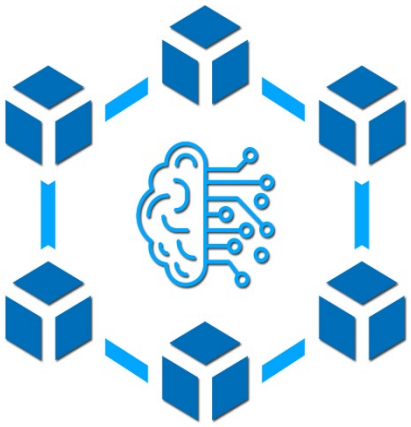

An Artificial Intelligent Aided Unified Network for Secure Beyond 5G Long Term Evolution [GA: 101096456]

Deliverable 5.3

Self-healing and Self-recovery Mechanisms

Programme: HORIZON-JU-SNS-2022-STREAM-A-01-06

Start Date: 01 January 2023

Duration: 36 Months

Co-funded by the European Union

6GSNS

NANCY 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 No 101096456.

Document Control Page

Deliverable Name	Self-healing and Self-recovery Mechanisms
Deliverable Number	D5.3
Work Package	WP5 ‘‘Security, Privacy, and Trust Mechanisms’’
Associated Task	T5.4 ‘Anomaly Detection, Self-healing & Self-recovery Mechanisms’
Dissemination Level	Public
Due Date	May 2025 (M29)
Completion Date	27 May 2025
Submission Date	27 May 2025
Deliverable Lead Partner	CRAT
Deliverable Author(s)	Emanuele De Santis (CRAT), Andrea Wrona (CRAT), Simone Gentile (CRAT), Stathis Mavridopoulos (MINDS), Dimitrios-Christos Asimopoulos (MINDS), Dimitrios Balompas (MINDS), Nikolaos Ntampakis (MINDS), Konstantinos Tsiamitros (MINDS), Christos Vasilakis (MINDS), Karypidis Paris-Alexandros (SID), Fountoukidis Eleftherios (SID), Karamitsiou Thomai (SID), Lytos Anastasios (SID), Andronikidis Georgios (SID), Kyranou Konstantinos (SID), Niotis Georgios (SID), Michoulis Georgios (SID), Tziolas Georgios (SID), Abir Yasser Barakat (TEI), Giuseppe Celozzi (TEI), Marco Tambasco (TEI), Daniel Casini (SSS), Francesco Paladino (SSS), Alessandro Biondi (SSS), Dimitrios Pliatsios (UOWM), Anna Triantafyllou (UOWM)
Version	1.0

Document History

Version	Date	Change History	Author(s)	Organisation
0.1	01 February 2025	Initial version and Table of Content	Emanuele De Santis, Andrea Wrona, Simone Gentile	CRAT
0.2	18 March 2025	MINDS contribution	Stathis Mavridopoulos, Dimitrios-Christos Asimopoulos, Dimitrios Balompas, Nikolaos Ntampakis, Konstantinos Tsiamitros, Christos Vasilakis	MINDS
0.3	25 April 2025	SID contribution	Karypidis Paris-Alexandros, Fountoukidis Eleftherios, Karamitsiou Thomai, Lytos Anastasios,	SID

			Andronikidis Georgios, Kyranou Konstantinos, Niotis Georgios, Michoulis Georgios, Tziolas Georgios	
0.4	05 May 2025	TEI contribution	Abir Yasser Barakat Giuseppe Celozzi Marco Tambasco	TEI
0.5	09 May 2025	CRAT contribution	Emanuele De Santis, Andrea Wrona, Simone Gentile, Antonio Pietrabissa, Alessandro Giuseppi, Danilo Menegatti, Francesco Delli Priscoli, Mohab Mahdy Helmy Atanasious, Valentina Becchetti, Federico Baldisseri, Kirolos Kamel	CRAT
0.6	09 May 2025	SSS contribution	Daniel Casini, Francesco Paladino, Alessandro Biondi	SSS
0.7	15 May 2025	Final version before internal review	Emanuele De Santis, Andrea Wrona, Simone Gentile	CRAT
0.9	19 May 2025	Integrated internal reviewer comments	Emanuele De Santis, Andrea Wrona, Simone Gentile	CRAT
1.0	27 May 2025	Final version after quality revisions	Dimitrios Pliatsios, Anna Triantafyllou	UOWM

Internal Review History

Name	Organisation	Date
Giorgos Kalantzis	DRAXIS	19 May 2025
Ramón J. Sánchez Iborra	UMU	16 May 2025

Quality Manager Revision

Name	Organisation	Date
Dimitrios Pliatsios, Anna Triantafyllou	UOWM	27 May 2025

Table of Contents

List of Figures

List of Tables

List of Acronyms

Acronym	Explanation
AI	Artificial Intelligence
AUC	Area under curve
B5G	Beyond 5G
BS	Base Station
CQI	Channel Quality Indicator
COC	Cell Outage Compensation
COD	Cell Outage Detection
DoS	Denial of Service
DL	Downlink
eMBB	Enhanced Mobile Broadband
eNodeB	Evolved Node B
E2E	End to End
ES	Edge Server
FL	Federated Learning
FN	False Negative
FP	False Positive
FPR	False Positive Rate
gNodeB	Next Generation Node B
HTTP	Hypertext Transfer Protocol
IDS	Intrusion Detection System
IoT	Internet of Things
IoV	Internet of Vehicles
LTE	Long Term Evolution
M2M	Machine to Machine
MEC	Multi-access Edge Computing
ML	Machine Learning
mMTC	Massive Machine Type Communications
mmWave	MillimeterWave
NF	Network Function
NLP	Natural Language Processing
NN	Neural Network
NV	Network Virtualization
O-RAN	Open Radio Access Network
PCA	Principal Component Analysis
PPA	Post-Processing Algorithm
PRB	Physical Resource Block
QoS	Quality of Service
RF	Random Forest
ROC	Receiver Operating Characteristic
RSRP	Reference Signal Received Power
RSRQ	Reference Signal Received Quality
SDN	Software Defined Network
TCP	Transmission Control Protocol
TN	True Negative
TP	True Positive
TPR	True Positive Rate
UE	User Equipment

UL	Uplink
UMTS	Universal Mobile Telecommunication System
URLLC	Ultra Reliable Low Latency Communications

Executive Summary

Deliverable 5.3 is the output of Task 5.4 “Self healing and self recovery mechanisms” and presents a suite of intelligent mechanisms that help B5G networks detect and recover from faults without relying on a central controller. As networks grow denser and more heterogeneous, with countless IoT devices, edge servers, and software-based components, traditional, centrally managed fault-handling becomes impractical and vulnerable. To address this, the NANCY team working on Taks 5.4 developed algorithms and tools based on federated learning, game theory and heuristics that are capable of automatically detecting anomalies, and counteracting to recover as much as possible the pre-fault performances.

To this aim, this deliverable presents a collection of decentralized, intelligence-driven methods designed to detect and correct faults across next-generation networks without relying on a central authority. It begins by outlining how distributed nodes can independently process their own operational data (whether application logs or traffic measurements) and collaboratively refine shared models through privacy-preserving methodologies (including learning). All the presented solutions and tools have been tested in simulation, to assess both their applicability in the context of the NANCY project and their performances. Across diverse test scenarios, these federated and learning-based techniques achieve faster convergence and greater resilience than classic, centrally managed solutions.

Extensive simulations both on real datasets and/ or realistic scenarios show these methods outperform standard baselines in accuracy, post-fault performances, convergence speed, and robustness. By embedding AI and decentralized solutions at the network layer, D5.3 paves the way for B5G systems that monitor, diagnose, and heal themselves in real-time.

1. Generalities on Anomaly Detection and Self-Healing Mechanisms

The advent of 5G has brought significant advancements over previous generations to meet the ever-increasing demands posed by modern technologies such as the Internet of Things (IoT), the Internet of Vehicles (IoV), and machine-to-machine (M2M) communication. These advancements have improved various communication services, including enhanced Mobile Broadband (eMBB), massive Machine-Type Communications (mMTC), and ultra-reliable and low-latency communications (URLLC), each tailored to distinct Quality of Service (QoS) requirements [1][2]. Additionally, 5G has facilitated the development of applications that were previously constrained by networking and infrastructure limitations, making real-time and high-throughput applications feasible due to increased data transmission rates, reduced end-to-end (E2E) latency, and near-optimal network availability and geographical coverage [3].

Beyond 5G (B5G) cellular networks extend these specifications further, offering even higher data rates and improved performance. However, they also introduce unprecedented complexity due to increased network densification, heterogeneous networks (HetNets), and software-driven functionalities [4]. B5G networks incorporate a mix of sub-6 GHz, millimeter-wave (mmWave), and terahertz (THz) spectrum bands, alongside the integration of artificial intelligence (AI) and edge computing for real-time processing and optimization.

In order to provide adequate availability and quality of service for those services, anomaly detection diagnosis and mitigation mechanisms have to be put in place, so to alert the system operators in case of malfunctioning and automatically mitigate/restore network functionalities. In particular, anomaly detection aims at performing fast and efficient detection of ongoing anomalies (without focusing on the reason that caused the anomaly). Then, the anomaly diagnosis, which investigates the kind of anomaly, may be applied. In the end, possibly leveraging on that information, mitigation mechanisms can automatically apply actions on the system, so to restore/recover the original performance.

1.1. Anomaly Detection

Anomaly detection refers to the process of identifying anomalies, thereby increasing the level of situational awareness of network operators and allowing the network to reconfigure itself once an anomaly is detected. Anomaly detection does not focus on determining the exact cause of an issue, but rather on identifying that something unusual is happening. This approach simplifies the overall process, making it faster and more efficient. As a result, early or automated responses to potential issues can be applied, even if the specific type of anomaly has not yet been identified. This ensures that an early response to anomalies can be applied as soon as possible, maintaining acceptable quality of service levels for users and ensuring network security, with an anomaly diagnosis task performed offline or in a non-real-time fashion.

In anomaly detection, machine learning-based approaches are commonly employed at different levels. In particular, anomaly detection of cellular radio access networks can be applied both at the radio level and at the computing infrastructure level (e.g., respectively in 5G-RAN and in 5G and beyond networks at Multi-Access Edge Computing (MEC) level)

1.1.1. Related Works

The literature on cell outage detection (COD) encompasses a diverse range of strategies. Some methods exploit geographic correlations among users, as in [5], while others rely on handover statistics to raise anomalies [6]. Measurement reports from neighboring cells have also been employed to

pinpoint faults [7], and several studies leverage user equipment Channel Quality Indicators (CQIs) for the same purpose [8].

Classic machine-learning algorithms have been applied: [9] used a Naive Bayes classifier to group cells and identify misbehaving ones in UMTS networks, whereas [10] adopted a Hidden Markov Model to predict 4G eNodeB status based on both healthy and faulty cell data. Unsupervised clustering techniques appear in [11], [12], distinguishing normal from anomalous cells, and [13] further used unsupervised learning to localize faults. More recently, autoencoder-based deep-learning architectures have been introduced by [14], [15] to detect outages using RSRP and RSRQ measurements.

Supervised deep-learning solutions such as those in [16], [17] distribute classification tasks to edge nodes for scalability. However, because each node trains on its own data without sharing learned parameters, these approaches can suffer from uneven performance when the quantity or quality of local datasets varies.

For what regards anomaly detection at computing infrastructure level (e.g., malicious attacks breaking confidentiality, availability or integrity, or hardware/software failures that can impact performance), machine learning methods soon gained traction by reducing reliance on detailed knowledge of attack vectors or critical infrastructure setups. Early work such as [18], [19], [20] applied Support Vector Machines and Bayesian Networks to identify system anomalies. More recently, Deep Learning architectures—as described in [21], [22]—have surpassed these techniques, particularly in uncovering network-based irregularities. This shift highlights these models' capacity to automatically extract intricate patterns from high-dimensional datasets. Among these methodologies, application logs have been widely used in several recent studies for anomaly detection at computing infrastructure level. The reason for using application logs is that in this case, the anomaly detectors can be application-specific, while system-level KPIs may not be informative if many applications run on the same physical or virtual machine. Among the works in literature that use logs to detect anomalies, [23] applied a combination of log parsing and embedding and then assessed a range of supervised and unsupervised learning models. Building on that work, [24] introduced a full pipeline that first uses a Temporal Convolutional Network to flag unusual log keys, then performs a secondary check on parameter values within those entries identified as normal. A more streamlined variant appears in [25], which drops the parameter-level anomaly stage and simplifies both parsing and embedding.

1.1.2. Challenges and Limitations

Despite significant progress, anomaly detection in cellular RANs and MEC environments still faces several hurdles. First, models that work well in one scenario often degrade under changing network conditions and have usually to rely on small dataset to be trained on (especially against rare anomalies and/or attacks that are usually less represented), highlighting the need for adaptive techniques. Moreover, most of the methods described above rely on aggregating log or KPI data into a central repository before training, which inherently requires sharing potentially sensitive information across network domains. Because supervised and unsupervised models demand a consolidated dataset, operators must expose their data (e.g., raw logs, often containing user behavior or configuration details), to a single processing point. This centralization not only creates a privacy bottleneck but also introduces a single point of failure and raises compliance concerns under increasingly strict data-protection regulations and industrial property rights, especially in current times where data gain increasing importance as companies' assets.

1.1.3. Future Research Directions

To address the privacy shortcomings of centralized anomaly detectors, future work should investigate privacy-preserving frameworks that keep raw logs on edge servers while still enabling collaborative learning. Federated Learning (FL) is a particularly promising avenue: by sharing only model updates rather than raw data, operators can train joint anomaly-detection models across multiple sites without exposing sensitive information. Indeed, Federated Learning has emerged as a groundbreaking paradigm in distributed machine learning that addresses critical challenges in data privacy and computational efficiency [26]. Unlike traditional centralized approaches where all data is aggregated at a single location for processing, federated learning enables model training across multiple decentralized devices or servers holding local data samples, without exchanging the data itself. This approach was initially introduced to enable mobile devices to collaboratively learn a shared prediction model while keeping all training data on the device, thereby addressing privacy concerns and reducing communication costs [27].

Research should explore how to adapt existing log-parsing and embedding pipelines to a federated setting, including techniques for heterogeneous feature spaces and variable client reliability. Beyond basic federated averaging [28], personalized FL methods could tailor a global model to each operator's unique traffic patterns, improving detection accuracy under varying load conditions. To counteract potential poisoning or unreliable participants, robust aggregation rules (e.g., see [29]) and robust consensus protocols will be essential. Finally, combining federated anomaly detection with on-device lightweight adaptation—such as continual learning or online fine-tuning—can help models remain accurate as networks evolve. By integrating privacy, robustness, and adaptability, next-generation anomaly detectors can meet the twin demands of effective real-time monitoring and strict data protection requirements.

1.2. Anomaly Diagnosis

Anomaly diagnosis refers to the process of identifying, analyzing, and mitigating irregular network behaviors that may impact performance, reliability, or security. These anomalies can stem from various factors, including hardware failures, network congestion, cyber-attacks, or misconfigurations [30]. Given the complexity of B5G networks, effective anomaly diagnosis is crucial for maintaining seamless connectivity, optimizing resource utilization, and ensuring network security.

B5G networks are heavily reliant on Software-Defined Networking (SDN), Network Virtualization (NV), and various Network Functions (NF). This intense digital transformation, also referred to as softwarization, has enabled the adoption of data-driven approaches for network monitoring and anomaly diagnosis. Traditional rule-based methods [31],[32] which were once the backbone of anomaly detection, are increasingly being replaced by more sophisticated Machine Learning (ML)-based and Big Data analytics-driven techniques [33].

In machine learning-based anomaly diagnosis, supervised, semi-supervised, and unsupervised learning paradigms are employed either independently or in combination. Deep neural networks (DNNs) and deep learning techniques are particularly popular, with most methodologies adopting a hybrid approach, integrating multiple techniques for different stages of anomaly detection.

1.2.1.Related Works

For instance, in [34], the authors leverage Referenced Signal Received Power (RSRP) user data to predict and classify anomalies using a combination of Convolutional Neural Networks (CNNs) and Random Forests. In [35], anomaly detection begins with Shannon entropy analysis to identify abnormal

behavior, followed by a hybrid method where Random Forests are used for feature extraction, and a combination of K-means++ and Adaboost is applied for threat classification. In [36], the authors propose an ensemble learning anomaly detection system for 6G networks, integrating feature selection and diverse classifiers (SVM, RF and Adaboost), achieving over 99% accuracy across multiple intrusion detection datasets. Similarly, [37] introduces a hybrid feature selection, utilizing genetic search, and rule-based selection to improve feature relevance. It further combines K-means, One-Class SVM, and DBSCAN for robust attack classification, attaining almost perfect accuracy with minimal false alarms. Furthermore, the work in [38] presents BoostLog, a boosted ensemble of Long Short-Term Memory (LSTM) classifiers designed for anomaly detection in 5G Radio Access Network (RAN) system logs. By learning functional behavior from normal execution logs, BoostLog effectively detects deviations, surpassing single LSTM models and reducing the reliance on manual log analysis.

The paper in [39] proposes a method for automatic fault detection and diagnosis in cellular networks and Beyond 5G by leveraging performance support system data and drive test data. It employs a plethora of unsupervised learning and classification algorithms to achieve a high fault diagnosis accuracy, with the Silhouette coefficient used for clustering evaluation. In [40], the authors assess the performance of deep semi-supervised learning techniques, such as Deep Belief Networks (DBN) and Stacked Auto-Encoders (SAE), to determine optimal hyper-parameter configurations (e.g., batch size, device type) for real-time anomaly diagnosis in B5G networks. In another study [41], they apply a Residual U-Net to classify anomalies in the CTU dataset, transforming the original time-series data into images for improved analysis.

In [42], anomaly detection in next-generation wireless networks is explored through the application of big data analytics on call detail records (CDRs). The authors employ unsupervised clustering techniques, including K-means and hierarchical clustering, to identify anomalous network behavior, followed by a neural network-based prediction model to demonstrate how transforming anomalous data into anomaly-free data improves prediction accuracy. Similarly, in [43], big data analytics is used to extract meaningful patterns from real user data provided by Telecom Italia Mobile, and the L-BFGS model is applied for predicting anomalous network traffic.

Moreover, there are other works that explore alternative approaches, which are less commonly used compared to ML-based and big data analytics methods. In [44], a novel approach for power grid anomaly detection is proposed, utilizing Time-Series Generative Adversarial Networks (Time-GAN) to generate synthetic datasets for training fault diagnosis models. This is combined with unsupervised clustering techniques such as DBSCAN and K-means to identify unknown anomalies, resulting in higher recognition rates and enhancing grid stability and operational efficiency. In addition, [45] introduces a graph-based approach for automating root cause analysis in cellular networks, specifically in the context of network softwarization and AI/ML-aided automation. By exploiting log files from negative-impacting events, this method identifies correlations and dependencies across network levels, facilitating more efficient anomaly diagnosis and tracking.

1.2.2. Challenges and Limitations

Anomaly diagnosis in B5G cellular networks faces a range of challenges [46] despite the success of the aforementioned methods. The wide variety of data sources, including IoT devices, autonomous vehicles, and edge computing nodes, among others, generates diverse data formats and volumes, complicating consistent anomaly detection. Additionally, the scarcity of real-world data further exacerbates this challenge, particularly when data is limited or unavailable due to privacy concerns. Furthermore, the lack of labeled data hampers the development of supervised learning models, making it difficult to detect new or evolving anomalies in real-time environments. The dynamic nature

of B5G networks [47], with features such as network slicing and edge computing, require scalable anomaly detection systems that can process large volumes of data with low latency. Another significant concern is resilience to adversarial attacks, as the increasing reliance on AI and machine learning models exposes these networks to manipulation by attackers attempting to evade detection. These data privacy and security concerns, emphasizing the need for robust and secure anomaly detection models that preserve user trust and comply with regulatory standards.

1.2.3. Future Research Directions

Considering these challenges, research could focus on developing synthetic data generation techniques [48] to resolve the issue of scarce real data or the occasion where data is available but cannot be utilized due to user privacy preservation. Multi-agent systems for distributed anomaly detection and graph-based models to capture the complex relationships between network elements could help improve scalability and resolve the issue of lack of labeled data exemplified by works such as [49]. Additionally, research into robust machine learning techniques is crucial for reinforcing model resilience to adversarial attacks and ensuring high detection performance. Adversarial robustness [50] can be improved through techniques such as adversarial training, ensemble methods, and differential privacy mechanisms, which help mitigate the impact of adversarial perturbations. Meanwhile, federated learning [51] plays a vital role in preserving data privacy by enabling decentralized model training without directly sharing sensitive information. Combining these approaches could enhance both security and performance in anomaly detection for B5G networks.

1.3. Self-Healing and Self-Recovery

Self-healing algorithms have emerged as a promising approach to ensure reliable and uninterrupted network operations [52]. In scenarios where immediate fault resolution is not feasible, compensation mechanisms and algorithms are specifically designed to provide temporary service continuity to users affected by outages. For this reason, usually self-recovery mechanisms are referred to as *Outage Compensation* ones.

In the context Self-Healing at radio level, Cell Outage Compensation (COC) techniques mainly focus on utilizing resources from nearby cells to temporarily supply services in the afflicted area These resources may include user associations and cell bandwidth, which can be adjusted using primary factors like antenna and cell/user equipment transmit powers, as well as secondary parameters like cell selection parameters and neighbor lists [52]. It follows that a complete outage compensation mechanism consists of two main elements:

1. A user association policy, through which users subject to fault events are assigned to neighboring cells.
2. An antenna resources adjustment policy, according to which it is possible to modify the primary factors (e.g. transmission power, beam downtilt, beam azimuth) to increase the users' QoS according to their prioritization.

In the context of Self-Healing at edge computing level, compensation techniques are mainly focused on re-routing traffic to healthy edge servers (ESs), so to balance the competing objectives of ES load balancing and preserving users/application performances.

Numerous strategies have been proposed in the literature to address this challenge in cellular networks.

1.3.1. Related Works

The study in [53] presents an algorithm that utilizes received power measurements from users in an outage-affected cell to iteratively adjust the antennas of neighboring cells, successfully restoring coverage and achieving effective compensation. Similarly, the authors in [54] investigate the impact of control parameters, such as signal power and antenna tilt, demonstrating that adjustments to the uplink power control parameter and antenna tilt significantly enhance both coverage and throughput. In [55], the authors propose advanced self-organizing and self-healing mechanisms for small-cell networks, which are conceptualized as intelligent distributed antenna systems. The self-organizing mechanism clusters cells to maximize resource utilization and system capacity, while the self-healing component dynamically reconfigures these clusters to address failures. Further optimization-based approaches are presented in works such as [56], where a non-convex immune algorithm is employed to compute the optimal uplink transmission power, thereby improving both coverage and channel quality. Similarly, [57] applies a particle swarm optimization algorithm to minimize coverage gaps and interference by adjusting downlink power levels, addressing a non-convex optimization challenge.

Recently, machine learning (ML) techniques have gained increasing attention for their potential to enhance self-healing capabilities in cellular networks [58]. A notable example is [59], which proposes a NOMA-based cell outage compensation scheme that maximizes spectral efficiency while maintaining service quality, particularly for cell-edge users. The approach combines heuristic user association with a deep neural network for power allocation, achieving a low-complexity solution close to optimal. In [60], a hybrid GA-ANN model is proposed for LTE cell outage compensation, where an optimization module identifies optimal parameters to meet key performance indicators (KPIs). An ANN is trained on outage scenarios to enable real-time decision-making while reducing computational complexity.

Reinforcement learning (RL) has also emerged as a promising approach for self-healing in cellular networks, enabling algorithms to learn optimal actions from environmental feedback and to adaptively improve over time [61]. RL techniques offer the ability to autonomously and dynamically respond to changing network conditions, addressing the limitations of static compensation systems and supporting the development of fully autonomous and resilient networks. An early RL-based study [62] explores outage compensation by adjusting the coverage of neighboring cells through antenna tilt optimization and power control, integrating fuzzy logic and RL to enhance Quality of Service (QoS). In [63], a Q-learning-based algorithm is proposed for COC, which adjusts neighboring cells' transmission parameters, namely power and antenna tilt, to compensate for coverage gaps in affected areas.

The integration of Deep Reinforcement Learning (DRL) has further expanded the potential of self-healing solutions. DRL combines the representational power of deep learning with the adaptive capabilities of RL, enabling agents to discover complex policies in high-dimensional environments. Building upon this, [64] introduces a Cell Outage Compensation (COC) algorithm using an Actor-Critic Temporal Difference (TD) learning framework, enabling eNodeBs to dynamically tune downlink power and antenna tilt. This is further enhanced in [65] through an actor-critic RL approach aimed at jointly improving coverage and capacity by optimizing antenna gain and transmission power of surrounding base stations. [66] introduces a DRL-based framework that enhances downlink Voice over LTE (VoLTE) performance indoors and manages faults in outdoor SON clusters by dynamically adjusting network parameters based on real-time user data and feedback. [67] presents a DRL-based compensation scheme for C-RAN that uses K-means clustering and a Deep Q-Network to optimize antenna downtilt and power, enhancing both energy efficiency and connectivity. Lastly, [68] proposes a DRL-based algorithm that accounts for the spatial relationship between the outage zone and surrounding antennas. This method learns an optimal antenna selection policy, contributing to more efficient and targeted outage compensation.

As for anomaly compensation at edge level, modern strategies for maintaining edge-computing performance often marry on-the-fly anomaly detection—leveraging, for example, graph neural networks [69] or other machine-learning classifiers—with adaptive rerouting schemes that shift traffic across underutilized edge nodes. Platforms such as Athena [70] and various vehicular-edge rerouting frameworks [71] proactively move workloads away from struggling servers, thereby curbing latency spikes and preventing service interruptions.

Another highly effective family of techniques draws inspiration from Wardrop's traffic equilibrium theory. In this view, each data flow (or user) selects the path that minimizes its own delay, and the system settles into a state where no single flow can improve its performance by switching routes unilaterally. Transposed onto edge networks, servers and links play the role of roadways whose "travel times" rise and fall with congestion or failures. By framing traffic assignment under anomaly conditions as a convex optimization or variational-inequality problem, these methods guarantee balanced load distribution and fair rerouting—critical in decentralized, multi-agent settings such as vehicle-to-infrastructure networks [72], [73]. When coupled with live anomaly alerts, Wardrop-inspired controllers steer the system toward resilient equilibrium points that tolerate failures without centralized coordination.

1.3.2. Challenges and Limitations

Self-healing and self-recovery algorithms, while promising for enhancing network resilience, face several challenges and limitations that must be addressed to ensure their successful implementation in next-generation mobile networks.

The demands of 5G mobile networks, including low latency, high capacity, high throughput, and reduced energy consumption, position them to be more focused on user QoE, in contrast to legacy networks that prioritized user quality of service. As a result, ensuring an optimal user QoE will be a central objective in future mobile networks, even in the case of outages. Given the anticipated rise in outages caused by failures and misconfigurations, maintaining user QoE will present a critical challenge for self-healing solutions.

Some of the works presented earlier had issues with sensitivity to initial conditions and parameter tuning. These algorithms heavily depend on initial values and require careful calibration of parameters. If these initial values or assumptions are incorrect, the algorithm may converge to a suboptimal solution, compromising the ability to restore network functionality properly.

A significant challenge is real-time decision-making. Self-recovery algorithms must act quickly to restore service during outages, but in some cases, centralized algorithms require data to be processed by a central unit, introducing delays in decision-making. This becomes problematic, especially as networks scale and generate increasing amounts of data. Additionally, the scalability of self-healing solutions remains a concern. In large, dense 5G networks, the volume of data and the number of devices involved can overwhelm centralized decision-making systems. Processing this vast amount of data in real-time without creating bottlenecks or single points of failure is a complex challenge.

Another important limitation is energy efficiency. Many self-healing algorithms involve actions that may lead to higher energy consumption, such as increasing transmission power, activating more cells, or rerouting traffic. While these actions are necessary to maintain service during outages, they can undermine the goal of energy efficiency, particularly in 5G networks, which are expected to support massive data traffic. This increased energy consumption raises operational costs and may impact the sustainability of the network, especially when self-healing actions are frequent. Finding a balance

between ensuring reliable service and maintaining energy efficiency is essential for future mobile networks.

1.3.3. Future Research Directions

Self-healing and self-recovery algorithms will play a critical role in ensuring that future networks, particularly 5G and beyond, can autonomously handle outages and maintain optimal service quality. While current self-healing approaches have made significant advancements, several research directions remain for improving their performance and applicability in large-scale, dynamic environments.

Future research should focus on improving the efficiency and scalability of self-healing algorithms, especially as 5G networks scale up. As networks grow in complexity, the volume of data and the number of devices involved can overwhelm current centralized algorithms. Research into distributed self-healing mechanisms and edge-computing-based solutions could help offload processing tasks and reduce latency, enabling real-time decision-making at the network's edge. Developing more scalable optimization techniques, such as decentralized learning, would help handle the large-scale operations typical of 5G networks without compromising performance.

Current machine learning-based self-healing solutions often require large amounts of data for training and are sensitive to initial conditions. Researchers could focus on developing models that require less training data and are more robust to dynamic changes in the network environment. Furthermore, deep reinforcement learning approaches hold great potential for enabling networks to continuously learn from real-time data, adapt to changing conditions, and optimize network performance.

Given the growing concerns about energy consumption in modern networks, energy-efficient self-healing solutions will be a key area of future research. Self-healing algorithms often lead to increased power consumption due to actions like increasing transmission power or activating additional cells to maintain coverage during outages. Future research could focus on developing algorithms that optimize self-healing actions while minimizing energy consumption.

2. Algorithms for Anomaly Detection and Self-Healing

The rapid expansion of 5G and beyond terrestrial cellular networks has created new issues for sustaining strong, secure, and high-performance infrastructures. In this regard, Task 5.4 focuses on building proactive, federated learning-based algorithms to detect abnormalities and enable network-wide self-healing and recovery procedures. The anomaly detection and self-healing methods developed in NANCY aim to anticipate future threats and disruptions by continuously monitoring and comparing historical and real-time network data, allowing for the early detection and mitigation of performance-degrading events.

This section describes in detail distributed algorithms capable of detecting anomalies both at radio (coverage) and at edge (computing and networking) level. Some of the presented algorithms are based on distributed/federated learning, with detection algorithms that are mostly based on the supervised learning paradigm, while self-healing ones rely on reinforcement learning. All these algorithms work across federated nodes to enable localized learning while maintaining data privacy and scalability, thus reducing the computational burden at the core network. The following subsections feature different distributed and federated learning-based architectures and algorithms devoted to addressing critical issues arising from having to first correctly and promptly detect anomalies, and then try to adopt countermeasures to restore the correct functioning of the network.

Table 1 summarizes the capabilities and the scope of each of the algorithms that are presented in the subsequent subsections.

Table 1: Summary of the algorithms for Anomaly Detection and Self-Healing

Algorithm	Section	Type	Domain	Approach	Distributed
Anomaly Detection in Multi-Access Edge Computing Environment via Federated Learning	2.1	Anomaly Detection	Edge	Federated Temporal Convolution Network	✓
Threat Detection using FL-IDS	2.2	Anomaly Diagnosis	Edge	Federated Deep Learning	✓
Federated Learning Algorithm for Improved Anomaly Detection in Cellular Networks	2.3	Anomaly Detection	Radio	Federated Random Forest + Consensus	✓
Memory Traffic Monitor Tool via ARM Performance Counters	2.4	Anomaly Detection	Edge	Heuristic on hardware counters	✓
Federated User Reallocation in Cellular Networks	2.5	Self-healing	Radio	Federated Deep Reinforcement Learning (D3QN)	✓
Deep Reinforcement Learning Control for Self-Healing and Anomaly Compensation in Cellular Networks	2.6	Self-healing	Radio	Reinforcement Learning (DDPG)	✓

Adversarial Dynamic Healing based on Wardrop Equilibrium	2.7	Self-healing	Edge	Game Theory (Wardrop)	✓

2.1. Anomaly Detection in Multi-Access Edge Computing Environment via Federated Learning

This work introduces a distributed anomaly detection framework that employs the principles of Federated Learning (FL) to inspect application logs for identifying potential irregularities at the software level. Accordingly, each participating agent transmits solely the insights or models derived from its local training phase. This design upholds robust security and confidentiality, enabling remarkably efficient distributed anomaly detection overall.

Furthermore, since application logs are often recorded in a human-readable natural language, appropriate log parsing methods—drawing on the insights from [24] and [25], which note that application logs generally conform to a finite set of templates—and embedding techniques are employed.

Additionally, this study introduces the use of weighted loss functions at the level of individual learning agents, guided by validation metrics, to enhance the overall FL training performance. In this context, we performed an evaluation of the anomaly detection system's performance across various validation metrics integrated into the custom loss functions to substantiate the proposed methodology.

The primary contributions of this approach are as follows:

- Utilization of a dynamic FL strategy tailored for the log anomaly detection problem;
- Deployment of a performance-driven weighted averaging method that adjusts for data distribution disparities and strengthens federation robustness;
- Enhancement of the training process through an adaptive loss function directly linked to validation metrics;
- A comprehensive comparative analysis of training performance based on specific metrics.

Specifically, we cast the challenge of identifying anomalies in distributed log data as a binary classification task, aimed at determining if each log entry signifies normal or abnormal activity.

Unlike traditional approaches that aggregate raw logs into a centralized repository for uniform pre-processing, our method addresses scenarios where logs remain on their original devices, which are responsible for their own parsing, sampling, and embedding. Figure 1 depicts the workflow of our proposed system.

The resulting distributed classification challenge is addressed through an adaptive Federated Learning strategy that highlights collaborative engagement among various devices. The proposed work is divided into log preprocessing, to make logs suitable for learning purposes, and neural network training (in a federated fashion). The software implementation of this proposed approach is publicly available here: https://github.com/StefanoF00/FL_Decentralized_Log_Anomaly_detection. Moreover, this proposed approach has been disseminated in [74].

2.1.1. Log Processing

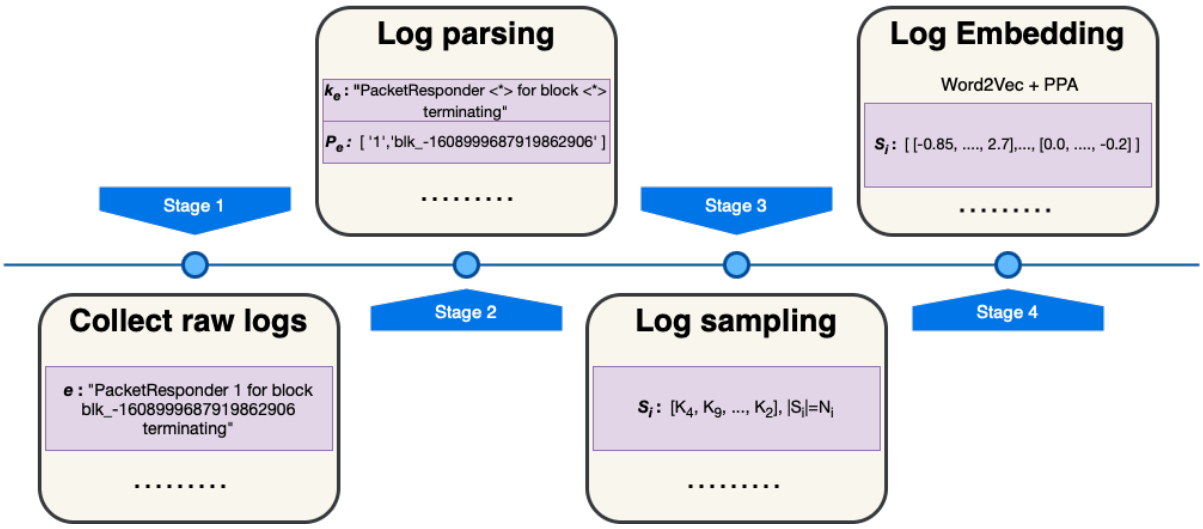

Figure 1: Log Processing workflow

In the log pre-processing phase, raw logs are transformed into meaningful representations that support model training. Our approach integrates various strategies described in [25] that enable real-time log mapping with minimal computational overhead by processing logs in sequential stages, as illustrated in Figure 1.

Because log entries are made up of textual data containing several fields—such as timestamps and details regarding system processes—it is vital to remove any extraneous information to isolate the necessary values. Therefore, after the initial collection (Stage 1), the logs are parsed in Stage 2.

Parsing methods like Spell [75], Drain [76], and IPLoM [77], [78] convert each raw log entry, denoted as e, into a structured template. This process identifies the recurring portion of the log, referred to as the log key k_e, and separates it from the variable components known as event parameters p_e.

Once each entry has been parsed, the sampling phase (Stage 3) is initiated, where the log keys k_e are organized into execution sequences S_i based on the characteristics of the data, with each sequence containing a different number N_i of keys. This grouping is achieved using session window methods, similar to those used for Hadoop Distributed File System logs [79] and OpenStack logs [24], where logs are grouped by block id or instance id, respectively. In cases where logs lack explicit session identifiers, such as in Blue Gene/L logs [80], grouping is performed based on timestamps.

Finally, the log embedding phase (Stage 4) concludes the pre-processing workflow. In the first step, the k-th structured sequence of log keys (S_i) is encoded using the widely recognized Word2Vec model [81]—a neural network that produces vector representations of words, also known as word embeddings. Word2Vec is trained on a large corpus of text, in this case log keys, to capture the contextual relationships between words by mapping semantically similar words to nearby points in a continuous vector space.

Using this technique, each log key (k_e) is projected into an M-dimensional vector (V_e^M), with M chosen to provide a comprehensive representation of the semantic information contained in the log key. As a result, the processed log sequences are formed as numerical vectors V_1^M, V_2^M, …,$V_{N_i}^M$. However, these sequences can become computationally demanding to handle, especially when dealing with extensive training data or large vocabularies.

To mitigate these computational challenges and boost efficiency, dimensionality reduction techniques—such as the Post-Processing Algorithm (PPA)—are often employed [82]. The primary goal of PPA is to lower the dimensionality of Word2Vec embeddings while preserving as much semantic content as possible. This reduction substantially decreases the computational load associated with processing and storing the embeddings, making them more suitable for downstream tasks like natural language processing (NLP) and machine learning (ML).

PPA, as proposed in [82], utilizes methods such as Principal Component Analysis (PCA) to achieve this dimensionality reduction. PCA determines the main components of the high-dimensional embedding space and projects the embeddings onto a lower-dimensional subspace ($n < M$), preserving as much variance as possible. Additionally, PPA removes dominant eigenvectors from the reduced embeddings (those contributing most to the variance) to eliminate potential noise or irrelevant details. The exact number of eigenvectors to discard is typically decided through empirical evaluation or cross-validation, depending on the data characteristics and the specific application.

2.1.2. Temporal Convolutional Network

In our work, as suggested in [24], [25], we employ a Temporal Convolutional Network (TCN) during the training phase for embedded data. Originally introduced in [83], this deep neural network has shown effectiveness in log anomaly detection [25] due to its aptitude for managing sequential data such as application logs.

The strength of these networks in capturing long-term dependencies lies in their use of dilated convolutions. Unlike a standard causal convolution, a dilated convolution incorporates a dilation rate r, effectively skipping certain input values at fixed intervals.

Mathematically, consider an input sequence $x = \{x_{[1]}, \dots, x_{[q]}\}$ with each $x_{[t]} \in \mathbb{R}^n$ for time $t = 1, \dots, q$. A causal convolution with a filter size F processes only the current and past time steps, producing an output

$$y_{[t]} = \sum_{f=0}^{F-1} w_f x_{[t-f]} \tag{1}$$

where w_f denotes the convolutional filter weights. In contrast, a dilated convolution with a dilation rate r alters the convolution operation defined in (1) to

$$y_{[t]} = \sum_{f=0}^{F-1} w_f x_{[t-rf]} \tag{2}$$

The outputs from the dilated convolutional layers, designed to capture temporal dependencies at various scales, are aggregated by global pooling layers. These layers summarize the temporal features from all time steps within a feature map before passing them to fully connected layers, which ultimately output the binary classification for the log sequence. Figure 2 illustrates a typical architecture featuring three dilated convolutional layers with a kernel size of $F = 3$.

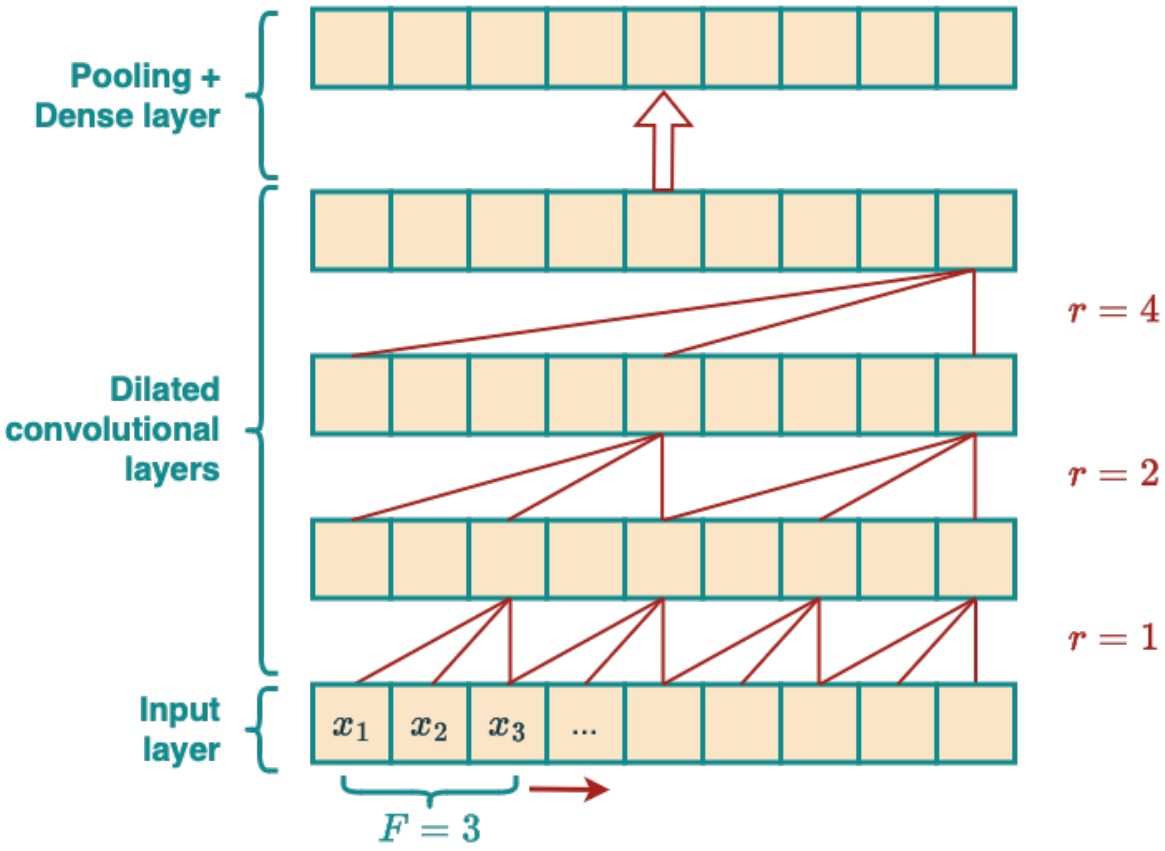

Figure 2: Temporal Convolutional Network architecture

2.1.3.Federated Learning Procedure

The proposed framework considers a federation of K local servers communicating with a coordinator, G, as indicated in Figure 3. This setup is common in critical infrastructures, where industrial control systems rely on devices (such as local servers) that both store and process data while being securely connected to external networks. The coordinating server may be one of the local nodes or reside in the cloud, potentially operated by a different party.

The FL paradigm enables sharing the same algorithm among multiple stakeholders because neither the logs nor the embeddings are exchanged among the servers. Each local server independently pre-processes its data, resulting in different embeddings, which mirrors real-world applications.

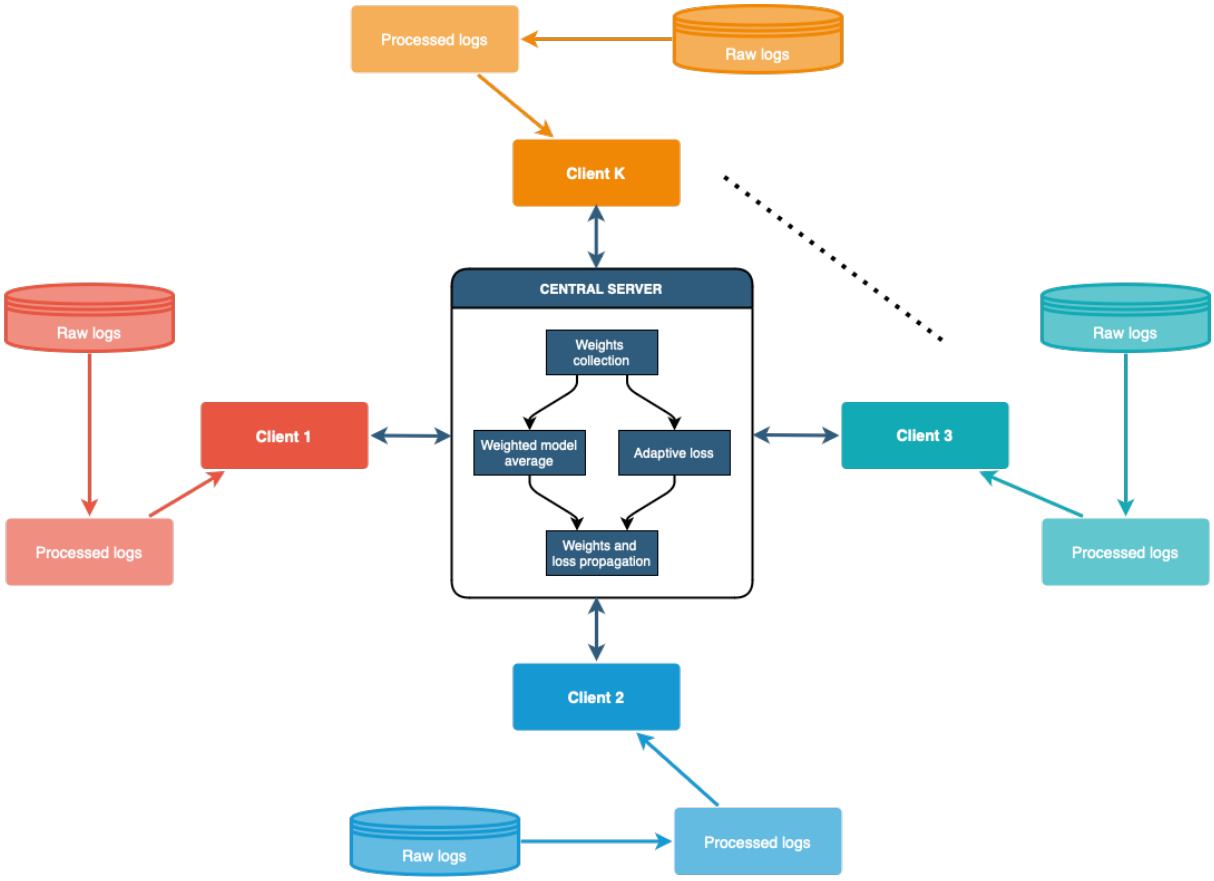

Figure 3: Federated Learning based Log Anomaly Detection

Another notable difference from other log-related FL implementations [84], [85] is the incorporation of an adaptive loss function derived from the use of AdaFed [86]. This adaptive loss function not only accelerates convergence but also evaluates whether a client has been subjected to a malicious attack. Through this performance evaluation mechanism, the coordinator G can potentially exclude an underperforming or compromised client from the training process.

Specifically, the training process unfolds as follows: during the c-th communication round (with $c = 1, \dots, E$), each client k (where $k = 1, \dots, K$) receives the TCN model from G with current weights W_G^c and trains it on its local dataset for T epochs (i.e., training rounds). The client then sends its updated weights W_k^c back to G, which evaluates their performance on a dedicated dataset—assumed to be representative of the overall machine learning task—and assigns a corresponding performance weight s_k^c. After collecting all the weights, G updates its global model weights by computing a weighted average:

$$W_g^{c+1} = \frac{\sum_{k=1}^{K} W_k^c s_k^c}{\sum_{k=1}^{K} s_k^c} \quad (3)$$

The idea behind this equation is to let clients with higher performance scores drive the learning process, while clients with lower performance—whether due to low-quality data or a malicious attack—benefit by learning from their better-performing counterparts. In the case of a malicious client, this mechanism allows for its exclusion without adversely affecting the overall training.

This performance evaluation approach also links the federation's learning process to the global model's performance. Because the data is distributed, allowing each client to concentrate on misclassified examples has been shown to enhance the global model's performance [86]. To support this, at the

beginning of each communication round, G sends out an updated TCN model along with a weighted loss function.

For the binary classification task at hand, each client minimizes a weighted categorical cross-entropy loss function defined as

$$\ell_{k(D_k,c)} = -\frac{1}{|D_k|}\sum_{j=1}^{|D_k|}\left[w^c_{normal} y_j^{D_k}\log(\hat{y}_j^{D_k}) + w^c_{abnormal}(1-y_j^{D_k})\log\left(1-\hat{y}_j^{D_k}\right)\right] \quad (4)$$

where D_k denotes the local dataset of the k-th client, $y_j^{D_k}$ and $\hat{y}_j^{D_k}$ are the true and predicted labels, respectively, and w^c_{normal} and $w^c_{abnormal}$ are the weights associated with the normal and abnormal classes.

Initially set to default values, these weights are updated at the end of each communication round based on a performance metric m representing the global model W_G^{c+1}'s effectiveness on a dedicated test set. The update rules are given by

$$w^{c+1}_{normal} = \frac{1}{m^{c+1}+\varepsilon} \quad (5)$$

$$w^{c+1}_{abnormal} = \frac{1}{(1-m^{c+1})+\varepsilon} \quad (6)$$

where $\varepsilon > 0$ is a small constant to avoid numerical issues.

2.1.4. Simulations

Our numerical simulations are carried out on the HDFS log dataset [79], which comprises 11,175,629 entries, approximately 2.58% of which contain anomalies. The parsing method selected is Spell [75], known for achieving 100% accuracy on the HDFS dataset. The federated framework is shown in Figure 3, with the number of local servers set to $K = 5$ in order to maximize parsing accuracy. As a result of dividing the dataset proportionally among the K servers, each partition holds roughly 2,235,050 (±50) log entries. Additionally, each partition contains about 21 (±2) different log keys.

For each client, logs are grouped into sequences based on their block ID. Sessions are marked as abnormal (value 1) if at least one log key in the session is flagged as anomalous; otherwise, they are labeled as normal (value 0). On average, each client has about 180,000 (±30,000) normal sessions and 4,500 (±500) abnormal sessions, which results in abnormal sessions accounting for approximately 2.45% of the total.

Log embedding then takes place, producing vectors of dimension $M = 300$ as proposed by Wang et al. [25]. By eliminating $d = 7$ dominating eigenvectors using the Post-Processing Algorithm (PPA), the embedding dimensionality is reduced to $n = N_5 = 19$. This choice is based on the fact that the fifth local server has the smallest number of log keys among all clients since the maximum number of components for PCA cannot exceed the number of log keys present in a client's dataset.

In terms of training, the simulations are executed over $E = 300$ global epochs. During each communication round, every local model is trained for $T = 3$ local epochs using the ADAM optimizer with a learning rate of 0.005 and the previously described adaptive loss function. The local TCNs are structured with four dilated convolutional layers, employing dilation rates of 1, 2, 4, and 8, followed by an Adaptive Average Pooling layer and a fully connected layer that outputs the binary classification of the log sequences.

The first simulation conducts a comparative analysis on the metric m used to evaluate the global model's performance on a dedicated test set—information that is essential for determining the weights of the adaptive loss function. Specifically, when assessing the score s_k for the k-th client based on accuracy over the test set, the influence of the adaptive loss function is compared by setting m to precision, recall, and F1-score. Although each metric provides insights into classification performance—accuracy (Figure 5) indicating overall correctness, precision (Figure 7) measuring how many predicted normals are truly normal, recall (Figure 6) assessing how many actual normals are correctly predicted, and F1-score (Figure 4) combining the latter two—the F1 metric-based model consistently outperforms the others.

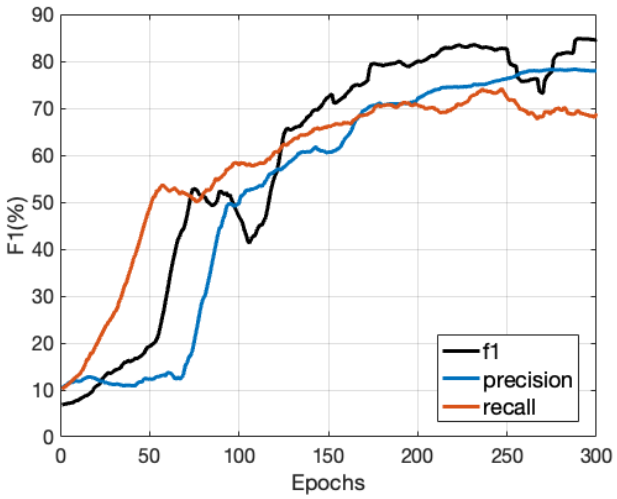

Figure 4: Comparison of metrics with different scores - F1 score

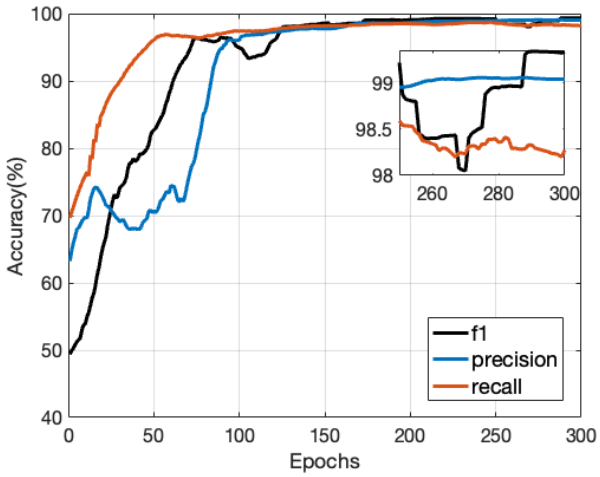

Figure 5: Comparison of metrics with different scores - Accuracy score

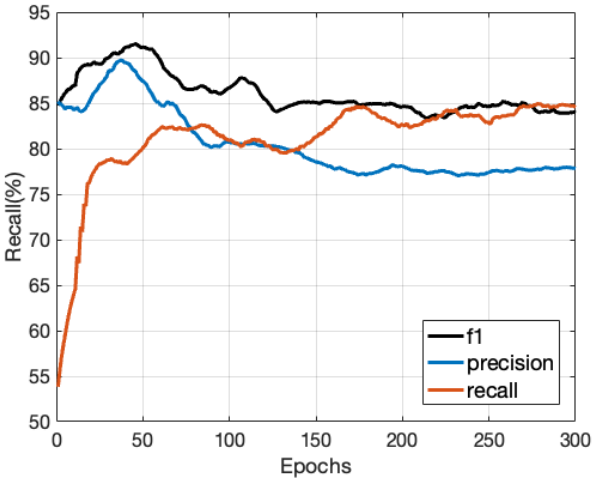

Figure 6: Comparison of metrics with different scores - Recall score

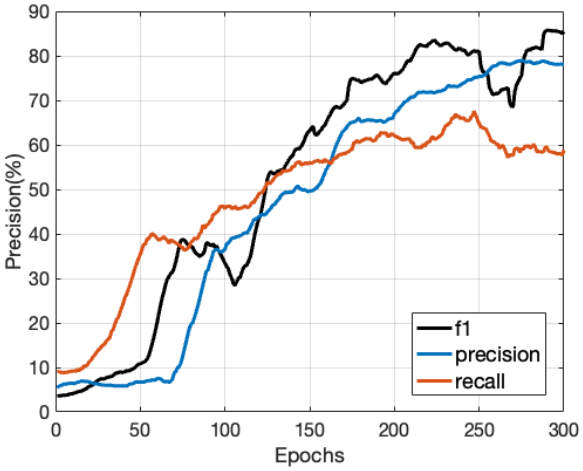

Figure 7: Comparison of metrics with different scores - Precision score

In a subsequent simulation, and with the global performance metric m fixed to the F1-score, we compare the evaluation score s_k for local clients across accuracy, F1-score, and precision. According to following Figures, the accuracy-based criteria generally yield superior performance over metrics F1 (Figure 8), Accuracy (Figure 9), and Precision (Figure 11), whereas in the recall measure (Figure 10), the F1-score outperforms accuracy by about 2%.

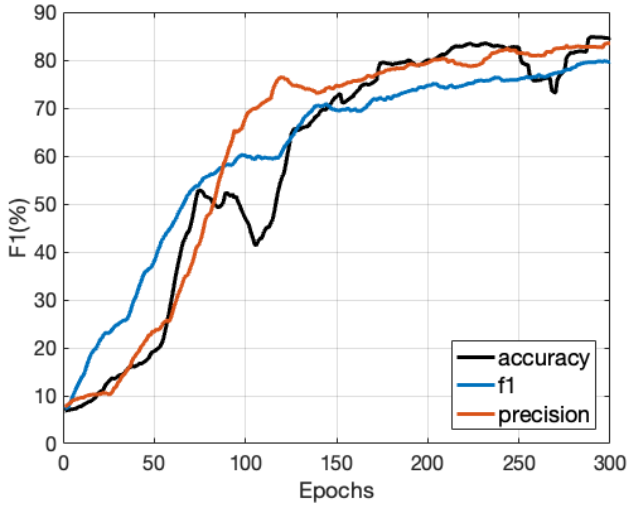

Figure 8: Comparison of scores with different metrics - F1 metric

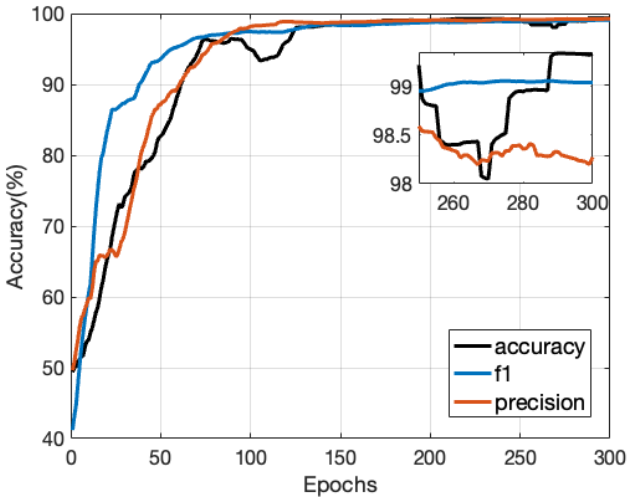

Figure 9: Comparison of scores with different metrics - Accuracy metric

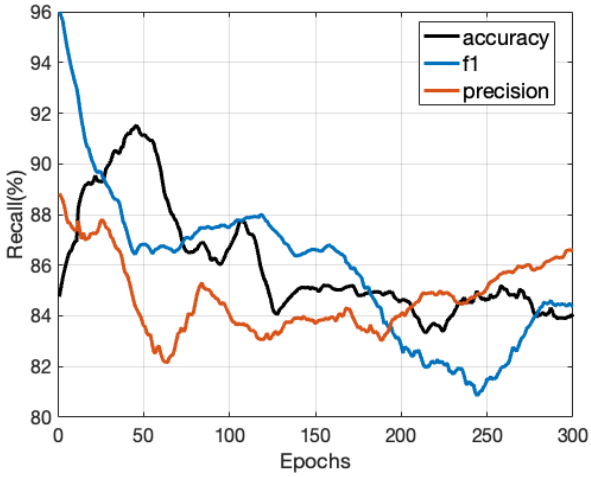

Figure 10: Comparison of scores with different metrics - Recall metric

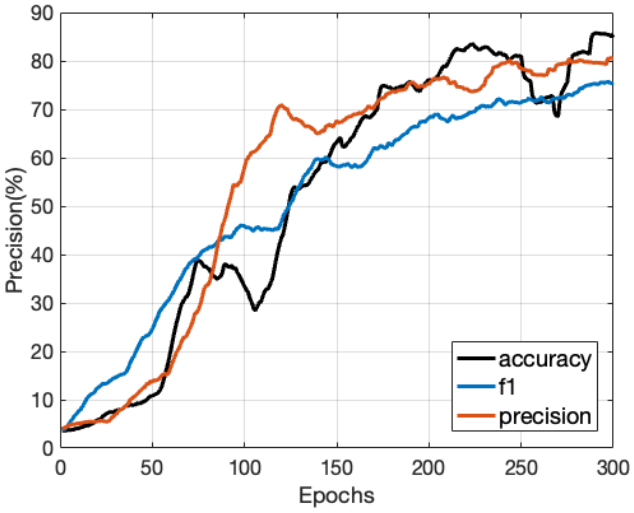

Figure 11: Comparison of scores with different metrics - Precision metric

Finally, having identified the F1-score as the most effective for m and accuracy for s_k, we compare the proposed algorithm, AdaLightLog, with the standard FedAvg approach. As demonstrated in Figure 12 AdaLightLog outperforms FedAvg across all the comparative performance metrics.

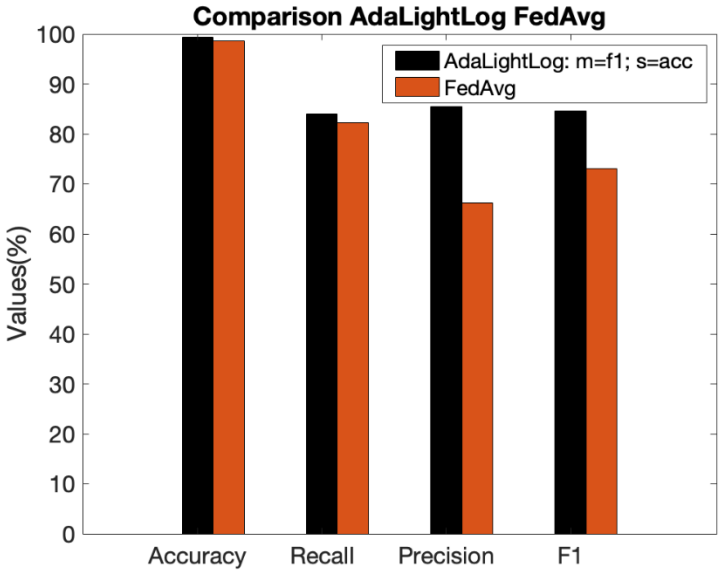

Figure 12: Comparison of performances between the proposed approach and FedAvg

2.2. Threat Detection using FL-IDS

In the context of network security, federated learning offers solutions to longstanding challenges in intrusion detection systems (IDS). Traditional IDS approaches typically rely on centralized data collection and analysis, creating bottlenecks in data processing, increasing network overhead, and raising significant privacy concerns [87]. As networks grow in scale and complexity, particularly with the advent of 5G and beyond technologies, these challenges become increasingly pronounced. The Open Radio Access Network (O-RAN) architecture represents a paradigm shift in telecommunications

infrastructure, moving from vendor-locked systems toward open interfaces. While this transformation offers benefits in terms of flexibility, cost-effectiveness, and innovation, it simultaneously expands the attack surface for cyber threats [88]. The distributed nature of O-RAN components, combined with the critical role of Beyond 5G (B5G) and 6G networks necessitates security mechanisms that can operate effectively across the network's distributed architecture.

Intrusion detection for telecommunications networks presents unique challenges due to the massive volume of data, real-time processing requirements, and diverse attack vectors targeting these critical infrastructures. Network-based attacks require sophisticated detection mechanisms that can identify unique patterns in network traffic while minimizing false alarms [89]. The evolution of this kind of attacks has necessitated a shift from signature-based detection to more advanced anomaly detection techniques leveraging machine learning and artificial intelligence.

To address these challenges, a Federated Learning-based Intrusion Detection System (FL-IDS) that can be found in the code repository: https://github.com/Metamind-Innovations/nancy_federated_learning_framework was developed which leverages distributed machine learning to detect network anomalies and security threats while preserving data privacy and minimizing communication overhead. This approach aligns perfectly with the distributed architecture of O-RAN, enabling security mechanisms to operate natively within the network's design philosophy. In this FL-IDS implementation, as shown in Figure 13, the federated learning process operates through a coordinated client-server architecture. The central server initializes a global machine learning model and coordinates the training process across distributed client nodes deployed throughout the network. Each client maintains local datasets of network traffic, performs local model training, and shares only model updates (parameters) with the central server. The server aggregates these updates to improve the global model, which is then redistributed to the clients for continued refinement. This iterative process continues until the model achieves satisfactory performance across all nodes. This FL-IDS approach has the ability to leverage the collective intelligence of distributed network components without compromising data privacy or creating bandwidth bottlenecks. Since raw network traffic data never leaves its origin, sensitive information remains protected. Furthermore, by transmitting only model parameters rather than raw traffic data, FL significantly reduces the bandwidth requirements for security operations, which is crucial for maintaining network performance while implementing robust security measures.

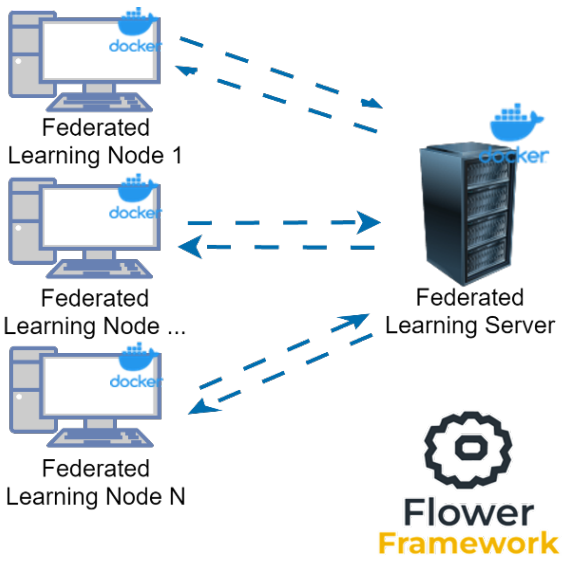

Figure 13: FL-IDS set-up

Our FL-IDS implementation is built upon the Flower (Flwr) framework [90], a specialized library for federated learning that provides the communication infrastructure necessary for distributed model training. The system comprises a central server component that coordinates the training process and multiple client nodes that perform local data processing and model training. The central server serves as the coordinator for the federated learning process. Its primary responsibilities include initializing the global model with an appropriate neural network architecture, coordinating training rounds across distributed clients, aggregating model updates from clients using advanced strategies, distributing the improved global model back to clients, evaluating global model performance, and saving the final trained model for deployment.

The implementation incorporates the FedAdagrad strategy [91] for model aggregation, providing functionality for saving the final model and tracking detailed performance metrics from each client. The client components represent the distributed intelligence of the FL-IDS system. Each client operates independently on its local network segment and is responsible for loading and preprocessing local network traffic data, training the IDS model on local data, evaluating model performance using local testing data, communicating model updates to the central server, and implementing the updated global model received from the server. One of the main points of intelligence of the FL-IDS system is the capability to facilitate deep neural network models designed to classify network flows.

One of the most critical components of the FL-IDS component is the data processing pipeline that could transform raw network traffic into structured features suitable for machine learning. This transformation process is implemented through a series of streaming techniques that operate efficiently on incoming data streams. Once the raw traffic data has been transformed into flows, our system implements a preprocessing pipeline that prepares the data for machine learning. This process includes feature selection to extract relevant attributes while discarding identifiers and metadata that could introduce bias or privacy concerns, data cleaning to handle anomalous values, and normalization to bring all features to a common scale, which is essential for neural network performance. To maintain consistency across distributed clients, a common scaler is trained on representative data from all clients and then distributed to ensure uniform feature scaling.

The implementation is flexible, supporting multiple federated aggregation strategies beyond just FedAdagrad. This flexibility allows network operators to select the most appropriate strategy based on

their specific requirements for convergence speed, model accuracy, and resilience against potential poisoning attacks. Scalability was a primary consideration also in the system design as the architecture can accommodate a variable number of clients with minimal configuration changes. This is achieved through an adaptive client management system that dynamically adjusts to available resources and ensures that the server can coordinate training effectively regardless of the number of participating nodes.

The data processing pipeline incorporates robust error handling and retry mechanisms to ensure reliability in dynamic network environments. Clients automatically attempt to reconnect to the server if connections are lost, with configurable retry intervals and maximum attempt thresholds. For deployment flexibility, the entire system has been containerized using Docker, allowing for seamless deployment across heterogeneous infrastructure environments. This containerization approach also facilitates integration with orchestration platforms like Kubernetes for production-scale deployments. Finally, and after the completion of FL, evaluation metrics are automatically calculated and reported for both local and global models, including accuracy, true positive rate (TPR), false positive rate (FPR), F1-score, and area under the ROC curve (AUC). The performance monitoring system tracks these metrics across training rounds, enabling administrators to visualize convergence patterns and identify potential anomalies in the learning process. Key features and capabilities of the FL-IDS are summarized in Table 2.

Table 2: Key features and capabilities of FL-IDS

Feature	Description	Technical Implementation
Aggregation Strategy Flexibility	Supports multiple federated aggregation methods	Custom *SaveModelStrategy* class extending Flower's strategy interfaces
Dynamic Client Scaling	Adapts to variable numbers of clients with minimal configuration	Parameterized *min_clients* setting with connection retry mechanisms
Robust Error Handling	Automatic reconnection and failure recovery	Time-based retry logic with exponential backoff in client connection code
Container-based Deployment	Streamlined deployment across heterogeneous environments	Docker and Docker Compose configurations with volume mounting for data persistence
Neural Network Architecture Support	Support of custom NN	Set-up yhe model in *create_model()*
Evaluation Metrics	Detailed performance monitoring	Integrated evaluation framework tracking accuracy, TPR, FPR, F1-score, and AUC
Streaming Data Processing	Efficient transformation from raw traffic to ML features	Pipeline utilizing tshark capture, CICFlowMeter conversion, and custom preprocessing

2.3. Federated Learning Algorithm for Improved Anomaly Detection in Cellular Networks

This section introduces a Federated Learning Anomaly Detection algorithm based on Federated Random Forests for Cell Anomaly Detection (COD) in 5G Radio Access Networks (RAN). Building on a previous work [92], we introduce a fully distributed, consensus-based approach for a federated Random Forest classifier for cell anomaly detection. As in the prior study, learning agents progressively eliminate features deemed less important by the federation. The proposed fully decentralized method leverages the benefits of federated learning while reducing the risks associated with central entity failures or attacks.

The implementation of this approach is available here: https://github.com/wronatime/fedUserAssignmentNANCY/blob/main/consensus_attack_EuCNC.ipynb. Moreover, this work has been disseminated in [92].

2.3.1. Network Scenario and Proposed Methodology

The cellular network scenario, as shown in Figure 14, consists of N base stations (BS), also known as next generation NodeBs (gNB), indexed by i. Each gNB i is responsible for managing C_i cells and providing wireless connectivity to the associated cells. It is important to note that each gNB may serve a different number of cells. Each gNB i is equipped with an edge server, referred to as ES i, which is responsible for performing anomaly detection. For this purpose, each edge server contains a dataset, denoted as D_i, which includes various information related to the cells managed by the connected base station. The cell data may include (i) timestamp, (ii) BS's Physical Resource Block (PRB) utilization, (iii) the number of connected users, and (iv) user activity data.

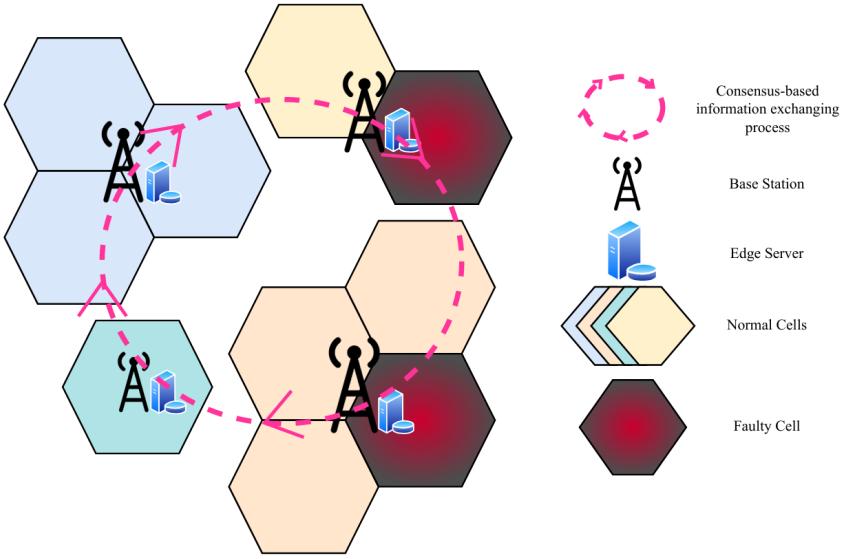

Figure 14: Cellular Network Scenario.

One of the key challenges in anomaly detection within cellular networks is identifying the optimal set of features that ensure high classification accuracy. Feature selection and elimination play a crucial role, particularly when handling large datasets that are often noisy and originate from numerous users. This study presents a collaborative, privacy-preserving feature selection approach based on Decision Trees and Random Forests. As previously mentioned, Random Forest algorithms construct an ensemble of decision trees. A decision tree follows a hierarchical structure composed of nodes, where leaf nodes represent predictions, and internal nodes correspond to decisions based on specific attributes. Each decision tree is trained on a randomly sampled subset of the dataset, following the bagging technique. For classification tasks such as anomaly detection in cellular networks, the Random Forest ensemble typically employs a majority voting strategy. In this approach, each tree casts a vote for a specific class, and the final classification is determined by the class receiving the most votes. One of the strengths of the Random Forest method is its ability to assess feature importance. The algorithm assigns an importance score to each feature based on how frequently it is selected for splitting across the ensemble. This capability aids in both feature selection and the evaluation of variable significance within the dataset.

In this study, importance scores computed at the local level are aggregated using a consensus-based approach to improve overall anomaly detection performance. Consensus theory explains how

autonomous agents can reach a common agreement through local interactions without depending on a central authority.

The dynamics of each node (or agent) governed by the consensus protocol are defined as follows:

$$\dot{x} = \sum_{j \in N_i} a_{ij}(x_j - x_i) \tag{7}$$

where:

- x_i denotes the state of agent i;
- N_i denotes the set of neighbors of agent i;
- a_{ij} denotes the weight representing the connectivity between agents.

In matrix form this can be expressed as:

$$\dot{x} = -Lx \tag{8}$$

Where L is the Laplacian matrix of the graph, obtained by the difference from the degree matrix D, which is a diagonal matrix containing the degrees of the nodes, and the adjacency matrix A, which represents the connections between the nodes.

Global consensus is achieved when the graph is connected, which occurs when the second smallest eigenvalue λ_2 of the Laplacian matrix is strictly positive [93].

By integrating consensus theory into the AI-based COD framework, the dependency on a central authority for aggregating data from local nodes is eliminated. This reduces the risks associated with single points of failure and supports scalability in large-scale cellular networks.

The consensus-based FedRF algorithm is defined in detail below:

In the initial phase, all ESs are included in the federation, where they independently train on their local datasets and compute feature importance scores. During each iteration, nodes communicate their feature importance scores only with their neighbors. By applying consensus theory, each node adjusts its scores based on the received information, achieving a global consensus on feature importance throughout the network. This can be mathematically represented as follows. Node i updates its importance scores x for the metrics m according to the following dynamical system:

$$\dot{x}_i^m(t) = \sum_{j=1}^{N_i} w_{j,i}(t)(x_j^m(t) - x_i^m(t)) \tag{9}$$

where N_i represents the number of neighbors of node i, and $w_{j,i}$ is the reliability weight of node j as perceived by node i, calculated as:

$$w_{j,i}(t) = \begin{cases} 1, if\ \left|x_i^m(t) - x_j^m(t)\right| \leq \tau \\ 0, otherwise \end{cases} \tag{10}$$

After reaching consensus, the feature with the lowest importance score is determined to be the least significant. Each node removes this feature and updates its Random Forest model accordingly. ESs that experience a decline in performance exit the federation while keeping the removed feature, whereas the others continue to the next iteration. The process continues iteratively until no agents remain in the federation.

2.3.2. Simulations

The anomaly detection dataset used in this study originates from [24] and represents a real-world LTE deployment. Data was collected every 15 minutes over two weeks from a set of 10 base stations (BSs), each managing a different number of cells. The dataset consists of 36,905 samples, with features and classification labels detailed in Table 3.

Table 3: Features of the Cell Outage Detection Dataset

Feature	Description	Measurement Unit
Time	Hour of the day	(hh:mm)
CellName	Unique identifier for BS and cell	Text String
PRBUsageUL	Resource utilization in uplink	%
PRBUsageDL	Resource utilization in downlink	%
meanThr_UL	Average carried traffic in uplink	Mbps
meanThr_DL	Average carried traffic in downlink	Mbps
maxThr_UL	Maximum carried traffic in uplink	Mbps
maxThr_DL	Maximum carried traffic in downlink	Mbps
meanUE_UL	Average number of active UEs in uplink	-
meanUE_DL	Average number of active UEs in downlink	-
maxUE_UL	Maximum number of active UEs in uplink	-
maxUE_DL	Maximum number of active UEs in downlink	-
maxUE_UL+DL	Maximum number of total active UEs	-
Anomaly	Label for supervised learning	0 or 1

To distribute the dataset, the “CellName” attribute was used to allocate data across the 10 edge servers (ESs). Specifically, each private dataset was divided into training and testing sets, with 80% reserved for training—serving as the primary input for model optimization—and the remaining 20% designated for testing. To implement the consensus procedure, the 10 Base Stations have been interconnected in such a way that each one is linked to two others, forming a ring structure, as represented in Figure 15.

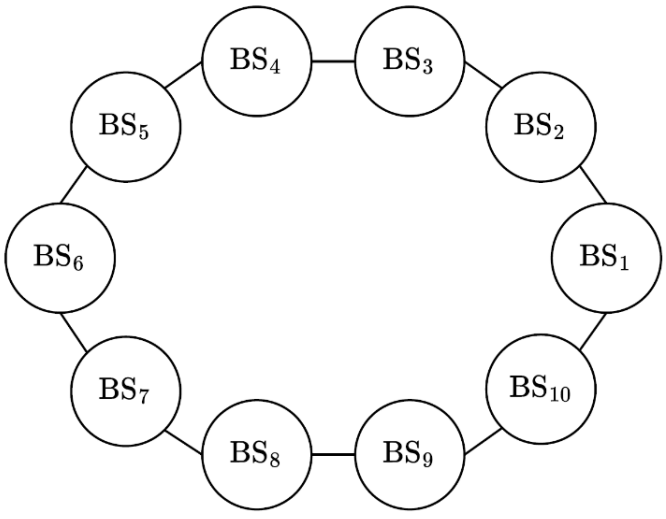

Figure 15: Simulation scenario with 10 Base Stations in a ring network topology

The performance comparison was carried out using four classification metrics, evaluated on the private test sets. Let TP (True Positives), TN (True Negatives), FP (False Positives), and FN (False Negatives)

represent the number of correctly detected anomalies, correctly identified normal samples, incorrectly detected anomalies, and missed anomalies, respectively. The evaluated metrics are as follows:

1. **Test Accuracy** – Measures the overall correctness of predictions and is computed as:

$$A = \frac{TP + TN}{TP + TN + FP + FN} \tag{11}$$

2. **Precision** – Assesses the accuracy of anomaly predictions and is defined as:

$$P = \frac{TP}{TP + FP} \tag{12}$$

3. **Recall** – Represents the model's ability to correctly identify all anomalies and is given by:

$$R = \frac{TP}{TP + FN} \tag{13}$$

4. **F1-Score** – Represents the harmonic mean of precision and recall, balancing both metrics:

$$F_1 = \frac{2\,(P \cdot R)}{(P + R)}. \tag{14}$$

The proposed approach has been compared against two alternative classification methods:

- **DisRF**: A local Random Forest classifier without collaborative feature selection.
- **FedRF**: A local Random Forest classifier with collaborative feature selection trough mathematical averaging proposed in our previous work [23].

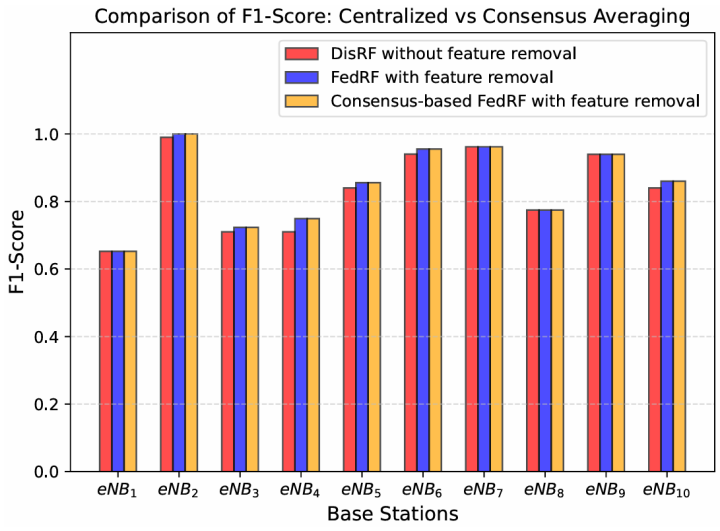

Figure 16: Performance Comparison.

The results shown in Figure 16 were obtained in the absence of attacks from malicious nodes. As expected, the F1-scores for consensus-based averaging and standard mathematical averaging are identical across all base stations. This equivalence demonstrates that consensus averaging achieves the same performance as traditional mathematical averaging performed by a central authority, validating its effectiveness and reliability as a distributed computation method. In contrast, the F1-scores in the no-cooperation scenario DisRF are consistently lower for certain base stations. This decline emphasizes the essential role of cooperation and information sharing in enhancing fault detection performance.

Consider now the case in which some nodes are malicious and inject false data about feature importance scores during the federated learning process. The attack strategy considered in this work follows these steps.

Assume that F features are used in a specific round of the federated process. The attacker computes the feature importance vector on its dataset and selects the feature with the highest importance score using

$$i^* = argmax\, f_{score} \tag{15}$$

The attack vector is then structured so that its i-th component is defined as follows:

$$f_{attack}^{i} = \begin{cases} \beta/F, & if\ i = i^* \\ \dfrac{1-\beta/F}{F-1}, & otherwise \end{cases} \tag{16}$$

where β is a free attack parameter controlled by the malicious node. This parameter must be set so that the importance assigned to i^* is lower than all others. By simple calculations, this condition is met when:

$$\beta < \frac{F}{F-2} \tag{17}$$

Through this approach, the attacker assigns the lowest importance score to the most crucial feature in its dataset. Notably, the sum of the attack vector remains equal to 1, making it appear as if the values represent legitimate importance scores.

The performances of BS2 and BS6 are evaluated in the presence of attacking nodes. However, similar observations apply to any other base station. The simulation results in the left panel of Figure 17 illustrate the degradation in BS2's performance as the number of attacking nodes increases. Performance is measured using the F1-score under two scenarios: (i) FedRF with standard averaging and (ii) consensus-based FedRF. As the number of attackers grows from 1 to 9, the F1-score begins to decline from 6 attackers onward when using centralized averaging. In contrast, with the proposed consensus algorithm, performance remains unaffected even with 9 attackers. This highlights the centralized approach's vulnerability to malicious interference, whereas the consensus-based method demonstrates maximum resilience by maintaining a stable F1-score regardless of the number of attackers.

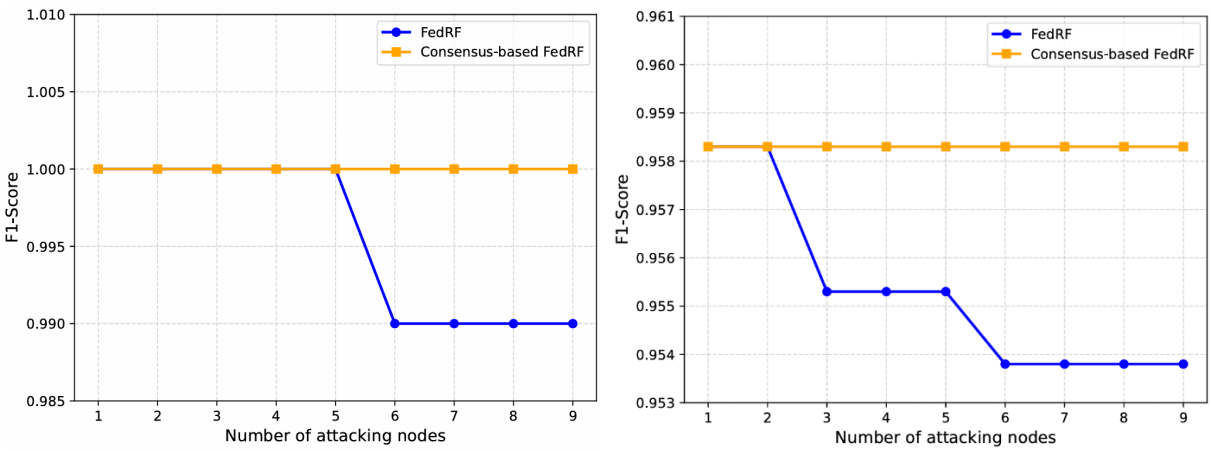

Figure 17: Variation of F1-score on BS2 (left) and BS6 (right) against increasing number of attackers.

A similar trend is observed for BS6, as shown in the right panel of Figure 17. In this case, the F1-score drops with just 2 attackers under centralized averaging, whereas the consensus-based approach

preserves full stability. This reinforces its effectiveness in adversarial settings across multiple base stations.

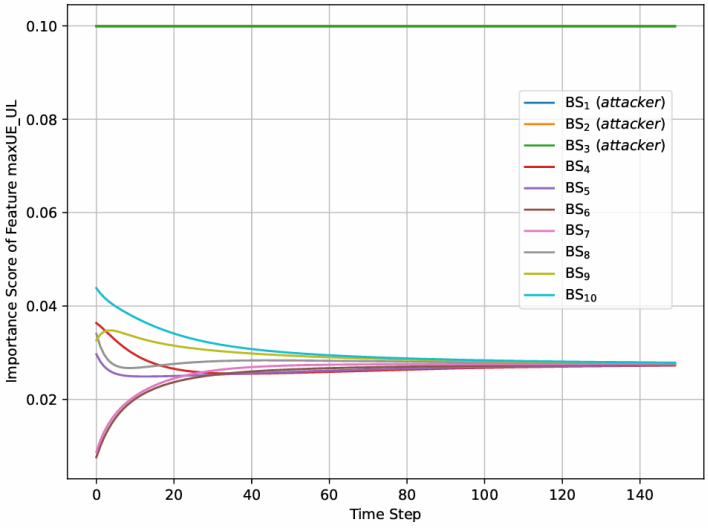

Figure 18: Time evolution of the feature maxUE_UL during the consensus procedure with 3 attackers and 7 normal nodes.

Eventually, Figure 18 illustrates how the consensus algorithm operates when handling the maxUE UL feature in the presence of 3 attacking nodes and 7 normal ones. The results confirm that the proposed algorithm enables normal nodes to reach consensus while effectively mitigating the false data injected by attackers, which are excluded from the consensus reached by the normal agents.

To conclude, it is possible to state that our federated cell anomaly detection procedure addresses the vulnerabilities of traditional centralized methods by eliminating the single point of failure and introducing a consensus process that mitigates the impact of malicious nodes. Specifically, the implemented solution prevents adversarial entities from manipulating feature importance scores, thus ensuring the integrity of the anomaly detection process. By adopting a decentralized approach, feature removal decisions are made collectively within the network, reducing the risk of biased or compromised outcomes.

2.4. Memory Traffic Monitor Tool via ARM Performance Counters

This section describes the Memory Traffic Monitor tool that helps detect Denial of Service (DoS) attacks against user applications with the use of ARM performance counters.

Modern ARM processors are equipped with *performance counters*, that are dedicated hardware for counting hardware events during the execution of a program. Performance counters can track the number of clock cycles and the amount of instructions executed by an application, thus providing a way to measure its performance. Additionally, ARM performance counters allow counting the total number of cache access events and the corresponding hit-and-miss events, which specifically help evaluate the memory performance of the application under analysis. These counters are accessible by the programmers through the Performance Monitoring Unit (PMU) registers.

The Memory Traffic Monitor tool leverages the ARM performance counters on the Texas Instruments AM69 board, equipped with an eight-core Cortex A72 processor, to analyze the memory traffic of any application running on the board. In detail, this monitoring tool reads the PMU registers of the board and helps determine if the application under analysis is experiencing a DoS attack by monitoring its memory traffic. The PMU registers involved in the monitoring process acquire the number of level-1 and level-2 cache memory accesses and their corresponding *miss* events: if the cache miss rate is

higher than the average for that application in an attack-free situation, this might be an indication of an ongoing DoS attack against the application under analysis.

The memory traffic monitor attaches to any application given its PID, so that to analyze its memory traffic. At the end of the monitoring, the tool writes on a file the memory traffic stats for the user to read and interpret them. The monitor reads from the PMU the following hardware events:

- L1-DCACHE-LOADS
- L1-DCACHE-LOAD-MISSES
- L1-ICACHE-LOADS
- L1-ICACHE-LOAD-MISSES
- L2D_CACHE_ACCESS
- L2D_CACHE_REFILL

The above events represent the number of memory accesses and misses performed by the application to data and instruction level-1 cache, and level-2 cache access and refill events. A sudden increase in cache miss rates can be a symptom of an ongoing DoS attack.

The monitoring tool is developed leveraging the PAPI library from the University of Tennessee, Knoxville (UTK) [https://github.com/icl-utk-edu] to read the ARM performance counters. This low-level API manages hardware events in user-defined groups called Event Sets. It is possible to monitor both native and preset events. Native events are all those events that are countable by the CPU and whose names are platform-dependent, while presets or predefined events are a common set of events deemed relevant and useful for application performance tuning. These are mappings from symbolic names (PAPI preset names) to machine-specific definitions for a particular hardware resource.

Two steps are required to execute the monitoring tool:

- compile and install the PAPI library by running the script "`compile.sh`" (this needs to be done only the first time the monitor is executed);
- run the traffic analyser with the script "`start-monitor.sh`" and pass as argument the PID of the process to monitor; example of usage: "`./start-monitor.sh 12345`" to monitor process with PID 12345.

The latter script is a simplified interface to execute the monitoring application to trace the six events described earlier. The general monitoring tool executed by the script accepts the following parameters:

- `-p <pid>`: instructs the tool to monitor the application whose process has PID `pid`.
- `-ne <nevts>`: sets the number of events (`nevts`) to be monitored by the tool (between one and six);
- `-e <evt1,evt2,...,evtn>`: configures the tool to monitor the provided events separated by commas;
- `-h`: prints a help message on the terminal with further details on the parameters.

The tool writes the values of the performance counters in a *.csv* file, whose name uniquely identifies the application that was monitored and the events that were traced.

2.5. Federated User Reallocation in Cellular Networks

This section will present a Federated, AI-based user reallocation in cellular networks, after the occurrence of a sleeping cell event. The situation is the one in which one cell within the network stops working, thus not providing anymore its services to the users located in that geographic area. Affected users shall be assigned to the neighboring cells, with the softest handover possible, i.e. guaranteeing the continuity of services. Our strategy aims to efficiently reassign affected users to neighboring cells, minimizing service interruption and maintaining stable network performance, thus reducing the impact of the reassignment process on the quality of service (QoS) of the users that were already connected to the neighboring cells.

The Federated assignment is demanded to a group of agents (each one associated with a single cell) based on the Reinforcement Learning paradigm, described in the next subsection.

The implementation of this approach is available here: https://github.com/wronatime/fedUserAssignmentNANCY.

2.5.1. Recalls on Markov Decision Processes and Reinforcement Learning

Reinforcement Learning (RL) is a subfield of Machine Learning (ML) focusing on training intelligent agents to perform actions over a dynamic environment, based on observations and rewards [94]. The environment is formalized in the form of a Markov Decision Process (MDP), a model for sequential decision-making when outcomes are uncertain. Said model sticks with the Markov property, i.e., its future evolution is independent of its history.

Formally, an MDP is defined as a tuple $< \mathcal{S}, \mathcal{A}, P, R, \gamma >$, where:

- $\mathcal{S}$ is the state space;
- $\mathcal{A}$ is the action space;
- $P(s'|s,a)$ is the transition probability function, i.e., the probability of going from s to s' when performing action a;
- $R(s,a)$ is the reward function the agent gets after performing action a when in s;
- γ is the discount factor, determining the agent 's preference between the immediate rewards ($\gamma \approx 0$) and future ones ($\gamma \approx 1$).

The goal in an MDP is to learn an optimal policy $\pi: \mathcal{S} \rightarrow \mathcal{A}$ to maximize the expected discounted sum of instantaneous rewards over a potentially infinite horizon, known as the action value function $Q(s,a)$:

$$Q(s,a) = E\left[\sum_{t=0}^{\infty} \gamma^t R_t |\, s_0 = s, a_0 = a\right]. \quad (18)$$

RL algorithms are typically exploited to solve MDP-related problems, without assuming the complete knowledge of P. In this setting, any RL procedure requires a training phase, in which the agent learns to apply actions according to the current state, and an evaluation phase, in which it is possible to appraise the algorithm's performance against some requirements or KPIs. Ingenious exploring techniques are necessary for the training phase of RL. One such technique is the ϵ-greedy, in which the quantity of exploration versus exploitation is controlled by a value $0 \leq \epsilon \leq 1$. Exploitation is selected with probability $1 - \epsilon$, and the agent selects the course of action that it thinks will have the best long-term impact (links between actions are broken evenly at random). As an alternative, exploration is selected with probability ϵ, and the action is randomly selected uniformly. Although ϵ is often a fixed

value, it can be changed adaptively based on heuristics or in accordance with a timetable, which would cause the agent to explore less and less.

DRL extends traditional RL by leveraging deep neural networks to efficiently approximate complex mappings between high-dimensional states, actions, and rewards. This enables more effective learning of optimal control policies, particularly in environments with intricate and large-scale state spaces [95]. In the following, some recalls of the DRL procedure involved in this study are provided.

2.5.2. Dueling Double Deep Q Networks

Deep Q-Networks (DQN) constituted an important innovation in the field of AI [96]. They have demonstrated remarkable performance in solving RL tasks, often beating humans in playing various games. DQN are an extension of the traditional —Q-learning algorithm, in which the action space is still finite, but it is possible to have continuous states instead of simple discrete observations. This is realized using a neural network as a function approximator. However, they suffer from overestimation bias due to the maximization step in Q-learning. Double Q-Networks (DDQN) and Dueling Q-Networks address this issue and enhance performance. The combination of these techniques results in the Double Dueling Deep Q-Network (D3QN), which refines Q-learning by mitigating overestimation and improving state-value estimation. Hence, D3QN integrates Double Q-learning and Dueling Network Architecture to improve the stability and performance of Q-learning in DRL. Let us define the following quantities:

- $Q(s, a; \theta)$: Q-value estimated by the online network with parameters θ;
- $Q(s, a; \theta^-)$: Q-value estimated by the target network with parameters θ^-;
- r: observed instantaneous reward;
- s: current state;
- s': next state;
- a: action taken at state s.

The target value used for updating the network is computed as:

$$y = r + \gamma Q'(s', argmax_{a'} Q(s', a'; \theta); \theta^-) \tag{19}$$

Here, the action is selected using the online network but evaluated using the target network, reducing overestimation bias. The loss function minimized during training is:

$$L(\theta) = E_{(s,a,r,s') \approx D}\left[\left(y - Q(s, a; \theta)\right)^2\right]. \tag{20}$$

where D is the experience replay buffer, i.e. a dataset containing all the 4-tuple records about state, action, reward, and next state. To realize the dueling structure, the Q-value is decomposed into a state-value function $V(s)$ and an advantage function $A(s, a)$:

$$Q(s, a) = V(s) + \left(A(s, a) - \frac{1}{|\mathcal{A}|}\sum_{a'}^{N} A(s, a')\right). \tag{21}$$

This formulation helps the agent learn which states are valuable, regardless of the specific action taken.

By combining these mechanisms, D3QN has proven to be able to achieve superior stability and efficiency with respect to its predecessors in many RL applications [97], [98].

2.5.3. Cellular Network Modeling

In this study, we consider a multi-cell 5G network comprising multiple MIMO-enabled macro base stations (BSs) deployed in a hexagonal grid topology. The network structure is illustrated in Figure 19

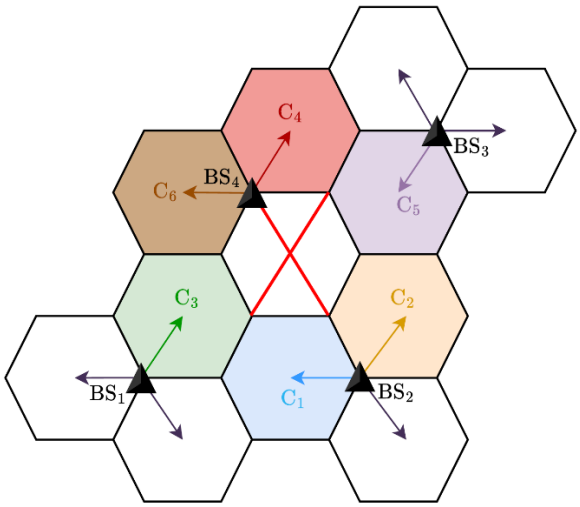

Figure 19: Illustration of a cellular network with a macro base station outage scenario

The cellular network consists of several hexagonal cells, each served by a macro base station (BS). The four base stations $BS_1, \dots, BS_4$ are positioned at the intersection of their respective cells. Each BS is equipped with multiple antennas to support MIMO communication and serves multiple users within its coverage area. Hence, each BS employs directional beamforming, as indicated by the colored arrows in the figure. These beams optimize signal transmission towards the users while minimizing interference to adjacent cells.

A key challenge in 5G networks is the occurrence of cell outages, which may result from hardware failures, power shortages, or unexpected environmental conditions that may disrupt power transformers, or physically damage hardware equipment. In Figure 19, the central hexagonal cell denoted with a red cross represents an outage area, which disrupts communication for users inside it.

To mitigate service degradation in the outage region, the 6 neighboring cells around the faulty one, denoted as $C_1, \dots, C_6$ shall compensate for the anomaly, providing connectivity services to the users within the sleeping cell. Hence, with the outage scenario in place, users in the affected region need to be reassigned to the neighboring cells. In particular, the relationship between BSs and cells is denoted as follows:

- BS_1 governs cells C_4, C_6 and the sleeping cell itself;
- BS_2 governs cell C_3;
- BS_3 governs cells C_1, C_2;
- BS_4 governs cell C_5.

This user assignment process must consider factors such as signal quality, load balancing, and interference management. In this work, we propose a federated reinforcement learning-based approach using the Federated Double Dueling Deep Q-Network (FedD3QN) to optimize user assignment while maintaining network performance and minimizing additional interference.

Let N be the number of users within the sleeping cell, and let $i = 1, \dots, N$ be the users' indexing variable. Similarly, let $j = 1, \dots, 6$ be the indexing variable for the neighboring cells. Each user is

assigned a quality of service (QoS) class q_i depending on its required datarateR_i^*, according to the following relation:

$$q_i = \begin{cases} 1, & if\ R_i^* = 1\ Mbps \\ 2, & if\ R_i^* = 5\ Mbps \\ 3, & if\ R_i^* = 10\ Mbps \end{cases} \tag{22}$$

The actual datarate received by users shall depend on the specific cell to which they connect. To model it, let the received-signal-received-power $RSRP_i^j$ be

$$RSRP_i^j = P^j + G_{h,i}^j + G_{r,i} - L_i^j, \tag{23}$$

The single terms of the previous equation are defined as follows:

- P^j is the transmitting power of the BS in the cell j;
- $G_{h,i}^j$ is the horizontal gain given by the following equation [99]:

$$G_{h,i}^j = G_{MAX} - \min\left(12\left(\frac{\phi_i^j - \phi_0^j}{HPBW_h}\right)^2, FBR_h\right), \tag{24}$$

  where ϕ_i^j is the horizontal angle between the BS position and the user i, ϕ_0^j is the beam direction of the cell j, $HPBW_h$ is the horizontal half-power beamwidth, and FBR_h is the front back ratio. In this work, we assume that the vertical gain, which depends on the elevation angle between the antenna beam and the user, can be approximated to 0 dB due to the small difference between the antenna's tilt angle and the elevation angle.
- $G_{r,i}$ is the receiver antenna gain;
- L_i^j is the path-loss between the user and the cell, computed as a 3D Urban Marco Path Loss based on 3GPP TR 38.901 [100]. It depends on the 5G signal frequency f and on the distance d_i^j between the user and the BS handling cell j.

Based on the RSRP, the signal to noise ratio (SNR) is

$$SNR_i^j = \frac{RSRP_i^j}{N_0}, \tag{25}$$

where N_0 is the thermal noise, usually defined as $N_0 = k_B TW$, with k_B being the Boltzmann's constant, T is the environment temperature, and W is the allocated bandwidth. By the Shannon-Hartley theorem, the spectral efficiency may be computed as

$$\eta_i^j = \log_2(1 + SNR_i^j). \tag{26}$$

In 4G and 5G networks, the number of Physical Resource Blocks (PRBs) required to serve a user depends on the desired data rate and the SNR. In particular, considering Eqs. (25) and (26), it follows that

$$B_i^{j,*} = \frac{R_i^*}{\eta_i^j}. \tag{27}$$

Each PRB, depending on the specific numerology μ adopted by the BS [101], has its own bandwidth B_{PRB}. Finally, the number of required PRBs to serve the user is simply computed as

$$PRB_i^{j,*} = \left\lceil \frac{B_i^{j,*}}{B_{PRB}} \right\rceil. \quad (28)$$

Now we are ready to define the cell load dynamics. This quantity varies based on new user requests. Let PRB_t^j be the total number of available PRBs in the cell j, and let $PRB^j(k)$ be the number of PRBs of the j-th cell occupied at the generic time k. Define the cell load as

$$L^j(k) = \frac{PRB^j(k)}{PRB_t^j}. \quad (29)$$

The load dynamics is then formulated according to the following discrete-time system

$$L^j(k+1) = \min\left(L^j(k) + \frac{\sum_{i=1}^{N} PRB_i^{j,*}(k)}{PRB_t^j}, 1 \right). \quad (30)$$

The $\min(\cdot)$ operation guarantees that the system variable does not exceed its upper bound, namely 1, or 100% of PRBs allocated. From a control system perspective, $PRB_i^{j,*}$ can be treated as a disturbance signal at time k, which cannot be changed at will. What can be changed by a decision support system is to decide the connectivity relationship between users and cells, i.e., which cell to be selected to serve each specific user.

The above-defined 5G cellular network system is translated into a MDP in the following way. Suppose that, at each time k exactly one user must be assigned. The state space $\mathcal{S}$ is

$$S = < q, L^1, \dots, L^6, RSRP^1, \dots, RSRP^6 >, \quad (31)$$

while the action space is

$$A = < 1, \dots, 6 >, \quad (32)$$

i.e. a discrete set made of 6 elements - the 6 different user assignment choices that can be done in the post-fault scenario.

Regarding the reward r, it shapes a trade-off between trying to guarantee the maximum RSRP possible to the users, while also keeping the cell loads as low as possible:

$$r(a) = \alpha\left(RSRP_i^a - \max_j RSRP_i^j \right) - \beta\left(L^a - \min_j L^j \right), \quad (33)$$

where a is the specific action, and α and β are weights that, if properly tuned, allow to prefer the maximization of the RSRP over the minimization of cell loads, and vice versa.

2.5.4. Federated D3QN for User Association

The main contribution of this work is the introduction of a novel federated strategy to assign users within a sleeping cell to one of the neighboring cells surrounding the faulty one. To do so, we extend the traditional RL framework going towards a federated RL (FRL) setting, where multiple agents collaboratively learn a shared policy while keeping their data decentralized. Unlike centralized RL, where all experiences are collected in a single location for training, FRL ensures privacy and reduces communication overhead by allowing local updates at distributed agents before aggregating the knowledge globally. Federated learning (FL) was originally proposed for supervised learning tasks, where multiple clients train local models and periodically aggregate their parameters at a central server [102]. In FRL, the goal is to train an RL agent in a distributed manner across multiple

environments without sharing raw experience data. Instead, each agent trains locally and sends model updates to a central server for aggregation, following the standard Federated Averaging (FedAvg) principle, i.e. the global weights are a uniform average of the local ones.

Now, suppose that each cell has its own D3QN agent. The training phase is performed over E episodes, and each episode e consists of N steps, mimicking a sequential user assignment process. At each step k of an episode, the k-th user demands the assignment to the cell towards which it has the highest RSRP value, and that cell becomes the cell at that time step. The associated agent computes an action, based on a ϵ-greedy strategy, and let the user connect consequently. It is worth noting that, in general, it is not true that the called agent suggests the user to connect to its own cell: the agent can freely choose any of the other cells depending on the situation.

To realize the federation, every f_{FED} episodes, the weights of the six agents are averaged (federated update) using a simple arithmetic mean without weighing each client:

$$\theta_{global} = \frac{1}{6}\sum_{i=1}^{6} \theta_i \,, \tag{34}$$

where θ_i is the online network parameters of agent i. Due to the inherent asynchronism of the training phase (at each step each agent is activated an unknown number of times), the federation process is performed also on the target networks of the D3QN agents. These operations require the presence of a central unit, capable of (i) retrieving the models from the clients, and (ii) redistributing the averaged model to all the agents. The pseudocode of the FedD3QN algorithm for the after-fault user assignment is reported in Figure 20.

2.5.5. Simulations and Results

The simulation scenario faithfully reproduces the cellular network depicted in Figure 19, with each cell being a regular hexagon with a size of 700 m. The other system parameters are reported in Table 4. Note how the fixed parameters assume the same numerical values independently of the specific user or cell. As per the signal frequency, the 5G NR frequency band n78 [103] has been chosen. It operates in the 3500 MHz range, commonly referred to as the 3.5 GHz band or C-band.

This band is designated for Time Division Duplexing (TDD) and is widely used for 5G deployments due to its relatively common availability and potential for high-capacity throughput. The n78 band spans from 3300 MHz to 3800 MHz, supporting channel bandwidths ranging from 10 MHz to 100 MHz. It has been used by many countries so far, including Australia, Canada, and Italy [104].

Algorithm 1 Federated D3QN for User Assignment in 5G Cellular Networks

Require: Number of episodes E, number of users N, federated update interval f_{FED}

Initialize D3QN agents $\{A_1, A_2, \ldots, A_6\}$ for the neighboring cells
for each episode $e = 1$ to E **do**
  Reset randomly the cellular network environment
  for each step (user) $k = 1$ to N **do**
    Measure $\text{RSRP}_k^j \ \forall j = 1, \ldots, 6$
    Identify active agent A_{j^*} where $j^* = \arg\min_j \text{RSRP}_k^j$
    Let s_k be the current state
    Agent A_{j^*} selects action $a_k \in \mathcal{A}$ using ϵ-greedy policy
    Assign user k to the cell corresponding to action a_k
    Observe reward r_k and next state s_{k+1}
    Store transition (s_k, a_k, r_k, s_{k+1}) in agent A_{j^*}'s replay buffer
    Perform D3QN update on agent A_{j^*}
  end for
  if $e \mod f_{\text{FED}} == 0$ **then**
    Compute global weights for the online network $\theta_{\text{global}} = \frac{1}{6}\sum_{i=1}^{6} \theta_i$
    Compute global weights for the target network $\theta^-_{\text{global}} = \frac{1}{6}\sum_{i=1}^{6} \theta_i^-$
    for each agent A_i **do**
      Update weights: $\theta_i \leftarrow \theta_{\text{global}}; \theta_i^- \leftarrow \theta^-_{\text{global}}$
    end for
  end if
end for

Figure 20: Federated D3QN for User Assignment in 5G Cellular Networks

Table 4: Numerical Parameters used in the simulation

Parameters [meas. unit]	Numerical Values
P [W]	10
Gr [dBi]	0
GMAX [dBi]	18
HPBWh [deg]	65
FBRh [dB]	30
f [GHz]	3.5
T [K]	293
W [MHz]	50
PRBt [–]	270
BPRB [kHz]	180

The D3QN agents consist of two neural networks having two hidden layers of 32 neurons each and they do share the same hyperparameters: $\gamma = 0.99$, batch size 64, replace parameter $\rho = 500$, learning rate $\zeta = 5 \times 10^{-4}$.

As per the exploration/exploitation strategy, a modified ϵ-greedy one has been adopted. Let E denote the number of training episodes, E_1 denote the number of exploration episodes and E_2 denote the number of exploitation episodes. Moreover, let ϵ_{min} be the minimum value assumed by ϵ throughout the learning episodes. Hence the ϵ function through the episode can be defined as

$$\epsilon(e) = \begin{cases} 1 & if\ e \leq E_1 \\ 1 + (\epsilon_{min} - 1)\dfrac{e - E_1}{E - E_2 - E_1} & if\ E_1 < e \leq E - E_2 \\ \epsilon_{min} & if\ E - E_2 < e \leq E \end{cases} \tag{35}$$

The parameters are chosen in the following way: $E = 200$, $E_1 = E_2 = 20$, $\epsilon_{min} = 1 \times 10^{-4}$. A visual demonstration of the epsilon curve is depicted in Figure 21.

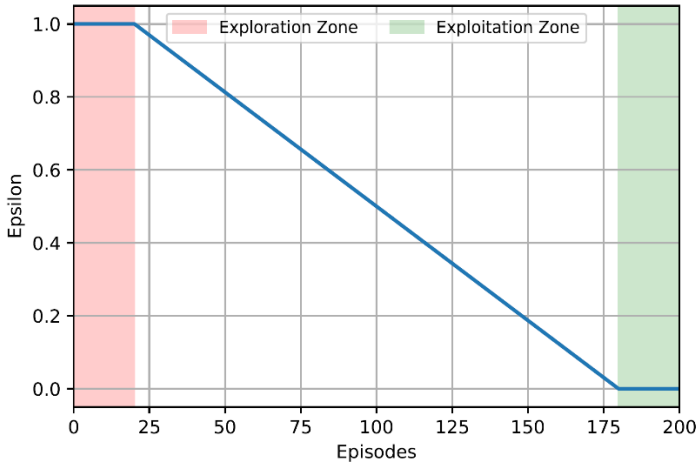

Figure 21: Epsilon curve during the training phase

Eventually, the federation parameter f_{FED} has been chosen equal to 25, for a total of 8 federation rounds during the D3QN training stage.

To evaluate the performance of the proposed Federated D3QN algorithm, three benchmark approaches were considered.

1. The first baseline is the **Max RSRP strategy**, where each user is assigned to a neighboring cell that provides the highest Reference Signal Received Power (RSRP), following a signal-strength-based selection similar to traditional handover logic.
2. The second benchmark is the **Least Loaded method**, in which the user is connected to the cell currently experiencing the lowest resource utilization, measured in terms of the number of occupied Physical Resource Blocks (PRBs), aiming to balance the load across neighboring cells.
3. The third comparison approach is a centralized variant of the learning-based method, namely **Centralized D3QN**, where a single agent oversees the decision-making for all user assignments, leveraging global state information without any federated structure. This setup allows for a direct comparison between distributed and centralized RL paradigms.

Figure 22 depicts the local scores obtained by the federation of agents. All of them demonstrate a general upward trend in cumulative rewards, indicating learning and improvement in decision-making over time. Moreover, variability in rewards early in training gradually decreases, suggesting increased stability and convergence, and the magnitude and rate of improvement vary slightly across agents, due to the fact that each agent is called a different amount of times during the training stage, depending on the users' location within the faulty cell. Hence, this suggests that the federated D3QN approach effectively enables agents to learn a meaningful policy for user assignment in their respective cells while benefiting from shared learning across the federation.

Figure 23, instead, presents a performance comparison between a Centralized D3QN agent and a Federated D3QN (FedD3QN) framework during the training phase. Both approaches show a clear upward trend, indicating learning and convergence over time. The Centralized D3QN starts off slightly better and maintains a marginal lead for most of the training duration. The FedD3QN demonstrates slightly higher variance in early episodes but steadily catches up. By episode ≈ 150 onward, both methods exhibit very similar performance, indicating the effectiveness of the federated method. This comparison illustrates that while the centralized approach benefits from access to global data and

converges slightly faster, the FedD3QN approach achieves comparable performance despite operating under distributed data constraints.

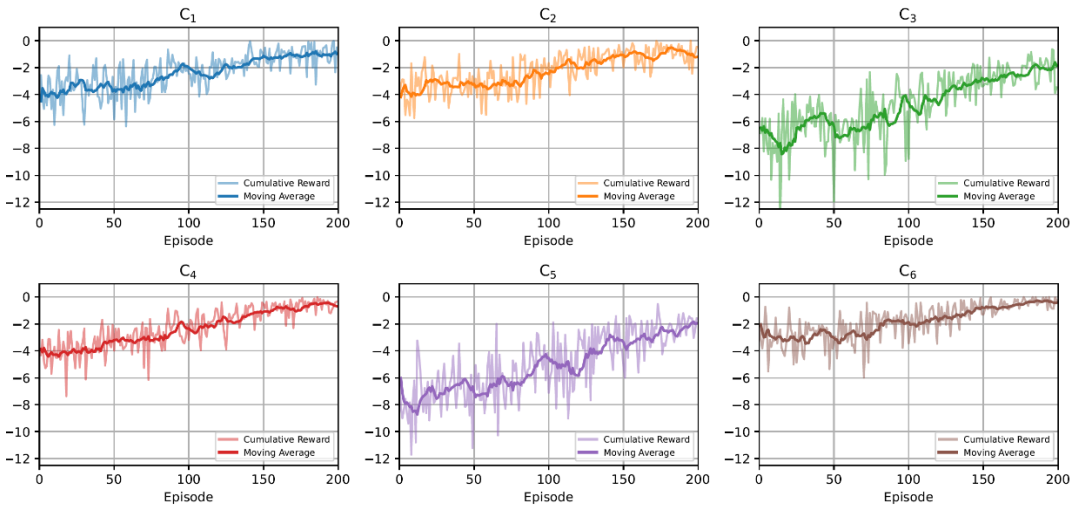

Figure 22: FedD3QN agents' local scores

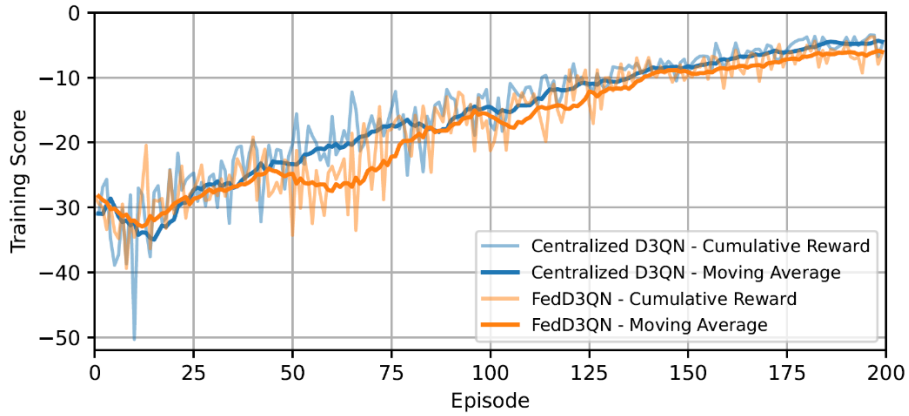

Figure 23: Centralized VS Federated Training

This result supports the viability of FRL in scenarios where data privacy or communication limitations prevent centralized training.

Figure 24 illustrates the evolution of load percentages over $N = 120$ time steps (and, hence, users) for each of the six neighboring cells under the four different user assignment strategies: Max RSRP, Least Loaded, Centralized D3QN, and Federated D3QN.

In the Max RSRP case, the load distribution is highly uneven. Some cells, such as C_5 accumulate significantly more users than others, resulting in a steep and disproportionate increase in load, while others like C_6 remain only moderately loaded. This confirms the typical behavior of signal-strength-driven strategies, where strong-RSRP cells attract most users regardless of their current load, leading to resource contention and potential congestion.

The Least Loaded strategy exhibits the most uniform load accumulation across all cells. The curves for all six cells are tightly clustered, increasing in near-parallel fashion. This validates the algorithm's intention: always assign users to the cell with the lowest current load, which minimizes disparity. However, while this method ensures fairness, it can result in poor signal quality, as users may be assigned to suboptimal RSRP cells simply to balance the load.

The Centralized D3QN approach balances load fairly well, though with slightly more variance than the Least Loaded strategy. The trajectories of the cells are closely grouped, but there is still some natural differentiation, indicating that the agent is optimizing both signal quality and load distribution instead of prioritizing just one. Importantly, this strategy relies on global information and centralized decision-making, which may limit scalability in practical deployments.

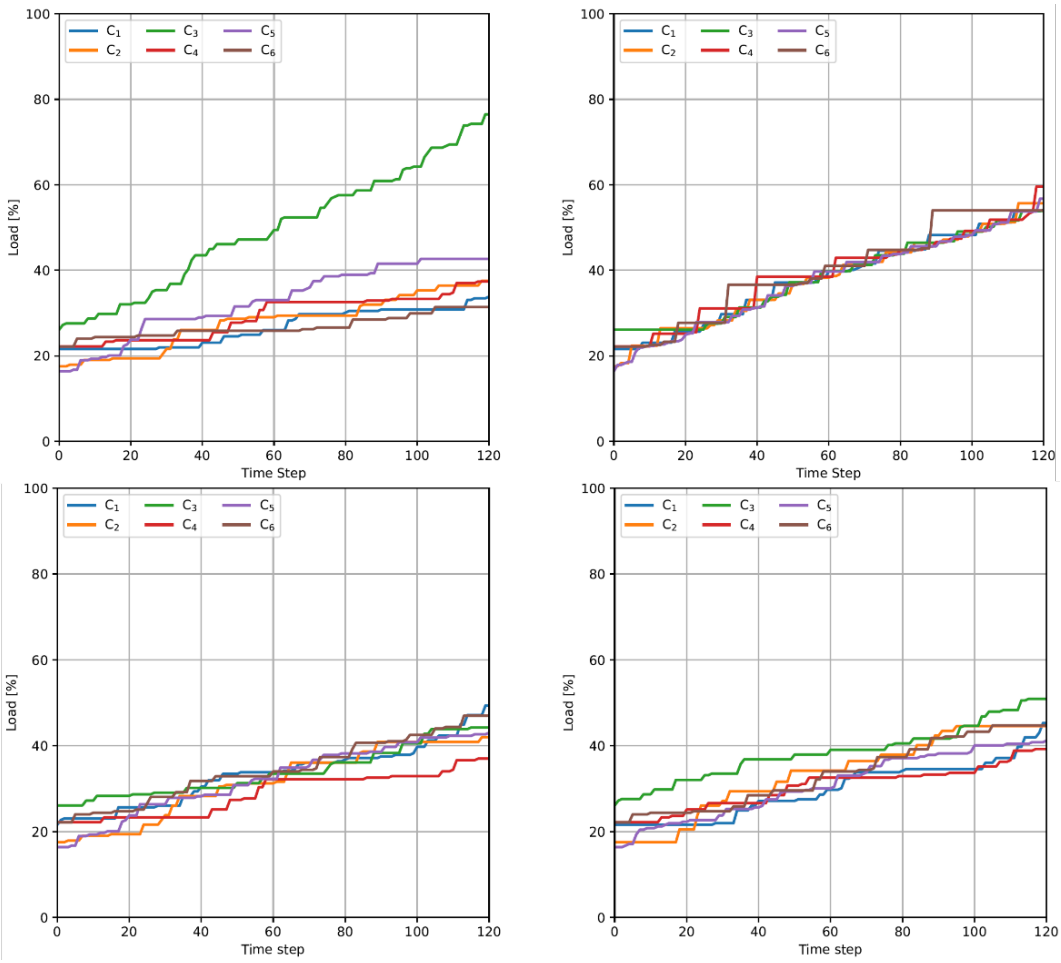

Figure 24: Comparison of cell loads with benchmark #1 (upper left), benchmark #2 (upper right), centralized D3QN (lower left) and FedD3QN (lower right).

The Federated D3QN demonstrates a good trade-off between performance and decentralization. While there's slightly more spread among the cell loads compared to the centralized case, the distribution remains well-behaved and relatively balanced. Unlike Max RSRP, it avoids overloading a single cell (e.g., no cell exceeds 65%), and unlike Least Loaded, it allows for intelligent differentiation to potentially respect RSRP constraints. This decentralized approach achieves effective coordination between agents, despite limited communication, showing its strength in practical 5G scenarios where central control may not be feasible.

Figure 25 provides a graphical representation of the user assignment process for the N users, represented as points colored with the corresponding color of the cell to which they have been assigned.

Table 5: KPI comparison across the four different methodologies

KPI	Max RSRP	Least Loaded	Centr. D3QN	Fed. D3QN
Avg RSRP (dBm)	**-89.8**	-99.1	-90.3	-90.6
Min RSRP (dBm)	**-95.3**	-119	-97.2	-101.7

Max Final Load (%)	76.4	60	49.3	**47.2**
Avg Final Load (%)	43.2	55.6	**43.8**	43.9
StdDev Final Load (%)	15.3	**2.0**	3.9	2.8
Training Score	-13.0	-13.6	-5.6	**-4.4**

The results presented in Table 5 provide a comprehensive evaluation of the performance of the proposed Federated D3QN (FedD3QN) algorithm against three benchmark methods: Max RSRP, Least Loaded, and Centralized D3QN. Each method exhibits distinct strengths and weaknesses across key performance indicators (KPIs), highlighting different trade-offs between signal quality, load balancing, and overall system performance.

Starting with signal quality, the Max RSRP method achieves the highest average RSRP at -89.8 dBm, closely followed by the centralized and federated D3QN variants. This result is expected, as Max RSRP explicitly selects the cell providing the strongest signal, ignoring all other considerations. Least Loaded, instead, led to poor minimum RSRP values (-119 dBm) compared to FedD3QN's (-101.7 dBm), and Max RSRP (-95.3 dBm), suggesting that Least Loaded heavily favors, as expected, load balancing at the cost of poor signal coverage. Interestingly, the FedD3QN performs competitively in average and minimum RSRP, maintaining good signal quality while accounting for other system-level factors.

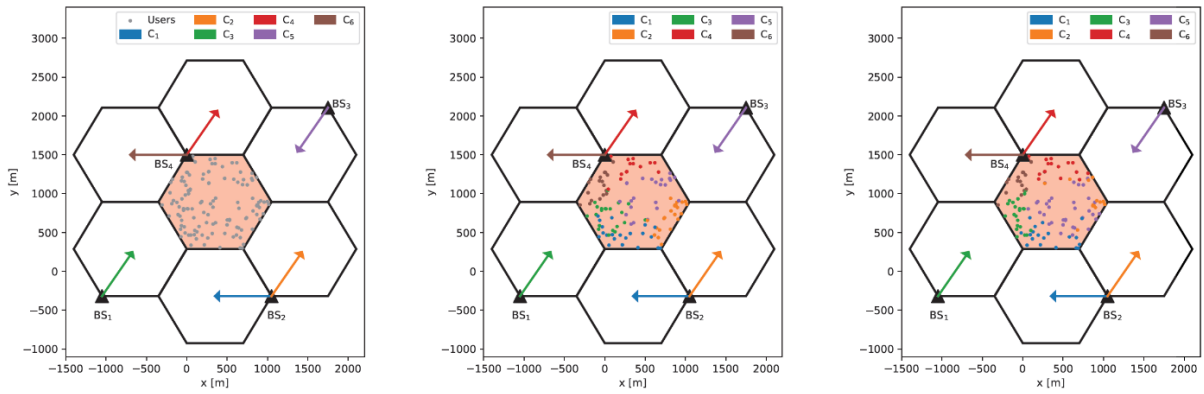

Figure 25: User assignment visual comparison

From a load balancing perspective, the Least Loaded approach excels in minimizing the standard deviation of final load across cells (2%), indicating highly uniform PRB utilization. FedD3QN follows closely (2.8%), showing that it can effectively distribute the load, even without having explicit knowledge of global cell load in a centralized fashion. The Max RSRP strategy, on the other hand, results in a significantly higher load imbalance, with a standard deviation of 15.3% and a high maximum final load of 76.4%. These results underscore its inefficiency in scenarios where fair resource distribution is critical. Both DRL-based methods (Centralized and Federated D3QN) clearly outperform traditional heuristics in this regard, with FedD3QN achieving the lowest maximum load (47.2%), ensuring no single cell becomes a bottleneck.

Looking at the final system-wide reward, defined here as the cumulative score, FedD3QN clearly outperforms all other methods with a value of -4.4, compared to -5.6 for Centralized D3QN, and much worse values for heuristic-based methods (-13 for Max RSRP and -13.6 for Least Loaded). This KPI likely encapsulates the cumulative benefits of well-balanced load, good signal quality, and optimal user-cell matching strategies over time. The superior score achieved by FedD3QN highlights its ability to generalize across diverse user distributions and dynamic network conditions, without relying on centralized knowledge or decision-making.

Overall, the Federated D3QN strikes an effective balance between optimizing RSRP, ensuring fair resource distribution, and maximizing overall network reward. While Max RSRP offers strong signal strength and Least Loaded achieves exceptional load uniformity, both fall short when evaluated holistically. The FedD3QN combines the strengths of these two paradigms while benefiting from the scalability and decentralization inherent in federated learning, making it a compelling solution for user assignment in real-world 5G networks, particularly under cell outage conditions.

We can conclude that, unlike traditional centralized or heuristic-based methods, our approach enables distributed agents to cooperatively learn optimal user allocation policies, preserving scalability and reducing dependency on global network knowledge. We were able to show that Federated D3QN achieves a strong balance between load fairness, signal quality, and overall reward, closely matching the performance of centralized RL while offering greater robustness and practical deployability.

In the next subsection, building on the proposed user assignment strategy, the subsequent step to achieve a complete self-healing process will be detailed. It will be shown how, after having assigned a user to a specific cell, it is possible to modify the antenna power, tilt and azimuth in order increase further the QoS of the users, taking care also about energy saving.

2.6. Deep Reinforcement Learning Control for Self-Healing and Anomaly Compensation in Cellular Networks

Considering the same scenario depicted in Figure 19 and assuming the user reassignment process described in the previous section has been completed, the next objective is to optimize the antenna configuration of the base stations to ensure adequate performance for the reassigned users.

Specifically, the proposed algorithm aims to dynamically adjust both the antenna orientation and the transmission power of each cell that has acquired new users due to the fault, while also maintaining efficient power usage. To achieve these goals, a continuous DRL algorithm is employed. This algorithm operates based on local information available at each base station, as well as the signal feedback received from the users to be served. Such an approach enables adaptive responses to real-time network conditions, optimizing service quality and enhancing overall system resilience.

The implementation of this approach is available here: https://github.com/SGentile7/selfHealingNANCY.

2.6.1. Some notions about Deep Deterministic Policy Gradient

The strategy adopted to realize the anomaly compensation module of the NANCY framework is based on the theoretical concepts of MDPs and RL, already discussed in Section 2.5.1 of this document. In this case, however, a different algorithm has been adopted. The Deep Deterministic Policy Gradient (DDPG) is an off-policy, model-free RL algorithm designed for continuous action spaces. It combines the strengths of Deterministic Policy Gradient (DPG) and Deep Q-Networks (DQN) by using deep neural networks to approximate the policy and value functions. DDPG uses an actor-critic architecture:

- The actor consists of a neural network that learns a deterministic policy $A(s|\theta^A)$, mapping states to a specific action.
- The critic is another neural network approximating the action-value function $C(s,a|\theta^C)$, estimating the expected return for taking action a when in state s.

The training procedure, as happens in many other DRL algorithms, involves the usage of an experience replay to store the transitions, which are sampled randomly to break correlations and improve data efficiency. Moreover, to stabilize learning, two separate *target* action and critic networks A' and C' are

used: they are slowly updated towards the main networks using a soft update mechanism through the parameter τ

$$\theta_{target} \leftarrow \tau\theta_{main} + (1-\tau)\theta_{target}. \quad (36)$$

The critic network is trained to minimize and Bellman error, while the actor is updated using the policy gradient computed via the critic. Since the policy is deterministic, noise injection is needed to carry out the exploration phase during the training stage.

DDPG is particularly useful in environments where actions are continuous-valued (e.g., controlling torques and steering angles), and where the task to solve is high-dimensional, thus requiring function approximation with neural networks. The DDPG pseudocode is provided in Figure 26.

Algorithm 2 DDPG algorithm

Randomly initialize critic network $\mathcal{C}(s,a|\theta^{\mathcal{C}})$ and actor $\mathcal{A}(s|\theta^{\mathcal{A}})$ with weights $\theta^{\mathcal{C}}$ and $\theta^{\mathcal{A}}$
Initialize target network $\mathcal{C}'$ and $\mathcal{A}'$ with $\theta^{\mathcal{C}'} \leftarrow \theta^{\mathcal{C}}$ and $\theta^{\mathcal{A}'} \leftarrow \theta^{\mathcal{A}}$
Initialize replay buffer R
for $e = 1, 2, \ldots, E$ **do**
    Define the standard deviation σ for action exploration
    Receive initial observation state s_1
    for $t = 1, 2, \ldots, T$ **do**
        Compute raw action $u_t = \mathcal{A}(s_t|\theta^{\mathcal{A}})$
        Generate noise $\varepsilon_t \sim \mathcal{N}(\mu, \sigma^2)$ with $\mu = -\sigma\, u_t$
        Apply noise $a_t = u_t + \boldsymbol{\varepsilon}_t$
        Execute a_t and observe r_t and s_{t+1}
        Store transition (s_t, a_t, r_t, s_{t+1}) in R
        Sample N-transitions random minibatch from R
        Set $y_i = r_i + \gamma\mathcal{C}'(s_{i+1}, \mathcal{A}'(s_{i+1}|\theta^{\mathcal{A}'})|\theta^{\mathcal{C}'})$
        Update critic minimizing the loss:
            $L = \frac{1}{N}\sum_i (y_i - \mathcal{C}(s_i, a_i|\theta^{\mathcal{C}}))^2$
        Update actor policy using the policy gradient:
$\nabla_{\theta^{\mathcal{A}}} J \approx \frac{1}{N}\sum_i \nabla_a \mathcal{C}(s, a|\theta^{\mathcal{C}})|_{s_i, \mathcal{A}(s_i)} \nabla_{\theta^{\mathcal{A}}} \mathcal{A}(s, |\theta^{\mathcal{A}})|_{s_i}$
        Update target networks, with $\tau \ll 1$:
            $\theta^{\mathcal{C}'} \leftarrow \tau\theta^{\mathcal{C}} + (1-\tau)\theta^{\mathcal{C}'}$
            $\theta^{\mathcal{A}'} \leftarrow \tau\theta^{\mathcal{A}} + (1-\tau)\theta^{\mathcal{A}'}$
    end for
end for

Figure 26: Pseudocode of the DDPG Algorithm

2.6.2. System Model

According to the 3GPP technical report (3GPP TR 36.814), the directional gain of an antenna element can be modeled by decomposing it into azimuth and elevation components, as shown in Figure 27 and Figure 28, each depending on the angular deviation from the antenna boresight. The total directional gain perceived by a user can be expressed as the sum of the horizontal and vertical gain components:

$$G_{tot} = G_h + G_v. \quad (37)$$

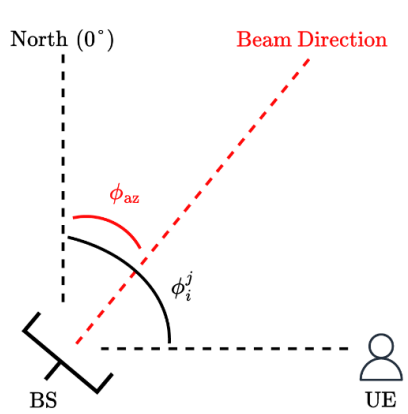

Figure 27: Antenna azimuth configuration

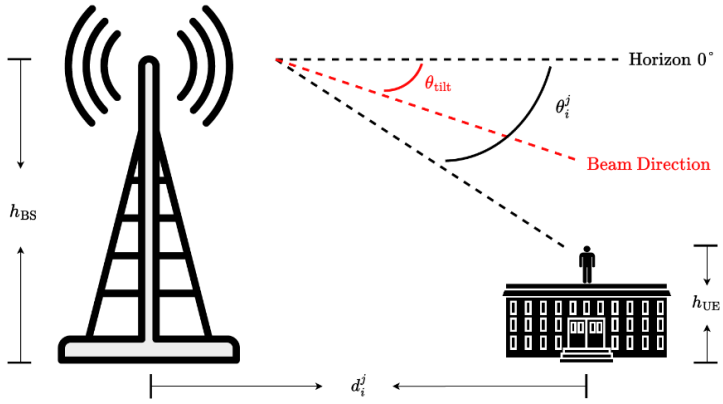

Figure 28: Antenna tilt configuration

The horizontal radiation pattern, also referred to as the azimuth pattern, is defined by the horizontal antenna gain, denoted as G_h. This gain primarily depends on the antenna's azimuth angle ϕ_{az} and the planar angle ϕ_i^j, which represents the relative angle between base station j and user terminal i, measured with respect to the geographic North.

The azimuth angle is allowed to vary around a nominal orientation ϕ_0, which denotes the central alignment of the antenna relative to North. Accordingly, the actual azimuth angle is selected within a predefined interval:

$$\phi_{az} \in [\phi_0 + \Delta\phi_{min}, \phi_0 + \Delta\phi_{max}], \tag{38}$$

where $\Delta\phi_{min} < 0 < \Delta\phi_{max}$ define the allowable angular deviation bounds from the nominal orientation.

The planar angle, on the other hand, can be computed based on the geographic coordinates of the BS and the user as follows:

$$\phi_i^j = \frac{\pi}{2} - \arctan\left(\frac{y_{user} - y_{bs}}{x_{user} - x_{bs}}\right) \tag{39}$$

The horizontal gain is then modeled as:

$$G_h(\phi_{az}) = G_h^{max} \min\left(\left(\frac{\phi_i^j - \phi_{az}}{\phi_{3dB,h}}\right)^2, A_m\right) \tag{40}$$

where G_h^{max} is the maximum antenna gain in the azimuth plane, $\phi_{3dB,h}$ represents the 3 dB beamwidth in the azimuth plane and A_{mh} is the maximum attenuation allowed in the horizontal domain.

The vertical radiation pattern characterizes the variation of antenna gain as a function of the elevation angle θ_i^j, which represents the inclination between the antenna and the position of the target user. The elevation gain is modeled analogously to the azimuth gain pattern and considers both electrical and mechanical tilt adjustments. Specifically, the elevation gain is expressed as:

$$G_v(\theta_{tilt}) = \max\left(12\left(\frac{\theta_i^j - \theta_{tilt}}{\theta_{3dB,v}}\right)^2, A_{mv}\right) \quad (41)$$

Where θ_{tilt} represents the downtilt angle applied to the antenna, $\theta_{3dB,v}$ is the 3 dB beamwidth in the elevation plane and A_{mv} is the maximum attenuation allowed in the vertical plane and θ_i^j is the elevation angle between the base station and the user terminal, computed as:

$$\theta_i^j = \arctan\left(\frac{z_{bs} - z_{user}}{d_{ij}^x}\right) \quad (42)$$

with d_{ij}^x that denotes the 2D horizontal distance between the user terminal and the base station.

This total directional gain has a direct impact on the received signal strength at the user terminal, which is typically quantified by the Reference Signal Received Power (RSRP). RSRP represents the long-term average of the received signal power, excluding the effects of interference and noise. This makes it particularly useful for making mobility and connectivity decisions within the network. Measured in dBm, RSRP is often reported by the user to aid the network in effective radio resource management. The RSRP for a user, relative to the serving base station, is mathematically expressed as:

$$RSRP = P + G_{tot} + G_{user} - L \quad (43)$$

where P is the transmit power allocated to the reference signal [dBm], G_{user} is the receive antenna gain of the user equipment, L is the total propagation loss between the base station and the user, which is calculated as:

$$L = I + 37.6\log_{10} R + P_L \quad (44)$$

where R is the distance in kilometers (km), I is a constant value related to the frequency of the radio signal and P_L is another constant term that accounts for additional losses due to the penetration of the signal through materials.

The cellular network comprises a collection of base stations, denoted by $B = \{1, \dots, m\}$. Each base station $j \in B$ is assigned to serve a group of users, represented by the set $N_j = \{1, \dots, n_j\} \subseteq N$, where N denotes the complete set of users within the network.

Each base station is characterized as a dynamic system, defined by the vector state:

$$x(k) = \begin{bmatrix} P(k) \\ \phi_{az}(k) \\ \theta_{tilt}(k) \end{bmatrix}, \quad (45)$$

with a control input vector that encodes the incremental actions applied to each state:

$$u(k) = \begin{bmatrix} \Delta P(k) \\ \Delta\phi_{az}(k) \\ \Delta\theta_{tilt}(k) \end{bmatrix}. \quad (46)$$

The base station dynamics evolve according to a discrete-time integrator model:

$$P(k+1) = P(k) + T_s \Delta P(k) \quad (47)$$
$$\phi_{az}(k+1) = \phi_{az}(k) + T_s \Delta \phi_{az}(k) \quad (48)$$
$$\theta_{tilt}(k+1) = \theta_{tilt}(k) + T_s \Delta \theta_{tilt}(k) \quad (49)$$

where $T_s > 0$ denotes the sampling time of the control loop.

On the other hand, the dynamics of the signal received by each user i evolves as:

$$RSRP_i(k+1) = P(k) + T_s \Delta P(k) + G_h^i\big(\phi_{az}(k), \Delta\phi_{az}(k)\big) + G_v^i(\theta_{tilt}, \Delta\theta_{tilt}) + c_i \quad (50)$$

Where the nonlinear functions $G_h^i(\cdot)$ and $G_v^i(\cdot)$ denote the horizontal and vertical antenna gain contributions, while $c_i \in \mathbb{R}$ models static propagation effects given by the device gain and path loss.

Based on the defined model, we can now formally specify the state space, action space, and reward function required by the DDPG algorithm. The state vector, which encapsulates the key characteristics of the network environment and antenna configuration at a specific time step, is defined as:

$$s = \left[P, \phi_{az}, \theta_{tilt}, RSRP_{min}, RSRP_{avg}, \Delta RSRP_{min}, FS\right] \quad (51)$$

In addition to the previously defined parameters, the new variables $RSRP_{min}$ and $RSRP_{avg}$ are the minimum and average values, respectively, among all users connected to the base station

$$RSRP_{min}(k) = \min_i(RSRP_i(k)) \quad (52)$$

$$RSRP_{avg}(k) = \frac{1}{N}\sum_{i}^{N} RSRP_i(k) \quad (53)$$

$\Delta RSRP_{min}$ is the variation in the minimum RSRP and FS is a binary flag identifying if the fault zone lies to the right or left of the antenna's nominal azimuth.

The agent interacts with the environment by continuously adjusting the antenna configuration through an action vector defined as:

$$a = [\Delta P, \Delta\phi_{az}, \Delta\theta_{tilt}] \quad (54)$$

where each element corresponds to an incremental variation applied to a specific control variable. These increments are constrained within predefined bounds reflecting the physical and regulatory limitations of the system, namely:

$$\Delta P \in \left[a_P^{min}, a_P^{max}\right] \quad (55)$$

$$\Delta\phi_{az} \in \left[a_{\phi_{az}}^{min}, a_{\phi_{az}}^{max}\right] \quad (56)$$

$$\Delta\theta_{tilt} \in \left[a_{\theta_{tilt}}^{min}, a_{\theta_{tilt}}^{max}\right] \quad (57)$$

The reward function serves as the objective function of an underlying optimization problem, crafted to balance two potentially conflicting goals: ensuring user quality of service and minimizing the energy consumption of the base station. The first objective is addressed by maximizing $RSRP_{min}$, which reflects the weakest signal level experienced among all connected users. This ensures that service

quality is improved for the most disadvantaged user, thereby enhancing the overall experience for the entire group of newly reassigned users. The second objective targets energy efficiency, by minimizing the base station's transmit power. Given the inherent trade-off between these two goals, the task can be framed as a multi-objective optimization problem, where the aim is to reach a compromise that optimally balances both service quality and power consumption. In reinforcement learning terms, this objective corresponds to a reward function of the form:

$$r = \alpha RSRP_{min} - \beta P \tag{58}$$

where α and β are weighting coefficients that regulate the relative importance of signal quality and energy efficiency; by adjusting their values, the learning agent can be guided to prioritize either enhanced service quality or reduced power consumption, depending on the system's operational objectives.

2.6.3. Simulation Setup

The actor network is implemented as a feed-forward deep neural network composed of two hidden layers with 64 and 32 units, respectively, followed by a final linear output layer of size 3, corresponding to the three action dimensions. The critic network adopts a similar architecture, but with larger hidden layers of 128 and 64 units, culminating in a single output neuron that estimates the Q-value for a given state–action pair. Both networks are trained using the Adam optimizer. The learning rate for the actor is set to 1×10^{-4}, while the critic uses a learning rate of 5×10^{-4} . Training samples are drawn from a replay buffer with a maximum capacity of 10,240 transitions. At each training step, a minibatch of 1,024 transitions is sampled uniformly at random from the buffer. To improve training stability, target network parameters are updated using a soft update mechanism with a smoothing factor of $\tau = 5 \times 10^{-3}$. The exploration noise standard deviation during training follows an exponentially decaying schedule, defined as

$$\sigma = \sigma_0 \exp\left(-\frac{e}{\gamma E}\right) \tag{59}$$

where $\sigma_0 = 0.3$, $\gamma = 0.6$, e denotes the actual training episode, and E represents the total number of episodes.

The training was conducted over a total of $E = 1500$ episodes, with each one initialized by fixing the base station at the origin of the coordinate system. The positions of the already-connected users are generated randomly within a predefined range of distances, centred around the nominal orientation of the antenna. Conversely, the users requiring compensation are placed at greater distances and with significant deviations from the antenna's main beam direction, alternating between positions on the left and right sides of the base station's azimuth. This setup creates a realistic scenario in which signal degradation is more pronounced for newly assigned users, encouraging the learning agent to optimize configurations that improve service for the disadvantaged group.

Based on this setup, the results are reported starting from the network scenario generated by the algorithm defined in the previous section. As shown in Figure 29, the objective is now to adjust the antenna configuration to improve the quality of service for the reassigned users. In the following simulation, the case of base station BS2 with respect to cell C4 is presented as an illustrative example. It should be noted that this choice was made solely for the sake of simplicity, and the proposed approach can be readily applied to the other cases as well.

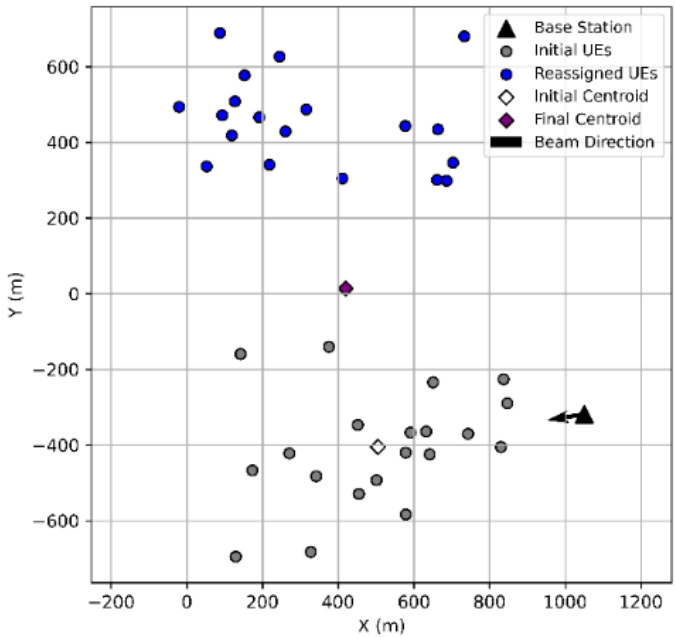

Figure 29: Initial Scenario after user assignment

The scenario under consideration is depicted in Figure 25. Under normal operating conditions, C1 serves a total of 20 users. However, following the failure of an adjacent cell and the subsequent user reassignment carried out by the federated algorithm, BS2 is required to accommodate an additional 19 users who were previously connected to the failed cell, bringing the total number of users served to 39.

In Figure 29, grey users represent those who were already connected to BS2, while blue users are those who were reassigned to BS2 by the allocation algorithm. BS2, represented as a black triangle, starts with a specific configuration deemed optimal for ensuring the QoS for the originally connected users. The antenna is initially oriented toward the centroid of this user group, calculated as the average position of the users and shown as a white rhombus in the figure. This centroid serves as a reference user, as it captures the average behavior and spatial distribution of the group.

It is important to note that the centroid is shown for illustrative purposes only and is not available to the algorithm during execution.

2.6.4. Results

The results obtained from the algorithm are presented in Figure 30, which illustrate the evolution of the transmission power, the azimuth angle, and the tilt angle of the base station antenna, respectively. The transmission power converges to 2.41 W in approximately 1 s, reflecting the algorithm's objective to reduce energy consumption, as specified by the reward function. The tilt angle converges to 0.44° in roughly 1 s, while the azimuth angle converges to 141.29° over about 5 s. This longer convergence time is reasonable given its larger range of adjustments, since the initial configuration was highly specialized for the originally connected users.

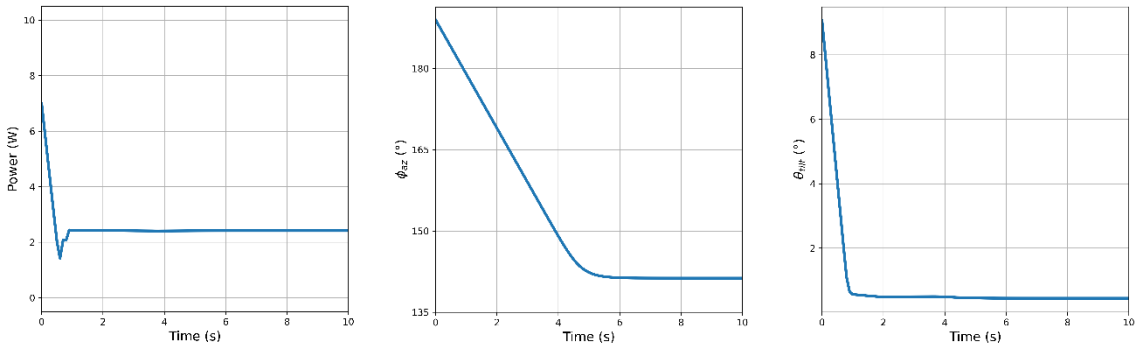

Figure 30: Evolution of the Base Station State

These adjustments are reflected in the RSRP values received by the users, as shown in Figure 31. The average RSRP value decreases and converges to –102.94 dBm, while the minimum RSRP increases and converges to –110.16 dBm. While the reduction in average RSRP may seem like a negative outcome, it is a direct consequence of the optimization strategy, which simultaneously pursues two main objectives: prioritizing the quality of service for reassigned users, who belong to a higher-priority category, and reducing energy consumption by lowering transmission power. The algorithm effectively shifts resources to better serve the reassigned users, even at the expense of slightly degraded performance for users originally connected to the base station. This trade-off is further amplified by the energy-saving objective, which leads to a reduction of nearly 5 W in transmission power, an adjustment that inevitably impacts overall signal quality.

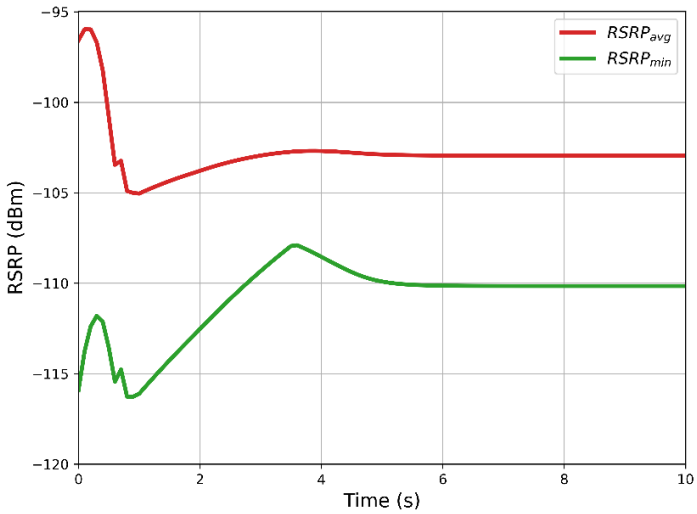

Figure 31: Minimum RSRP and Average RSRP evolution

The preference given to the reassigned user group is further highlighted in Figure 32, which shows the evolution of the average RSRP for both user groups throughout the simulation. As illustrated, the average RSRP of the originally connected users decreases and converges to –102.72 dBm, while the average RSRP of the reassigned users increases, converging to –103.18 dBm. This outcome is particularly significant considering that the reassigned users are located at a considerably greater distance from the base station and would, under normal conditions, belong to a different cell. Despite this, the optimization process succeeds in raising their signal quality to a level comparable to that of the original users, underscoring the system's prioritization strategy.

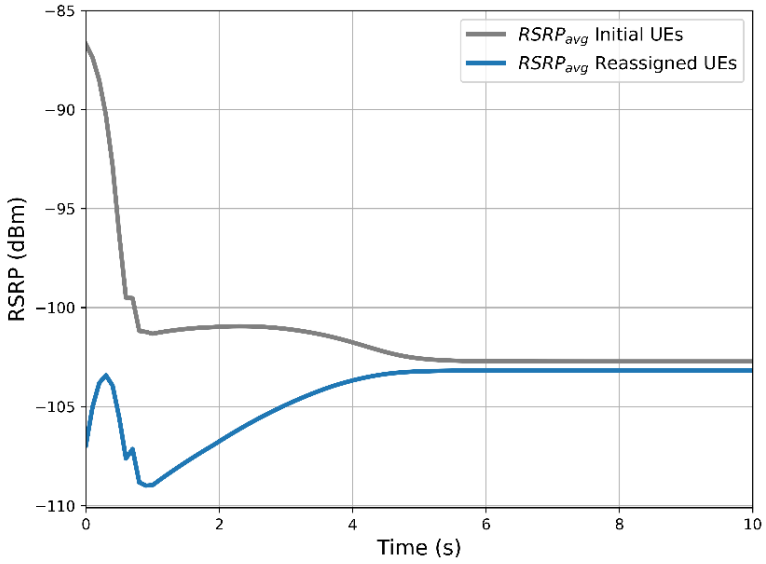

Figure 32: Comparison of the average RSRP between the two groups

Finally, Figure 33 presents the scenario after the compensation, illustrating the antenna's orientation. Since the representation is two-dimensional, only the azimuth angle can be visually inferred. Nevertheless, this visualization is useful for observing that the antenna is directed around the purple rhombus, which represents the centroid of all users currently connected to the base station. This highlights how the antenna orientation resulting from the optimization process effectively aligns to target the group's average user, thereby ensuring balanced service quality for both originally connected and reassigned users.

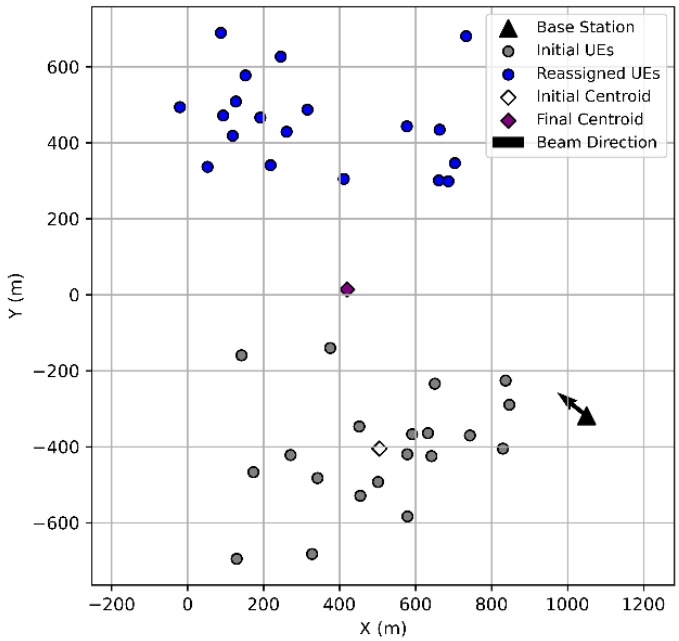

Figure 33: Final Scenario

These results demonstrate the effectiveness of the proposed DDPG-based control strategy in dynamically adapting the antenna configuration of the base station in response to user reassignment events. By balancing service quality for disadvantaged users and energy efficiency objectives, the algorithm successfully reconfigures transmission parameters to restore coverage, minimize disparities in signal strength, and preserve network performance in the presence of infrastructure faults.

2.7. Adversarial Dynamic Healing based on Wardrop Equilibrium

Self-healing and load balancing are fundamental tasks in network control, effectively amounting to routing traffic over a set of providers (viewed as distinct paths) whose performance is captured by a latency function, that not necessarily represents the network round-trip-time, but can be an arbitrary function to be equalized. In settings where each user routes selfishly, seeking only to minimize its own experienced delay without any cooperation, the network naturally moves toward special equilibrium states. One such state, known in mean-field game theory as the Wardrop equilibrium (equivalent to a Nash equilibrium for infinitely many participants), occurs when all active routes exhibit the same delay, so no single user can reduce its latency by changing paths. Here, we examine the case of load balancing and self-healing for a mobile network providing IoT services. Our proposed control law therefore works to equalize latency across every provider, by considering possible performance drops due to anomalies (e.g., due to malfunctioning or attacks). This leads to a generalized equilibrium notion—often called the Beckmann user equilibrium—that extends Wardrop's idea to networks.

The implementation of this approach is available here: https://github.com/trunk96/wardrop-nancy.

2.7.1. Preliminaries on Wardrop equilibrium and Lyapunov stability

Consider a collection of service providers $\mathcal{P}$ and a set $\mathcal{I}$ of $|\mathcal{I}|$ commodities. Each commodity $i \in \mathcal{I}$ generates work at a rate $\lambda^i > 0$, which must be processed by the subset of providers $\mathcal{P}^i \subseteq \mathcal{P}$ that support commodity $i \in \mathcal{I}$. The total workload is $\lambda = \sum_{i\in\mathcal{I}|p\in\mathcal{P}^i} \lambda^i$, and we view this demand as being handled by an infinite population of agents, each choosing its own allocation strategy [105].

Let x_p^{ki} denote the fraction of commodity i's load that agent k assigns to provider p. Aggregating across agents gives

$$x_p^i = \sum_{k\in\mathcal{K}} x_p^{ki} \tag{60}$$

and summing over commodities yields

$$x_p = \sum_{i\in\mathcal{I}} x_p^i \tag{61}$$

the total share of all workloads directed to provider $p \in \mathcal{P}$. We bundle these variables into the flow vector $x = \left[x_p^{ki}\right]_{p\in\mathcal{P},i\in\mathcal{I},k\in\mathcal{K}}$.

Definition 1. The corresponding feasible state space (as the set of feasible flow vectors) with a load demand vector $\lambda = \left[\lambda^i\right]_{i\in\mathcal{I}}$ is defined as:

$$\mathcal{X} \coloneqq \left\{ x = \left[x_p^{ki}\right]_{p\in\mathcal{P},i\in\mathcal{I},k\in\mathcal{K}} \middle| x_p^{ki} \geq 0, \forall p \in \mathcal{P}^i, \sum_{p\in\mathcal{P}^i,k\in\mathcal{K}} x_p^{ki} = \lambda^i, \forall i \in \mathcal{I} \right\} \tag{62}$$

Moreover, the maximum load of a provider $p \in \mathcal{P}$ is defined as $\lambda_p \coloneqq \sum_{i\in\mathcal{I}|p\in\mathcal{P}^i} \lambda^i$.

Effective load balancing hinges on a suitable performance metric that reflects how well the system distributes work. In the Wardrop setting, this role is played by *latency functions*, which quantify system performance—often in terms such as average response time under a given load.

Following standard practice [106], [107], we require each latency function (one per provider $p \in \mathcal{P}$) to be nonnegative and strictly increasing with respect to that provider's total allocated load x_p. Because

these functions vary widely depending on the specific application and performance goals, we do not assume any analytical form. Instead, we restrict our attention to the general class of latency functions that satisfy the conditions below.

Assumption 1. The latency functions $l_p: [0, \lambda] \to \mathbb{R}_{\geq 0}$ are Lipschitz continuous for all $p \in \mathcal{P}$, strictly increasing over $[0, \lambda]$ and with Lipschitz constant K_p.

A stable flow vector is achieved at what's known as a Wardrop equilibrium: a state where no individual commodity can, acting alone, improve its own performance by shifting its load to a different provider.

Definition 2. [107] For a fixed total load λ, we call a flow vector $x \in \mathcal{X}$ a Wardrop equilibrium if, for every commodity i and any two providers $p, m \in \mathcal{P}^i$, the following holds: whenever $x_p^i > 0$, the latency at p does not exceed that at m, i.e.,

$$l_p(x_p) \leq l_m(x_m) \tag{63}$$

This condition must be satisfied for all $i \in \mathcal{I}$, ensuring no single commodity can reduce its own latency by shifting load unilaterally.

We call a provider $p \in \mathcal{P}^i$ as *loaded* for commodity i whenever $x_p^i > 0$. In a Wardrop equilibrium—following the original definition in [108]—all loaded providers for the same commodity exhibit identical latency. The collection of all such equilibria, $\mathcal{W}$, can be described as

$$\mathcal{W} \coloneqq \{x \in \mathcal{X} | l_p(x_p) - l_m(x_m) \leq 0, \forall p, m \in \mathcal{P}^i \, s.t. \, x_p^i > 0, \forall i \in \mathcal{I}\} \tag{64}$$

Later on, we make use of the Beckmann potential [109], a standard tool in this context, defined by

$$\Phi(x) \coloneqq \sum_{p \in \mathcal{P}} \int_0^{x_p} l_p(\xi) d\xi \tag{65}$$

which features the following key properties:

Property 1. Under Assumption 1, Beckmann potential of (65) is continuous and the following hold:

1. There is exactly one feasible flow vector $w \in \mathcal{X}$ that achieves the minimum of the Beckmann potential $\Phi(x), \; x \in \mathcal{X}$.
2. The minimum value Φ is positive and unique, and it is denoted as $\Phi_{min} = \Phi(w) > 0$.
3. This minimizer w coincides with the Wardrop equilibrium, so the equilibrium set reduces to the single point $\mathcal{W} = \{w\}$.

Below, we reformulate the load-balancing challenge as that of guiding a nonlinear dynamical system—with state $x \in \mathcal{X}$—into an invariant region surrounding the equilibrium w. Convergence is then established via LaSalle's invariance principle, which in turn relies on the construction of appropriate Lyapunov-like functions.

Definition 3. A candidate function $\mathcal{L}: \mathcal{X} \to \mathbb{R}$ called a weak Lyapunov function for the dynamics $\dot{x} = f(x)$ if it meets all of the following:

1. $\mathcal{L}$ is continuous on $\mathcal{X}$, differentiable everywhere (i.e., C^1), and $\mathcal{L}(x) \to \infty$ as $|x| \to \infty$.
2. For every $x \in \mathcal{X}$, $\mathcal{L}(x) \geq 0$.
3. Along any solution $x(t)$, the time derivative $\dot{\mathcal{L}}(x(t)) \coloneqq \nabla \mathcal{L}(x) f(x) \leq 0$.

Theorem 1. Let $\dot{x} = f(x)$ define a nonlinear dynamical system on the state space $\mathcal{X}$. Suppose $\mathcal{L} : \mathcal{X} \to \mathbb{R}$ is a weak Lyapunov function and define $\mathcal{D} = \{x \in \mathcal{X} \mid \dot{\mathcal{L}}(x) = 0\}$. If there exists a compact set $\mathcal{M} \subset$

$\mathcal{D}$ that is the largest positively invariant set, then every solution starting from any $x(0) \in \mathcal{X}$ will converge to $\mathcal{M}$ as $t \to \infty$. This result follows directly from LaSalle's invariance principle.

2.7.2. IoT requests Allocation in B5G Networks as a Dynamic Load-balancing Problem

In Beyond-5G systems, we anticipate an explosion of IoT endpoints, all relying on the network's edge computing resources for tasks like data aggregation, pre-processing, and real-time control. To illustrate this, consider the setup in Figure 34: multiple IoT devices send different types of tasks (indicated by various arrow colors) over a B5G radio link to a pool of edge servers. It is essential that each device distributes its workload across these servers efficiently; overloading a single node can lead to long processing delays or even dropped messages if the server's capacity is exceeded. While one could place a central load balancer between devices and servers, doing so introduces a single point of failure, something next-generation networks aim to avoid since any disruption (from faults, attacks, or overload) could severely degrade IoT service availability.

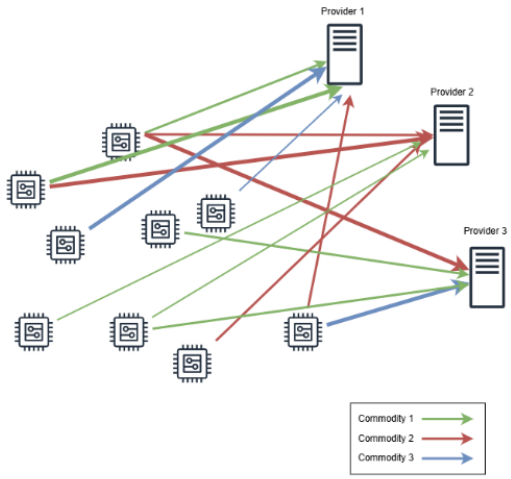

Figure 34: IoT network scenario for the proposed approach. Different line colors indicate different commodities, while thicker lines indicate higher request rate

Each IoT node offers distinct services—meaning their sensing capabilities aren't identical—and they may generate requests at varying rates (for instance, due to differences in sensing resolution). At the same time, edge servers differ in the set of service types they support, so not every server can process every kind of IoT request.

2.7.3. Load Balancing Algorithm, Stability and Convergence to a Wardrop Equilibrium

In this section, we derive a decentralized feedback rule that adjusts allocations in real time to spread the load across providers. Beginning from any admissible starting point $x(0) \in \mathcal{X}$, the proposed control law guarantees that the system will reach, within a finite time, a prescribed neighborhood of equilibrium—denoted by $\mathcal{W}_{\delta,\varepsilon}$—whose precise characterization appears below.

Definition 4. For given parameters $\delta \in (0,1)$ and $\varepsilon > 0$ we define the (δ,ε)-Wardrop equilibrium region as the set of all flow vectors $x \in \mathcal{X}$:

$$\mathcal{W}_{\delta,\varepsilon} \coloneqq \left\{x \in \mathcal{X} \middle| l_p(x_p) - l_m(x_m) \leq \varepsilon, \forall p,m \in \mathcal{P}^i \; s.t. \; x_p^i > \delta\lambda^i, \forall k \in \mathcal{K}, \forall i \in \mathcal{I}\right\} \supset \{w\} \quad (66)$$

In this context, the parameter ε represents the maximum allowable deviation between latency values, whereas δ defines the minimum share of commodity i's load that must be allocated to provider p for it to be considered "loaded" by that commodity—such providers are termed δ-loaded. Hence, at any

(δ, ε)-Wardrop equilibrium, all providers $p \in \mathcal{P}^i$ satisfying $x_p^{ki} > \delta\lambda^i$ for a given agent k and commodity i, will have latencies that differ by no more than ε. Because the defining inequalities in (66) depend continuously on x, the set $\mathcal{W}_{\delta,\varepsilon}$ is both closed and bounded within $\mathcal{X}$.

Each agent $k \in \mathcal{K}$, handling commodity $i \in \mathcal{I}$, exercises control by transferring portions of its traffic from one provider p to another m, both drawn from P^i. Denote by $r_{pm}^{ki}(t)$ the instantaneous rate at which agent k shifts load from p to m. The resulting time evolution of x_p^{ki} is then

$$\dot{x}_p^{ki}(t) = \sum_{m \in \mathcal{P}^i} \left[r_{mp}^{ki}(t) - r_{pm}^{ki}(t) \right] \tag{67}$$

Summing over all agents gives the dynamics for the total load of commodity i on provider p:

$$\dot{x}_p^{i}(t) = \sum_{k \in \mathcal{K}} \dot{x}_p^{ki} = \sum_{m \in \mathcal{P}^i} \sum_{k \in \mathcal{K}} \left[r_{mp}^{ki}(t) - r_{pm}^{ki}(t) \right] \tag{68}$$

for all $p \in \mathcal{P}^i$ and $i \in \mathcal{I}$.

The controller we adopt extends the dynamic control from [110], where each agent's migration rate from provider p to provider m is given by

$$r_{pm}^{ki}(t) = \sigma x_p^{ki}(t) \mu_{pm}^{ki}(t) \tag{69}$$

with $\sigma > 0$ as a tuning parameter and $\mu_{pm}^{ki}(t)$ indicating whether (when equal to 1) or not (when 0) a portion of commodity i is shifted from p to m. Our choice for the policy $\mu_{pm}^{ki}(t)$ is driven by the observed latencies

$$\mu_{pm}^{ki}(t) = \begin{cases} 1 & if\ l_p(x_p) - l_m(x_m) > \varepsilon\ and\ x_p^{ki}(t) > \delta\lambda^i \\ 0 & otherwise \end{cases} \tag{70}$$

Defining the total load of agent k on provider p as $x_p^k(t) = \sum_{i \in \mathcal{I}} x_p^{ki}(t)$ and its aggregate migration rate as $r_{pm}^k(t) = \sum_{i \in \mathcal{I}} r_{pm}^{ki}(t)$, we can write the overall flow dynamics across all providers as

$$\begin{aligned} \dot{x}(t) = \sum_{p \in \mathcal{P}} \dot{x}_p(t) = \sum_{p \in \mathcal{P}} \sum_{k \in \mathcal{K}} \dot{x_p^k}(t) = \sum_{p \in \mathcal{P}} \sum_{m \in \mathcal{P}} \sum_{k \in \mathcal{K}} \left[r_{mp}^k(t) - r_{pm}^k(t) \right] = \\ = \sum_{p,m,k} x_m^k(t) \sigma \mu_{mp}^k(t) - x_p^k(t) \sigma \mu_{pm}^k(t) \end{aligned} \tag{71}$$

We begin demonstrating the combined dynamics given by these equations are well-posed.

Lemma 1. Under Assumption 1, $\mathcal{X}$ is a positively invariant set for the nonlinear system (71).

Proof. We now verify that, for any initial conditions $x(0) \in \mathcal{X}$, the trajectory $x(t)$ remains in $\mathcal{X}$ as $t \to \infty$. This requires showing (i) the conservation of the total commodities flow $\sum_{p \in \mathcal{P}} x_p^i(t) = \lambda^i\ \forall i \in \mathcal{I}$; (ii) the nonnegativity of the individual allocations $x_p^{ki}(t) \geq 0$ for all $k \in \mathcal{K}$, $i \in \mathcal{I}$, and $p \in \mathcal{P}$.

i) From (69) we obtain

$$\sum_{p \in \mathcal{P}^i} \dot{x}_p^i(t) = \sum_{p \in \mathcal{P}^i} \sum_{m \in P^i} \sum_{k \in \mathcal{K}} \left[r_{mp}^{kı}(t) - r_{pm}^{kı}(t) \right] = \sum_{k \in \mathcal{K}} \left(\sum_{p,m \in \mathcal{P}^i} r_{mp}^{ki}(t) - \sum_{p,m \in \mathcal{P}^i} r_{pm}^{ki}(t) \right) = 0 \tag{72}$$

which immediately implies $\sum_{p \in \mathcal{P}} x_p^i(t) = \lambda^i$ for all $t \geq 0$ and $i \in \mathcal{I}$

ii) From (67), we have

$$\dot{x}_p^{ki}(t) = \sum_{m\in\mathcal{P}^i}\left[r_{mp}^{ki}(t) - r_{pm}^{ki}(t)\right] \geq -\sum_{m\in\mathcal{P}^i} r_{pm}^{ki}(t) \tag{73}$$

The right-hand side can only be negative if $r_{pm}^{ki}(t) > 0$ for some m, which by (69) requires $x_p^{ki}(t) > 0$. Since $x_p^{ki}(0) \geq 0$, this argument by induction ensures $x_p^{ki}(t) \geq 0$ for all $t \geq 0$. ■

To establish convergence of the controlled dynamics (71), we invoke LaSalle's invariance principle. We introduce a candidate Lyapunov function based on the Beckmann potential (65):

$$\mathcal{L}\big(x(t)\big) \coloneqq \Phi\big(x(t)\big) - \Phi_w \tag{74}$$

Lemma 2. Lyapunov candidate $\mathcal{L}(x)$ defined above is a weak Lyapunov function for the system $\dot{x} = f(x)$ given by (71), with any $\varepsilon > 0$.

Proof. By Assumption 1, $\Phi(x)$ is continuous, continuously differentiable, and radially unbounded, so $\mathcal{L}$ shares these properties. Property 1 of the Beckmann potential guarantees $\mathcal{L}(x) \geq 0$, with equality only at the unique minimizer w, establishing positive definitiveness around w. It remains only to check that $\dot{\mathcal{L}}\big(x(t)\big) \leq 0$ with respect to the set $W_{(\delta,\varepsilon)}$ for all trajectories. Using $\dot{\mathcal{L}} = \dot{\Phi}$ and the definition of Φ in (65), we compute

$$\begin{aligned}
\dot{\mathcal{L}}(t) = \dot{\Phi}(t) &= \sum_{p\in\mathcal{P}} \dot{x}_p(t) l_p\left(x_p(t)\right) = \sum_{p\in\mathcal{P}}\sum_{k\in\mathcal{K}} \dot{x}_p^k(t) l_p\left(x_p(t)\right) = \\
&= \sum_{p\in\mathcal{P}}\sum_{m\in\mathcal{P}}\sum_{k\in\mathcal{K}} \left(r_{mp}^k(t) - r_{pm}^k(t)\right) l_p\big(x(t)\big) = \\
&= \sum_{p\in\mathcal{P}}\sum_{m\in\mathcal{P}}\sum_{k\in\mathcal{K}} r_{mp}^k(t) l_p\left(x_p(t)\right) - \sum_{p\in\mathcal{P}}\sum_{m\in\mathcal{P}}\sum_{k\in\mathcal{K}} r_{pm}^k(t) l_m\big(x_m(t)\big) = \\
&= \sum_{m\in\mathcal{P}}\sum_{p\in\mathcal{P}}\sum_{k\in\mathcal{K}} r_{pm}^k(t) l_m\big(x_m(t)\big) - \sum_{m\in\mathcal{P}}\sum_{p\in\mathcal{P}}\sum_{k\in\mathcal{K}} r_{mp}^k(t) l_p\left(x_p(t)\right) = \\
&= \sum_{i\in\mathcal{I}}\sum_{p\in\mathcal{P}}\sum_{m\in\mathcal{P}}\sum_{k\in\mathcal{K}} r_{pm}^{ki}(t)\Big(l_m\big(x_m(t)\big) - l_p\left(x_p(t)\right)\Big)
\end{aligned} \tag{75}$$

In (69), $x_p^{ki}(t) \geq 0$ as from Lemma 1, and σ and $\mu_{pm}^{ki}(t)$ are nonnegative. Moreover, if $l_p(x_p) - l_m(x_m) \leq \varepsilon$, then by migration policy $\mu_{pm}^{ki}(t) = 0$, so that term vanishes, while if $l_p(x_p) - l_m(x_m) > \varepsilon$, then $\mu_{pm}^{ki} = 1$ makes $r_{pm}^{ki}(t) \geq 0$ and $l_m - l_p < 0$, so the product is nonpositive. Hence, every summand of (75) is ≤ 0, and $\dot{\mathcal{L}} \leq 0$, proving the lemma. ■

Theorem 2. All trajectories of the controlled system reach the (δ, ε)-approximate equilibrium set $\mathcal{W}_{\delta,\varepsilon}$ in finite time

Proof. With Lemma 2 in hand, LaSalle's principle applies once we show that (i) $\mathcal{W}_{\delta.\varepsilon}$ is the largest positively invariant set where $\dot{\mathcal{L}} = 0$; (ii) the trajectories enter $\mathcal{W}_{\delta.\varepsilon}$ within finite time

i) If $x \in \mathcal{W}_{\delta.\varepsilon}$, then by definition ε-tolerance and δ-loading imply $\mu_{pm}^{ki} = 0$ everywhere from (69), so $\dot{x} = 0$ (using (71)) and $\dot{\mathcal{L}} = 0$. Conversely, if $x \notin \mathcal{W}_{\delta.\varepsilon}$, there exists some i, k, p, m with $l_p(x_p) - l_m(x_m) > \varepsilon$ and $x_p^{ki} > \delta\lambda^i$, so $\mu_{pm}^{ki} = 1$ and $r_{pm}^{ki}(t) = \sigma x_p^{ki}(t) > \sigma\delta\lambda^i > 0$. Moreover, since $l_p(x_p) - l_m(x_m) > \varepsilon$, then $l_m(x_m) - l_p(x_p) < 0$, thus the inner part of the summation of (75) is negative. This means that $\dot{\mathcal{L}} < 0$. This demonstrates that $\mathcal{W}_{\delta.\varepsilon}$ is the largest positively invariant set where $\dot{\mathcal{L}} = 0$

ii) Since $\dot{\mathcal{L}}(x)$ from the previous point is bounded away from zero whenever $x \notin \mathcal{W}_{\delta.\varepsilon}$, the state must enter $\mathcal{W}_{\delta.\varepsilon}$ in finite time. ■

2.7.4. Simulation Setup

We validated our control scheme in MATLAB, modeling a network of seven providers $p \in \mathcal{P}$ (edge servers), three distinct commodity types $i \in \mathcal{I}$ (reflecting different sensing functions), and twenty IoT devices $k \in \mathcal{K}$. To avoid any isolated clusters of service, we randomly assigned each provider a subset of commodities under the constraint that no disjoint provider set serves exactly the same commodity set. Concretely:

- Providers $p = \{1, 3, 4\}$ handle commodities $i = \{2, 3\}$.
- Providers $p = \{2, 6\}$ handle commodities $i = \{1\}$.
- Provider $p = \{5\}$ handles $i = \{1, 2\}$.
- Provider $p = \{7\}$ handles $i = \{3\}$.

Each device's initial request rate was routed to one randomly chosen provider from its corresponding $\mathcal{P}^i$. Initial loads were drawn so that the total demand per commodity was $\lambda^1 = 45$ req/s, $\lambda^2 = 1$ req/s, and $\lambda^3 = 14$ req/s for the first experiment, and $\lambda^1 = \lambda^2 = \lambda^3 = 20$ req/s for the second. The approximate equilibrium parameters were fixed at $\delta = 0.01$ and $\varepsilon = 0.01$. We integrated the dynamics using a backward-Euler scheme with a step size $dt = 10$ ms. According to the previous section 2.7.2, latency at each provider was modeled by

$$l_p\left(x_p(t)\right) \coloneqq \frac{x_p(t)}{c_p} \tag{76}$$

where $x_p(t)$ denotes the current request rate at provider p, and $c_p$represents the capacity threshold of the provider p. In this linear form, the Lipschitz constant of l_p is simply $K_p = \frac{1}{c_p}$.

2.7.5. Simulation Results

Each simulation begins by assigning each IoT device's entire initial request rate to a randomly chosen provider $p \in \mathcal{P}^i$ that supports its commodity i. In both experiments, the system reaches the (δ, ε)-Wardrop equilibrium in under 20 seconds – specifically, within 10 seconds for the first scenario (Figure 35) and 20 seconds for the second scenario (Figure 38).

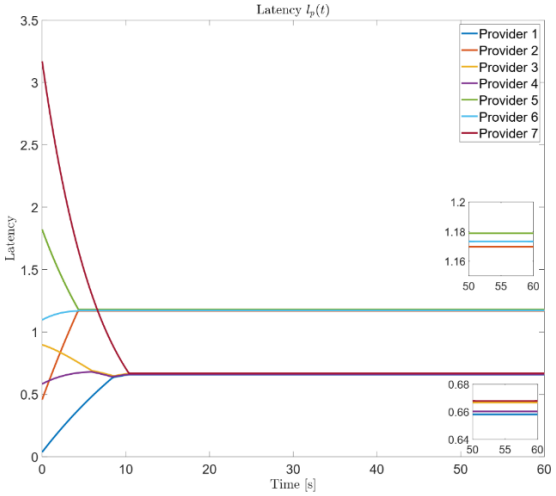

Figure 35: Simulation 1: latency of the providers over time

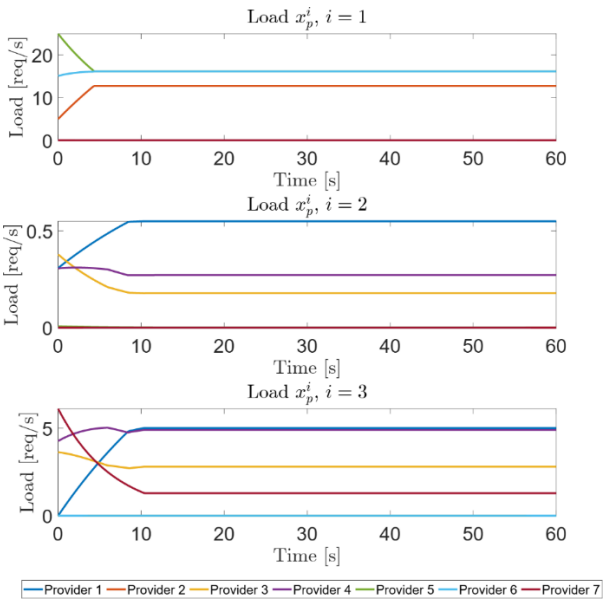

Figure 36: Simulation 1: load over time of the commodities, divided by provider

In the first simulation set, the total demands λ^i differ markedly across commodities, so the steady-state latencies split into two clusters: providers $p = \{2, 5, 6\}$ settles around a latency value of 1.17, while providers $p = \{1, 3, 4, 7\}$ converge near 0.67 (see Figure 35). Provider 5, in particular, becomes δ-unloaded for commodity 2 (i.e., $x_5^{2,k} \leq \delta\lambda^2$ for every device $k \in \mathcal{K}$)., which is possible to notice in Figure 36 and Figure 37.

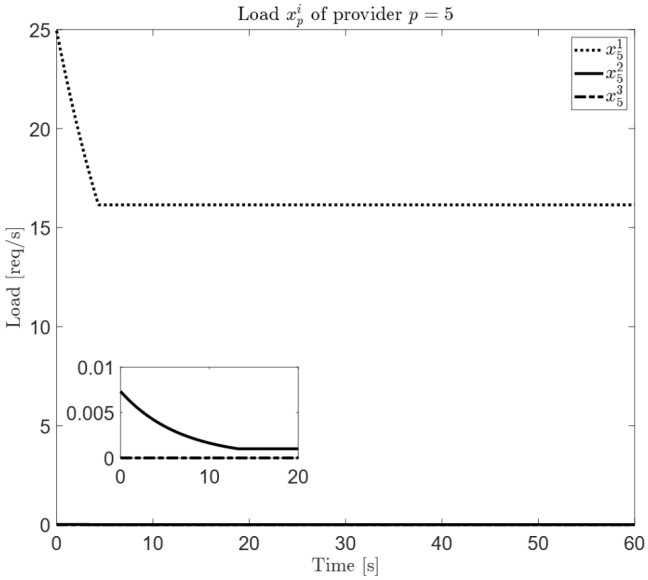

Figure 37: Simulation 1: load over time of provider p = 5 for the three commodities.

Although its overall latency drops as x_5^2 is reduced, this load can no longer be shifted after roughly 13 seconds, because the δ-threshold is reached. Nonetheless, all provider latencies end up within ε of their respective equilibrium values, as illustrated by the subplots in Figure 35.

In the second set of simulations, with $\lambda^1 = \lambda^2 = \lambda^3$, no provider ever falls below the δ threshold, so traffic continues to migrate until every server's latency equalizes. In this case, all latencies converge to about 0.98.

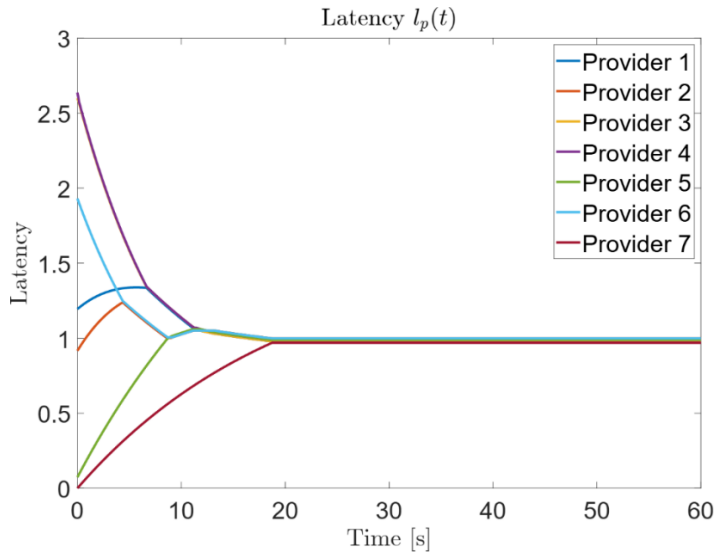

Figure 38: Simulation 2: latency of the providers over time

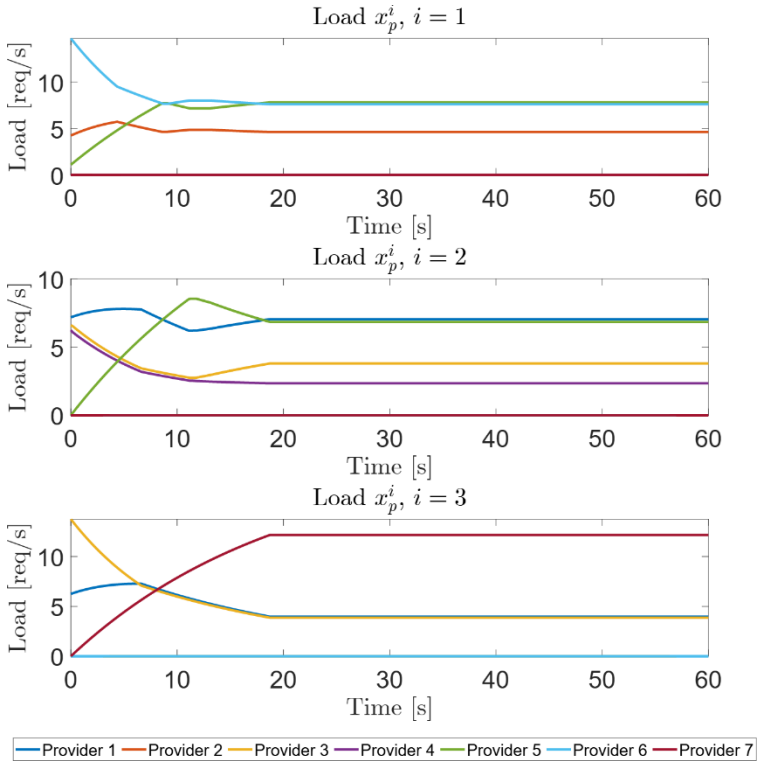

Figure 39: Simulation 2: load over time of the commodities, divided by provider

While simulations 1 and 2 refers to nominal cases where there are no anomalies, this proposed algorithm acts as load balancing (and in particular, given the latency function (76), the number of requests from IoT devices of different IoT applications is balanced among the edge servers). Instead, a third set of simulations is presented here to address the case of anomalies (e.g., loss of providers' capacity due to attacks or malfunctioning). This third simulation set starts from the equilibrium reached by simulation 2, and at time $t = 30$ seconds, the anomaly appears in provider $p = 1$ as a substantial reduction (80%) of its capacity c_1. This makes the equilibrium change, as shown in Figure 40, to a latency of 1.1, while in Figure 41 it is possible to notice that provider 1 is unloaded as to match the new reduced capacity.

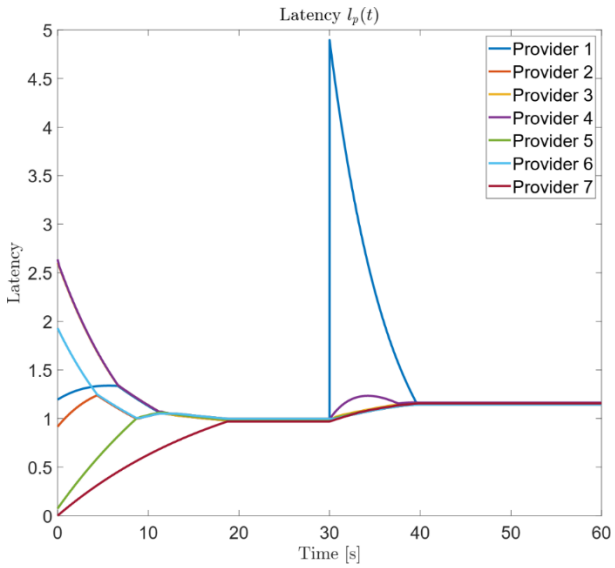

Figure 40: Simulation 3: latency of the providers over time

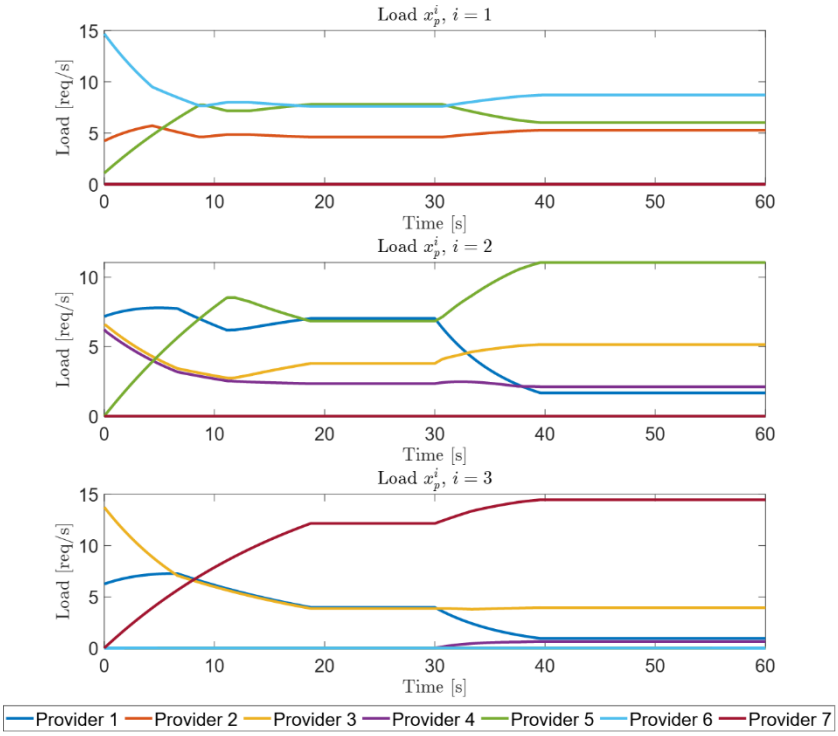

Figure 41: Simulation 3: load over time of the commodities, divided by provider

3. Real-time Data Ingestion and Processing for Big Data

A Big Data Platform in the context of 5G/6G networks is a critical foundation that supports the ingestion, processing, storage, and analysis of high-velocity, high-volume network data. These platforms are engineered to enable intelligent network management, including Self-Healing and Self-Recovery, by harnessing real-time analytics, machine learning, and automation.

With the increasing complexity of 5G/6G systems, autonomous capabilities like self-detection, diagnosis, correction, and prevention of faults are essential. Big data platforms act as the decision-making engine behind these autonomous functions.

To support these demanding functions, a Big Data Platform must fulfill several essential requirements.

First and foremost, scalability and reliability are fundamental. A platform must be able to ingest and process data from a large number of devices and nodes across the network. It should support horizontal scaling to manage growing data volumes and dynamic workloads without degradation in performance. Reliability ensures the system continues to operate effectively even in the presence of faults or disruptions, which is critical in environments where service continuity is essential.

Equally important is the platform's ability to support real-time data ingestion and processing. This capability enables the system to analyze incoming data streams as they arrive, facilitating timely detection of anomalies or failures and allowing for prompt initiation of appropriate response actions.

A robust data storage strategy is essential to ensure efficient access, scalability, and durability of data across the platform. The storage system should support high performance for data retrieval and analysis, accommodate large volumes of diverse data, and provide mechanisms for managing evolving data structures over time.

For advanced use cases, integration with ML and AI frameworks is essential. The platform must be able to train, validate, and deploy models capable of predicting failures, identifying root causes, and recommending or executing actions. These models rely on historical data as well as real-time features to detect anomalies, classify fault patterns, and even learn from past remediation efforts. Frameworks like TensorFlow, PyTorch, and MLflow are commonly used to support the full ML lifecycle.

The envisioned solution is aligned with the architecture developed in Task 3.3 and described in Deliverable D.3.3 for Self-Evolving AI Model Repository (SEMR) that provides essential functionalities for machine learning automation.

One of the core functionalities developed as part of this task is the capability to ingest and process data in real-time. This allows the platform to continuously receive and analyze streaming data as it is generated, enabling timely insights, rapid detection of critical events, and prompt decision-making within dynamic and data-intensive environments. In the following subsections, the design of such functionality is presented.

3.1. Real-time Data Ingestion and Processing

The real-time data ingestion and processing functionality enables the platform to continuously collect, transport, and analyze data as it is generated, without significant delay. This capability is critical in modern platforms that operate in fast-changing environments, such as 5G/6G networks, where timely access to information is essential for operational intelligence, automation, and responsiveness.

Data ingestion refers to the process of acquiring data from various distributed sources, including sensors, logs, network elements, user equipment, or external systems. In a real-time context, data is

ingested in the form of streams (sequences of data records that arrive continuously). The ingestion layer ensures reliable and scalable collection, while maintaining message order, consistency, and minimal latency.

Data processing in real time involves transformation, filtering, enrichment, aggregation, and analysis of data streams as they flow through the system. Instead of waiting for data to be stored and processed in batches, real-time processing pipelines act on each piece of data immediately as it arrives. This enables the platform to detect anomalies, compute metrics, trigger alerts, or initiate automated actions with minimal delay.

To effectively support time-sensitive and mission-critical operations, the real-time data ingestion and processing functionality is designed with several key characteristics that ensure high performance, resilience, and responsiveness.

Low-latency operation is fundamental to this functionality. The system is engineered to minimize the time between data generation and actionable insight, allowing data to be captured, analyzed, and acted upon with minimal delay. This is particularly important in environments where delayed responses could lead to service degradation, missed opportunities, or failure to meet operational requirements.

Another essential characteristic is scalability and fault tolerance. The system must be capable of ingesting and processing large volumes of data originating from multiple, geographically distributed sources without becoming a bottleneck. To achieve this, the architecture supports dynamic resource allocation and redundancy, ensuring continuous operation even in the presence of failures or spikes in data load.

Finally, the functionality is built around event-driven logic, which enables the system to monitor incoming data for predefined patterns, thresholds, or anomalies. When such events are detected, the system can trigger automated responses or alerts in real time. This capability is crucial for use cases that require a rapid reaction to changes in network conditions, user behavior, or system performance.

Together, these characteristics ensure that the platform can operate efficiently and reliably in highly dynamic and data-intensive environments.

3.2. Big Data Platform Architecture

This section presents an architecture that combines Apache Kafka for reliable data streaming and Ray for distributed computation, both orchestrated within a Kubernetes environment. Integrating streaming platforms with distributed computing frameworks is essential for building scalable, real-time processing pipelines.

As part of the system's architectural design, Apache Kafka was selected as the core component for **real-time data ingestion** due to its proven capability to support high-throughput, low-latency data streaming in distributed environments. Kafka is an open-source platform with a mature and extensive ecosystem, offering seamless integration with various tools for stream processing, storage, monitoring, and data governance. Its publish-subscribe model and its support for horizontal scalability make it particularly well-suited for handling continuous data flows from diverse sources such as network elements, log streams, and telemetry systems. Additionally, Kafka's features, such as message durability, partitioning, and fault tolerance, align well with the platform's requirements for reliability and operational resilience. This design choice ensures a robust, flexible, and future-proof foundation for building real-time data pipelines in dynamic and mission-critical environments.

Several alternative technologies were considered for the real-time data ingestion layer, including RabbitMQ, Apache Pulsar, and Amazon Kinesis. RabbitMQ, while lightweight and suitable for message queuing, is generally more focused on traditional messaging patterns and may not scale as efficiently under high-throughput streaming workloads. Apache Pulsar offers similar capabilities to Kafka and introduces features like native multi-tenancy and built-in message storage tiering, but its ecosystem is less mature and widespread. Amazon Kinesis provides a managed streaming service with strong integration into the AWS ecosystem, yet it introduces vendor lock-in and may not offer the same level of control and flexibility as open-source solutions. In comparison, Apache Kafka was chosen for its balance of performance, maturity, active community support, and rich ecosystem of connectors and tools, making it a reliable and extensible choice for building a scalable, real-time ingestion pipeline.

For **data stream processing**, Ray was selected as the primary framework due to its ability to unify distributed computing across diverse workloads, including real-time data processing, machine learning, and reinforcement learning. Unlike traditional stream processing engines such as Apache Flink or Spark Streaming, which are optimized primarily for event-time and stateful stream computations, Ray provides a more general-purpose and flexible execution model that supports both stateless and stateful tasks with fine-grained control. Its actor-based programming model, low-latency task scheduling, and support for scalable microservice patterns make it particularly suitable for dynamic environments like 5G/6G networks, where real-time inference, decision-making, and autonomous behaviors (e.g., self-healing mechanisms) are critical. Moreover, Ray's open-source nature and growing ecosystem (e.g., Ray Serve, Ray Train, and integrations with ML libraries) offer a cohesive foundation for building intelligent, responsive applications that go beyond traditional stream analytics. This design choice enables the platform to process real-time data while seamlessly integrating advanced analytics and AI-driven automation within the same execution environment.

In evaluating stream processing frameworks, several alternatives were considered, including Apache Flink and Spark Streaming. Apache Flink offers powerful capabilities for stateful, event-time-based stream processing and is well-suited for complex event pattern detection and exactly-once semantics. However, it has a steeper learning curve and is tightly focused on traditional stream processing tasks, which may limit its flexibility in supporting broader workloads such as online machine learning or reactive microservices. Spark Streaming, while widely adopted and integrated with the larger Apache Spark ecosystem, follows a micro-batch model that can introduce latency and is less optimal for low-latency or real-time use cases. In contrast, Ray provides a more general-purpose and unified framework that supports both streaming and non-streaming (e.g., ML inference, distributed training, decision-making) workloads under a single abstraction. Its native support for heterogeneous compute workloads, along with a lightweight and flexible execution model, makes it particularly well-suited for AI-enhanced, real-time systems such as those enabling self-healing, predictive analytics, or adaptive resource control in modern network environments. These advantages led to the selection of Ray as the most adaptable and forward-looking solution for the platform’s real-time processing needs.

Kubernetes (K8s) was selected as the deployment platform for both Kafka and Ray due to its scalability, flexibility, and robust orchestration capabilities. As a container orchestration system, Kubernetes enables efficient management of containerized applications, which is critical for handling the dynamic and distributed nature of modern data processing pipelines. For Kafka, Kubernetes provides automated scaling, load balancing, and fault tolerance, ensuring that the data ingestion layer can handle high-throughput, real-time streams with minimal downtime. Kafka's distributed architecture benefits from Kubernetes' ability to manage microservices, maintain cluster health, and handle failover seamlessly. Similarly, Ray, which supports distributed workloads for both real-time processing and

machine learning, leverages Kubernetes' orchestration to scale its computational resources dynamically, allowing flexible management of resources across clusters and ensuring high availability.

Additionally, Kubernetes provides an ideal solution for managing the heterogeneity of deployment environments, especially when combining edge and cloud infrastructures. In the context of a 5G network, where ultra-low latency, high throughput, and high availability are crucial, Kubernetes facilitates the deployment of containers on edge nodes, allowing for near-instantaneous processing of real-time data. Meanwhile, Kubernetes also supports cloud resources for large-scale batch processing, data aggregation, and advanced analytics, providing a seamless extension from edge to cloud. The ability to dynamically distribute workloads based on the needs of the 5G/6G environment ensures that the platform can efficiently manage diverse data sources and processing requirements, providing high performance and reliability.

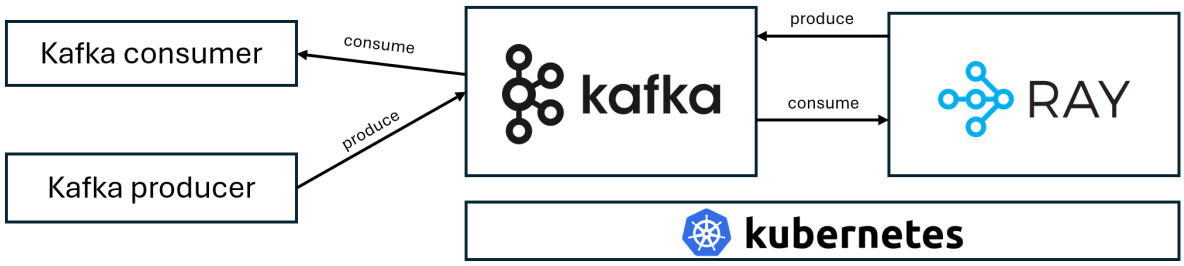

Figure 42: Simplified architecture - Integration of Apache Kafka and Ray

Figure 42 illustrates a distributed architecture where Apache Kafka and Ray are deployed on a Kubernetes cluster, enabling scalable data processing and streaming. Apache Kafka, deployed as a stateful set within Kubernetes, serves as the central **real time data ingestion** component, handling high-throughput, fault-tolerant communication between producers and consumers. Ray, a distributed compute framework also deployed on Kubernetes, is used for parallel and distributed **data stream processing**. Kafka producers (e.g., an IoT device, application backend, or batch ingestion system) send data to Kafka topics hosted within the cluster. Inside the cluster, Ray applications act both as Kafka consumers and producers. Ray consumes incoming messages from Kafka topics to perform real-time computations or machine learning inference tasks. After processing, Ray produces results back to Kafka, potentially to new topics, enabling downstream systems to consume enriched or transformed data. External Kafka consumers can subscribe to Kafka topics to retrieve the processed results produced by Ray.

4. Selection of Anomaly Detection and Self-Healing Tools to be integrated into NANCY testbeds

In the previous sections, a set of tools for Anomaly Detection, Self-Healing and Self-Recovery have been presented. All these methodologies address one specific domain which is of interest to the NANCY project. Moreover, all the presented algorithms have been validated through simulation and in realistic scenarios in order to demonstrate their usefulness in the context of the project.

Among these algorithms, some of them will be also implemented in the project testbeds to demonstrate them in operative or pre-operative contexts. This will allow us to increase the overall TRL from 2-3 to 5-6 of the produced algorithms, thus adding value to the NANCY project and demonstrators. In the following subsections, a detailed report of the algorithms to be implemented or finalized in the WP6 testbeds is presented. These subsections will include motivation, architectures, and in some cases preliminary results obtained in the context of WP6 activities.

4.1. Greek in-lab testbed

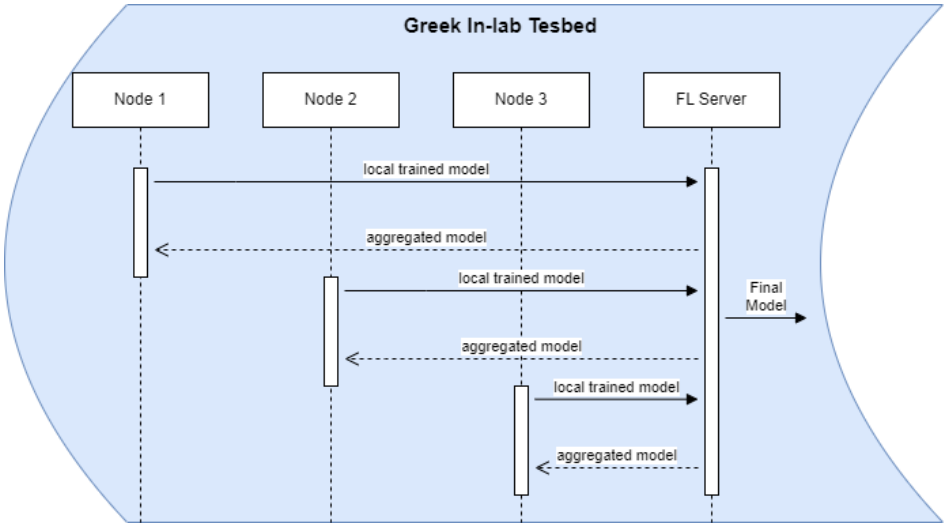

Figure 43: FL-IDS system under Greek In-lab testbed

The FL-IDS (Figure 43) was trained using a dataset that was generated using the Greek in-lab testbed. The utilized network topology is illustrated in Figure 44. In more detail, the *tcpdump* software was deployed in the Micro-operator and was used to capture the network traffic. During the experiment, seven attacks were carried out against the main operator and micro-operator services, namely Reconnaissance Attack, UDP Scan, TCP Connect Scan, SYN Scan, SYN Flood, ICMP Flood, HTTP Flood, and Slow-rate DoS. After the collection of network traffic, the network flows were extracted into a .csv file using the *CICFlowMeter* software.

More details about the aforementioned process are provided in Section 3.2 of D6.7 “Greek in-lab testbed dataset 2”, while the respective dataset can be found on Zenodo and IEEE DataPort.

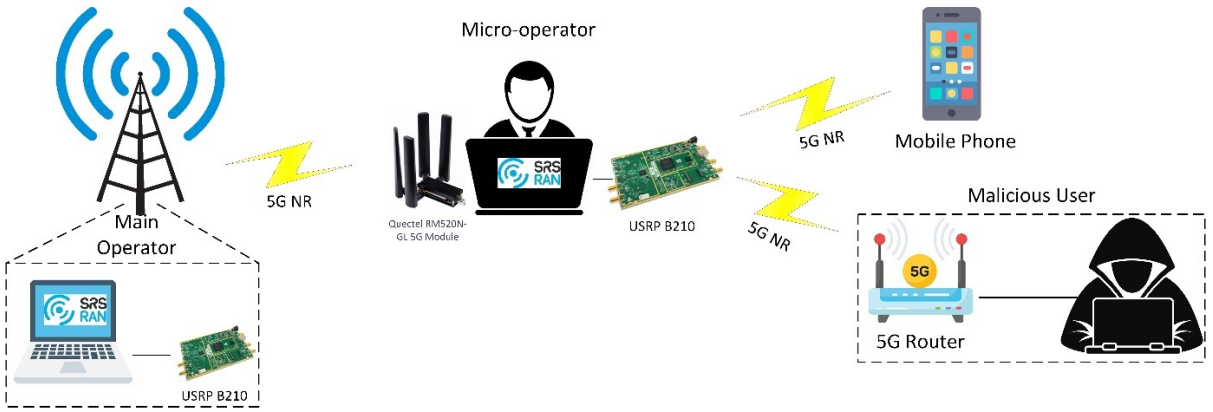

Figure 44: Topology for the dataset generation

The implementation of the FL-IDS in the Greek in-lab testbed represents a critical step in validating the system's performance in a near-operational environment. Initially, the containerized FL-IDS system is deployed across 3 nodes in the testbed, with the central server running on a dedicated high-performance computing node and client instances distributed across various O-RAN components. This deployment leverages the Docker Compose configuration included in our implementation, which greatly simplifies the orchestration of multiple components.

The neural network model at the core of the FL-IDS implementation consists of an input layer that accepts network flow features, followed by multiple dense layers with varying dimensions and activation functions, as described in Table 6.

Table 6: FL-IDS Greek In-lab NN Architecture

Layer	Type	Output Shape	Activation Function
Input	Input	(76,)	-
Hidden 1	Dense	(32)	ReLU
Hidden 2	Dense	(64)	Tanh
Hidden 3	Dense	(32)	ReLU
Hidden 4	Dense	(32)	ReLU
Hidden 5	Dense	(64)	ReLU
Hidden 6	Dense	(64)	ReLU
Output	Dense	(7)	Softmax

The model strikes a balance between detection accuracy and computational efficiency, making it suitable for distributed deployment across nodes with varying capabilities. The model is compiled using parameters of Table 7.

Table 7: FL-IDS Greek In-lab NN training parameters

Parameter	Value
Number of FL rounds	10
Epochs	5
Minimum connected Clients	3
Batch Size	32
Optimizer	Adam
Learning Rate	0.001

A preprocessing approach is applied across all clients, including feature selection to focus on relevant attributes and normalization using a shared standard scaler. This consistency in preprocessing is critical

for the federated learning process, ensuring that model updates from different clients are compatible and meaningful during aggregation. The target of the final dataset produced for classification was to classify traffic into 7 distinct categories as in Table 8.

Table 8: FL-IDS Greek In-Lab NN classification categories

Parameter	Description
Benign	Normal network traffic without malicious intent
Reconnaissance	Probe activities attempting to gather information about the network
TCP Scan	Systematic scanning of TCP ports to identify open services
SYN Scan	Specialized TCP scanning using SYN packets without completing handshakes
SYN Flood	Denial of service attack that sends many SYN packets without completing connections
HTTP Flood	Application layer attack targeting web servers with numerous HTTP requests
Slowrate DoS	Low-bandwidth attack that holds connections open by sending partial requests

The implementation supports multiple federated aggregation strategies, including FedAvg, FedProx, FedAdam, FedAdagrad, and FedYogi, allowing for comparative evaluation under testbed conditions. Data allocation across clients follows a stratified partitioning approach that ensures each client receives instances of all attack classes, maintaining representative class distribution for effective local training. The evaluation in the Greek testbed has provided valuable insights into the performance characteristics of different federated learning strategies in realistic deployment scenarios, as in Figure 45 and Figure 46

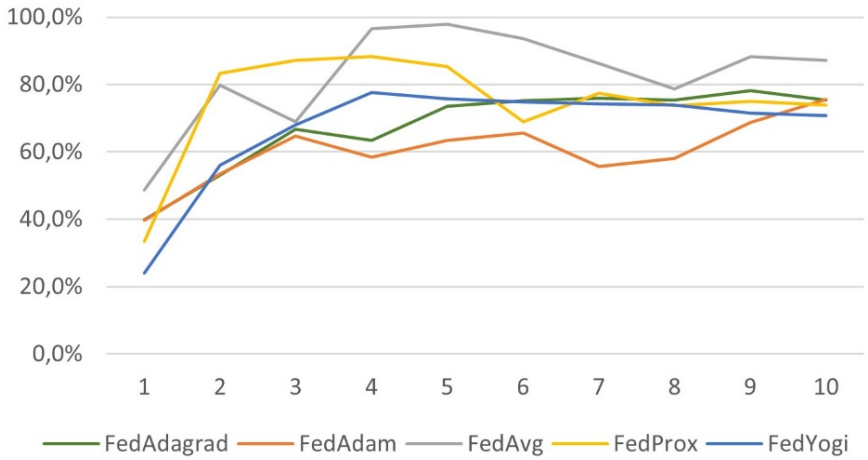

Figure 45: FL-IDS Greek In-lab accuracy during training rounds

.

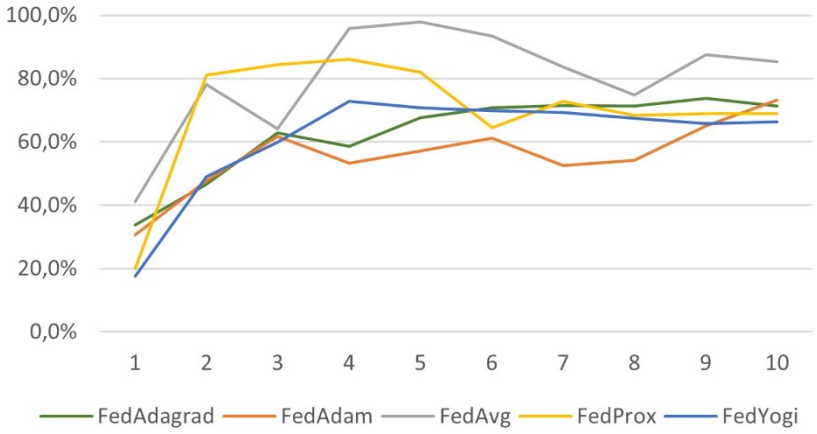

Figure 46: FL-IDS Greek In-lab f1-score during training rounds

The FedAvg strategy demonstrates the highest overall performance, achieving excellent accuracy and F1-score metrics while maintaining the privacy benefits inherent to federated learning. The FedProx strategy shows enhanced resilience when network conditions are less than ideal, making it a valuable alternative for deployments with connectivity challenges. Table 9 summarizes the results.

Table 9: FL-IDS Greek-In-Lab results

Strategy	Accuracy	Precision	Recall	F1-score	Notes
FedAvg	97.89%	98.12%	97.64%	97.85%	Best overall performance
FedProx	88.35%	87.42%	84.95%	86.11%	More resilient to network issues
FedAdam	73.93%	74.12%	73.85%	70.28%	Faster initial convergence
FedAdagrad	72.74%	73.25%	72.15%	69.16%	Less effective for this task
FedYogi	76.00%	76.32%	75.84%	71.72%	Moderate performance

The system maintains strong detection capabilities across most attack categories, with particularly effective performance in identifying reconnaissance activities and scanning attacks. As observed, some attack types such as Slowrate DoS present greater detection challenges, indicating areas for future refinement of the neural network architecture or training methodology.

The federated learning approach ensures that raw network traffic data never leaves its point of origin, addressing privacy concerns inherent in traditional centralized intrusion detection systems. The results demonstrate that with appropriate configuration, particularly when using the FedAvg and FedProx strategies, the system can maintain overall model accuracy even when facing potential adversarial inputs.

4.2. Italian in-lab testbed

4.2.1. Memory Traffic Monitoring Tool

We assessed the suitability of the Memory Traffic Monitoring tool in the Italian in-lab testbed with an experiment that showed how the performance counters change by varying the memory traffic generated by the application under analysis. To this end, we used a micro-benchmark that can be found in the IsolBench suite [https://github.com/CSL-KU/IsolBench] called Latency. This is a memory-killer program whose objective is to stress the memory and see how the system behaves. In detail, Latency accesses a randomly shuffled single linked list of fixed size, which is a parameter of the benchmark. In our experiments, the allocated memory size was varied in the range [16384KB,

131072KB]. The results are shown in Figure 47, which reports the plots for the six performance counters (reported in the label above each plot) handled by the monitoring tool.

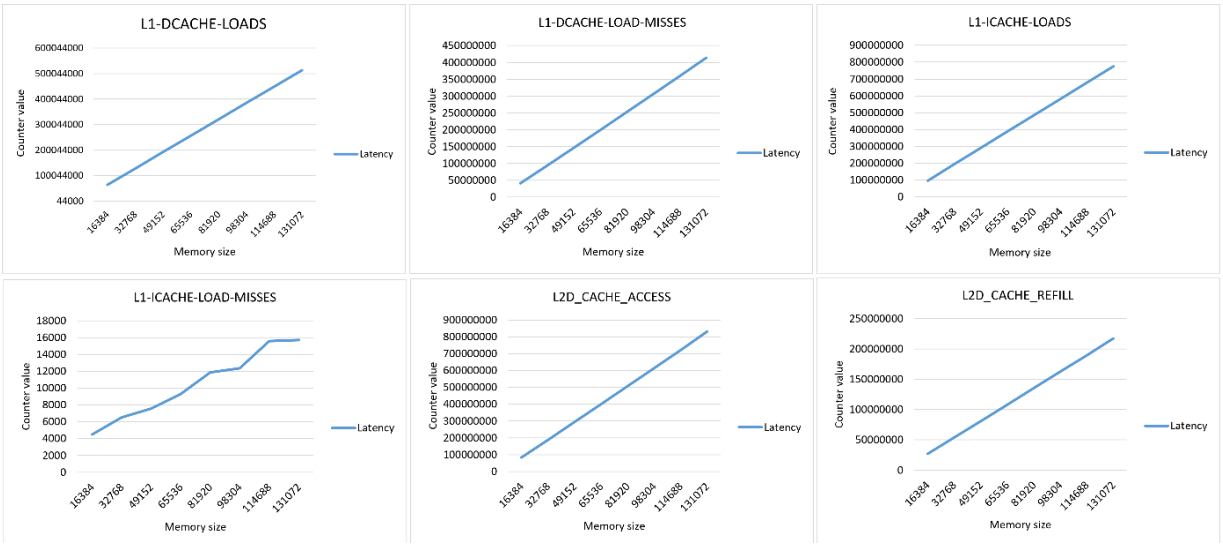

Figure 47: Evaluation of the performance counters for Latency benchmark with varying memory size

The results show that the number of events is directly proportional to the size of the allocated memory. This experiment hints that the monitoring tool can help with the detection of DoS attacks in the Italian in-lab scenario. As a matter of fact, given a user application to monitor with the tool, if the monitor observes that the values of the six ARM performance counters are higher than those expected in a scenario without any attacks, then the application is likely experiencing a DoS attack. The tool can then be used as a trigger for countermeasures against the attack.

4.2.2. Anomaly Detection Procedure

Italian in-lab testbed will also integrate the Anomaly Detection procedure described in Section 2. In particular, given the architecture from D6.8 represented in Figure 48, the anomaly detection module should be able to inspect edge computing-related features (like CPU usage, RAM usage, disk space, and network activity) to assess whether an anomaly in edge computing performances is ongoing. This anomaly may be caused by some malicious traffic generated by unsecure applications running at ARM edge, that reduce the edge resources for the healthy applications (e.g., video streaming application in the testbed).

To this end, we choose the algorithm presented in Section 2.3 to be implemented in the Italian in-lab testbed. The motivation for this choice is that, even if the algorithm is presented to detect anomalies at radio level, its concept can be easily translated into the detection of edge computing anomalies. Indeed, by using the dataset provided by ITL (output of D6.8, which represents the normal activity of the testbed) and the dataset generated by ITL in the third round of data collection (which instead represents the execution of the testbed in the presence of malicious traffic), it is possible to train offline the anomaly detection module so to predict online ongoing anomalies.

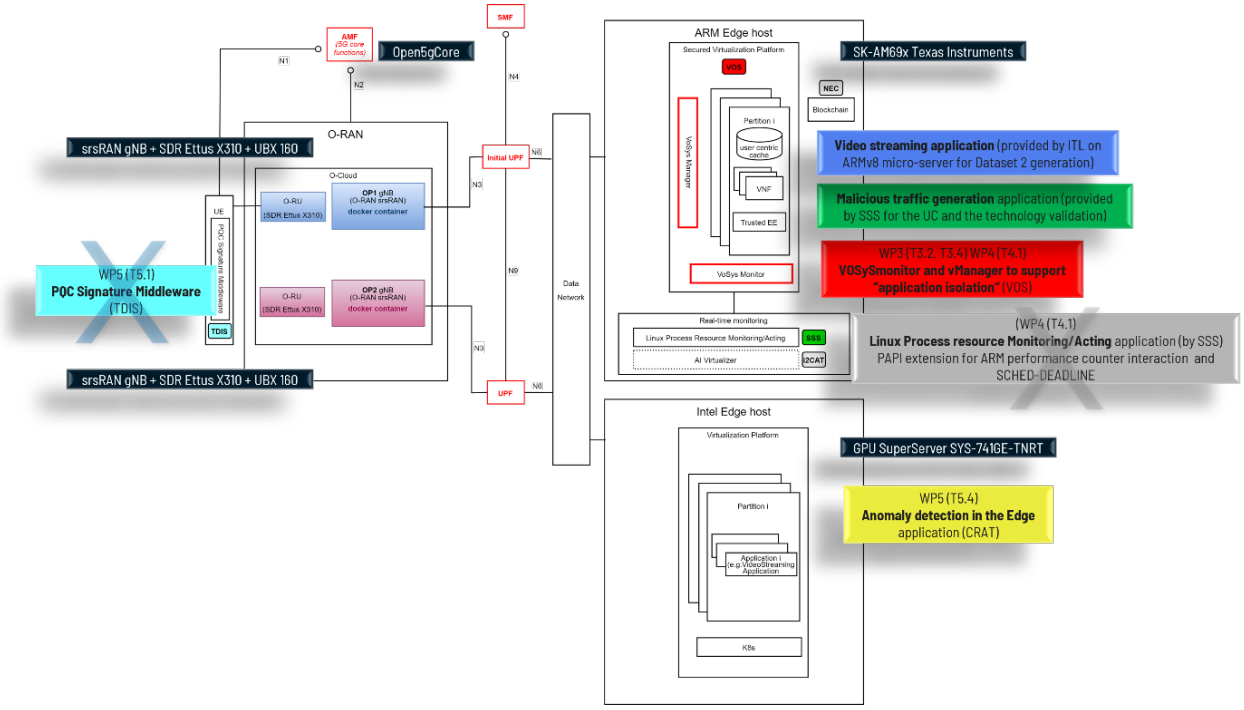

Figure 48: Italian in-lab testbed architecture

The architecture of the proposed anomaly detection module inside Italian in-lab Testbed is represented in Figure 49 (taken from D6.8). In particular, given a single edge server to be monitored, the Anomaly Detection algorithm of Section 2.3 will be implemented as a centralized agent, without any federation possible. The Anomaly Detection module will also report its activity to a Dashboard component provided by ITL, so to visualize alerts for ongoing anomalies. All the interfaces presented in Figure 49 have been defined in the context of WP6 and will be detailed in the relevant WP6 deliverables.

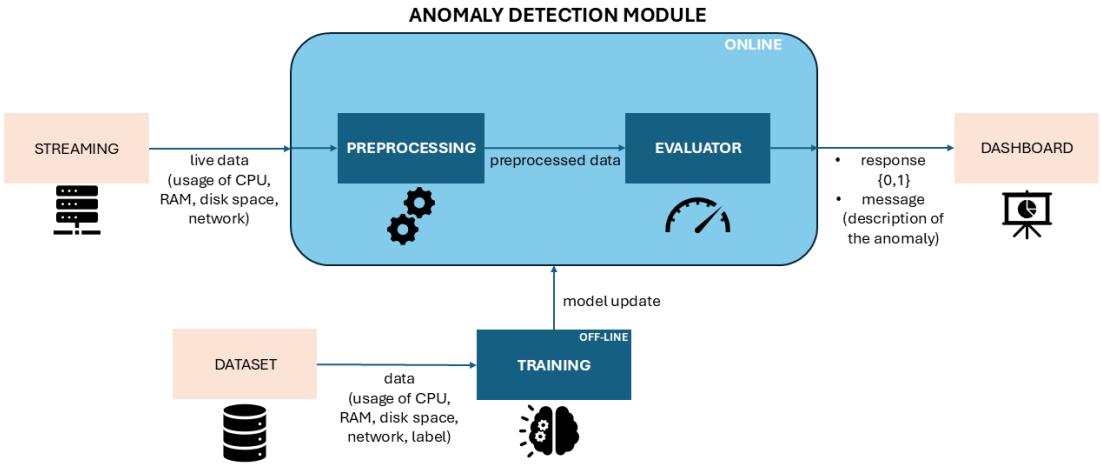

Figure 49: Anomaly Detection Module in the Italian in-lab testbed

4.3. Italian Massive IoT testbed

The Italian Massive IoT testbed is structured around a fixed topology fronthaul network with direct connectivity, designed to demonstrate key characteristics and practical applications of IoT technologies. This configuration serves as a platform to explore the efficient integration and management of IoT systems within a networked environment, with an emphasis on their scalability and suitability for real-world deployment scenarios with a special focus on self-healing techniques.

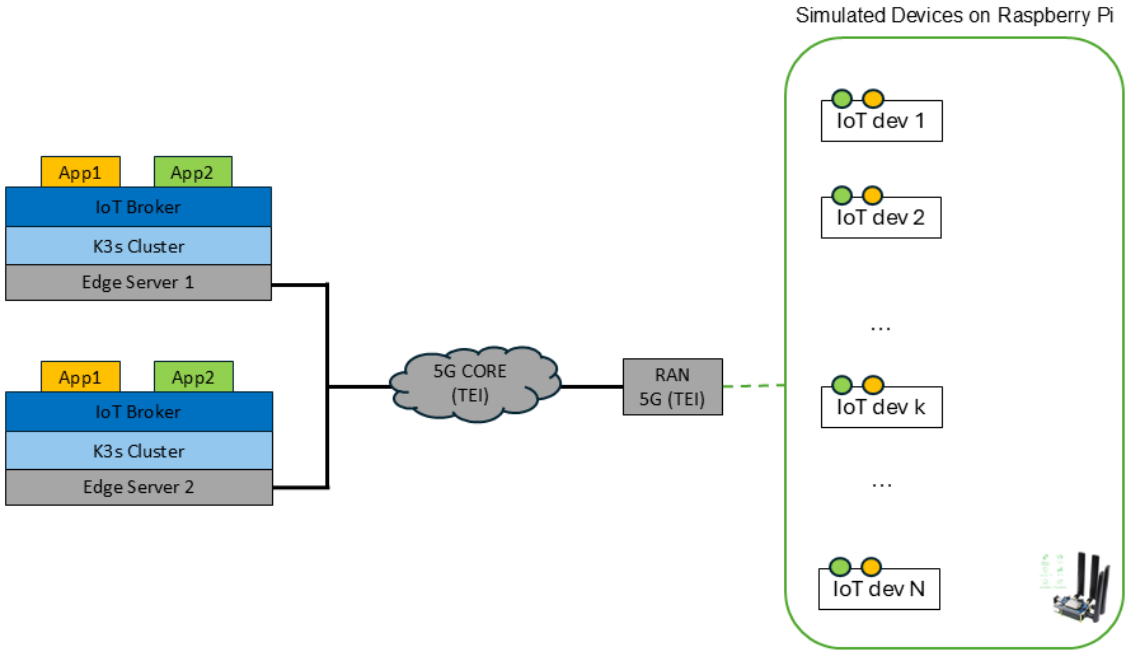

Figure 50: Italian Massive IoT testbed

Figure 50 presents the high-level architecture of the Italian Massive IoT demonstrator testbed deployed at TEI's facilities in Genoa (Italy), highlighting the applications to be deployed to evaluate adversarial dynamic healing based on Wardrop Equilibrium as described in Section 2.7. This commercial-grade 5G testbed includes Ericsson 5G RAN and Core, used to connect IoT devices (simulated on a Raspberry Pi equipped with 5G connectivity) to IoT applications deployed at the edge of the network on two Edge Servers. The two Edge Servers share a similar architecture, utilizing K3s, a lightweight Kubernetes distribution, for container deployment and orchestration. An IoT message broker is deployed on top of Kubernetes, acting as a middleware layer to enable communication between IoT devices and applications via the MQTT protocol. The open-source Eclipse Mosquitto was selected as the broker due to its efficiency and reliability. Multiple IoT applications can be deployed and seamlessly integrated with the message broker to support diverse use cases.

The self-healing algorithm based on Wardrop methodology has been chosen to be implemented in this Testbed since its computationally light logic to be run in an IoT device (i.e., described by equation (69)). This means that each IoT device will independently compute its control action to distribute the load among the available Edge Servers (2 in the case of the Italian Massive IoT Testbed), but at the end reaches a common equilibrium for the desired latency metric. The only centralized information required by the IoT applications to compute such a control action is the desired latency metric measure (or its estimation) at the Edge Servers level (i.e., l_p in (70)).

This algorithm, in the context of the Italian Massive IoT Testbed, will have a two-fold purpose: on one hand it will guarantee balancing on the desired latency metric (e.g., the total number of IoT requests

to each Edge Server, as reported in section 2.7.2) during its normal operation, while on the other hand, it will re-balance the desired latency metric if any anomaly that impacts this latency metric occurs.

The other algorithms chosen as self-healing procedures in Section 2 are not suitable to be implemented in this testbed, given the limitation of the testbed itself in controlling radio parameters like horizontal and vertical gain. Moreover, given the fact that IoT devices will be simulated by a Raspberry Pi, the 5G radio network will actually have only one device connected, which does not fully demonstrate the contribution of the other proposed self-healing and self-recovery tools.

5. Conclusion

In summary, Deliverable 5.3 brings together a range of lightweight, distributed routines that turn the vision of self-healing B5G networks into reality. By placing intelligence at the edge, whether through federated learning models that scan local logs, consensus schemes that spot misbehaving cells, or reinforcement-learning agents that reroute traffic on the fly, this work shows that networks can diagnose and repair themselves without leaning on a single, vulnerable control point. Across multiple simulated and real-world scenarios, these solutions consistently bounce back to pre-fault levels faster than traditional, centralized approaches, all while keeping sensitive data in its place and staying resilient against malicious tampering.

Equally important, the methods laid out here fit neatly into NANCY's broader architecture and can be updated or swapped out as new algorithms emerge. That flexibility will prove critical as we move into NANCY testbeds under the umbrella of WP6 activities. Thus, these tools will be validated in live environments and integrated with other NANCY services. At that stage, we will refine aspects like timing, resource overhead and multi-vendor coordination, but the core takeaway is clear: A truly self-healing B5G network is no longer a distant aim, but a practical blueprint we are ready to build upon.

References

[1] A. Gupta and R. K. Jha, “A survey of 5G network: Architecture and emerging technologies,” IEEE Access, vol. 3, pp. 1206–1232, 2015.

[2] J. G. Andrews, S. Buzzi, W. Choi, S. V. Hanly, A. Lozano, A. C. K. Soong, and J. C. Zhang, “What will 5G be?,” IEEE Journal on Selected Areas in Communications, vol. 32, no. 6, pp. 1065–1082, Jun. 2014.

[3] H. Fourati, R. Maaloul, and L. Chaari, “A survey of 5G network systems: challenges and machine learning approaches,” International Journal of Machine Learning and Cybernetics, vol. 12, no. 2, pp. 385–431, Aug. 2021.

[4] W. Saad, M. Bennis, and M. Chen, “A vision of 6G wireless systems: Applications, trends, technologies, and open research problems,” IEEE Network, vol. 34, no. 3, pp. 134–142, Oct. 2019.

[5] W. Wang, Q. Liao, and Q. Zhang, “COD: A cooperative cell outage detection architecture for self-organizing femtocell networks,” IEEE Transactions on Wireless Communications, vol. 13, no. 11, pp. 6007–6014, Sep. 2014.

[6] I. de-la-Bandera, R. Barco, P. Munoz, and I. Serrano, “Cell outage detection based on handover statistics,” IEEE Communications Letters, vol. 19, no. 7, pp. 1189–1192, Apr. 2015.

[7] C. M. Mueller, M. Kaschub, C. Blankenhorn, and S. Wanke, “A cell outage detection algorithm using neighbor cell list reports,” Self-Organizing Systems: Third International Workshop (IWSOS), 2008, pp. 218–229.

[8] Q. Liao, M. Wiczanowski, and S. Stańczak, “Toward cell outage detection with composite hypothesis testing,” IEEE international conference on communications (ICC), 2012, pp. 4883–4887.

[9] R. M. Khanafer, B. Solana, J. Triola, R. Barco, L. Moltsen, Z. Altman, and P. Lázaro, "Automated Diagnosis for UMTS Networks Using Bayesian Network Approach," IEEE Transactions on Vehicular Technology, vol. 57, no. 4, pp. 2451-2461, Jul. 2008.

[10] M. Alias, N. Saxena, and A. Roy, “Efficient cell outage detection in 5G HetNets using hidden Markov model,” IEEE Communications Letters, vol. 20, no. 3, pp. 562–565, 2016.

[11] O. Onireti, A. Zoha, J, Moysen, A. Imran, L. Giupponi, M. A. Imran, and A. Abu-Dayya, "A Cell Outage Management Framework for Dense Heterogeneous Networks," in IEEE Transactions on Vehicular Technology, vol. 65, no. 4, pp. 2097-2113, Apr. 2016.

[12] L. Bodrog, M. Kajo, S. Kocsis, and B. Schultz, “A robust algorithm for anomaly detection in mobile networks,” IEEE 27th Annual International Symposium on Personal, Indoor, and Mobile Radio Communications (PIMRC), 2016, pp. 1–6.

[13] W. Xue, M. Peng, Y. Ma, and H. Zhang, “Classification-based approach for cell outage detection in self-healing heterogeneous networks,” IEEE Wireless Communications and Networking Conference (WCNC), 2014, pp. 2822–2826.

[14] P.-C. Lin, “Large-scale and high-dimensional cell outage detection in 5G self-organizing networks,” in 2019 Asia-Pacific Signal and Information Processing Association Annual Summit and Conference (APSIPA ASC), 2019, pp. 8–12.

[15] Y.-H. Ping and P.-C. Lin, “Cell outage detection using deep convolutional autoencoder in mobile communication networks,” Asia-Pacific Signal and Information Processing Association Annual Summit and Conference (APSIPA ASC), 2020, pp. 1557–1560.

[16] B. Hussain, Q. Du, S. Zhang, A. Imran, and M. A. Imran, “Mobile edge computing-based data-driven deep learning framework for anomaly detection,” IEEE Access, vol. 7, pp. 137656–137667, 2019.

[17] W. H. L. Pinaya, S. Vieira, R. Garcia-Dias, and A. Mechelli, “Convolutional neural networks,” Machine learning, Elsevier, 2020, pp. 173–191.

[18] C. Kruegel, D. Mutz, W. Robertson, and F. Valeur, “Bayesian event classification for intrusion detection,” 19th Annual Computer Security Applications Conference, 2003, pp. 14-23.

[19] W. Hu, Y. Liao, and V. R. Vemuri, “Robust Anomaly Detection Using Support Vector Machines,” 2003. [Online]. Available: https://api.semanticscholar.org/CorpusID:2433033

[20] Applications of Data Mining in Computer Security. Springer US, 2002. doi: 10.1007/978-1-4615-0953-0.

[21] S. Hawkins, H. He, G. Williams, and R. Baxter, “Outlier Detection Using Replicator Neural Networks,” Lecture Notes in Computer Science, Springer Berlin Heidelberg, 2002, pp. 170–180.

[22] G. Poojitha, K. N. Kumar, and P. J. Reddy, “Intrusion Detection using Artificial Neural Network,”2nd International conference on Computing, Communication and Networking Technologies, 2010, pp. 1-7.

[23] S. He, J. Zhu, P. He, and M. R. Lyu, “Experience Report: System Log Analysis for Anomaly Detection,” IEEE 27th International Symposium on Software Reliability Engineering (ISSRE), 2016, pp. 207–218.

[24] M. Du, F. Li, G. Zheng, and V. Srikumar, “DeepLog: Anomaly Detection and Diagnosis from System Logs through Deep Learning,” ACM SIGSAC Conference on Computer and Communications Security (CCS ’17), 2017, pp. 1285–1298.

[25] W. Zumin, T. Jiyu, F. Hui, C. Liming, and Q. Jing, “LightLog: A lightweight temporal convolutional network for log anomaly detection on the edge,” Elsevier, vol. 203, p. 108616, Feb. 2022.

[26] I. Makris, N. Ntampakis, T. Lagkas, P. Radoglou-Grammatikis, S. K. Goudos, V. Argyriou, E. Fountoukidis, A. F. Skarmeta, P. F. Saura, and P. Sarigiannidis, “Elevating 5G Network Security: A Profound Examination of Federated Learning Aggregation Strategies for Attack Detection,” IEEE Future Networks World Forum (FNWF), 2023, pp. 1–6.

[27] J. Bhanbhro, S. Nisticò, and L. Palopoli, “Issues in federated learning: some experiments and preliminary results,” Scientific Reports, vol. 14, no. 1, pp. 1–15, Dec. 2024.

[28] H. B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. y Arcas, “Communication-Efficient Learning of Deep Networks from Decentralized Data,” 20th International Conference on Artificial Intelligence and Statistics (AISTATS), 2017.

[29] T. Wang, Z. Zheng, and F. Lin, “Federated learning framework based on trimmed mean aggregation rules,” Expert Systems with Applications, vol. 270, p. 126354, Apr. 2025.

[30] V. Chandola, A. Banerjee, and V. Kumar, “Anomaly detection: A survey,” ACM Computing Surveys (CSUR), vol. 41, no. 3, pp. 1–58, 2009.

[31] N. Duffield, P. Haffner, B. Krishnamurthy and H. Ringberg, "Rule-Based Anomaly Detection on IP Flows," IEEE INFOCOM, pp. 424-432, 2009

[32] S. Hela, B. Amel, and R. Badran, “Early anomaly detection in smart home: A causal association rule-based approach,” Artificial Intelligence in Medicine, vol. 91, pp. 57–71, Sep. 2018.

[33] B. Li, S. Zhao, R. Zhang, Q. Shi, and K. Yang, “Anomaly detection for cellular networks using big data analytics,” IET Communications, vol. 13, no. 20, pp. 3351–3359, Nov. 2019.

[34] J. B. Porch, C. H. Foh, H. Farooq, and A. Imran, “Machine learning approach for automatic fault detection and diagnosis in cellular networks,” IEEE International Black Sea Conference on Communications and Networking (BlackSeaCom), 2020, pp. 1–5.

[35] J. Li, Z. Zhao, and R. Li, “Machine learning-based IDS for software-defined 5G network,” IET Networks, vol. 7, no. 2, pp. 53–60, Mar. 2018.

[36] M. M. Saeed, R. A. Saeed, M. Abdelhaq, R. Alsaqour, M. K. Hasan, and R. A. Mokhtar, “Anomaly detection in 6G networks using machine learning methods,” Electronics (Basel), vol. 12, no. 15, p. 3300, Jul. 2023.

[37] E. Jaw and X. Wang, “Feature selection and ensemble-based intrusion detection system: an efficient and comprehensive approach,” Symmetry (Basel), vol. 13, no. 10, p. 1764, Sep. 2021.

[38] T. Sundqvist, M. Bhuyan, J. Forsman, and E. Elmroth, “Boosted Ensemble Learning for Anomaly Detection in 5G RAN,” IFIP International Conference on Artificial Intelligence Applications and Innovations, 2020, pp. 15–30.

[39] A. Kumar, S. Rezaei, A. Javadpour, F. Miri, W. Zhang, and D. Wang, “Automatic Fault Detection and Diagnosis in Cellular Networks and Beyond 5G: Intelligent Network Management,” Algorithms, vol. 15, p. 432, Apr. 2022.

[40] L. Fernández-Maimó, F. J. García Clemente, M. Pérez, and G. Martinez Perez, “On the Performance of a Deep Learning-Based Anomaly Detection System for 5G Mobile Networks,” 2017 IEEE SmartWorld, Ubiquitous Intelligence & Computing, Advanced & Trusted Computed, Scalable Computing & Communications, Cloud & Big Data Computing, Internet of People and Smart City Innovation, 2017, pp. 1-8.

[41] M. Doan and Z. Zhang, “Deep learning in 5G wireless networks-anomaly detections,” 29th Wireless and Optical Communications Conference (WOCC), 2020, pp. 1–6.

[42] M. S. Parwez, D. B. Rawat, and M. Garuba, “Big data analytics for user-activity analysis and user-anomaly detection in mobile wireless network,” IEEE Transactions on Industrial Informatics, vol. 13, no. 4, pp. 2058-2065, Aug. 2017.

[43] Q. Zhu and L. Sun, “Big data driven anomaly detection for cellular networks,” IEEE Access, vol. 8, pp. 31398–31408, 2020.

[44] W. Liu, P. Lei, D. Xu, and X. Zhu, “Anomaly recognition, diagnosis and prediction of massive data flow based on time-GAN and DBSCAN for power dispatching automation system,” Processes, vol. 11, no. 9, p. 2782, Sep. 2023.

[45] D. Canastro, R. Rocha, M. Antunes, D. Gomes, and R. L. Aguiar, “Root cause analysis in 5G/6G networks,” 8th International Conference on Future Internet of Things and Cloud (FiCloud), 2021, pp. 217–224.

[46] G. Fernandes, J. J. P. C. Rodrigues, L. F. Carvalho, J. F. Al-Muhtadi, and M. L. Proença, “A comprehensive survey on network anomaly detection,” Telecommun Systems, vol. 70, pp. 447–489, 2019.

[47] S. Thudumu, P. Branch, J. Jin, and J. Singh, “A comprehensive survey of anomaly detection techniques for high dimensional big data,” Journal of Big Data, vol. 7, pp. 1–30, Jul. 2020.

[48] V. Gudepu et al., “GEN-DRIFT: Generative AI-driven drift handling for beyond 5G networks,” Computer Networks, vol. 263, p. 111237, 2025.

[49] A. Hasan, C. Boeira, K. Papry, Y. Ju, Z. Zhu, and I. Haque, “Root Cause Analysis of Anomalies in 5G RAN Using Graph Neural Network and Transformer,” arXiv preprint arXiv:2406.15638, 2024.

[50] D. Han, Z. Wang, Y. Zhong, W. Chen, J. Yang, S. Lu, X. Shi, and X. Yin, “Evaluating and improving adversarial robustness of machine learning-based network intrusion detectors,” IEEE Journal on Selected Areas in Communications, vol. 39, no. 8, pp. 2632–2647, Aug. 2021.

[51] S. Agrawal, S. Sarkar, O. Aouedi, G. Yenduri, K. Piamrat, M. Alazab, S. Bhattacharya, P. K. R. Maddikunta, and T. R. Gadekallu, “Federated learning for intrusion detection system: Concepts, challenges and future directions,” Computer Communications, vol. 195, pp. 346–361, Nov. 2022.

[52] A. Asghar, H. Farooq, and A. Imran, “Self-healing in emerging cellular networks: Review, challenges, and research directions,” IEEE Communications Surveys & Tutorials, vol. 20, no. 3, pp. 1682–1709, Apr. 2018.

[53] M. Z. Asghar, S. Hämäläinen, and T. Ristaniemi, “Self-healing framework for LTE networks,” iIEEE 17th International Workshop on Computer Aided Modeling and Design of Communication Links and Networks (CAMAD), 2012, pp. 159–161.

[54] M. Amirijoo, L. Jorguseski, R. Litjens, and R. Nascimento, “Effectiveness of cell outage compensation in LTE networks,” IEEE Consumer Communications and Networking Conference (CCNC), 2011, pp. 642–647.

[55] E. Chu, I. Bang, S. H. Kim, and D. K. Sung, “Self-organizing and self-healing mechanisms in cooperative small-cell networks,” IEEE 24th annual international symposium on personal, indoor, and mobile radio communications (PIMRC), 2013, pp. 1576–1581.

[56] Z. Jiang, P. Yu, Y. Su, W. Li, and X. Qiu, “A cell outage compensation scheme based on immune algorithm in LTE networks,” 15th Asia-Pacific Network Operations and Management Symposium (APNOMS), 2013, pp. 1–6.

[57] W. Li, Y. Peng, J. Zhengxin, and Z. Li, “Centralized Management Mechanism for Cell Outage Compensation in LTE Networks,” International Journal of Distributed Sensor Networks, vol. 8, no. 11, p. 170589, Jan. 2012.

[58] T. Zhang, K. Zhu, and E. Hossain, “Data-driven machine learning techniques for self-healing in cellular wireless networks: Challenges and solutions,” Intelligent Computing, vol. 2022, Sep. 2022.

[59] E. Vaezpour, L. Majzoobi, M. Akbari, S. Parsaeefard, and H. Yanikomeroglu, “A deep learning-based approach for cell outage compensation in NOMA networks,” IEEE Open Journal of Vehicular Technology, vol. 3, pp. 149–166, 2022.

[60] M. Yonan, A. M. Hassan, and E. Emary, “Cell outage compensation scheme based on hybrid genetic algorithms and neural networks,” IET Communications, vol. 17, no. 4, pp. 387–397, 2023.

[61] Y. Chen, Y. Liu, M. Zeng, U. Saleem, Z. Lu, and X. Wen, “Reinforcement learning meets wireless networks: A layering perspective,” IEEE Internet Things Journal, vol. 8, no. 1, pp. 85–111, Jan. 2021.

[62] A. Saeed, O. G. Aliu, and M. A. Imran, “Controlling self healing cellular networks using fuzzy logic,” IEEE Wireless Communications and Networking Conference (WCNC), 2012, pp. 3080–3084.

[63] S. Adel, K. Muhammed, A. Abdallah, M. Rida, A. Morsy, G. Nasser, A. Khattab, A. Taha, and H. El-Akel,, “Cell Outage Compensation Using Q-learning for Self-Organizing Networks,” International Conference on Microelectronics (ICM), 2021, pp. 102–105.

[64] J. Moysen and L. Giupponi, “A reinforcement learning based solution for self-healing in LTE networks,” IEEE 80th Vehicular Technology Conference (VTC2014-Fall), 2014, pp. 1–6.

[65] O. Onireti et al., “A cell outage management framework for dense heterogeneous networks,” IEEE Trans Veh Technol, vol. 65, no. 4, pp. 2097–2113, 2015.

[66] F. B. Mismar, J. Choi, and B. L. Evans, “A framework for automated cellular network tuning with reinforcement learning,” IEEE Transactions on Communications, vol. 67, no. 10, pp. 7152–7167, 2019.

[67] P. Yu, X. Yang, F. Zhou, H. Li, L. Feng, W. Li, and X. Qiu, “Deep reinforcement learning aided cell outage compensation framework in 5G cloud radio access networks,” Mobile Networks and Applications, vol. 25, no. 5, pp. 1644–1654, Jul. 2020.

[68] M. Iwamoto, A. Suzuki, and M. Kobayashi, “Deep Reinforcement Learning Based Antenna Selection for Cell Outage Compensation,” IEEE International Conference on Communications (ICC), 2023, pp. 3945–3950.

[69] P. Tam, I. Song, S. Kang, S. Ros, and S. Kim, “Graph Neural Networks for Intelligent Modelling in Network Management and Orchestration: A Survey on Communications,” Electronics (Basel), vol. 11, no. 20, p. 3371, Oct. 2022.

[70] S. Lee, J. Kim, S. Shin, P. Porras, and V. Yegneswaran, “Athena: A Framework for Scalable Anomaly Detection in Software-Defined Networks,” 47th Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN), 2017, pp. 249–260.

[71] K. Jurczenia and J. Rak, “A Survey of Vehicular Network Systems for Road Traffic Management,” IEEE Access, vol. 10, pp. 42365–42385, 2022.

[72] D. Bis, N. Bix, B. Gruman, S. Guenette, A. Hauge, H. Moser, J. Paul, and G.Trajcevski, “LaCAVR: Load and Constraints Aware Vehicle Rerouting,” 20th IEEE International Conference on Mobile Data Management (MDM), 2019, pp. 359–360.

[73] K. I. Adeyinka and T. I. Adeyinka, “Real-Time Traffic Management Using Graph Models,” in Neural Networks and Graph Models for Traffic and Energy Systems, IGI Global, 2025, pp. 231–258.

[74] D. Menegatti, E. De Santis, S. Felli, and A. Giuseppi, “AdaLightLog: Enhancing Application Logs Anomaly Detection via Adaptive Federating Learning,” 19th International Conference on Critical Information Infrastructures Security (CRITIS), 2024, pp. 289–305

[75] D. Min and L. Feifei, “Spell: Streaming Parsing of System Event Logs,” IEEE, 2016, doi: 10.1109/ICDM.2016.160.

[76] P. He, J. Zhu, Z. Zheng and M. R. Lyu, "Drain: An Online Log Parsing Approach with Fixed Depth Tree," IEEE International Conference on Web Services (ICWS), 2017, pp. 33-40.

[77] A. Makanju, A. N. Zincir-Heywood, and E. E. Milios, “Clustering Event Logs Using Iterative Partitioning,” 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 2009, pp. 1255–1264.

[78] A. Makanju, A. N. Zincir-Heywood, and E. E. Milios, “A lightweight algorithm for message type extraction in system application logs,” IEEE Transactions on Knowledge and Data Engineering, vol. 24, no. 11, pp. 1921-1936, Nov. 2012.

[79] W. Xu, L. Huang, A. Fox, D. Patterson, and M. I. Jordan, “Detecting large-scale system problems by mining console logs,” ACM SIGOPS 22nd symposium on Operating systems principles (SOSP ’09), 2009, pp. 117–132.

[80] X. Liu, W. Liu, X. Di, J. Li, B. Cai, W. Ren, and H. Yang, “LogNADS: Network anomaly detection scheme based on semantic representation,” Future Generation Computer Systems, vol. 124, pp. 390–405, Nov. 2021.

[81] A. Joulin, E. Grave, P. Bojanowski, M. Douze, H. Jégou, and T. Mikolov, “FastText.zip: Compressing Text Classification Models,” arXiv preprint arXiv:1612.03651, 2016, [Online]. Available: https://arxiv.org/abs/1612.03651

[82] V. Raunak, “Effective Dimensionality Reduction for Word Embeddings,” Workshop on Representation Learning for NLP (RepL4NLP) at ACL, 2019, pp. 235–243.

[83] C. Lea, R. Vidal, A. Reiter, and G. D. Hager, "Temporal Convolutional Networks: A Unified Approach to Action Segmentation," Computer Vision – ECCV 2016 Workshops, 2016, pp. 47-54.

[84] B. Li, S. Ma, R. Deng, K.-K. R. Choo, and J. Yang, "Federated Anomaly Detection on System Logs for the Internet of Things: A Customizable and Communication-Efficient Approach," IEEE Transactions on Network and Service Management, vol. 19, no. 2, pp. 1705–1716, Jun. 2022.

[85] Y. Guo, Y. Wu, Y. Zhu, B. Yang, and C. Han, "Anomaly Detection using Distributed Log Data: A Lightweight Federated Learning Approach," International Joint Conference on Neural Networks (IJCNN), 2021, pp. 1-8.

[86] A. Giuseppi, L. Della Torre, D. Menegatti, and A. Pietrabissa, "AdaFed: Performance-based Adaptive Federated Learning," The 5th International Conference on Advances in Artificial Intelligence (ICAAI), 2021, pp. 1-6.

[87] M. Devine, S. P. Ardakani, M. Al-Khafajiy, and Y. James, "Federated Machine Learning to Enable Intrusion Detection Systems in IoT Networks," Electronics (Basel), vol. 14, no. 6, p. 1176, Mar. 2025.

[88] M. Alavirad, U. S. Hashmi, M. Mansour, A. Esswie, R. Atawia, G. Poitau, and M. Repeta, "O-RAN architecture, interfaces, and standardization: Study and application to user intelligent admission control," Frontiers in Communications and Networks, vol. 4, p. 1127039, Mar. 2023.

[89] U. Ahmed, M. Nazir, A. Sarwar, T. Ali, E.-H. M. Aggoune, T. Shahzad, and M. A. Khan, "Signature-based intrusion detection using machine learning and deep learning approaches empowered with fuzzy clustering," Scientific Reports, vol. 15, no. 1, p. 1726, Jan. 2025.

[90] D. J. Beutel et al., "Flower: A friendly federated learning framework," arXiv preprint arXiv: 2007.14390, 2007.

[91] S. Reddi et al., "Adaptive federated optimization," arXiv preprint arXiv:2003.00295, 2020.

[92] A. Wrona, S. Gentile, E. De Santis, A. Giuseppi, A. Pietrabissa, and F. Delli Priscoli, "A Cooperative Feature Removal Mechanism for Cell Outage Detection in Wireless Telecommunication Networks," International Conference on Critical Information Infrastructures Security, 2024, pp. 84–95.

[93] J. Vidal, "Anomaly detection in cellular networks," 2020, Kaggle. [Online]. Available: https://kaggle.com/competitions/anomaly-detection-in-4g-cellular-networks

[94] R. S. Sutton and A. G. Barto, Reinforcement learning: An introduction. MIT press, 2018.

[95] V. Mnih et al., "Human-level control through deep reinforcement learning," Nature, vol. 518, no. 7540, pp. 529–533, Feb. 2015.

[96] V. Mnih et al., "Playing atari with deep reinforcement learning," arXiv preprint arXiv:1312.5602, 2013.

[97] B. Peng, Q. Sun, S. E. Li, D. Kum, Y. Yin, J. Wei, and T. Gu, “End-to-end autonomous driving through dueling double deep Q-network,” Automotive Innovation, vol. 4, pp. 328–337, Jul. 2021.

[98] W. Cheng, X. Liu, X. Wang, and G. Nie, “Task offloading and resource allocation for industrial Internet of Things: A double-dueling deep Q-network approach,” IEEE Access, vol. 10, pp. 103111–103120, 2022.

[99] F. Gunnarsson, M. N. Johansson, A. Furuskär, M. Lundevall, A. Simonsson, C. Tidestav, and M. Blomgren, “Downtilted base station antennas-a simulation model proposal and impact on HSPA and LTE performance,” IEEE 68th Vehicular Technology Conference, 2008, pp. 1–5.

[100] 3rd Generation Partnership Project (3GPP), “3GPP TR 38.901 V16.1.0: Study on channel model for frequencies from 0.5 to 100 GHz,” Jan. 2020.

[101] J. Vihriälä, A. A. Zaidi, V. Venkatasubramanian, N. He, E. Tiirola, J. Medbo, E. Lähetkangas, K. Werner, K. Pajukoski, A. Cedergren, and R. Baldemair, “Numerology and frame structure for 5G radio access,” IEEE 27th annual international symposium on personal, indoor, and mobile radio communications (PIMRC), 2016, pp. 1–5.

[102] L. Li, Y. Fan, M. Tse, and K.-Y. Lin, “A review of applications in federated learning,” Computers & Industrial Engineering, vol. 149, p. 106854, Nov. 2020.

[103] H.-Y. Liu, C.-J. Huang, and others, “Wideband MIMO antenna array design for future mobile devices operating in the 5G NR frequency bands n77/n78/n79 and LTE band 46,” IEEE Antennas Wireless Propagation Letters, vol. 19, no. 1, pp. 74–78, 2019.

[104] FrequencyCheck, “5G NR Band n78 (3500 MHz),” 2025. [Online]. Available: https://www.frequencycheck.com/bands/5g-nr-band-78-3500

[105] A. Pietrabissa and V. Suraci, “Wardrop equilibrium on time-varying graphs,” Automatica, vol. 84, pp. 159–165, 2017.

[106] A. Giuseppi and A. Pietrabissa, "Stability and Wardrop Equilibria of Noncooperative Routing With Time-Varying Load," IEEE Transactions on Automatic Control, vol. 68, no. 6, pp. 3814-3821, Jun. 2023.

[107] S. Fischer, L. Olbrich, and B. Vöcking, “Approximating Wardrop Equilibria with Finitely Many Agents,” Lecture Notes in Computer Science, Springer Berlin Heidelberg, 2008, pp. 238–252.

[108] J. G. Wardrop, “Road Paper: Some Theoretical Aspects of Road Traffic Research,” Proceedings of the Institution of Civil Engineers, vol. 1, no. 3, pp. 325–362, 1952.

[109] M. Beckmann, C. B. McGuire, and C. B. Winsten, “Studies in the Economics of Transportation,” Econometrica, vol. 26, no. 1, p. 183, 1958.

[110] A. Pietrabissa et al., “Lyapunov-Based Design of a Distributed Wardrop Load-Balancing Algorithm With Application to Software-Defined Networking,” IEEE Transactions on Control Systems Technology, vol. 27, no. 5, pp. 1924–1936, Sep. 2019.