6GSNS

Co-funded by the European Union


*SEamless integratioN of efficient 6G WirelesS tEchnologies for Communication and Sensing*

# D4.2 Final design and implementation of Wireless Edge Caching solutions

**April 2026**

**6G-SENSES project has received funding from the Smart Networks and Services Joint Undertaking (SNS JU) under the European Union's Horizon Europe research and innovation programme under Grant Agreement 101139282**

**Project Start Date**: 2024-01-01 **Duration**: 30 months

**Call**: HORIZON-JU-SNS-2023 **Date of delivery**: 2026-04-01

**Topic**: HORIZON-JU-SNS-2023-STREAM-B-01-02 **Version**: 0.1



**Dissemination Level**: Public (PU)

| | |
|---|---|
| **Grant Agreement Number:** | 101139282 |
| **Project Name:** | SEamless integratioN of efficient 6G WirelesS tEchnologies for Communication and Sensing |
| **Project Acronym:** | 6G-SENSES |
| **Document Number:** | **D4.2** |
| **Document Title:** | Final design and implementation of Wireless Edge Caching solutions |
| **Version:** | 1.0 |
| **Delivery Date:** | 2025-12-31 (**2026-04-01**) |
| **Responsible:** | *Sapienza, University of Rome* (**UNIROMA1**) |
| **Editor(s):** | Federico Trombetti (**UNIROMA1**) |
| **Authors:** | Federico Trombetti, Salvatore Pontarelli, Novella Bartolini (**UNIROMA1**),<br>Pavlos Doanis, Markos Anastasopoulos, Anna Tzanakaki (**IASA**), |
| **Keywords:** | 6G, Sensing, Wireless Edge Caching (WEC). |
| **Status:** | Draft |
| **Dissemination Level** | Public (PU) |
| **Project URL:** | https://www.6g-senses.eu |

# Revision History

| Rev. N | Description | Author | Date |
|---|---|---|---|
| 0.1 | First draft layout<br>Table of Contents | Federico Trombetti (UNIROMA1) | 2025-11-12 |
| 0.2 | Updated ToC<br>General content update | Federico Trombetti (UNIROMA1) | 2026-01-19 |
| 0.25 | Updated use case section | Federico Trombetti (UNIROMA1) | 2026-01-20 |
| 0.3 | Revised all sections<br>Document ready for review | Federico Trombetti (UNIROMA1) | 2026-01-23 |
| 0.5 | Revised version | Valerio Frascolla (INT) | 2026-01-30 |
| 0.51 | Merged Review from INT | Federico Trombetti (UNIROMA1) | 2026-01-30 |
| 0.6 | Added content based on INT comments | Federico Trombetti (UNIROMA1) | 2026-02-04 |
| 0.61 | Added content to chapter 4 | Anna Tzanakaki, Markos Anastasopoulos, Pavlos Doanis (IASA) | 2026-02-05 |
| 0.65 | UC Review | Ramón Agüero, Luis Díez (UC) | 2026-02-09 |
| 0.7 | Merged Review from UC | Federico Trombetti (UNIROMA1) | 2026-02-11 |
| 0.8 | Updated chapter 4<br>General content update | Anna Tzanakaki, Markos Anastasopoulos, Pavlos Doanis (IASA) | 2026-02-19 |
| 0.9 | Revision of the chapters | Ramón Agüero (UC),<br>Valerio Frascolla (INT) | 2026-03-03 |
| 0.95 | Revised all sections | Federico Trombetti (UNIROMA1),<br>Anna Tzanakaki (IASA),<br>Jesús Gutiérrez (IHP) | 2026-03-24 |
| 1.0 | Final revision and submission to the Participant Portal | Jesús Gutiérrez (IHP) | 2026-04-01 |

# Table of Contents

# List of Figures

# List of Tables

# Executive Summary

This deliverable provides a complete overview of the work carried out by 6G-SENSES partners in the context of Wireless Edge Caching (WEC) technologies. It elaborates on the strategies developed to perform eviction decisions in the context of wireless sensing, along with the implementation methodologies that were chosen to carry out experiments and performance evaluations.

**D4.2** provides updated content regarding the State-of-the-Art (SotA) research in the WEC domain, with a more detailed overview of topics such as Multi-Access Edge Computing (MEC) and Integrated Sensing and Communication (ISAC) technologies. This deliverable aligns with and extends the work of deliverables **D5.1** [1] and **D2.1** [2], by providing additional insights over the small-scale demo related to 6G-SENSES Proof of Concept #1 (PoC#1), and brings the caching strategies presented in deliverable **D4.1** [3] to a more refined and final version.

To enable practical implementation of the solutions presented herewith, this deliverable describes a set of open source libraries that have been made available to test not only the caching strategies, but also to lay the foundation of a new testing methodology of applications in Robotics and Internet of Things (IoT) environments in the ISAC field [4].

The simulation toolset is based on the technologies already introduced in deliverable **D4.1** [3], i.e. Robot Operating System (ROS) and Gazebo. It provides a new and standardized approach that can be generalized to more use cases and an extensible wireless plugin for Gazebo, providing a more flexible solution to the simulation environment. With the libraries, we implement the use case of the small-scale demo of PoC#1, and evaluate the performances of the Closest In, Farthest Out (CIFO) caching strategy in two different use case contexts.

In addition, the role of WEC in optimizing networked ISAC services is investigated by integrating WEC functionality in the Open RAN (O-RAN)-compliant 6G-SENSES architecture with the aim to support moving-target detection. WEC is used to store sensing outputs specifically range-Doppler radar maps for detecting moving targets and to run Machine Learning (ML) models that estimate target trajectories. It also manages metadata to support both distributed and centralized ML training and inference. By processing and storing data locally at the edge, the approach reduces the need to send large volumes of raw sensing data to central servers, thereby lowering network traffic and computational load.

# 1 Introduction

In recent years, data generated by Radio Access Network (RAN) components has seen exponential growth. With modern applications reaching mobile users everywhere in the world, the demand for a high-data-rate and low-latency services is higher than ever.

Due to these reasons, ISAC [5][6][7] has recently emerged as one of the most actively investigated technologies. ISAC enables sensing capabilities within wireless systems by jointly integrating communication and environmental sensing into a single framework, allowing networks to simultaneously transmit data and perceive their surroundings, using shared spectral and hardware resources. This novel paradigm is slowly establishing itself as a fundamental building block of forthcoming 6G wireless systems, enabling networks to perform sensing tasks that can significantly enhance application performance.

In this context, Work Package (WP) 4 has explored the State-of-the-Art (SotA) of Wireless Edge Caching (WEC) technologies, with an application to a WEC use case. In this deliverable, we present the final design of the caching strategies introduced in deliverable **D4.1** [3] and we detail the full implementation used to carry out testing and evaluation.

Along with the final design of the WEC solutions, we present a lightweight set of libraries, ISAC Libs [4], to enable the development and evaluation of ISAC applications in realistic, dynamic environments, primarily for IoT and robotic applications. The proposed libraries provide a standardized and extensible framework for testing ISAC solutions beyond high-level numerical simulations, thereby enabling assessment of system-level behavior under realistic mobility, environmental interactions, and sensing-communication aspects.

Our libraries are composed of two main components:

- The first one is based on the Robot Operating System (ROS)[1], a well-established set of open-source libraries for developing robotic software solutions. This first part handles the application aspects of the simulations, exposing a simple set of libraries in a Python workspace.
- The second main component is based on Gazebo[2], an open-source simulation platform that helps bringing robotic solutions to life by integrating advanced simulation tools into the development pipeline. In Gazebo, we implement an ISAC Antenna plugin that simulates basic communication between robots by tracking and exposing relevant sensing telemetry values, which can then be used by ISAC applications to enrich and improve application performance at different implementation levels.

The two environments communicate with each other using specific bridge interfaces, allowing for solutions to be implemented seamlessly. ROS and Gazebo together offer an interesting solution to test ISAC applications, not only because they allow to strongly simulate robotic interfaces, but also given the modular nature of the ROS paradigm, allowing structured hardware components to be seamlessly integrated through existing libraries and encapsulated inside the simulation workflow. This development fits very well in the context of development of modern network applications, in particular those pertaining to IoT networks, where hardware heterogeneity is strongly relevant. In such context, providing a standardized approach among different environments is fundamental for the success of new solutions.

To prove the robustness of our solution, we use our libraries to implement a simple Multi-Access Edge Computing (MEC) network. On top of this network, we run a WEC application based on our Closest In, Farthest Out (CIFO) caching strategy, that makes eviction decisions of the data on edge servers, based on the

[1] Robot Operating System (ROS), https://www.ros.org/

[2] Gazebo, https://gazebosim.org/

perceived signal of the communication channel that the ISAC Libs simulates. We additionally provide a simple example demonstrating how the libraries can be used to leverage passive sensing of the communication channel.

In addition to the development of ISAC libraries for ROS and Gazebo, the role of WEC in optimizing networked ISAC services is also investigated. In this context, WEC is integrated into the O-RAN-compliant 6G-SENSES architecture to host the spatio-temporal outputs of sensing applications for moving-target detection. Specifically, sensing outputs are stored in the form of range-Doppler radar maps that capture the location and dynamics of moving targets. These data are subsequently processed by Machine-Learning (ML) models deployed at the edge to estimate target trajectories. To support this processing pipeline, WEC also maintains metadata associated with the ML models, enabling both distributed and centralized training and inference pipelines. By localizing sensing data storage and ML processing at the edge, the proposed approach significantly reduces network traffic and computational overheads that would otherwise arise from transmitting large volumes of raw sensing measurements to a centralized processing location.

To summarize, the contents of the deliverable are;

- Chapter 2 provides a concise overview of the SotA in WEC, with an updated perspective focused on sensing scenarios. It also reviews the SotA in ISAC and discusses the current status of studies that focus on the simulation aspect of ISAC.
- Chapter 3 formulates the details of the small-scale demo for PoC#1 and explores two different use cases for WEC scenarios, one pertaining to a sensing use case already explored in deliverable **D4.1** and another to an MEC application. It also presents a finalized formulation of the CIFO caching strategy, under two different formulations, each of which focuses on one of the aforementioned use-case applications. Moreover, it delves into the details of the ROS and Gazebo simulation framework, with implementation guidelines for testing ISAC applications. ROS provides a structured interface for application logic, while an ad-hoc Gazebo plugin models ISAC sensing and communication related mechanisms. Finally, it reports on the performances of the proposed caching strategies in both the Sensing and the MEC contexts. It confronts, just like in deliverable **D4.1**, the performances of the final solutions against traditional caching strategies such as Random Replacement (RR), First In, First Out (FIFO), Least Recently Used (LRU) and Least Frequently Used (LFU)
- Chapter 4 leverages WEC within the O-RAN-compliant 6G-SENSES architecture to host and process spatio-temporal ISAC sensing outputs. Range–Doppler radar maps are stored and analyzed at the edge using ML models to estimate moving-target trajectories, supporting both centralized and distributed learning.
- Chapter 5 summarizes the content of the deliverable.

# 2 State-of-the-Art of WEC in Sensing and ISAC Applications

## 2.1 WEC and Sensing

Sensing has become a fundamental component of modern edge and IoT systems, enabling real-time data collection and rapid, responsive decision making in domains such as environmental monitoring, smart cities [8], and wildlife monitoring [9][10][11].

WEC refers to a novel approach to distributed architectures, aimed at storing content at the edge of a wireless network, using devices such as base stations (BSs) and user terminals [12][13]. This method is designed to accommodate the recent surge in mobile devices and data-hungry applications. With recent infrastructure advancements, the deployment of WEC techniques has become an appealing solution.

Efficiently deploying WEC solutions means tackling multiple challenges to meet specific application requirements. The way caching should behave in this context deeply differs from traditional caching paradigms which tend to retain data based only on past access; effective caching in this scenario must be context-aware and adapt to the characteristics of its application.

In the context of sensing, WEC has had limited application. This is mainly due to the stringent requirements of the sensing tasks, where data are typically highly sensitive to time [14], making caching a less attractive solution. However, the amount of data produced by such sensing applications is often extremely high [8][15].

Differently from previous works, our location-based eviction policy represents a more general and adaptable approach, enabling support for a wider range of applications beyond the specific scenarios considered beforehand.

## 2.2 ISAC and WEC

ISAC has recently emerged as a compelling paradigm for future wireless systems, mainly driven by the increased efficiency requirements of 6G technologies. This research has spanned multiple topics and focus areas, depending on the research scope.

A significant part of the ongoing work in the SotA of ISAC study highly emphasizes waveform design and signal processing algorithms. Works in [16][17][18] demonstrate the theoretical trade-offs between sensing and communication objectives and practical strategies for waveform superposition to achieve dual functionality in shared time-frequency resources. These works demonstrate that a significant boost can be obtained by intelligently modulating the communication signal.

Separated from the algorithmic waveform study of ISAC, a few recent efforts, like the work in [19], recognize the need for a simulation environment that supports detection and localization within ISAC systems. Such a framework aims to approximate sensing and communication under stochastic channel models. These abstraction-based approaches represent a crucial step toward bridging the gap between theoretical ISAC designs and practical system evaluations, enabling the study of performance trade-offs and resource allocation strategies at a higher level of abstraction.

Beyond the study of physical (PHY)-layer waveform and signal processing, a recent ISAC research direction has steered towards the exploration of network-level aspects, such as resource allocation and sensing-assisted scheduling [20]. In such works, sensing information is exploited to support higher-layer decision-making.

However, many of these works often rely on abstract channel and mobility models, limiting their ability to capture tightly the coupling of different factors such as sensing accuracy, device mobility, and environmental interaction. As a result, the evaluation of these ISAC systems often remains disconnected from realistic deployments that involve complex, dynamic scenarios.

The work of Yunhao et al. [21] discusses how ISAC is gradually driving a paradigm shift in IoT system design. Traditionally, IoT devices would have separate sensing and communication subsystems, each implemented with specialized hardware and independent processing pipelines. In such architectures, sensing data is collected first and then transmitted, often with little cooperation between the two functionalities. ISAC changes this by enabling joint design and operation of sensing and communication on shared hardware and spectrum. This integration can reduce hardware redundancy, lower system costs, and improve energy efficiency by reusing the same physical components, such as antennas, for both functions, rather than maintaining separate stacks for each task. Moreover, the convergence of sensing and communication layers toward a unified layer facilitates mutual assistance between the two functions, allowing sensing measurements to inform communication strategies and vice versa, which can improve overall system performance and responsiveness in IoT environment.

ISAC applications share several common traits and are often applied to use cases that are deeply dependent on user mobility [22], with a strong emphasis on the interactions between devices and their surrounding environment. The performance of the sensing task in such an application is inherently influenced by these characteristics. This tight coupling between the application and its operational context makes it challenging to bridge formal, abstract studies with real-world deployment and evaluation.

Existing tools often lack support for realistic mobility, environmental interaction, and cross-layer experimentation. For this reason, robotic simulators such as Gazebo, when combined with the modular software abstractions provided by ROS, offer an attractive environment for evaluating ISAC solutions in scenarios that more closely resemble real-world deployments.

# 3 6G-SENSES WEC in sensing

## 3.1 Testing strategy and small-scale PoC

In **D5.1** [1] we presented the layout of the small-scale demo associated with 6G-SENSES PoC#1 to demonstrate the developed WEC strategies. In what follows, we briefly recall its layout for completeness. In addition, we further elaborate on the application use cases to which these strategies are applied, while also considering more general use-case scenarios.

To demonstrate the applicability of our strategies, we perform different levels of testing across different simulation instances:

1. **Phase#1 - Custom Simulation:** the first set of iterations, which focuses on delivering a solid caching mechanism in terms of numerical evaluation. Its main focus is to provide a lightweight simulator capable of comparing caching strategies in a more abstract environment, without taking into consideration the physical characteristics of the RAN sensing use case.
2. **Phase#2 - ROS-Gazebo Simulation:** the second set of testing iterations, which introduces a more realistic environment to the simulation. With the integration of ROS and Gazebo, the simulation is executed in a more standardized environment, with physically accurate objects. This allows the simulation of closer to reality scenarios, with multiple interacting actors taking part, thus creating a more realistic environment for evaluating and comparing the caching solutions.
3. **Phase#3 - ROS-Gazebo + RAN Signal Simulation:** the final phase of testing, which extends the work of the previous phase to also simulate the RAN wireless sensing signal, and the cache data produced by the sensing use case. It aligns with the objectives of 6G-SENSES of providing solutions directly tied to the small-scale demo of PoC#1.

The first two phases have been addressed so far in deliverable **D4.1** [3]. In the present deliverable, we build upon the already presented caching strategies by first performing a set of tests on the environment of Phase#1, and then moving on directly to Phase#3 by implementing the same strategies in the context of MEC networks and ISAC sensing.

The proposed solution in **Phase#3** will be executed on a specific ROS xApp deployed on the Radio Intelligent Controller (RIC). As shown in Figure 3-1, the xApp takes in input sensing data coming from PoC#1 radar-based technologies, and uses such data to perform more accurate simulations, i.e. with higher fidelity. We collect data with a running ad-hoc application able to communicate with ROS topics. These topics share the same namespace of the Gazebo environment, and enable a seamless link between the real deployment and the simulated environment. This allows for a coupled solution that is able to test more elaborate instances and where smart caching decision can be of great help to the sensing application, without the need to allocate real, expensive, hardware. More details on this demo will be provided in deliverables **D5.2** and **D5.3**.

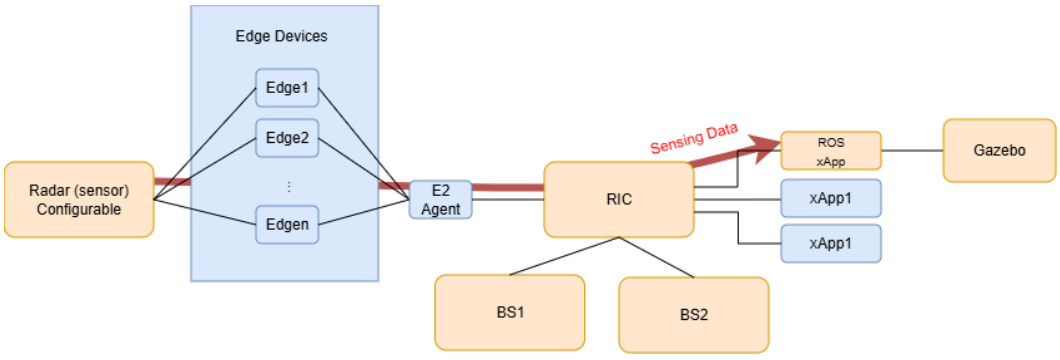



**Figure 3-1 WEC small scale demo integration in PoC#1**

## 3.2 Use cases

### 3.2.1 Wireless Sensing Use Case

The objective of the sensing use case is the continuous monitoring of a set of targets moving within a given area. This monitoring paradigm applies to a wide range of tracking applications, including wildlife monitoring, asset tracking, and mobile object surveillance, where information about target location or status must be collected and made available to a central entity.

In this context, sensing devices distributed over the field generate data associated with one or more targets, while a BS periodically issues queries to retrieve the most recent information available. Given the mobility of the targets and the limited storage and communication capabilities of edge devices, it is not always possible to guarantee that the requested data is locally available at the time of the query.

We consider an optimization problem in terms of HIT and MISS rates. A HIT event occurs when a request from the BS for a data from a particular sensor is satisfied, otherwise, it is considered a MISS.

The targeted KPIs are:

- Energy Consumption: we measure the amount of energy consumed in terms of sensor activations, i.e., how many times sensors are activated by a given caching policy.
- End-to-End (E2E) Delay: we measure the average time it takes to satisfy BS requests, given standardized times on tasks such as data retrieval and cache interrogation.

We perform the sensing task through a set of sensing devices, which is deployed across the area of interest. Each sensor is capable of continuously collecting environmental data and broadcasting the gathered information in a pre-defined, limited, communication range. In addition to the sensors, a set of edge devices is deployed to interact with the sensing infrastructure and to respond to data requests issued by a central BS.

Sensing activities are performed continuously, with data requests evaluated at fixed, discrete time intervals. At each of these intervals, the BS issues a sensing query related to a specific target of interest in the monitored area. The objective of each query is to retrieve the most recent and non-expired sensing data generated by the sensor located closest to the target.

To reduce latency and communication overhead, sensing data can be cached at edge devices. When a query is issued, it can be served by any edge device that already stores the required data in its cache, provided that the data has not expired. Each cached data item is associated with a freshness indicator that reflects its temporal validity; once this freshness falls below a predefined threshold, the data are considered outdated and cannot be used to satisfy requests.

The performance of the system is evaluated in terms of cache efficiency, focusing on HIT and MISS metrics. We define a cache HIT event when a query issued by the BS is successfully satisfied by at least one edge device using cached data. A cache MISS occurs instead when none of the edge devices is able to provide the required data.

Figure 3-2 shows a view of the simplified environment running the demo simulation. The environment offers live view of edge devices (orange dots), sensors (blue dots) and targets (green dots), while the right-most part of the view displays in real time the cache content of the edge devices.

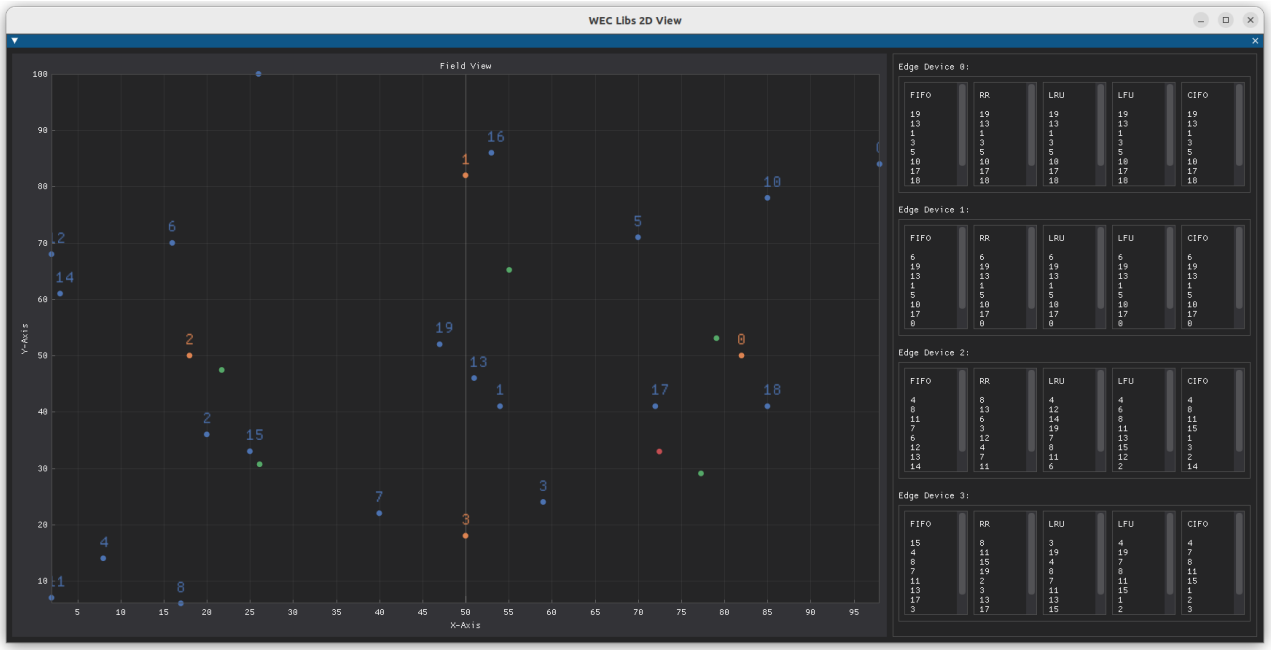



**Figure 3-2 A 2D view of the Phase#1 simulation, evaluating different caching strategies**

Our proposed solution addresses key aspects of WEC architectures in the context of sensing. Although the concept of caching in mobile-assisted environments is not new, it has largely been confined to theoretical explorations. The technologies necessary to fully implement these ideas, particularly those that support the dynamic and flexible nature of mobility, are still in the early stages of development. As a result, practical applications have been limited.

Our WEC solutions highlight the importance of domain-specific caching strategies in edge-assisted IoT systems, paving the way for more adaptive and efficient architectures in data-intensive sensing environments.

### 3.2.2 MEC Application Use Case

In this use case, we aim to enhance the performance experienced by end-user devices operating on a field, by leveraging edge computing caching mechanisms. With this objective, we not only aim to reduce latency through edge computing, but to improve the cache eviction strategies at the MEC layer under dynamic wireless and mobility conditions. While conventional MEC solutions rely mostly on content popularity, they often overlook the real-time state of wireless channels and the mobility of user devices.

We demonstrate how location-aware caching, enabled by sensing information exposed through the ISAC libraries, can be exploited to adapt cache contents to the current communication context. Specifically, the proposed approach leverages channel-quality indicators and proximity information to prioritize the storage of data that is more likely to be accessed by nearby users, thereby reducing cache MISS events.

The primary performance metric (KPI) considered in this use case is the **MISS RATE**, defined as the fraction of requests that cannot be satisfied by cached data at the edge. The evaluation focuses on analyzing how the proposed CIFO caching strategy performs under different system configurations and operating conditions by varying multiple parameters. A lower MISS rate directly reflects improved caching efficiency and better application-level performance.

We use the ISAC libraries to model a case of MEC infrastructure, where edge devices make eviction decision on content stored, depending on ISAC sensed data when communicating with the end users.

Figure 3-3 shows an example instance of Gazebo running the use case. In this particular instance, entities are spawned close together for a better visualization. The figure shows edge devices (the blue box), antenna relays, and users taking part in the simulation.

Our architecture (Figure 3-4) implements three different ISAC devices, with a central application controller evaluating the performance of different cache strategies.

- **Edge Device**: the device responsible for capturing data from the antenna relays. It serves as an intermediary between the end user and the main application controller. Its task is to cache the most "useful" data for the application, depending on the requests from the ground users.
- **Antenna Relay**: the end-point device serving applications to users. It processes user requests within its coverage area and forwards them to its associated edge device. Upon startup, an antenna relay broadcasts messages to all edge devices and negotiates its associated edge device based on the received signal strength of their responses.
- **User**: the end user of the application. A set of users is spawned randomly on the field and moves towards random objectives inside the area covered by the application. At random intervals, they communicate with the closest antenna relay and request data from the application. The user periodically sends HELLO packets in broadcast, notifying other devices of its presence, and allowing the other device to keep up-to-date information of the telemetry data.

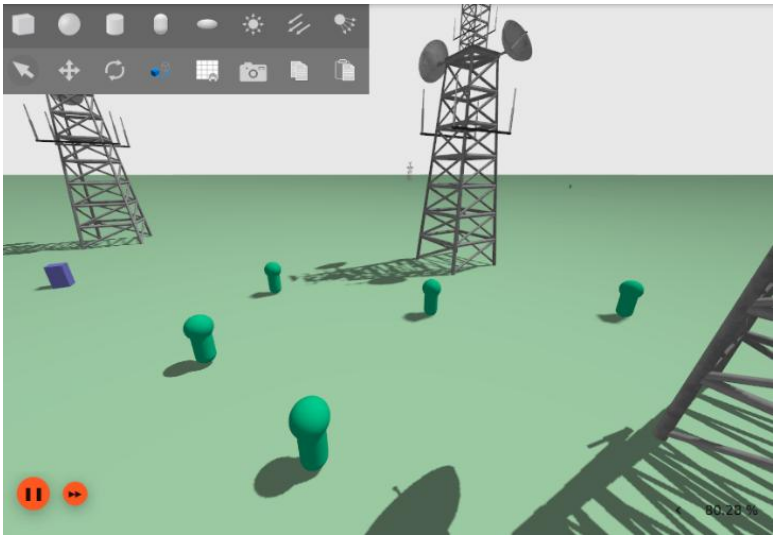


**Figure 3-3 MEC Application running in Gazebo**

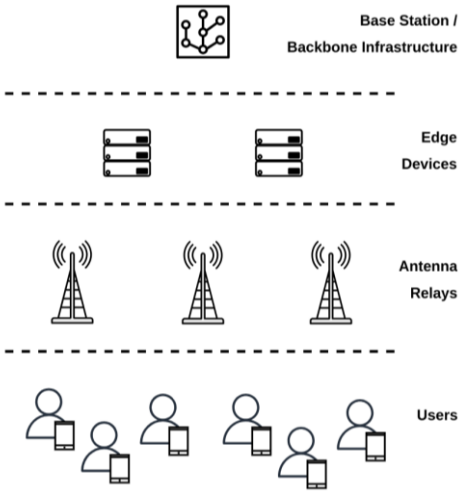



**Figure 3-4 MEC architecture deployed**

The proposed MEC Application use case leverages the CIFO caching strategy already presented, enabling a new class of optimizations for ISAC-sensed data at the wireless edge. Unlike traditional caching approaches that rely solely on content popularity or request frequency, the proposed solution integrates sensing-aware and context-aware information into the caching decision process.

By exploiting and analyzing signal strength within the MEC framework, the proposed solution demonstrates how wireless edge caching can evolve from static, content-centric mechanisms to intelligent, context-driven systems suitable for future 6G networks.

### 3.3 Location-Aware Caching Strategy

In this section, we present the CIFO caching strategy that is adopted across both the sensing and the MEC application use cases. The proposed strategy is designed to efficiently manage sensing data at the edge by considering data location. By prioritizing caching decisions that improve data availability while respecting freshness constraints, CIFO aims to reduce cache misses and enhance the responsiveness of MEC applications. The following subsections describe the main design principles of the strategy.

We present the CIFO strategy in Algorithm 1. The algorithm is run by a caching device, $\boldsymbol{e}$, which keeps in memory its caching table, $\boldsymbol{C_e}$, the latest request from the BS, $\boldsymbol{r^t}$, and a view of the sensors on the field, $\boldsymbol{S}$.

**Algorithm 1:** CIFO Caching Algorithm

```
1  C_e ← cache table of device e;
2  r^t ← latest request from BS;
3  S ← set of sensors on the field;
4  Event: Data received e_s^t
5      if s ∈ C_e then
6          update entry for sensor s;
7      else
8          if C_e is full then
9              s_farthest ← arg max_{s∈C_e} ||s − r^t||;
10             C_e = C_e \ s_farthest;
11         C_e = C_e ∪ e_s^t;
12 Event: Request received r^{t_last}
13     r^t ← r^{t_last};
14     s_closest ← arg min_{s∈S} ||s − g||;
15     if s_closest ∈ C_e then
16         return s_closest;
17     else
18         request sensing task to s_closest;
```

The execution of the policy is driven mainly by two events:

- The first event is triggered when a sensor broadcasts new data (line 4). This broadcast could either be triggered by a passive sensing activity by the sensor or actively by an edge device previously requesting the sensing data. When the data is received, the edge device checks if it already has old sensing data for that sensor (line 5), and updates it accordingly. If the data is not present, it means that a new entry for it must be generated. If the cache is full, the device first finds the farthest sensing data from the latest queried position, $\boldsymbol{r^t}$, (line 9) and evicts it.
- The second event is triggered when the edge device receives a query request from the BS (line 12). When this happens, the edge device first updates its last queried position, and then verifies which sensor data is needed to satisfy the BS's request, which is given by the closest sensor (line 14). If the data for that sensor is present in its memory, $\boldsymbol{C_e}$, then it is promptly returned; otherwise, a sensing task is requested to that sensor.

### 3.3.1 CIFO Extension to Multiple Targets in a Sensing Use Case

The CIFO algorithm presented is useful for tracking the position of a single target accurately when applied in an active sensing context. When multiple targets need to be tracked, and the data needed for the tracking is generated by multiple sensors together, requests for distant targets are likely to result in cache misses, unless some portion of the caching storage is reserved for them.

Partitioning the cache is one possible strategy to ensure such coverage. We hereby define the data handling strategy used to support multiple-target scenarios. Note that this introduces some parametrization which is application-specific, and needs to be tuned on a defined use-case scenario, depending on how the main BS is interested in querying the different targets. We partition the storage of an edge device into multiple sections, up to $|G|$, where each section behaves as a single CIFO cache table on its own. The way memory is allocated among these partitions is given by a focus factor $\lambda \in [0, 1]$, which determines the normalized amount of space allocated for the most recent requested target, with the remaining space equally shared between other targets. A $\lambda$ value of 1 means that all the space is reserved for the latest queried target, and a value of 0 means that the space is evenly distributed among the G targets. Formally, the fraction for the allocated space of the latest target ls is computed as follows:

$$l_s = \lambda + \frac{1-\lambda}{|G|} \tag{3-1}$$

The maximum allocated space for each target expands and shrinks, following the queries from the BS. This maximum allocated space for each target is considered a ”soft” bound, meaning that eviction from other targets happens only when the latest target partition does not have enough space to store the newly received data.

### 3.3.2 CIFO Formulation in a MEC Use Case

The CIFO caching strategy is applicable across different application scenarios. For the second use case introduced in Chapter 3, CIFO is employed to manage the caching of user session information at the edge. This session data includes both application data and ISAC sensing data, which are used to enhance application performance and support sensing functionalities.

In this context, instead of computing distances based on the location of the data source, CIFO uses directly the wireless signal strength perceived by the antenna devices. These signal strength measurements are used as the metric for CIFO cache eviction decisions. This approach, also thanks to the simulated wireless signals, simplifies the sensing model and enables the deployment of CIFO in a more realistic and practical scenario.

Furthermore, data requests in this use case are generated directly by users operating in the field, rather than being issued by a central BS. Each user is associated with a session data object that can be cached at edge devices to improve responsiveness and reduce access latency.

For this use case, data generated is associated directly with an end user, rather than with individual sensing devices, making an explicit formulation of CIFO for multiple sensors no longer required. This design choice allows for a more light-weight implementation of the caching strategy, further improving its efficiency and overall performance. The results of this improvement can be appreciated in the section 3.5.

## 3.4 Simulation Framework

In this chapter, we present the implementation details of the ISAC libraries, showing how the use cases are set up for testing. We explain how the different components work together in the simulation environment and how the system is configured to evaluate performance in realistic scenarios.

### 3.4.1 ROS & Gazebo Simulation Framework

Figure 3-5 shows the workflow of the libraries, mostly common to ROS and Gazebo development pipelines. ROS acts as the central component, handling application behavior and carrying implementation methodologies. On the right, the simulation part of Gazebo is handled using the ad-hoc ***ISACAntenna*** sensor plugin. On the left, ROS communicates with real hardware, through custom interfaces, using the same exact behavior implemented for the simulated environment.

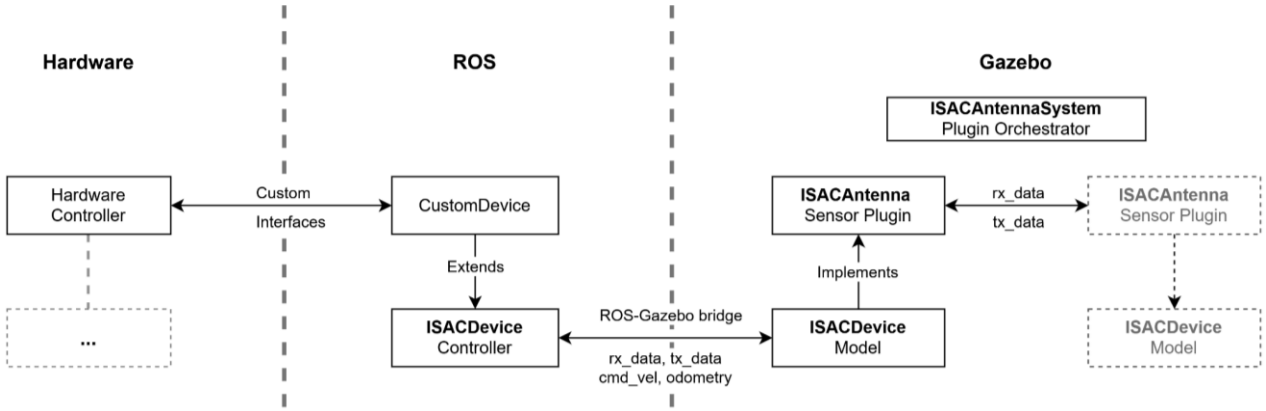



**Figure 3-5 ISAC Libs Infrastructure**

Every simulated ISAC device is built on top of an ***ISACDeviceController*** ROS node, which is the base class in the ISAC libraries. When extended, this class automatically connects to a Gazebo entity implementing an ***ISACAntenna*** sensor, bridging ROS topics for ISAC messages (***rx_data*** and ***tx_data***) as well as for movement (***cmd_vel*** and ***odometry***).

This behavior is implemented mainly in two ROS packages:

- ***isac_libs_main***: This is the main package that contains the ROS code for implementing application behavior. It includes the ***ISACDeviceController*** ROS node class, along with utilities for dynamically spawning an ISAC device and a corresponding model within the simulation.
- ***isac_libs_resources***: This package contains all the resources used by Gazebo to simulate the ISAC hardware. It includes the ***ISACAntenna*** sensor plugin and the ***ISACAntennaSystem*** plugin, which orchestrates the simulation, along with predefined Gazebo models.

Additionally, all behavior related to the MEC caching application is stored in the ***isac_libs_mec_example*** package, while the ***isac_libs_sensing_example*** package contains the passive sensing example.

The following tables present the ISAC Libs ROS Packages.

**Table 3-1 isac_libs_main**

| *Package: isac_libs_main* |
|---|
| **Package Description:**<br>The main package hosting ROS code for implementing application behavior. It contains the ***ISACDeviceController*** ROS node class, along with utilities for dynamically spawning an ISAC device and a model within the simulation. |
| **Package Content:**<br>• **ISACDeviceController**: the main class of the libraries. When used to spawn an ISAC device, it automatically links the controller with all the topics used in the simulation. |

- ***launch_utils.py***: Contains some function to support a ROS 2 Python Launch File.
  - ***create_gazebo_launch_description()***: Used to return a list of ROS 2 Launch Nodes which start Gazebo along with important bridges for world clock and world control.
  - ***create_isac_device_launch_description()***: Used to launch a ROS 2 node which extends an **ISACDeviceController** class, it automatically sets ups all namespaces correctly, and starts bridges for the topics used by the device (tx_data, rx_data, cmd_vel, odometry)
  - ***spawn_sdf***(): Used to dynamically spawn a Gazebo sdf model at startup, automatically renaming conflicting instances.
- ***event_scheduler.py***: Contains the ***EventScheduler*** class. This class serves as a Gazebo-synced clock utility, which is to be used for events that need to happen in actual simulation time; as Gazebo may run slower or faster than real time depending on simulation complexity and world parameters.

**Table 3-2 isac_libs_resources**

| *Package: isac_libs_resources* |
|---|
| **Package Description:**<br>The package containing all the resources used by Gazebo to simulate the ISAC hardware. It contains the previously mentioned *ISACAntenna* sensor plugin and *ISACAntennaSystem*, the plugin orchestrator, along with predefined Gazebo models. |
| **Package Content:**<br>• ***ISACAntennaPlugin:*** the sensor plugin attached to entities that participate in the ISAC message exchange<br>• ***ISACAntennaSystem:*** the orchestrator system plugin which handles message exchange between ISACAntenna-enabled devices<br>• Different Gazebo models to spawn directly inside a simulated environment. |

**Table 3-3 isac_libs_mec_use_case**

| *Package: isac_libs_mec_use_case* |
|---|
| **Package Description:**<br>Contains the MEC Example demo. The demo can be started directly with the following launch file:<br>`ros2 launch isac_libs_mec_example main.launch.py`<br>The default simulation spawns 1 edge device, 3 relays, and 1 user. Parameters can be given to change multiple simulation settings. |
| **Package Content:**<br>• **ISAC Devices**:<br>  ○ **EdgeDeviceController**: defines the ISAC device for an Edge Device, handling caching behavior for received data.<br>  ○ **AntennaRelayController**: defines the ISAC device for an Antenna Relay, handling communication between users and edge devices, and forwarding messages accordingly. Upon startup, an Antenna Relay chooses its assigned edge devices, by picking the one with the strongest perceived signal. |

- **AgentController**: defines the ISAC device for a User, the final end-user leveraging the application on the field. A user moves randomly on the field, and queries the main application at random intervals, interrogating always the closest Antenna Relay.

- **Orchestrators**:
  - **ApplicationController**: listen to messages of Edge Devices, and logs performances of the different cache strategies accordingly.
  - **AgentNetworkManager**: coordinates the set of users in the field, and instructs actions to all of them in a centralized approach.

**Table 3-4 isac_libs_sensing_use_case**

| *Package: isac_libs_sensing_use_case* |
|---|
| **Package Description:**<br>Contains the Passive Sensing Example demo. The demo can be started directly with the following launch file:<br>`ros2 launch isac_libs_sensing_example main.launch.py` |
| **Package Content:**<br>• **AgentController**: the controller which allows for the users to simply moves back and forth in the simulated world<br>• **IsacSignalDisplay**: a 2d view which shows with a live plot the signal strength of one of the moving users. |

### 3.4.2 RAN Gazebo Plugin

**Table 3-5 ISACAntennaSystem Plugin**

| *Gazebo Plugin: ISACAntennaSystem* |
|---|
| **Plugin Initialization:**<br>The plugin can be initialized simply by adding it to the system plugins of a simulated world. |
| **Plugin Description:**<br>This is the orchestrator system plugin which handles message exchange between ***ISACAntenna***-enabled devices. This orchestrator listens to the tx data topics of the enabled devices and properly transmits data in the simulation to the receiving rx data interfaces in range. The plugin estimates the received signal strength at the destination as a function of inter-node distance, transmit power, path-loss attenuation, and small-scale fading; additionally, we characterize achievable channel bandwidth by using Shannon theorem. All this computation is handled by a ***computeReceivedSignal*** function of the **ISACAntennaSystem** plugin, so further modifications are possible depending on how much detail is needed in modeling the studied use case.<br>The plugin additionally keeps tracks of solid objects in the simulation, which further attenuate the signal when between source and destination. |

**Table 3-6 ISACAntenna Plugin**

| *Gazebo Plugin: ISACAntenna* |
|---|
| **Plugin Initialization:**<br>An **ISACAntenna** sensor can be configured with the following parameters: |

- ***tx_power_dBm***: The transmitted power of the antenna, expressed in dBm, which determines the signal strength sent into the environment.
- ***path_loss_exponent***: A factor that models the attenuation of the signal with distance, typically dependent on the environment (e.g., free space, urban, indoor).
- ***fading_std***: The standard deviation of the small-scale fading, representing random fluctuations of the signal due to multipath propagation.
- ***bandwidth_Hz***: The operational bandwidth of the antenna.
- ***noise_figure_dB***: The additional noise introduced by the receiver electronics, expressed in dB, used to compute the effective noise power for SNR and packet reception calculations.

**Plugin Description:**

This is the sensor plugin attached to entities that participate in the ISAC message exchange. An object implementing this sensor advertises two main topics, *tx_data* for sending messages, and *rx_data*, for receiving. All data received in *rx_data* is additionally enriched with ISAC-sensed data to measure the quality of the channel on which the message was received. These topics are bridged to the ROS control plane and can be used by any application running on top of it. All data sent to *tx_data* is immediately broadcast inside the simulated environment and propagated according to the properties of the antenna.

Broadcast and propagation behavior is defined by the ***ISACAntennaSystem*** plugin.

#### 3.4.2.1 Using the plugins in Gazebo

The plugin can be used inside a Gazebo simulation by adding first the **ISACAntennaSystem** system plugin to the Gazebo sdf world.

```
<plugin
 filename="ISACAntennaSystem"
 name="custom::ISACAntennaSystem">
</plugin>
```

After that, any model inside the simulation can use its own **ISACAntenna** Sensor, which can be attached to a link of the model and configured with multiple parameters as follows:

```
<sensor name="ISACAntennaSensor" type="custom" ignition:type="ISACAntenna">
 <always_on>1</always_on>
 <update_rate>30</update_rate>
 <visualize>true</visualize>
 <ignition:ISACAntenna>
  <tx_power_dBm>15</tx_power_dBm>
  <path_loss_exponent>3.2</path_loss_exponent>
  <fading_std>4.0</fading_std>
  <bandwidth_Hz>1000000</bandwidth_Hz>
  <noise_figure_dB>5.0</noise_figure_dB>
 </ignition:ISACAntenna>
</sensor>
```

Models inside ***isac_libs_resources*** already come with their **ISACAntenna** sensor configured, with different parameters set depending on the device.

#### 3.4.2.2 Wireless Signal

The wireless signal is simulated by the ***ISACAntennaSystem*** orchestrator system plugin which handles message exchange between **ISACAntenna**-enabled devices. The orchestrator listens to the *tx_data* topics of the enabled devices and properly transmits data in the simulation to the receiving *rx_data* interfaces in range.

The plugin estimates the received signal strength at the destination as a function of inter-node distance, transmit power, path-loss attenuation, and small-scale fading. Additionally, it characterizes achievable channel bandwidth using the Shannon theorem.

All of this computation is handled by the *computeReceivedSignal* function of the ***ISACAntennaSystem*** plugin, allowing further modifications depending on how much detail is needed for modeling the studied use case. The plugin also keeps track of solid objects in the simulation, which further attenuate the signal when positioned between source and destination.

We compute the total signal strength of a message $P_{rx}$ outgoing from a source interface tx as follows:

1. Distance between transmitter and receiver:

$$d = \| \mathbf{p}_{\text{rx}} - \mathbf{p}_{\text{tx}} \| \tag{3-2}$$

2. Received power:

$$P_{\text{rx,dBm}} = P_{\text{tx,dBm}} - (10\, n \log_{10}(d) + L_{\text{obs}}) + X_{\sigma} \tag{3-3}$$

2. Small-scale fading:

$$X_{\sigma} \sim \mathcal{N}(0, \sigma^2) \tag{3-4}$$

3. Receiver noise power:

$$P_{\text{noise,dBm}} = 10\log_{10}(k_B T B) + 30 + \text{NF} \tag{3-5}$$

4. Signal-to-noise ratio in dB:

$$\text{SNR}_{\text{dB}} = P_{\text{rx,dBm}} - P_{\text{noise,dBm}} \tag{3-6}$$

5. SNR in linear scale:

$$\text{SNR}_{\text{linear}} = 10^{\text{SNR}_{\text{dB}}/10} \tag{3-7}$$

6. Achievable rate:

$$R = B\log_2(1 + \text{SNR}_{\text{linear}}) \tag{3-8}$$

In the above equations:

- $d$ is the Euclidean distance between transmitter and receiver positions.
- $P_{\text{tx,dBm}}$ is the transmit power in dBm, and $P_{\text{rx,dBm}}$ is the received power after accounting for path loss, obstacles, and fading.
- The path loss exponent $n$ depends on the environment.
- $L_{\text{obs}}$ is the attenuation due to obstacles.
- $X_{\sigma} \sim \mathcal{N}(0, \sigma^2)$ models small-scale fading.
- $P_{\text{noise,dBm}}$ is computed from thermal noise and the receiver noise figure (NF), with $k_B$ as Boltzmann's constant, $T$ as absolute temperature (Kelvin), and $B$ as channel bandwidth (Hz).
- The SNR in dB is converted to linear scale for computing the achievable rate using the Shannon-Hartley theorem.

The total attenuation due to obstacles along the propagation path is modeled as:

$$L_{\text{obs}} = \sum_{i=1}^{N_{\text{obs}}} t_i \ \beta_i \tag{3-9}$$

where:

- $N_{\text{obs}}$ is the number of obstacles intersecting the line-of-sight between transmitter and receiver.
- $t_i$ is the effective thickness of the $i$-th obstacle along the path.
- $\beta_i$ is the attenuation coefficient in dB/m.

To speed up collision calculations, all obstacles in the simulation are treated as Axis-Aligned Bounding Boxes (AABB). This allows for simple determination of which objects lie between the source and destination.

## 3.5 Performance Evaluation

### 3.5.1 Sensing Application

We test our new CIFO in a setting of sensing of tracking, as described by the sensing use case. A single target is initially deployed on the field, which moves during the simulation by progressively choosing a new random destination in the covered area.

To assess the effectiveness of the proposed CIFO caching algorithm, we conduct a series of simulations across different scenarios. Our evaluation focuses on key performance metrics such as energy consumption and end-to-end delay. To highlight the advantages of our solution in handling location-sensitive data, we compare CIFO against other widely used standard caching strategies: LRU, LFU, FIFO, and RR.

Each simulation instance reflects the dynamics of wireless sensing applications, where sensor-generated data is continuously produced and queried by a central BS via intermediate edge devices. Different factors are varied in the tests, including the number of edge devices and their cache size, the query rate of the BS, and the speed of the tracked targets.

All tests are run for a duration of four minutes, starting from the initial deployment of edge devices with empty cache tables. Four different sets of tests are performed, with each set varying a single parameter. The parameters for each configuration are reported in Table 3-7. Each bar of the box plot refers to the distribution of fifteen different results.

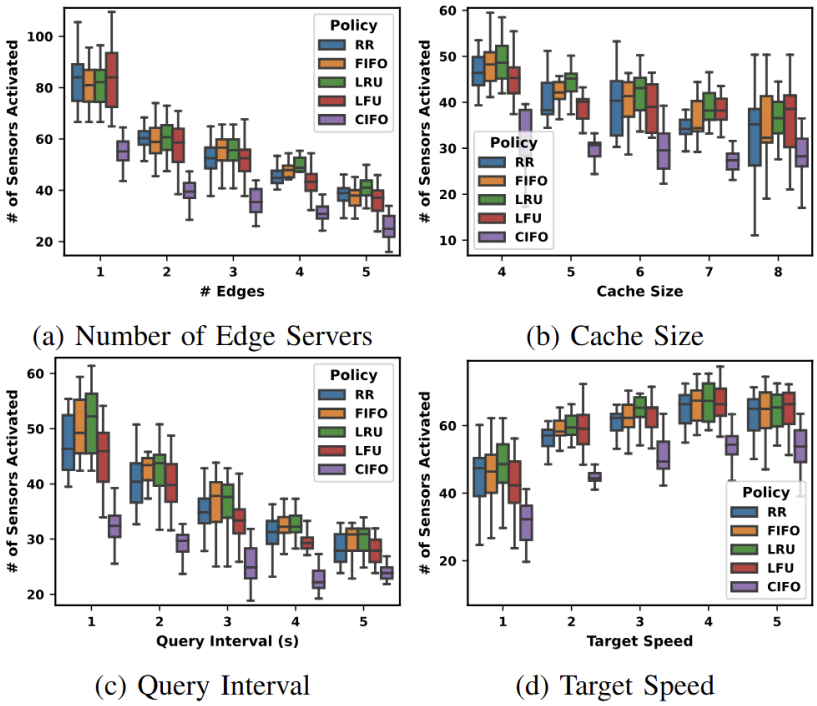
**Table 3-7 Evaluation Setup**

| Parameter | Default Value | Varying Range |
|---|---|---|
| **Simulation Time** | 240 s | N/A |
| **Edge Servers** | 4 | 1–5 |
| **Sensors** | 40 | N/A |
| **Cache Size** | 4 | 4–8 |
| **Query Interval** | 1 s | 1–5 |
| **Target Speed** | 1 m/s | 1–5 |
| **Freshness Threshold** | 10 s | N/A |

#### 3.5.1.1 Energy Consumption

We analyze the performance of all strategies in terms of energy consumption. Energy is computed based on the number of times sensing devices are activated, either through passive sensing or active requests triggered by a cache MISS from the BS.

The results show that the CIFO caching strategy consistently achieves lower energy consumption across all evaluation settings, reducing sensor activation by roughly 30% compared to other policies (see Figure 3-6).



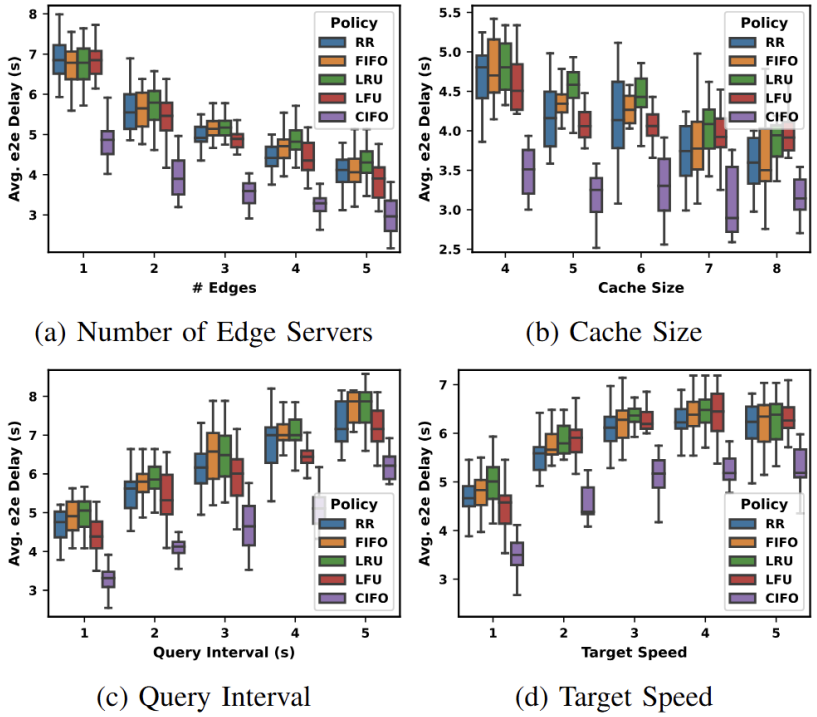

(a) Number of Edge Servers (b) Cache Size

(c) Query Interval (d) Target Speed

**Figure 3-6 Sensing Application, Evaluation Results – Energy Consumption**

### 3.5.1.2 E2E Delay

The average E2E delay is evaluated by assigning a 1-sec delay to cache HIT events and a 10-sec delay to MISS events, reflecting the higher latency of MISS events. Across all configurations, according to Figure 3-7, CIFO consistently achieves the lowest end-to-end delay, mirroring the trends observed in the energy consumption evaluation.




(a) Number of Edge Servers (b) Cache Size

(c) Query Interval (d) Target Speed

**Figure 3-7 Sensing Application, Evaluation Results – E2E Delay**

#### 3.5.1.3 Multiple Targets

We also evaluate CIFO when monitoring multiple targets, applying the memory partitioning described previously. For this test, 40 sensors and a single edge caching device with a memory size of 16 are deployed.

We study how the focus value, λ, influences caching results across three query orders from the BS:

1. **Random Query:** The BS selects a random target at each step.
2. **Round Robin Query:** The BS selects targets in a predefined, sequential order.
3. **Focused Query:** The BS selects a target in round-robin fashion but monitors that target for 10 consecutive queries.

Performance is evaluated based on the raw MISS rate. Due to integer rounding, the allocated memory for each target only changes at fixed λ values of 0, 0.25, 0.5, 0.75, and 1.0.

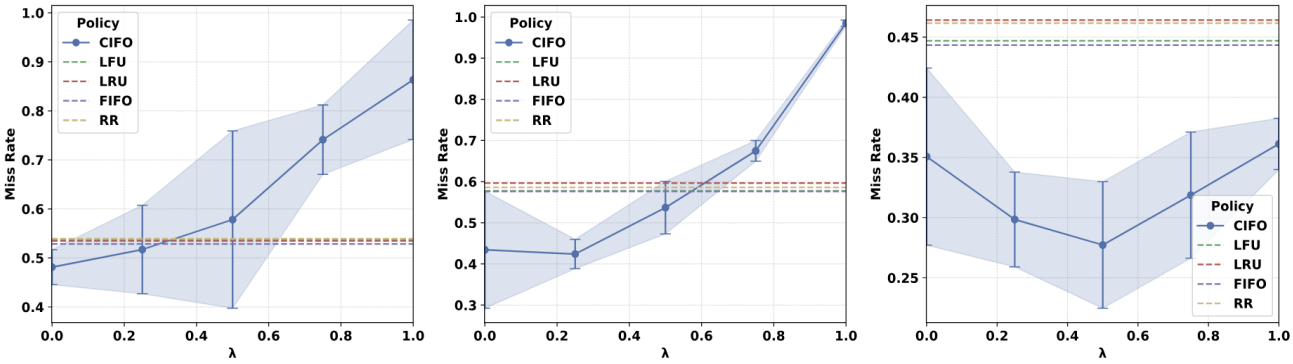
The results of our tests are shown in Figure 3-8. The results of the other baseline caching strategies are displayed with horizontal dashed lines, which denote their average value across all the experiments. Their results are fixed through the execution, as they are not influenced by the focus value λ.

For the case of random query (Figure 3-8.a), we can see that giving equal space to each target yields the best results, as a λ value of 0 means that every target has a reserved space of 4 in the cache of 16 spaces.

A similar trend can be observed for Round Robin Query (Figure 3-8.b), where higher λ values drastically increase the MISS rate.

Different results can be seen, as expected, for the case of Focused Query (Figure 3-8.c), where the best performance can be seen for larger values of λ; although too large values, 0,75 and 1.0, tend to degrade the effectiveness of the CIFO caching strategy, which, however, manages to perform better than all the other strategies, even in the worst case.

It has to be noted that these results are additionally influenced by the density of the sensors in the covered area, the query rate, and all the configuration variations discussed in the past sections. We omit the results of such tests due to space constraints, highlighting, however, how the choice of an appropriate λ value is important to achieve the optimal performance.




(a) Random Query (b) Round Robin Query (c) Focused Query

**Figure 3-8 Sensing Application, Evaluation Results – Multiple Targets**

### 3.5.2 MEC Use Case

In this section, we present the results of the simulation experiments. We tested the CIFO strategy against traditional caching strategies, including RR, FIFO, LRU, and LFU.

For each test, we deployed a baseline scenario consisting of 3 edge devices, 10 antenna relays, and 30 users, randomly placed in a 2-by-2 Km field. Starting from this baseline, we varied the following parameters for each

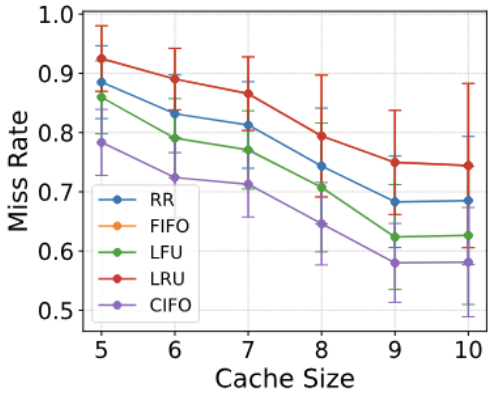
set of tests: cache size, number of users, number of edge devices, number of antenna relays, user speed, and field size. The results of these tests are shown in Figure 3-9.

Across all settings, CIFO outperforms the traditional caching strategies, particularly in scenarios where the amount of storage is limited relative to the number of users in the field. For example, setups with a small number of cache edge devices (Figure 3-9.c) or small cache size (Figure 3-9.a) demonstrate the most significant performance improvements.

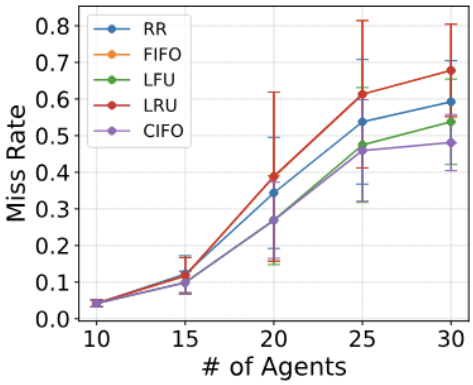
From Figure 3-9.c, it is noticeable that increasing the number of edge devices slightly decreases performance. This occurs because having multiple edge devices increases the likelihood that users move from one assigned edge device to another, which can trigger a MISS event. Such situations could be mitigated by predicting user movement and proactively moving data to the new device, but current caching strategies do not implement this behavior.

Increasing the number of antenna relays (Figure 3-9.d) leads to results with smaller standard deviations and more predictable performance. This is likely due to improved coverage of the field and a more balanced assignment of relays to edge devices.

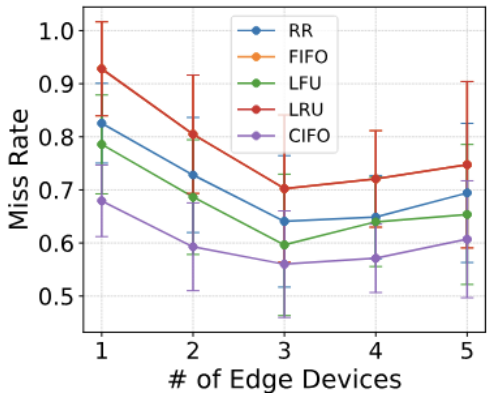
Interestingly, changing the size of the covered field or the speed of the users does not significantly affect the results. Only small variations are observed, with slightly worse performance for faster-moving users. This is explained by the fact that, even in these scenarios, users tend to switch more frequently between relays, often changing their designated edge device.



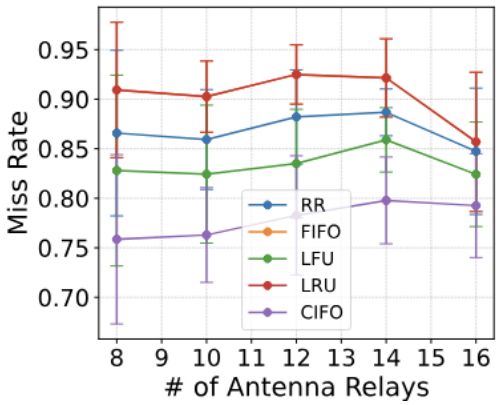

*a) Variable Cache Size*



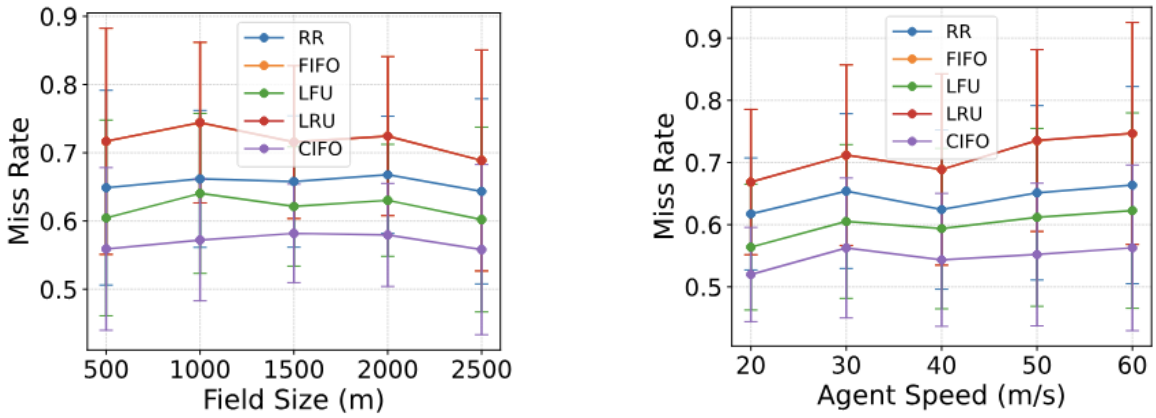

*b) Variable Number of Agents*




*c) Variable No. Edge Devices*




*d) Variable No. of Relays*




*e) Variable Field Size*

*f) Variable Agent Speed*

**Figure 3-9 MEC Application, Evaluation Results in terms of MISS rate**

# 4 Federated Learning-Enabled Networked ISAC with Edge Caching

The present use case considers the concept of caching at the local controllers of the 6G-SENSES architecture to improve the accuracy of sensing services. To achieve this, we consider, as baseline scenario, the centralized O-RAN architecture shown in Figure 4-1. Based on this approach, ISAC services are supported by extracting, at the Open Distributed Unit (O-DU), lower layer physical reference signals such as I/Q samples. A sensing function hosted at the near Real-Time (RT) RIC can be then used to process the corresponding sensing data to detect moving targets.

Caching at Near RT RIC level can be used to store range-doppler radar maps, which can be post-processed to estimate trajectories of moving targets. Then, suitable policies can be applied to ensure sensing service continuity. The same concept can be extended to perform distributed training of the Artificial Intelligence (AI) models, adopting Federated Learning (FL) for sensing services as shown in Figure 4-2. Following this approach, multiple nodes with sensing capabilities, such as Open Remote Units (O-RUs) connected with their corresponding O-DUs, can collaboratively train a sensing model without sharing raw sensing data. Instead, sensing functions hosted at distributed RICs can train locally their own measurements (I/Q samples, range–Doppler maps, micro-Doppler features, point clouds) and share model updates (e.g., gradients or weights) with an aggregator that produces a global model. This model can be then used to perform trajectory prediction for the moving targets [26].

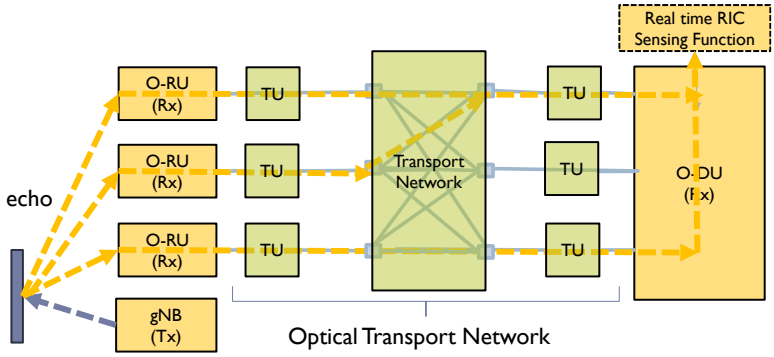



**Figure 4-1 Centralized O-RAN enabled Networked ISAC**

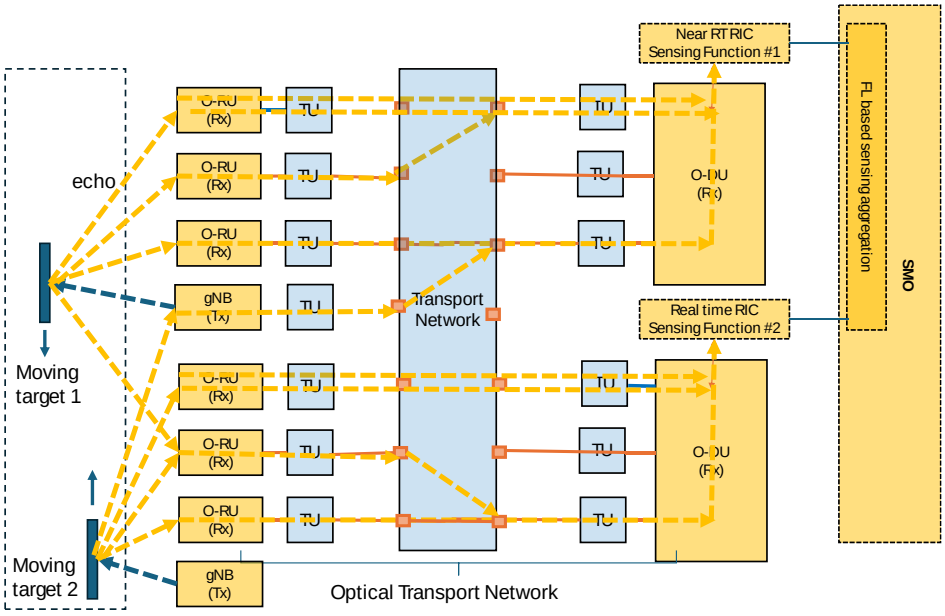



**Figure 4-2 O-RAN enabled Networked ISAC using FL**

Therefore, the present use case considers multiple O-RUs observing the environment from a different spatial viewpoint. These I/Q samples are extracted at the corresponding O-DU and processed by the Near RT RIC sensing function, extracting sensing representations, as range-Doppler radar images. These spatio-temporal images are cached at the local controller and are then used by an AI-based solution e.g., Convolutional Neural Network (CNN)/ Long Short-Term Memory (LSTM)/Transformer, to train a local model for moving target trajectory prediction. The metadata of the ML models are combined by the Non-Real-Time (Non-RT) RIC, which performs federated aggregation. The Service Management and Orchestration (SMO) manages ML model versions and policies, selects which RUs participate in the federated process and defines the number of federation rounds to achieve a specific sensing accuracy.

### 4.1 Data and AI models caching

AI/ML techniques are core in optimizing the operation of the integrated infrastructure. Intelligent operations for the RAN, Core Network (CN) and transport network domains are facilitated through various controllers installed at the different segments of the system. For example, intelligent RAN optimization is supported by the Non-RT RIC through the provisioning of policy-based guidance, ML model management and enrichment information offered to the Near-RT RIC function. Intelligent radio resource management can be also performed within a non-real-time interval (i.e., greater than 1 second). The Non-RT RIC can determine the RAN optimization actions through data analytics and AI/ML training/inference leveraging SMO services. These services include data collection and provisioning of the O-RAN nodes through **O1** and **O2** interfaces. Non-RT RIC can provide AI/ML capabilities through rApps. Similarly, near-real-time control and optimization of RAN elements and resources is provided through the Near-Real-Time RAN Intelligent Controller, also adopting AI/ML techniques. These are achieved through high granularity data collection and other actions supported by the **E2** interface. Overall, the Near-RT RIC functionalities are supported through xApps. The SMO intelligence management processes are shown in Figure 4-3.

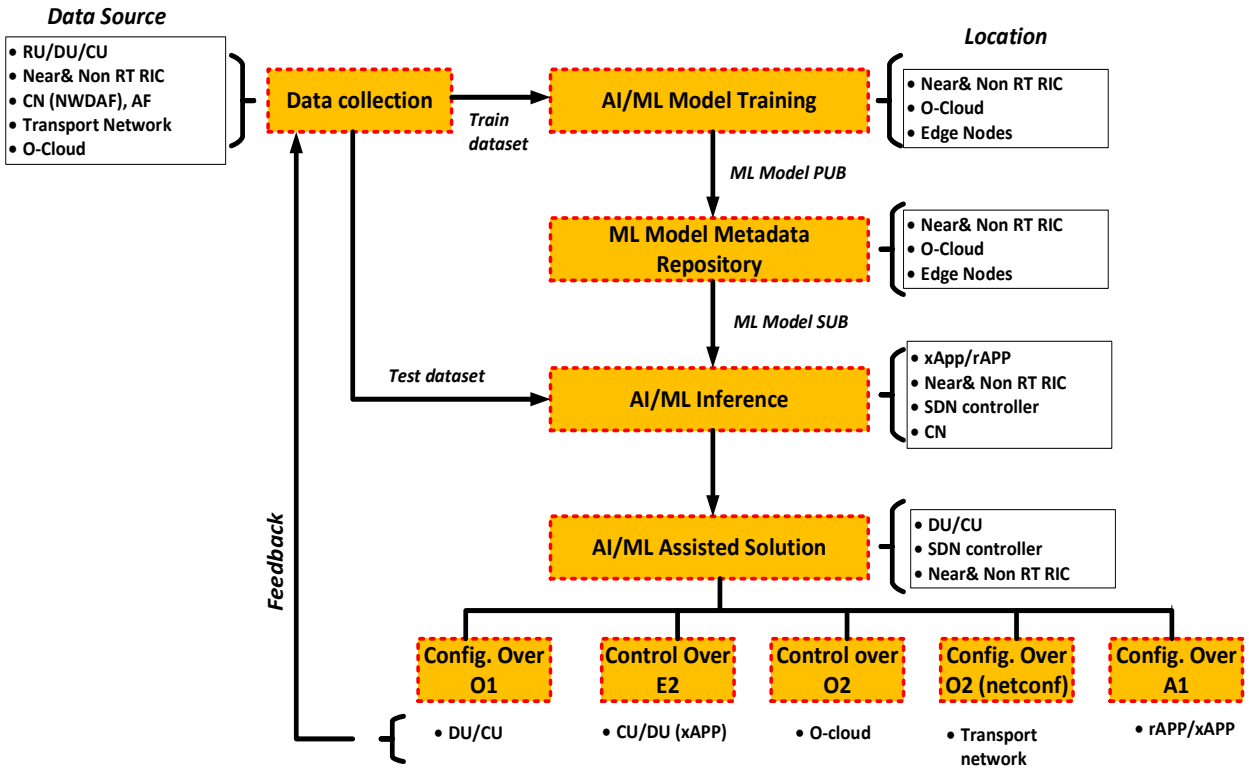



**Figure 4-3 SMO Intelligence Management: Supported AI/ML workflows and caching locations for data, ML inference and training metadata**

In addition to RAN Network Functions (NFs), the SMO can also aggregate data from the User Equipment (UE), the CN, the application function and the transport network. The collected data can be either regular monitoring information or enriched statistics that can then be utilized for ML model training and inference. Data, ML training and inference can be hosted at different locations, depending on the available physical resources and service specifications. The trained ML model is stored in an ML repository that can then be selected for inference. The output of the inference can be used by an Actor, i.e., DU, CU, Non/near RT RIC, Software Defined Networking (SDN) controller, CN controller, to provide the "AI/ML assisted Solution" optimizing the performance of a specific network segment. The relevant actions (i.e., network configurations, management policies, scheduling and routing decisions, etc.) are applied to the targeted entity using appropriate SMO interfaces (**O1**, **O2**, **E2**, **A1**).

Finally, depending on the location of the ML training and inference, the time scale where decisions need to be taken, or data availability and computational complexity, multiple ML deployment options can be considered [25]:

- **Option 1**: Offline training and inference. In this case the ML workflow is used to support non-real time decisions (with response time >1s) and therefore, ML training and inference can be hosted in the SMO and/or in the domain specific orchestrators (i.e., Non-RT RIC for RAN, transport network orchestrator, CN orchestrator, etc.), as shown in the top layer of Figure 4-3.
- **Option 2**: Offline training and online inference. In this deployment option ML training is performed at the SMO whereas inference is typically run close to the edge (i.e., in the Near-RT RIC). This option is suitable for scenarios requiring fast decision times and global knowledge of the status of the network (data collected from all network entities are stored in a LSTM centralized entity). Due to the large volume of training dataset, the SMO needs to be supported by significant computational resources.
- **Option 3**: Online training and inference where both ML training and inference are performed at the edge using limited (local) information. In this scenario data used for inference are stored locally (i.e., in the Near RT RIC) mitigating privacy risks.
- **Option 4**: Distributed training and online inference. This approach, also known as FL, combines the benefits of Option 2 and 3 as it allows multiple AI/ML entities to train an ML model collaboratively, whereas inference is performed at the edge. Instead of gathering training data into a central server, FL keeps the training data decentralized to mitigate privacy risks. AI/ML models are trained locally in distributed entities, and the local models are aggregated by a central server that orchestrates the FL.

Although the 6G-SENSES approach supports all ML deployment options, the management framework used in the present use case adopts and implements Option 4 using FL to jointly optimize the system. The present use case considers an FL-based optimization framework to predict trajectories for the moving targets and ensure sensing service continuity in an optimized manner. This is achieved by optimizing networked ISAC services by identifying the optimal match action rules to the transport network and compute resources and instantiate suitable network slices for sensing and communication services.

### 4.2 Sensing and comms datasets used for FL training

To train the ML models, a small-scale testbed was deployed in a lab environment to collect sensing and communication data.

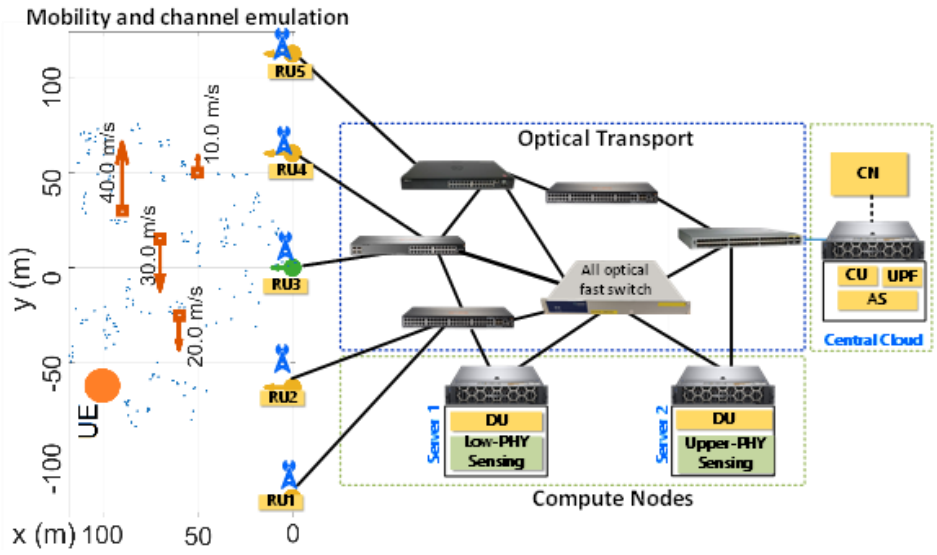


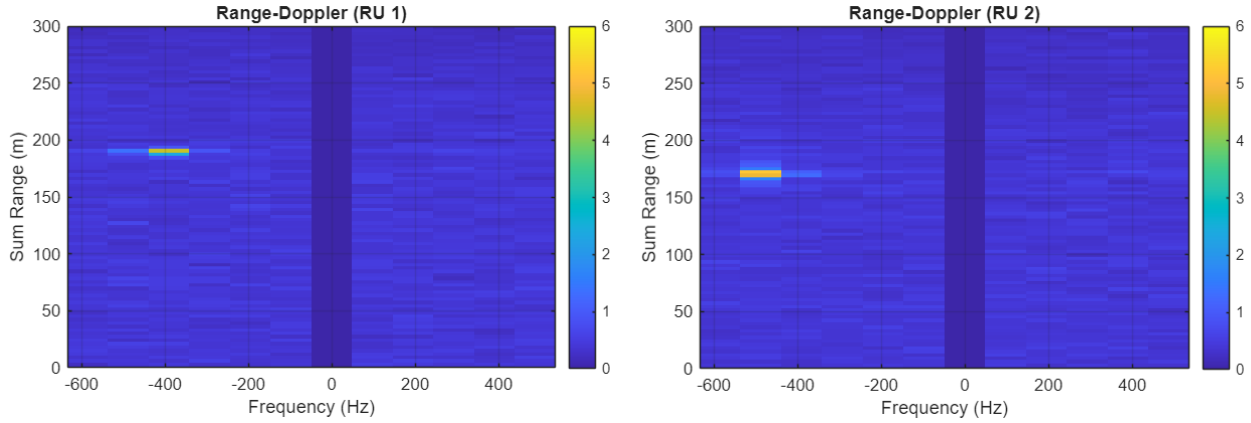

**Figure 4-4 ISAC configuration with emulated air interfaces. Snapshot of 3 moving targets, blue dots are static reflectors**

The testbed architecture is shown in Figure 4-4. It exploits an optoelectronic transport network interconnecting the RAN with the core functions located at the MEC servers. The hierarchical structure of the proposed architecture offers RAN connectivity and collects and aggregates transport network and sensing traffic streams from the RUs, while it transports these to the servers. Both the RAN and aggregation transport network segments are implemented through multi-vendor SDN-controlled optoelectronic switches with different capabilities (port no, capacity per port & latency). The access network is equipped with low energy consuming switching nodes with limited number of input ports and relatively small capacity (1/10GbE/port).

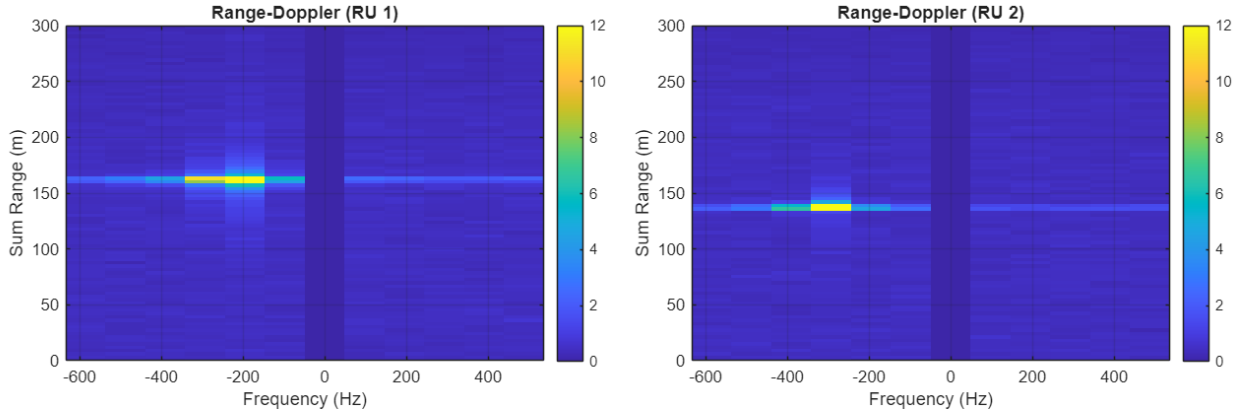
The aggregation transport network comprises high-end optoelectronic and all-optical fast switches with higher capacity and density (10G/40G/100GbE/ports), offering much higher energy efficiency. All-optical fast switches are a key enabler that facilitates transparent routing of I/Q sensing flows to the MEC servers needed to achieve acceptable transport delays. For the RAN segment, a Software Defined Radio (SDR) platform (based on USRP B210 and N310) transmitting 5G NR OFDM modulated signals is used to support the functionality of the RUs. For the compute domain, edge servers (attached to access switches) and central cloud servers (connected to the aggregation/core switches) are adopted. These servers are used to host containerized RAN elements (i.e. DU/CU) and sensing functions (storage of I/Q echoes and processing).

The sensing app implements a passive OFDM-based radar that uses, as input, signals reflected on objects in the surrounding area, when connectivity between BSs and UEs is established. The sensing app evaluates the quality of the reflected signals captured per BS and, if these are above a specific SNR threshold, they are used to estimate the position of the corresponding object. These results are then communicated to the SMO that calculates the optimal resource allocation policy needed to accommodate both communication and sensing services. This is achieved by identifying the appropriate capacity for the paths implementing “RU-DU-CU” and “RU-Sensing” function connectivity, executed by the SDN transport network controller. Finally, for the wireless channel emulation, a dedicated virtual machine has been deployed, hosting the open-source GNU Radio platform, which emulates channel impairments and objects in the surrounding area using the “Noise Source” and “Static Target Simulator” blocks, respectively.

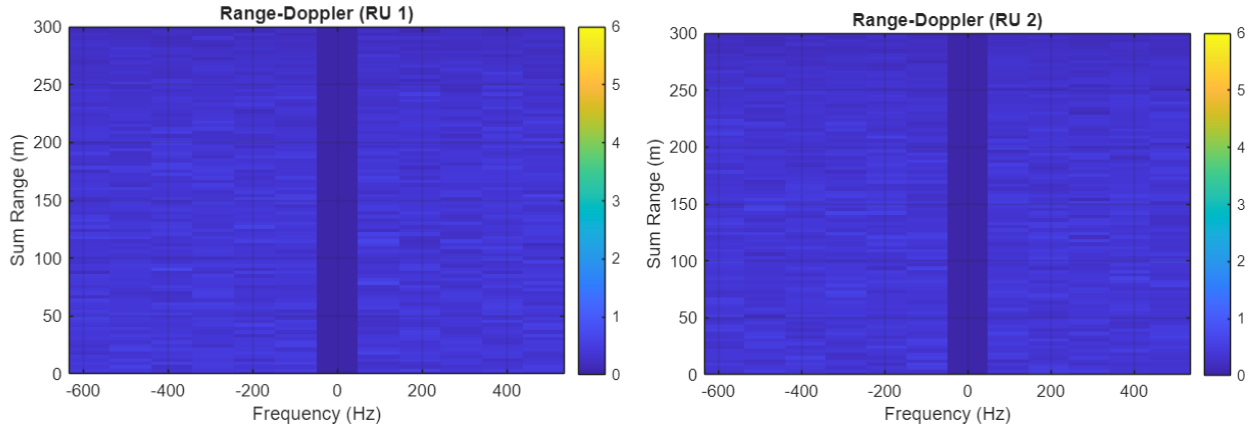
The cached dataset comprises geometric characteristics of the monitored environment along with specifications of the communication system. This includes coordinates for RUs along with antenna orientation and beamforming patterns, communication system parameters such as carrier frequency, bandwidth, number of subcarriers etc. In addition to this, the sensing function produces a set of series space-temporal doppler radar images per RU, as shown in Figure 4-5.



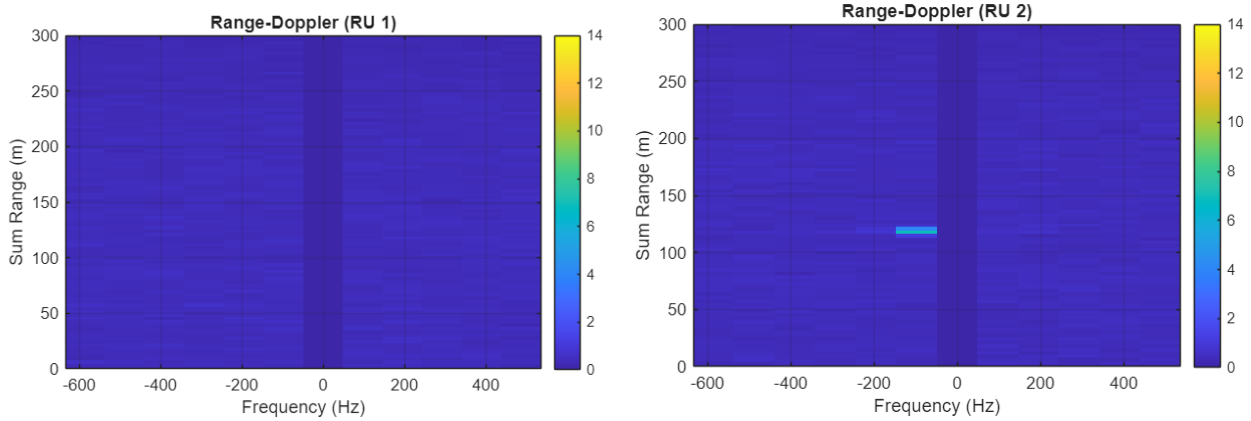

a) T0




b) T1




c)T2




d)T3

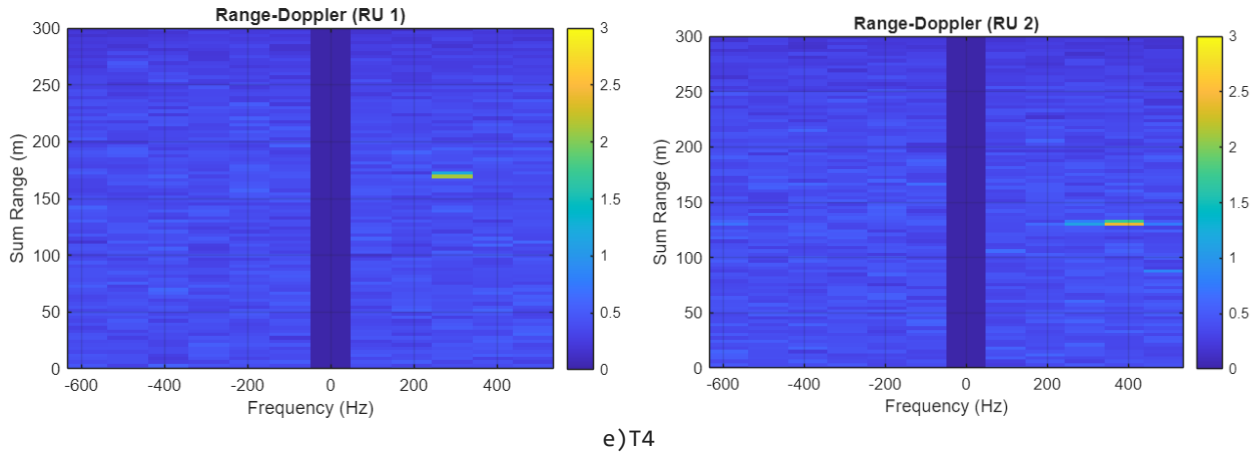



**Figure 4-5 Range doppler images captured by RU1 and RU2 at different time instants. The image represents the detection of a moving target**

### 4.3 FL management

The O-RAN compliant architecture allows both non-real time controllers hosted in the SMO and near real time domain specific controllers to support data collection, training and inference of ML functions. This feature unlocks the option to implement a decision support framework with distributed training and online inference capabilities for the ML models, allowing each network segment to be managed autonomously through AI-assisted solutions. As shown in Figure 4-3, the AI/ML-assisted solutions can apply policies and control strategies, taking decisions for a variety of tasks including management of radio and compute resources, prediction of mobile network traffic, etc. Information from local domain managers can be combined in a network-wide context. This capability can be exploited by the SMO to globally optimize the overall system performance.

To implement "network intelligence" functionality, we adopt a distributed approach, where the main decisions are taken in a hierarchical manner. The main parameters of interest for each domain for sensing and communication are stored in local databases (DBs). Information stored at each local DB is accessible only by the local (domain) controllers. Based on the local datasets, each domain controller individually trains an ML model that performs trajectory prediction for moving targets based on range doppler images. Then the locally trained ML models in the form of prescriptive analytics are transmitted to a centralized server that aggregates the received models creating a new global ML model.

The role of the central server aggregating the ML model is served by the SMO, which continuously interacts with local controllers. The global model is transmitted back to domain controllers which, based on their local datasets, create an updated ML model that is transmitted back to the SMO. The process is repeated for several rounds until certain prediction accuracy is achieved. The predicted trajectories are used by the orchestrator to set the network to ensure service continuity.

#### 4.3.1 FL procedure

The procedure of FL among real time and non-RT entities hosted at the local controllers and the SMO, respectively, can be summarized into following steps:

**Step 1**: Predictive analytics at the local controllers: In the first step, a set of local AI models are trained performing sensing and estimating trajectories for the moving targets. The local models are created adopting a FL technique where for each RAN NF their associated AI models are trained using local data samples. The hyperparameters (e.g. the weights and biases) of the trained models are exchanged between the local nodes on a regular basis generating a global model which is shared by all elements of the system. Sharing of the ML models among multiple sensing functions is performed using the SMO, responsible to orchestrate the

different steps of the learning process and coordinate the operation of the whole system. The server is also responsible for the selection of the near real time controllers that will participate in the training process and for the extraction of the central (global) AI models. The learning procedure adopted is summarized below (Figure 4-6):

- **Initialization**: The servers identify the ML model, e.g. LSTM, CNN, Multilayer Perceptron (MLP), which will be trained on the local range doppler images.
- **Client selection**: Following this, the SMO selects a random subset of sensing functions at the real time controllers that will train their ML models using their local datasets. The sensing functions that have not been selected wait for the next federated round.
- **Training process**: The training process is initiated and the ML models for each function are calculated. The hyperparameters are stored in an appropriate format, i.e. Open Neural Network Exchange (ONNX).
- **Reporting**: each selected sensing device/element sends its local model to the server for aggregation. The central server combines the received models and sends back the model updates to the sensing functions hosted at the local near RT RICs. The next federated round is started returning to the client selection phase. Model exchange among elements is performed using O1 protocol.
- **Termination**: once a pre-defined termination criterion is met (e.g., a maximum number of iterations is reached or the model accuracy is greater than a threshold) the central server aggregates the updates and finalizes the global model.

The predicted network trajectory can be used to estimate the necessary resources to configure the network and ensure service continuity for sensing services.

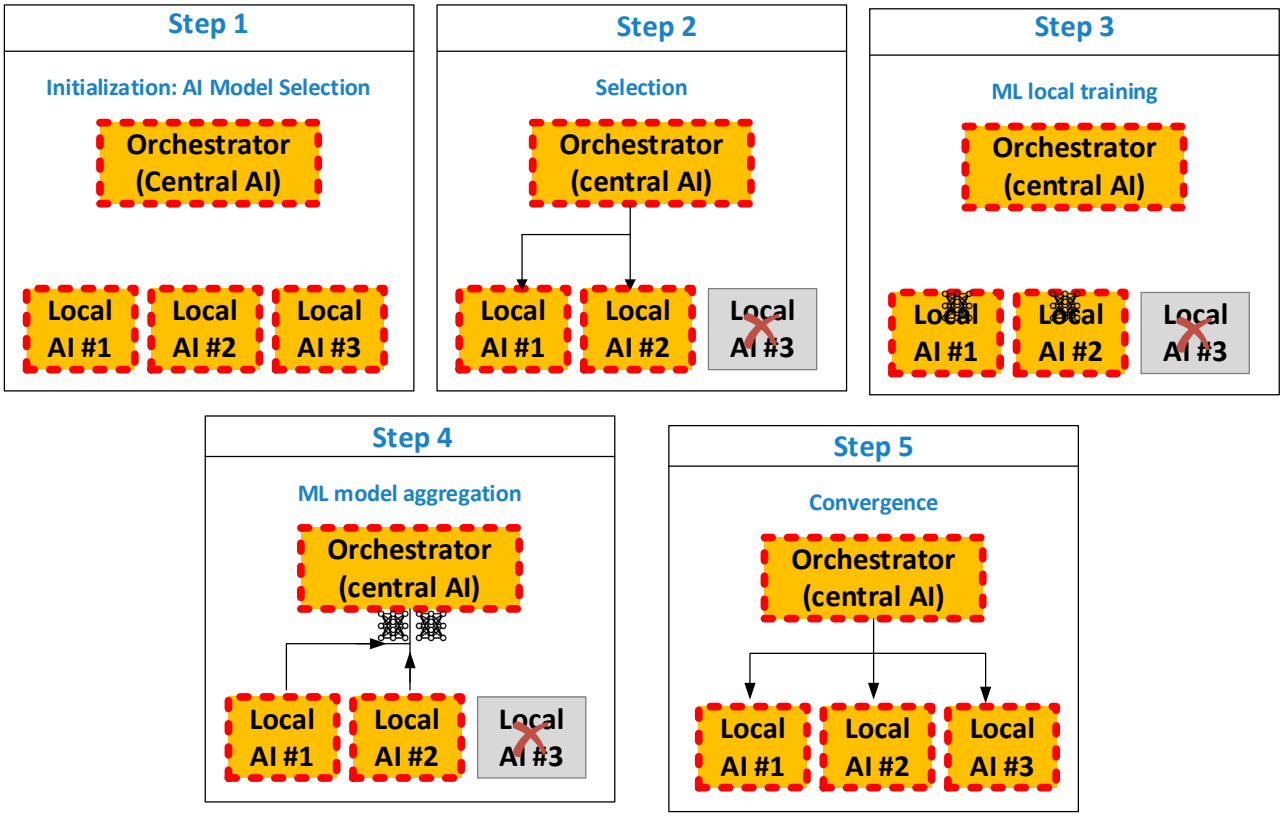



**Figure 4-6 AI model update using FL**

**Step 2: Orchestrator-Triggered Reconfiguration**: In the second stage, the output of the predictive analytics performed at local level scheme are exposed to the E2E orchestrator that prescribes optimal decisions across all layers of the system. This is achieved applying standard optimization techniques with the goal of identifying optimal strategies (ensure sensing accuracy), minimizing a cost function, for instance the number of links/modes used. The relevant problem can be written as a constraint optimization problem of the form: $\boldsymbol{min}\, \mathcal{E}(\mathbf{x}), s.t.\ \mathbf{c}(\mathbf{x}) \leq \mathbf{\Lambda}$. where $\mathcal{E}(\mathbf{x})$, is the cost function under routing strategy $\mathbf{x}$ and $\mathbf{c}(\mathbf{x}) = [\boldsymbol{c_1}(\mathbf{x}), \dots, \boldsymbol{c_m}(\mathbf{x})]$, $\mathbf{\Lambda} = [\boldsymbol{\lambda_1}, .., \boldsymbol{\lambda_m}]$ is the set of inequality constraints that need to be satisfied for the system to operate [24].

### 4.3.2 Use case description, service definition, technical solution and Results

This section presents the use case description, service definition, technical solution and results of the FL-Enabled Networked ISAC with Edge Caching. The objective is to improve the accuracy, robustness, and continuity of ISAC sensing services by exploiting edge caching and FL across distributed O-RAN controllers, while preserving data privacy and minimizing transport overhead.

#### 4.3.2.1 Use case description

The use case considers multiple O-RUs observing the environment from different spatial viewpoints. I/Q samples are extracted at the O-DU and processed by sensing functions hosted at the Near-RT RIC to generate range–doppler radar representations. These space-temporal sensing features are cached locally and used to train AI models to perform moving-target trajectory prediction. Centralized and distributed learning are compared in terms of sensing accuracy and computational cost.

#### 4.3.2.2 Service definition

The service delivers AI-assisted ISAC capabilities for moving-target detection, tracking, and trajectory prediction, ensuring sensing service continuity under dynamic radio, transport, and compute resource conditions. The service is implemented on top of an O-RAN–compliant architecture, comprising O-RUs, O-DUs, Near-RT RICs, Non-RT RICs, and the SMO.

At the RAN level, I/Q samples and physical-layer reference signals are extracted at the O-DU and forwarded to sensing functions hosted at the Near-RT RIC. These functions process the measurements to generate sensing representations, enriched with contextual information such as RU location, antenna orientation, beamforming patterns, and communication parameters (carrier frequency, bandwidth, number of subcarriers). The resulting space-temporal sensing data are cached locally at the Near-RT RIC.

The service employs AI/ML models (e.g., CNNs, LSTMs, Transformers) for target detection and trajectory prediction. Model training follows a FL paradigm, where distributed local training is performed at Near-RT RICs using cached sensing data, while only model updates and metadata (e.g., weights, gradients) are exchanged with a central aggregation entity hosted at the SMO/Non-RT RIC. This approach minimizes raw data transfer, reduces transport overhead, and preserves data privacy. The SMO coordinates the FL process by selecting participating nodes, managing model versions, defining federation rounds, and enforcing termination criteria.

#### 4.3.2.3 6G-SENSES WEC Technical Solution/Innovation

The solution introduces edge-cached sensing representations combined with FL over the O-RAN compliant 6G-SENSES architecture. This architectural approach enables distributed intelligence for ISAC services. Through sensing, trajectory patterns can be predicted, and can be then used to orchestrate transport and compute resources. For resource efficiency purposes, sensing accuracy and network resource allocation is jointly optimized by dynamically selecting the FL process the I/Q samples per RUs that will be included in the FL process.

#### 4.3.2.4 Results

Initial results on the accuracy of the proposed framework adopting the concept of local caching at the near RT RIC is shown in Figure 4-7 and Figure 4-8. In the initial implementation trajectory prediction is based on the hybrid CNN-LSTM architecture shown in Figure 4-7. This architecture performs trajectory prediction by combining spatial feature extraction from range–Doppler radar frames with temporal sequence modeling across consecutive sensing snapshots. For each sensing node (e.g., RU1 and RU2), a dedicated CNN range–Doppler frame encoder transforms each RD image into a compact latent representation that captures target-related signatures. The per-RU latent features are then combined in a feature fusion module to exploit multi-view spatial diversity affecting individual RUs. The fused feature sequence is fed into an LSTM temporal model, which learns motion dynamics across frames (e.g., smooth motion, maneuvers, and speed changes). Finally, a regression head maps the LSTM hidden state to the desired trajectory output, thus estimating the next location of the moving target in the map.

Figure 4-8 (a) shows a sequence of range-Doppler radar images captured by two spatially separated RUs at different time instants. Each image is independently processed by the CNN-based encoder at the RU side to extract compact spatial features that characterize the target's instantaneous range-Doppler signature.

Figure 4-8 (b) presents the predicted trajectory of the moving target, which is compared against the known trajectory. By integrating the fused multi-RU feature sequences over time, the figure evinces that the predicted track closely follows the actual path, demonstrating that the LSTM temporal model successfully learns the underlying motion dynamics. Minor deviations are observed during high-dynamic segments (e.g., turns or acceleration changes), where the target motion becomes less predictable from instantaneous range doppler map observations. This can be addressed by fusing signals from multiple RUs and optimizing the structure of the ML model.

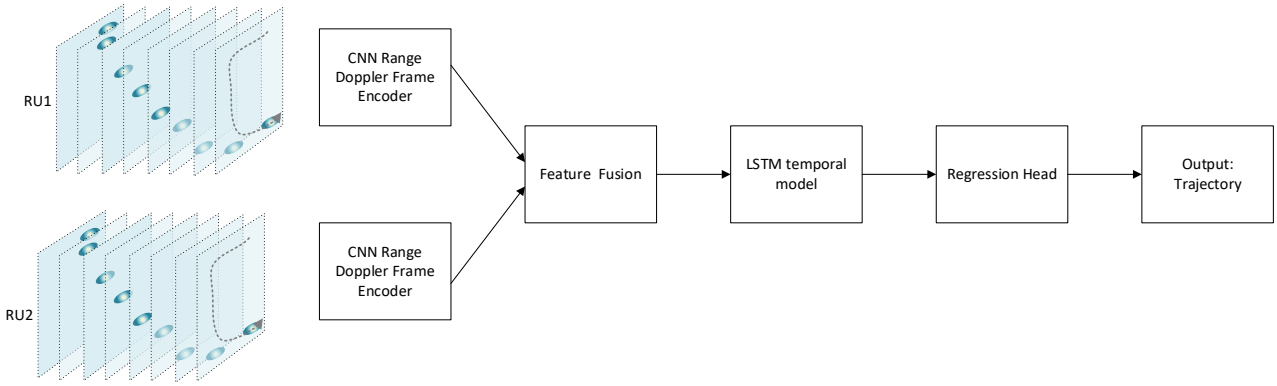
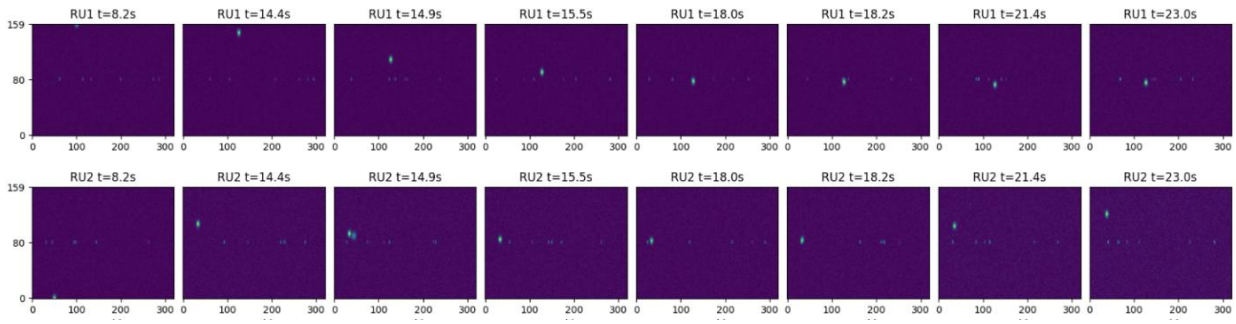
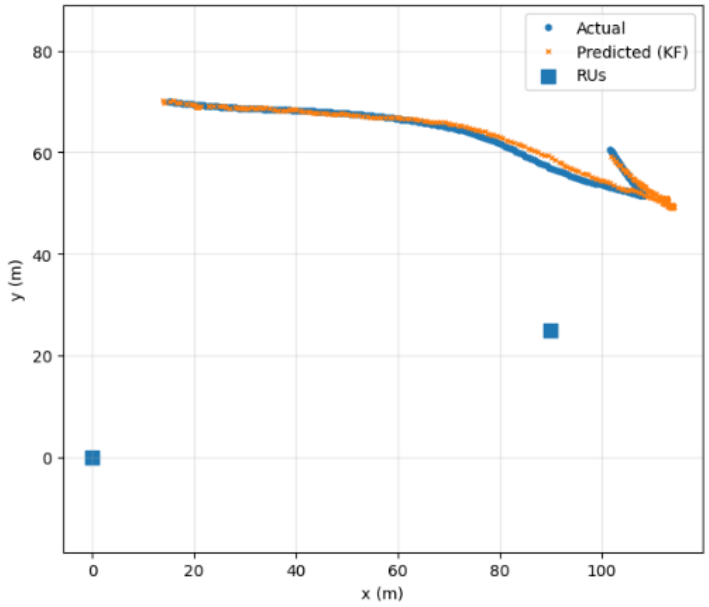


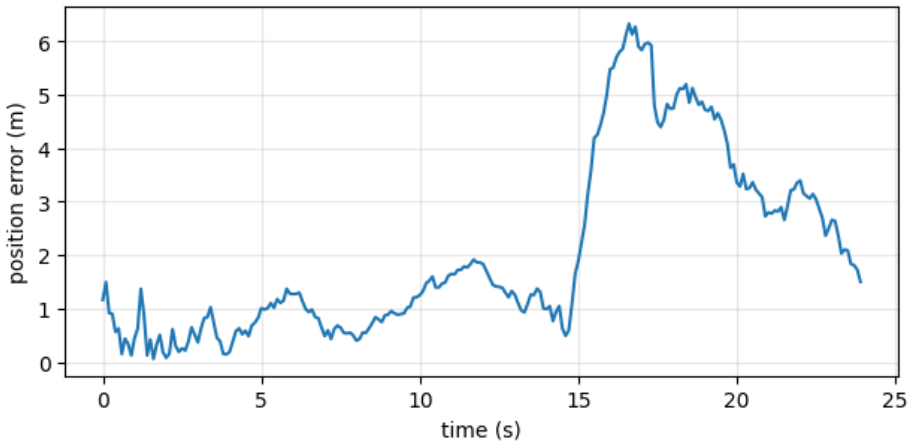

**Figure 4-7 Hybrid CNN-LSTM model for trajectory prediction**




a)




b)




c)

**Figure 4-8 a) Sequence of range doppler radar images for two RUs, b) predicted and actual trajectory for the moving target, c) trajectory prediction error**

Figure 4-8(c) depicts the trajectory prediction error as a function of time. The error remains low during steady-motion intervals and increases during maneuvering phases, reflecting the increased uncertainty in the sensed range–doppler measurements and the temporal prediction process. Overall, the results confirm that the proposed CNN–LSTM architecture can effectively exploit spatio-temporal information from distributed RD sensing to achieve accurate trajectory prediction.

# 5 Summary and Conclusions

This deliverable presents the final design and implementation of the WEC solutions developed within 6G-SENSES. Starting from the results reported in deliverable **D4.1**, we improved the caching strategies already presented, leading to the completion of the CIFO caching strategy and its application to both sensing and MEC scenarios.

The ISAC Libs were developed by integrating ROS and Gazebo, resulting in a flexible and practical framework for testing ISAC applications in realistic environments. This framework supports a wide range of use cases and pertains particularly well to those oriented towards IoT and robotic scenarios.

The small-scale demonstration of PoC#1 shows how edge caching strategies can improve data availability while reducing energy consumption. The performance of CIFO was compared with commonly used caching approaches, including RR, FIFO, LRU, and LFU, and the results highlight its benefits in both sensing and MEC contexts.

In addition, the proposed simulation framework enables reproducible evaluations across different scenarios. We plan to expand upon the small-scale demo in the coming deliverables, where a more detailed run-through of integration in PoC#1 will be given.

In parallel, WEC has been integrated into the O-RAN-compliant 6G-SENSES architecture, enabling the development of a flexible and practical framework for the design, deployment, and evaluation of caching-enabled ISAC applications in realistic and dynamic environments. The developed framework supports the generation, caching, and edge-level processing of spatio-temporal sensing data enabling use cases in which sensing, communication, and computation are tightly coupled. By leveraging WEC resources, this framework facilitates low-latency access to sensing information and efficient data reuse across ISAC services, reducing both network load and centralized processing requirements.

In summary, this deliverable brings together the design and implementation aspects of WEC strategies and establishes a solid basis for future work on ISAC applications. The tools and methodologies introduced are designed to be flexible and can therefore be adapted to additional use cases, providing a practical way to assess and enhance WEC solutions in complex and dynamic environments.

# 6 References


[1] 6G-SENSES Deliverable D5.1 **“**Report on the testing methodologies and testbed setup,” November 2025, [D5.1].

[2] 6G-SENSES Deliverable D2.1 “Report on 6G-SENSES use cases, requirements and KPIs, and key technological advancements,” [D2.1].

[3] 6G-SENSES Deliverable D4.1, “Initial State of the Art and Design of Wireless Edge Caching Solutions,” [D4.1].

[4] Fede3751. “ISAC Libs: A set of open-source isac libraries for ROS and Gazebo,” https://github.com/Fede3751/isac_libs, 2026. GitHub repository, Apache-2.0 licensed.

[5] J. Wang, N. Varshney, C. Gentile, S. Blandino, J. Chuang and N. Golmie, "Integrated Sensing and Communication: Enabling Techniques, Applications, Tools and Data Sets, Standardization, and Future Directions," in *IEEE Internet of Things Journal*, vol. 9, no. 23, pp. 23416-23440, 1 Dec.1, 2022, doi: 10.1109/JIOT.2022.3190845.

[6] S. Lu et al., "Integrated Sensing and Communications: Recent Advances and Ten Open Challenges," in *IEEE Internet of Things Journal*, vol. 11, no. 11, pp. 19094-19120, 1 June1, 2024, doi: 10.1109/JIOT.2024.3361173.

[7] X Y. Cui, F. Liu, X. Jing and J. Mu, "Integrating Sensing and Communications for Ubiquitous IoT: Applications, Trends, and Challenges," in *IEEE Network*, vol. 35, no. 5, pp. 158-167, September/October 2021, doi: 10.1109/MNET.010.2100152.

[8] C. Perera, A. Zaslavsky, P. Christen, and D. Georgakopoulos, “Sensing as a service model for smart cities supported by Internet of Things,” in *IEEE Transactions on Emerging Telecommunications Technologies*, 25: 81-93. https://doi.org/10.1002/ett.2704

[9] M. L. Casazza, A. A. Lorenz, C. T. Overton, E. L. Matchett, A. L. Mott, D. A. Mackell, F. McDuie, “AIMS for wildlife: Developing an automated interactive monitoring system to integrate real-time movement and environmental data for true adaptive management,” in *Journal of Environmental Management*, 345, 118636. https://doi.org/10.1016/j.jenvman.2023.118636

[10] V. Dyo, S. A. Ellwood, D. W. Macdonald, A. Markham, N. Trigoni, R. Wohlers, C. Mascolo, B. Pásztor, S. Scellato, and K. Yousef, “WILDSENSING: Design and deployment of a sustainable sensor network for wildlife monitoring,” in *ACM Transactions on Sensor Networks*, 8, 4, Article 29 (September 2012), 33 pages. https://doi.org/10.1145/2240116.2240118

[11] J. Hodgson, *et al.*, “Precision wildlife monitoring using unmanned aerial vehicles,” *Sci Rep*, 6, 22574 (2016). https://doi.org/10.1038/srep22574

[12] Y. Fu, Y. Zhang, Q. Zhu, H. -N. Dai, M. Li and T. Q. S. Quek, "A New Vision of Wireless Edge Caching Networks (WECNs): Issues, Technologies, and Open Research Trends," in *IEEE Network*, vol. 38, no. 1, pp. 247-253, Jan. 2024, doi: 10.1109/MNET.124.2200003.

[13] T. X. Vu, E. Baştuğ, S. Chatzinotas, and T. Q. S. Quek, “Wireless Edge Caching: Modeling, Analysis, and Optimization,” in *Cambridge University Press*, 2021.

[14] R. D. Yates, Y. Sun, D. R. Brown, S. K. Kaul, E. Modiano and S. Ulukus, *"Age of Information: An Introduction and Survey,"* in *IEEE Journal on Selected Areas in Communications*, vol. 39, no. 5, pp. 1183-1210, May 2021, doi: 10.1109/JSAC.2021.3065072.

[15] S. Suryavansh, *et al*., “A data-driven approach to increasing the lifetime of IoT sensor nodes,” *Sci Rep* 11, 22459 (2021). https://doi.org/10.1038/s41598-021-01431-y

[16] W. Zhou, R. Zhang, G. Chen and W. Wu, "Integrated Sensing and Communication Waveform Design: A Survey," in *IEEE Open Journal of the Communications Society*, vol. 3, pp. 1930-1949, 2022, doi: 10.1109/OJCOMS.2022.3215683.

[17] D. Tagliaferri et al., "Integrated Sensing and Communication System via Dual-Domain Waveform Superposition," in *IEEE Transactions on Wireless Communications*, vol. 23, no. 5, pp. 4284-4299, May 2024, doi: 10.1109/TWC.2023.3316888.

[18] Z. Wei et al., "Waveform Design for MIMO-OFDM Integrated Sensing and Communication System: An Information Theoretical Approach," in *IEEE Transactions on Communications*, vol. 72, no. 1, pp. 496-509, Jan. 2024, doi: 10.1109/TCOMM.2023.3317258.

[19] A. Ramos, S. Inca, M. Ferrer, D. Calabuig, S. Roger, J. F. Monserrat, " Simulation Framework for Detection and Localization in Integrated Sensing and Communication Systems," *Telecom 2025*, *6*, 4. https://doi.org/10.3390/telecom6010004.

[20] K. Meng, Q. Wu, W. Chen and D. Li, *"Intelligent Surface Enabled Sensing-Assisted Communication,"* in *IEEE International Conference on Communications*, Rome, Italy, 2023, pp. 4236-4241, doi: 10.1109/ICC45041.2023.10279221.

[21] Y. Cui, F. Liu, X. Jing and J. Mu, "Integrating Sensing and Communications for Ubiquitous IoT: Applications, Trends, and Challenges," in *IEEE Network*, vol. 35, no. 5, pp. 158-167, September/October 2021, doi: 10.1109/MNET.010.2100152.

[22] Aman, M., Zubair, M., Alharbey, R. et al. "ISAC based seamless handover solution for SAGIN in high mobility environments," *Sci Rep 16*, 330 (2026). https://doi.org/10.1038/s41598-025-29697-6.

[23] M. Anastasopoulos, A. Tzanakaki, G. Kaponis, Y. Jian, L. Lopacinski, J. Gutiérrez Teran, I. Mesogiti, E. Theodoropoulou, G. Lyberopoulos, “A 6G Transport Network Converging THz and Optical Network Technologies Empowered by Federated Learning Techniques”, in *European Conference on Optical Communication*, 459 (2024).

[24] M. Anastasopoulos et al., "Demonstration of a multi-technology 6G transport network integrating THz and optical network technologies empowered by federated learning," in *Journal of Optical Communications and Networking*, vol. 17, no. 8, pp. C156-C169, August 2025, doi: 10.1364/JOCN.551626.

[25] O-RAN.WG2.AIML-v01.03, O-RAN AI/ML workflow description and requirements 1.03.

[26] C. Wang, X. Chen, J. Wang and H. Wang, "ATPFL: Automatic Trajectory Prediction Model Design under Federated Learning Framework," in *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, New Orleans, LA, USA, 2022, pp. 6553-6562, doi: 10.1109/CVPR52688.2022.00645

# 7 Acronyms

| Acronym | Description |
|---|---|
| AABB | Axis-Aligned Bounding Boxes |
| AI | Artificial Intelligence |
| BS | Base Station |
| CN | Core Network |
| CNN | Convolutional Neural Network |
| DB | DataBase |
| FIFO | First In, First Out |
| FL | Federated Learning |
| I/Q | In-phase and Quadrature |
| IoT | Internet of Things |
| ISAC | Integrated Sensing and Communication |
| LFU | Least Frequently Used |
| LOC | Location |
| LRU | Least Recently Used |
| LSTM | Long Short-Term Memory |
| MEC | Multi-access Edge Computing |
| MLP | Multilayer Perceptron |
| MRU | Most Recently Used |
| NF | Network Function |
| O-RU | Open Remote Unit |
| RIC | Radio Intelligent Controller |
| RT | Real-Time |
| O-DU | Open Distributed Unit (O-DU) |
| ONNX | Open Neural Network Exchange |
| RAN | Radio Access Network |
| ROS | Robot Operating System |
| RR | Random Replacement |
| SDN | Software Defined Networking |
| SDR | Software Defined Radio |
| SMO | Service Management and Orchestration |
| SNS JU | Smart Networks and Services Joint Undertaking |
| SotA | State-of-the-Art |
| UE | User Equipment |
| WEC | Wireless Edge Caching |
| WP | Work Package |