0.000309	6.12243	3.79383	12.20	524.0	36.47

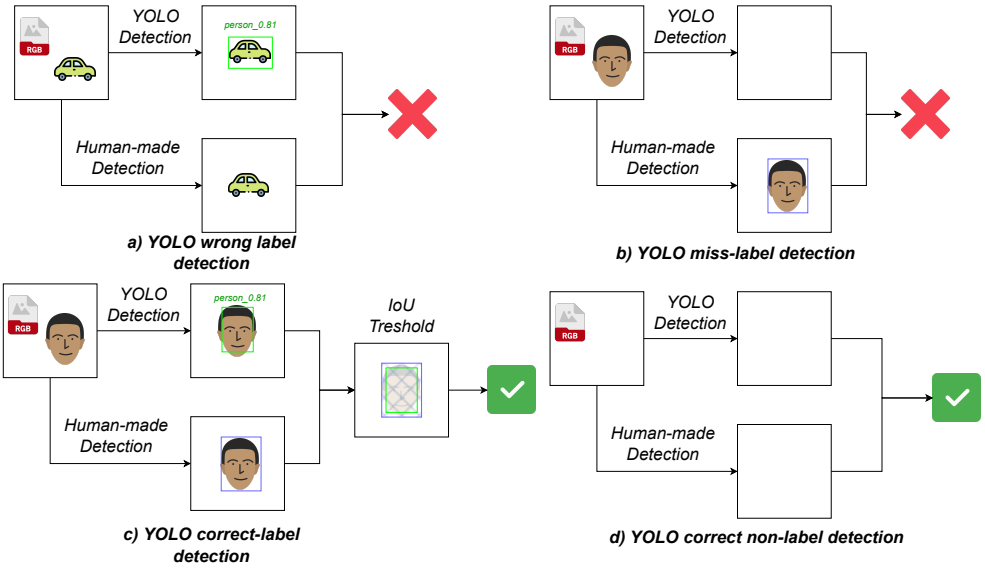
object detection metric that measures the average precision across all classes and recall levels, providing a single score that summarizes the detector's accuracy over the evaluated dataset. Higher mAP values indicate better detection performance.

Fig. 32 presents an overview of this evaluation. Correct detections are those where the model correctly identifies the "person" label (the only class evaluated in this scenario) and correctly returns no detections when no person is present in the scene. Incorrect detections include false positives, where the model predicts a person in a location containing another object, and false negatives, where no detection is made despite a person being present in the frame.

Two mAP variants were evaluated: mAP.5 and mAP.5:0.95. The first considers a fixed Intersection of Union (IoU) threshold of 0.5, whereas the second averages the mAP over IoU thresholds ranging from 0.5 to 0.95 in steps of 0.05. In this evaluation, both metrics yielded the same value of 55.6%, as reported in Table 10. This occurs because the detector's bounding box

predictions are either well above the 0.95 IoU threshold for correct detections or far below the 0.5 threshold for incorrect detections, resulting in identical scores for both evaluation criteria.

Figure 32 – Evaluation criteria applied in the YOLO mAP computation for each frame of the video stream, using a human-annotated ground truth for the "person" class.



Source: The author (2025)

Table 10 – mAP values obtained in the YOLO evaluation for this NTN case.

mAP@.5	0.5560
mAP@.5:0.95	0.5560

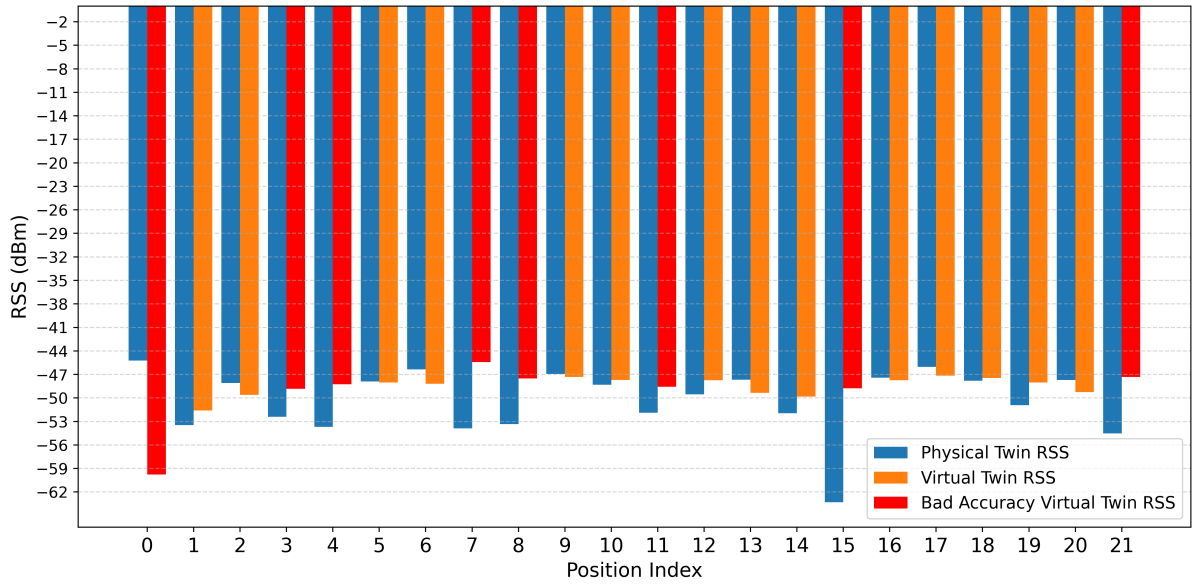
5.2.3 Indoor Results

Before conducting a general evaluation and comparing performance in terms of time and Received Signal Strength (RSS), it is essential to first assess the fidelity of the virtual twin in comparison to the physical twin. To achieve this, a comparison was made between the values measured in the same room and the values obtained in the virtual environment (using Sionna and the pre-computed codebook). Fig. 33 demonstrates the comparison between 22 points within the indoor scenario. The results are presented in RSS values, which are an estimate of the received power. These values are estimated by converting from Received Signal Strength Indicator (RSSI) to RSS using the formula provided by (SONG et al., 2023):

$$\text{RSS} = 0.0651 \times \text{RSSI} - 74.3875, \tag{5.2}$$

where RSS represents the strength of the received signal, and RSSI, where RSSI $\in$ N, is an integer representing the RSSI value. In the evaluations, 14 out of the 22 points were valid, with 10 points (from 10 different positions) deemed invalid. A threshold of -3 dB was applied, as

Figure 33 – Comparison among the physical twin RSS and the virtual twin RSS for fidelity evaluation.



Source: The author (2025)

a difference of 3 dB corresponds to a halving of the received signal power. Thus, only those positions with high accuracy were selected for evaluation.

In this context, both position-based and codebook-based beamforming methods are evaluated. Position-based beamforming is assessed by considering the fixed position of the transmitter in the scenario. For each new beamforming period (an arbitrary value of 50 ms), a new combination of azimuth and zenith angles is calculated, which is then used to determine the codeword. The accurate azimuth angle based on the position is calculated using:

$$\phi = \tan^{-1}\left(\frac{d_y}{d_x}\right), \tag{5.3}$$

where ϕ represents the azimuth angle, d_y and d_x are the differences in the y-axis and x-axis between the receiver and transmitter, respectively. The zenith angle based on the relative position is given by:

$$\theta = \cos^{-1}\left(\frac{d_z}{\sqrt{d_x^2 + d_y^2 + d_z^2}}\right), \tag{5.4}$$

where d_z is the difference in the z-axis, and θ is the zenith angle.

Using these relative position angles, the codeword is calculated using the wave vector and the UPA architecture, assuming a normalized inter-element spacing. The codeword is given by:

$$w = \left[e^{-j\frac{2\pi}{\lambda}d_n \cdot k}\right]_{n=0}^{N-1} \tag{5.5}$$

where w is the codeword vector, and each element corresponds to the phase weight applied at the n_{th} antenna element. Here, λ denotes the wavelength, d_n is the position vector (x, y, z) of the n_{th} antenna element in the planar array of N elements, and k is the wave vector in Cartesian coordinates, expressed as:

$$k = [\sin(\theta)\cos(\phi),\ \sin(\theta)\sin(\phi),\ \cos(\theta)] \tag{5.6}$$

After calculating the codeword, the RSS in dBm can be estimated using the multi-path site-specific channel model as follows:

$$\text{RSS} = P_{\text{tx}} + 10\log_{10}\left(\sum_{i=0}^{MPC} |H_i \cdot w^{\text{T}}|^2\right), \tag{5.7}$$

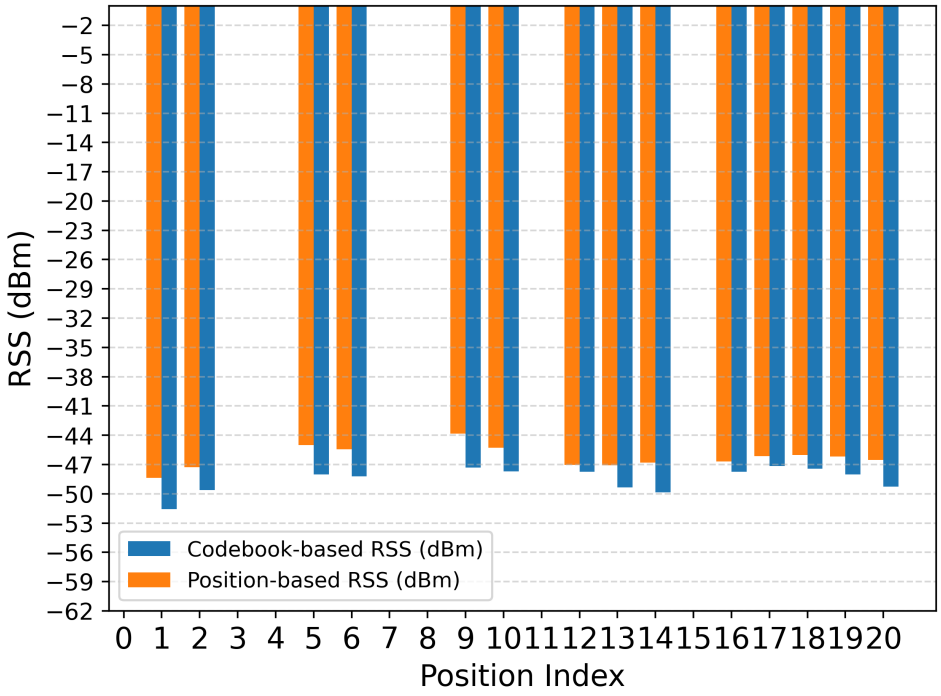
where MPC represents the number of multi-path components, each i_{th} MPC being evaluated as a H_{th} channel matrix in the same shape as the receiver and transmitter antennas (in this case, 1x32). P_{tx} is the transmission power in dBm (41 dBm). A similar calculation is performed for the beam sweeping approach, but here the maximum value among the 64 codewords in the codebook is selected.

Fig. 34 presents the evaluation of both position-based and beam sweeping (codebook-based) beamforming methods. Across all assessments, the position-based method consistently achieves higher RSS values, reaching almost twice the values in 42.85% of the valid indexes. In contrast, at the other indexes, the difference between the methods is less than 3 dB. However, when comparing the time overhead, as shown in Table 11, the edge node's time to estimate the correct user position and the RAN's time to calculate the corresponding codeword reach a maximum of nearly 8 ms, with an average of approximately 4 ms. On the other hand, the codebook-based method achieves its optimal value in as little as 0.82 ms, which is lower than the minimum time of the position-based approach. The average time for the codebook-based approach is 0.48 ms, with a maximum of 0.82 ms.

Table 11 – Processing time values for beamforming procedure assuming codebook-based and position-based.

Method	Maximum	Minimum	Average
Position-based	7.935 ms	1.0360 ms	3.9174 ms
Codebook-based	0.8240 ms	0.2940 ms	0.4858 ms

Figure 34 – Position-based and codebook-based (beam sweeping) RSS comparison.



Source: The author (2025)

6 CONCLUSION

This work presented the development and evaluation of the updated version of the CAVIAR framework, a modular and hybrid co-simulator designed to integrate 3D, communications, mobility, and artificial intelligence within ultra-realistic digital twin environments. By supporting multi-modal sensing, stochastic and site-specific channel integration within 5G-simulated and emulated end-to-end scenarios, and real-time orchestration, CAVIAR enables the investigation of next-generation 5G-Advanced and 6G/B6G use cases, including Open-RAN resource allocation and ISAC scenarios.

Three representative use cases were implemented and evaluated. In the rural scenario, the allocation of PRBs using a RL scheduler ensured high video streaming quality, maintaining throughput above the minimum requirements and achieving consistent user-perceived QoE, as measured by SSIM. In contrast, a baseline round-robin scheduler resulted in lower average throughput and occasional visual degradation. The urban NTN use case demonstrated the capability of CAVIAR to evaluate LEO satellite links for both S-band and Ka-band transmissions, enabling the assessment of channel capacity, spectral efficiency, and resource utilization with high fidelity. In the indoor ISAC scenario, multi-modal sensing combining LiDAR and RGB imagery allowed precise estimation of a human target's 3D position, enabling position-based beamforming to outperform traditional beam sweeping approaches in terms of received signal strength, while maintaining manageable computational overhead.

Across all scenarios, the evaluation of computational resources revealed that GPU and RAM were predominantly utilized for ray tracing and 3D rendering, whereas CPU utilization remained consistent across simulations, with minor variations observed in high-interaction scenarios. The real-time factor analysis confirmed that CAVIAR can execute simulations in near-real time, with deviations primarily introduced by computationally intensive ray-tracing calculations in indoor and rural scenarios. Furthermore, the use of YOLO for object detection and mAP evaluation demonstrated that CAVIAR can integrate AI-driven perception seamlessly within the co-simulation loop.

In summary, CAVIAR provides a comprehensive environment for evaluating and optimizing network performance, resource allocation, and AI integration in digital twin-based simulations. The results of this dissertation confirm the framework's capability to reproduce realistic behavior across diverse scenarios, supporting reproducible and scalable research for emerging 5G-Advanced and 6G systems, mainly under the 3GPP work and study cases.

6.1 Future Work

For future work, it would be valuable to propose new use cases using the same methodology as the 3GPP study cases evaluated in this work, but employing a synchronous execution approach. This could enable the construction of new datasets tailored for these use cases. Additionally, scenarios involving traffic management, such as vehicular or pedestrian flows, could be implemented using platforms like CARLA (DOSOVITSKIY et al., 2017). Specifically, for the proposed indoor use case, creating a closed-loop environment that directly affects the physical twin—for example, a two-way street where physical attributes reflect actions performed virtually—could improve network reliability, especially when employing position-based beamforming. Extending the indoor scenario to include XR would further demonstrate the advantages of 3D simulation.

In the rural scenario, integrating RL for monitoring and decision-making has a clear impact on service quality. However, the current use of fixed thresholds, such as allocating 70% of the available resource blocks, could be replaced by an online training approach in which decisions are only released after reaching an optimal value threshold. This modification would align the framework more closely with current state-of-the-art techniques, making it robust and adaptive. This would enable the network to respond dynamically to changing conditions while maintaining high service quality. Beyond this, the overall integration was made using a single computer, which can lead to multiple overheads during the execution time, leading to a bad overall RTF. These implementations, using NATS, could be deployed in a distributed manner, which decreases the monolithic overhead and enhances the practicality of the co-simulation in the DTcontext.

References

3GPP. *3rd Generation Partnership Project; Technical Specification Group Radio Access Network; Study on New Radio Access Technology; Channel Model for Frequencies from 0.5 to 100 GHz (Release 14)*. [S.l.], 2017. Citado na página 38.

ALKHATEEB, A.; JIANG, S.; CHARAN, G. Real-time digital twins: Vision and research directions for 6g and beyond. *IEEE communications magazine*, IEEE, v. 61, n. 11, p. 128–134, 2023. Citado na página 1.

AOUDIA, F. A. et al. Sionna rt: Technical report. *arXiv preprint arXiv:2504.21719*, 2025. Citado na página 17.

BEHRISCH, M. et al. Sumo–simulation of urban mobility: an overview. In: THINKMIND. *Proceedings of SIMUL 2011, the third international conference on advances in system simulation*. [S.l.], 2011. Citado na página 2.

BORGES, J. et al. Caviar: Co-simulation of 6g communications, 3-d scenarios, and ai for digital twins. *IEEE Internet of Things Journal*, v. 11, n. 19, p. 31287–31300, 2024. Citado 3 vezes nas páginas 3, 12, and 13.

DAO, N.-N. et al. A review on new technologies in 3gpp standards for 5g access and beyond. *Computer Networks*, Elsevier, p. 110370, 2024. Citado na página 1.

DEVELOPERS, T. Tensorflow. *Zenodo*, 2022. Citado na página 17.

DIWAN, T.; ANIRUDH, G.; TEMBHURNE, J. V. Object detection using yolo: challenges, architectural successors, datasets and applications. *multimedia Tools and Applications*, Springer, v. 82, n. 6, p. 9243–9275, 2023. Citado na página 21.

DOSOVITSKIY, A. et al. Carla: An open urban driving simulator. In: PMLR. *Conference on robot learning*. [S.l.], 2017. p. 1–16. Citado 2 vezes nas páginas 2 and 59.

FOWLER, M. *Python Concurrency with asyncio*. [S.l.]: Simon and Schuster, 2022. Citado na página 22.

GIULIANO, R. From 5g-advanced to 6g in 2030: New services, 3gpp advances and enabling technologies. *IEEE Access*, IEEE, 2024. Citado na página 1.

HOYDIS, J. et al. Sionna rt: Differentiable ray tracing for radio propagation modeling. In: IEEE. *2023 IEEE Globecom Workshops (GC Wkshps)*. [S.l.], 2023. p. 317–321. Citado na página 2.

HOYDIS, J. et al. Toward a 6g ai-native air interface. *arXiv preprint arXiv:2012.08285*, 2020. Citado na página 7.

JIA, D. et al. Integrated simulation platform for conventional, connected and automated driving: A design from cyber–physical systems perspective. *Transportation Research Part C: Emerging Technologies*, v. 124, p. 102984, 2021. ISSN 0968-090X. Disponível em: <https://www.sciencedirect.com/science/article/pii/S0968090X2100019X>. Citado na página 3.

KEMPTON, B.; RIEDL, A. Network simulator for large low earth orbit satellite networks. In: IEEE. *ICC 2021-IEEE International Conference on Communications*. [S.l.], 2021. p. 1–6. Citado na página 2.

KLAUTAU, A. et al. Generating mimo channels for 6g virtual worlds using ray-tracing simulations. In: IEEE. *2021 IEEE Statistical Signal Processing Workshop (SSP)*. [S.l.], 2021. p. 595–599. Citado 2 vezes nas páginas 3 and 12.

LIN, X. 3gpp evolution from 5g to 6g: A 10-year retrospective. *arXiv preprint arXiv:2412.21077*, 2024. Citado na página 32.

LIN, X. The bridge toward 6g: 5g-advanced evolution in 3gpp release i9. *IEEE Communications Standards Magazine*, IEEE, v. 9, n. 1, p. 28–35, 2025. Citado na página 32.

LIU, W.-C. et al. Theory and implementation of a three-dimensional spatial channel model for the 3gpp tr 36.873 long-term evolution system. *DEStech Transactions on Engineering and Technology Research*, 2018. Citado na página 8.

MANALASTAS, M. et al. From simulators to digital twins for enabling emerging cellular networks: A tutorial and survey. *IEEE Communications Surveys & Tutorials*, IEEE, 2024. Citado 2 vezes nas páginas 11 and 10.

MANN, T. L. *A network system level simulator for investigating the interworking of wireless LAN and 3G mobile systems*. Tese (Doutorado) — Virginia Tech, 2003. Citado na página 7.

MARCHESE, M.; MOHEDDINE, A.; PATRONE, F. Uav and satellite employment for the internet of things use case. In: *2020 IEEE Aerospace Conference*. [S.l.: s.n.], 2020. p. 1–8. Citado na página 38.

MIAO, W. et al. Position-based beamforming design for uav communications in lte networks. In: IEEE. *ICC 2019-2019 IEEE International Conference on Communications (ICC)*. [S.l.], 2019. p. 1–6. Citado na página 2.

MODESTO, C. et al. Accelerating ray tracing-based wireless channels generation for real-time network digital twins. *arXiv preprint arXiv:2504.09751*, 2025. Citado na página 30.

MOONEY, C. Z. *Monte carlo simulation*. [S.l.]: Sage, 1997. Citado na página 7.

MUSA, D. A.-A. Towards the unraveling of zombie effect in the linux kernel. *Available at SSRN 5020907*, 2024. Citado na página 24.

NAHUM, C. et al. Safeguard urllc services in industry 4.0 through reinforcement learning scheduling. In: *2023 Workshop on Communication Networks and Power Systems (WCNPS)*. [S.l.: s.n.], 2023. p. 1–7. Citado na página 34.

OTTENS, M. et al. Performance evaluation of ns-3 real-time emulation. *IEEE Access*, IEEE, 2025. Citado na página 20.

PEDERSEN, K. I. et al. A tutorial on radio system-level simulations with emphasis on 3gpp 5g-advanced and beyond. *IEEE Communications Surveys & Tutorials*, v. 26, n. 4, p. 2290–2325, 2024. Citado na página 1.

PEGURRI, R. et al. Van3twin: the multi-technology v2x digital twin with ray-tracing in the loop. *arXiv preprint arXiv:2505.14184*, 2025. Citado na página 2.

PEGURRI, R. et al. Toward digital network twins: Integrating sionna rt in ns3 for 6g multi-rat networks simulations. *arXiv preprint arXiv:2501.00372*, 2024. Citado na página 2.

PENG, Y. et al. Integrated multimodal sensing and communication: Challenges, technologies, and architectures. *arXiv preprint arXiv:2506.22507*, 2025. Citado na página 9.

PENG, Y. et al. *Integrated Multimodal Sensing and Communication: Challenges, Technologies, and Architectures*. 2025. Disponível em: <https://arxiv.org/abs/2506.22507>. Citado 2 vezes nas páginas 31 and 42.

QAZZAZ, M. M. et al. Machine learning-based xapp for dynamic resource allocation in o-ran networks. In: IEEE. *2024 IEEE International Conference on Machine Learning for Communication and Networking (ICMLCN)*. [S.l.], 2024. p. 492–497. Citado na página 8.

RANAWEERA, C. et al. 4g to 6g: disruptions and drivers for optical access [invited]. *Journal of Optical Communications and Networking*, v. 14, p. A143–A153, 01 2022. Citado na página 6.

RAVIGLIONE, F. et al. ms-van3t: An integrated multi-stack framework for virtual validation of v2x communication and services. *Computer Communications*, Elsevier, v. 217, p. 70–86, 2024. Citado na página 2.

REYNDERS, B.; WANG, Q.; POLLIN, S. A lorawan module for ns-3: Implementation and evaluation. In: *Proceedings of the 2018 Workshop on ns-3*. [S.l.: s.n.], 2018. p. 61–68. Citado na página 32.

REZAZADEH, F. et al. *GenOnet: Generative Open xG Network Simulation with Multi-Agent LLM and ns-3*. 2024. Disponível em: <https://arxiv.org/abs/2408.13781>. Citado na página 8.

RILEY, G. F.; HENDERSON, T. R. The ns-3 network simulator. In: *Modeling and tools for network simulation*. [S.l.]: Springer, 2010. p. 15–34. Citado 2 vezes nas páginas 2 and 3.

SANDERS, A. *An introduction to Unreal engine 4*. [S.l.]: AK Peters/CRC Press, 2016. Citado na página 3.

SANDRI, M. et al. Implementation of a channel model for non-terrestrial networks in ns-3. In: *Proceedings of the 2023 Workshop on ns-3*. ACM, 2023. (WNS3 2023), p. 28–34. Disponível em: <http://dx.doi.org/10.1145/3592149.3592158>. Citado na página 32.

SHAH, S. et al. Airsim: High-fidelity visual and physical simulation for autonomous vehicles. In: SPRINGER. *Field and service robotics: Results of the 11th international conference*. [S.l.], 2017. p. 621–635. Citado 2 vezes nas páginas 2 and 15.

SONG, Y. et al. 2ace: Spectral profile-driven multi-resolutional compressive sensing for mmwave channel estimation. In: *Proceedings of the Twenty-fourth International Symposium on Theory, Algorithmic Foundations, and Protocol Design for Mobile Networks and Mobile Computing*. [S.l.: s.n.], 2023. p. 41–50. Citado 2 vezes nas páginas 42 and 54.

VARGA, A. Omnet++. In: *Modeling and tools for network simulation*. [S.l.]: Springer, 2010. p. 35–59. Citado na página 7.

VARGAS-MUNOZ, J. E. et al. Openstreetmap: Challenges and opportunities in machine learning and remote sensing. *IEEE Geoscience and Remote Sensing Magazine*, IEEE, v. 9, n. 1, p. 184–199, 2020. Citado na página 12.

VIRDIS, A.; STEA, G.; NARDINI, G. Simulte-a modular system-level simulator for lte/lte-a networks based on omnet++. In: IEEE. *2014 4th International Conference On Simulation And Modeling Methodologies, Technologies And Applications (SIMULTECH)*. [S.l.], 2014. p. 59–70. Citado na página 8.

WALL, A. Systems tool kit (stk). In: *Space System Architecture Analysis and Wargaming*. [S.l.]: CRC Press, 2024. p. 11–32. Citado na página 3.

WIESER, V.; PŠENÁK, V. Wcdma mobile radio network simulator with hybrid link adaptation. *Advances in Electrical Engineering*, p. 200–205, 2005. Citado na página 7.

ZHANG, Z. et al. Artificial intelligence-enabled sensing technologies in the 5g/internet of things era: from virtual reality/augmented reality to the digital twin. *Advanced Intelligent Systems*, Wiley Online Library, v. 4, n. 7, p. 2100228, 2022. Citado na página 6.

ZHAO, Y. et al. *A Communication-Latency-Aware Co-Simulation Platform for Safety and Comfort Evaluation of Cloud-Controlled ICVs*. 2025. Disponível em: <https://arxiv.org/abs/2506.07696>. Citado na página 3.

ZOU, H. et al. Genainet: Enabling wireless collective intelligence via knowledge transfer and reasoning. *IEEE Access*, IEEE, 2025. Citado na página 8.

ZUBOW, A. et al. *Ns3 meets Sionna: Using Realistic Channels in Network Simulation*. 2024. Disponível em: <https://arxiv.org/abs/2412.20524>. Citado na página 8.